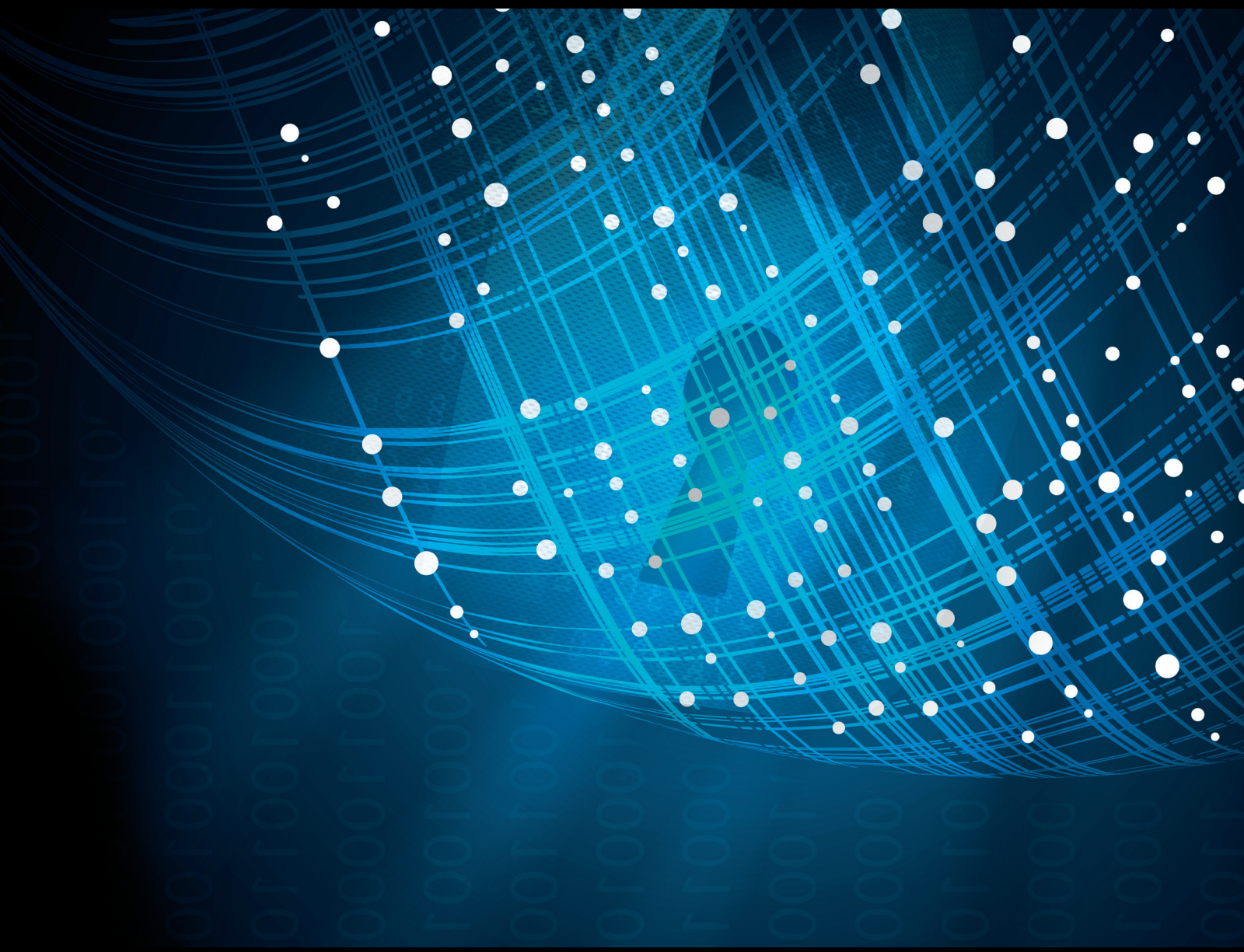
Security and Communication Networks

Privacy Protection and Security in Multimedia Processing and Artificial Intelligence

Lead Guest Editor: Zhihua Xia
Guest Editors: Athanasios V. Vasilakos, Neal N. Xiong, Xinpeng Zhang, and Yun-Qing Shi



Privacy Protection and Security in Multimedia Processing and Artificial Intelligence

Security and Communication Networks

Privacy Protection and Security in Multimedia Processing and Artificial Intelligence

Lead Guest Editor: Zhihua Xia
Guest Editors: Athanasios V. Vasilakos, Neal N. Xiong, Xinpeng Zhang, and Yun-Qing Shi

Copyright © 2020 Hindawi Limited. All rights reserved.

This is a special issue published in “Security and Communication Networks.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chief Editor

Roberto Di Pietro, Saudi Arabia

Associate Editors

Jiankun Hu, Australia
Emanuele Maiorana, Italy
David Megias, Spain
Zheng Yan, China

Academic Editors

Saed Saleh Al Rabaee, United Arab Emirates
Shadab Alam, Saudi Arabia
Goutham Reddy Alavalapati, USA
Jehad Ali, Republic of Korea
Jehad Ali, Saint Vincent and the Grenadines
Benjamin Aziz, United Kingdom
Taimur Bakhshi, United Kingdom
Spiridon Bakiras, Qatar
Musa Balta, Turkey
Jin Wook Byun, Republic of Korea
Bruno Carpentieri, Italy
Luigi Catuogno, Italy
Ricardo Chaves, Portugal
Chien-Ming Chen, China
Tom Chen, United Kingdom
Stelvio Cimato, Italy
Vincenzo Conti, Italy
Luigi Coppolino, Italy
Salvatore D'Antonio, Italy
Juhriyansyah Dalle, Indonesia
Alfredo De Santis, Italy
Angel M. Del Rey, Spain
Roberto Di Pietro, France
Wenxiu Ding, China
Nicola Dragoni, Denmark
Wei Feng, China
Carmen Fernandez-Gago, Spain
AnMin Fu, China
Clemente Galdi, Italy
Dimitrios Geneiatakis, Italy
Muhammad A. Gondal, Oman
Francesco Gringoli, Italy
Biao Han, China
Jinguang Han, China
Khizar Hayat, Oman
Azeem Irshad, Pakistan
M.A. Jabbar, India
Minho Jo, Republic of Korea
Arijit Karati, Taiwan
ASM Kayes, Australia
Farrukh Aslam Khan, Saudi Arabia
Fazlullah Khan, Pakistan
Kiseon Kim, Republic of Korea
Mehmet Zeki Konyar, Turkey
Sanjeev Kumar, USA
Hyun Kwon, Republic of Korea
Maryline Laurent, France
Jegatha Deborah Lazarus, India
Huaizhi Li, USA
Jiguo Li, China
Xueqin Liang , Finland
Zhe Liu, Canada
Guangchi Liu, USA
Flavio Lombardi, Italy
Yang Lu, China
Vincente Martin, Spain
Weizhi Meng, Denmark
Andrea Michienzi, Italy
Laura Mongioi, Italy
Raul Monroy, Mexico
Naghmeh Moradpoor, United Kingdom
Leonardo Mostarda, Italy
Mohamed Nassar, Lebanon
Qiang Ni, United Kingdom
Mahmood Niazi, Saudi Arabia
Vincent O. Nyangaresi, Kenya
Lu Ou, China
Hyun-A Park, Republic of Korea
A. Peinado, Spain
Gerardo Pelosi, Italy
Gregorio Martinez Perez, Spain
Pedro Peris-Lopez, Spain
Carla Ràfols, Germany
Francesco Regazzoni, Switzerland
Abdalhossein Rezai, Iran
Helena Rifà-Pous, Spain
Arun Kumar Sangaiah, India
Nadeem Sarwar, Pakistan
Neetesh Saxena, United Kingdom
Savio Sciancalepore, The Netherlands

De Rosal Ignatius Moses Setiadi, Indonesia
Wenbo Shi, China
Ghanshyam Singh, South Africa
Vasco Soares, Portugal
Salvatore Sorce, Italy
Abdulhamit Subasi, Saudi Arabia
Zhiyuan Tan, United Kingdom
Keke Tang, China
Je Sen Teh, Australia
Bohui Wang, China
Guojun Wang, China
Jinwei Wang, China
Qichun Wang, China
Hu Xiong, China
Chang Xu, China
Xuehu Yan, China
Anjia Yang, China
Jiachen Yang, China
Yu Yao, China
Yinghui Ye, China
Kuo-Hui Yeh, Taiwan
Yong Yu, China
Xiaohui Yuan, USA
Sherali Zeadally, USA
Leo Y. Zhang, Australia
Tao Zhang, China
Youwen Zhu, China
Zhengyu Zhu, China

Contents

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8847741, 9 pages
https://doi.org/10.1155/2020/8847741

Research Article

DRHNet: A Deep Residual Network Based on Heterogeneous Kernel for Steganalysis

Yang Xu, Zixi Fu, Guiyong Xu, Sicong Zhang, and Xiaoyao Xie

Key Laboratory of Information and Computing Science of Guizhou Province, Guizhou Normal University, Guiyang 550001, China

Correspondence should be addressed to Sicong Zhang; gs.sczhang16@gzu.edu.cn

Received 31 July 2020; Revised 5 November 2020; Accepted 18 November 2020; Published 12 December 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Yang Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Convolutional neural networks as steganalysis have problems such as poor versatility, long training time, and limited image size. For these problems, we present a heterogeneous kernel residual learning framework called DRHNet—Dual Residual Heterogeneous Network—to save time on the networks during the training phase. Instead of using the image as an input of the network, we extract and merge the images into a feature matrix using the rich model and use the generated feature matrix as the real input of the network. The architecture we proposed has good versatility and can reduce the computation and the number of parameters while still getting higher accuracy. On BOSSbase 1.01, we evaluate the performance of DRHNet in the setting of the spatial domain and frequency domain. The preliminary experimental results show that DRHNet shows excellent steganalysis performance against the state-of-the-art steganographic algorithms.

1. Introduction

As the most commonly used scheme of modern steganography, the least significant bit (LSB) will inevitably change the correlation between adjacent pixels of the image and the correlation of adjacent pixels of the residual image (high-frequency component) of the image [1, 2]. Before the renaissance of the neural network, the mainstream steganographic analysis method extracts the statistic that can describe the correlation of adjacent pixels of the residual image as a steganographic analysis feature and then uses the machine learning tool to train the steganographic analysis classifier [3, 4].

Convolutional neural network (CNN) has been widely used in the field of image classification [5–7]. Since steganalysis can be regarded as a two-class problem for images, the goal is to determine whether an image is embedded with the ciphertext. Steganalysis began to use convolutional neural networks to attack steganography. Qian et al. [8] first proposed the application of convolutional neural networks to steganalysis. They described a neural network steganalyzer with a Gaussian activation function equipped with a fixed preprocessing high-pass KV filter. The high-pass KV filter was used to suppress the image content, thus improving the signal-to-noise ratio (SNR) between the stego signal and the host image. Ye et al. [9] proposed a new network in which rather than a random strategy, the weights in the first layer of the proposed CNN are initialized with the basic high-pass filter set used in the calculation of residual maps in the spatial rich model (SRM), which acts as a regularizer to suppress the image content effectively. To better capture the structure of embedding signals, which usually have extremely low SNR (stego signal to image content), a new activation function called truncated linear unit (TLU) is adopted in their CNN model. Boroumand et al. [10] described a deep residual architecture, SRNet, designed to minimize the use of heuristics and externally enforced elements that are universal in the sense that it provides state-of-the-art detection accuracy for both spatial domain and JPEG steganography. The key part of the proposed architecture is a significantly expanded front part of the detector that "computes noise residuals" in which pooling has been disabled to prevent suppression of the stego signal.

The problem with using neural networks as a steganographic analysis tool is that it is impossible to analyze larger sized images due to limitations in computer resources. And the versatility of such steganographic analysis tools is not good; that is, a network trained with a steganographic algorithm cannot analyze images with another steganographic algorithm. Finally, the training time of the neural network is too long.

In this paper, we propose a heterogeneous kernel residual learning framework to save time on the networks during the training phase. Experimental results show that the DRHNet detection error achieves less than 10% when using S-UNIWARD [11] as the steganography algorithm, and the payload is 0.4 bpp. In summary, we make the following contributions in this paper:

(i) We address the accuracy-computation-time problem by introducing a versatile deep residual learning for steganalysis.

(ii) Instead of using the image as an input to the network, we extract and merge the images into a feature matrix through the rich model and use the generated feature matrix as the real input of the network.

(iii) The heterogeneous kernel is used as the convolution kernel of the network that we proposed. The heterogeneous kernel is adopted to reduce network parameters and reduce computational complexity.

2. Preliminaries

2.1. Feature Selection Method. The spatial rich model (SRM) [12] is a typical image steganographic analysis method. It designs a wide variety of spatial high-pass filters and uses these filters to filter the image to obtain a rich variety of residual images; then, it separately counts the frequency of occurrence of each adjacent residual sample pattern in a residual image. The cooccurrence matrix of the residual image is obtained. Finally, the elements of the cooccurrence matrix are rearranged into vectors as a steganographic analysis feature. The JPEG rich model (JRM) is the image steganographic analysis method that is widely used in the JPEG domain. JRM is similar to SRM. The only difference is that the features of JRM consist of the second-order cooccurrence matrices of the block coefficients of JPEG and their residual. However, the features of SRM are composed of the fourth-order cooccurrence matrices of different kinds of filter residuals. SRM is used in the spatial domain and JRM is used in the JPEG domain. The images of JPEG format are the carriers of the steganographic algorithms in the frequency domain.

Ma et al. [13] proposed a general feature selection method based on decision rough set α-positive region reduction to reduce the dimension of steganalysis features. Their results show that the reduced feature set can obtain the detection ability comparable to the original feature set, which effectively decreases the computation cost.

2.2. The Deep Learning Method. Because of gradient vanishment/explosion, the deep networks are normally difficult to train. He et al. [14] proposed ResNets, which solve the problem of gradient vanishment/explosion in deeper networks. This means the deeper network can show better accuracy rather than degradation. However, better accuracy comes with much more computation and time.

To reduce computational complexity, Singh et al. [15] presented a deep learning architecture in which the convolution operation leverages heterogeneous kernels. They improved the convolution kernel and achieved 3× to 8× FLOPs based improvement in speed while still maintaining (and sometimes improving) the accuracy.

Currently, deep learning is widely applied to improve the steganalysis performance. Hu et al. [16] proposed a new self-seeking steganalysis method based on visual attention and deep reinforcement learning. The visual attention method selects a region from the image and deep reinforcement learning is utilized to yield a summary region. Then, the summary regions are adopted to replace the misclassified training images to improve the steganalysis performance. Their experimental results show that their method can achieve steganalysis performance comparable to the state-of-the-art steganographic detection algorithms.

3. DRHNet

The proposed network architecture is called DRHNet–Dual Residual Heterogeneous Network. The "residual" here has two meanings, one of which means that 34 layers of ResNet are used as the main network structure, and the other meaning is that the residual of the image is treated as the object. Firstly, we explain the method of preprocessing, that is, how to get the feature matrix, and the principle of discriminating embedded images with residuals and then demonstrate the architecture of the network. At last, we describe the details of the experiment.

3.1. Method and Principle. The steganographic embedding process makes subtle changes to the image, which is similar to introducing weak noise (stealth noise) into the image. At the same time, the steganographic embedding process not only changes the adjacent pixel correlation of the natural image but also changes the adjacent pixel correlation of the residual image (noise component) of the natural image. SRM and other residual image based steganographic analysis methods [17, 18] model the residual image instead of directly modeling the image itself, mainly to weaken the interference of the image content on the steganalysis feature.

The residual of the cover image or stego image is extracted with k high-pass filters to form k submodels. Then, quantize, round, and truncate each submodel and extract the cooccurrence matrix in both horizontal and vertical directions. At this time, $2k$ cooccurrence matrices are generated for each picture. Cooccurrence matrices with similar properties are symmetrically merged and all elements are rearranged into feature vectors. At this point, the feature has been obtained, and its form is as follows [12]:

$$\overrightarrow{F_{k_{c_x}}} \leftarrow \text{Range}\left(\text{Merge}\left(M_h\left(c_x\right), M_v\left(c_x\right)\right)\right), \tag{1}$$

where $\overrightarrow{F_{k_{c_x}}}$ is the feature of the x^{th} cover image c_x calculated by using the k^{th} submodel. Merge$(\cdot)$ is to make two matrices merge to be one by combining elements having the same or similar statistical laws in the horizontal cooccurrence matrices $M_h(\cdot)$ and vertical one $M_v(\cdot)$. Range$(\cdot)$ is a function to rearrange the merged matrices into a feature vector. Among them, $x = 1, 2, \ldots, z$, where z is the last image of the train set; the feature of the stego image s_x can also be calculated. $c_x, s_x \in \mathbb{Z}^{n_1 \times n_2}$ are spatial domain images of size $n_1 \times n_2$ in which each value in the matrix is between 0 and 255.

$M_h(\cdot)$ can be obtained by the following formula [12], and $M_v(\cdot)$ can be obtained in the same way:

$$M_h\left(c_{x_{i,j}}\right) \leftarrow \text{Extr}_{h_d}\left(\text{Trunc}_T\left(\text{Round}\left(\frac{\text{HP}_k\left(c_{x_{i,j}}\right)}{q}\right)\right)\right), \tag{2}$$

where the positive vertex q is a quantization factor and the positive integer T is a truncation threshold; there are two important parameters that affect the dimensionality and steganalysis performance of SRM features. d is the order of the symbiotic matrix. If d is too large, sparse features will appear. If d is too small, the statistical diversity is not rich enough. $\text{HP}_k(\cdot)$ demonstrates the residual extracted by the kth high-pass filter; the specific definition is as follows:

$$\text{HP}_k(\cdot) \leftarrow \widehat{x}_{i,j}\left(\mathcal{N}_{i,j}\right) - \partial x_{i,j}, \tag{3}$$

where $x \in \mathbb{Z}^{n_1 \times n_2}$ is the pixel value of the cover or stego image at (i, j), $\text{HP}_k(\cdot) \in \mathbb{R}^{n_1 \times n_2}$ is the residual value of the residual f obtained by using a high-pass filter on the cover or stego image at (i, j), and $\partial \in \mathbb{Z}$ is the coefficient before the pixel value $x_{i,j}$. $\mathcal{N}_{i,j}$ is the pixel value of $x_{i,j}$ neighborhood $x_{i,j} \notin \mathcal{N}_{i,j}$, and $\left\{x_{i,j}, \mathcal{N}_{i,j}\right\}$ is the support set of image residuals. $\widehat{x}_{i,j}(\cdot)$ is the calculation result of the correlation between $x_{i,j}$ and it is the neighborhood $\mathcal{N}_{i,j}$ on different filters. Round$(\cdot)$ means rounding up by element, and $\text{Trunc}_T(\cdot)$ means a truncation operation by element. $\text{Extr}_{h_d}(\cdot)$ extracts the residual as a cooccurrence matrix. The common filters used in SRM are shown in Figure 1.

We choose $q = 0.5, 1, 2$, $T = 2$, and $d = 4$ and use all the merge rules designed by SRM. The obtained SRM feature instance is called the SRMQ3 (SRM feature using 3 kinds of quantization factors). It has 106 features, 17 of which are 338-dimensional features and 89 of which are 325-dimensional features. The dimension of the RMQ3 feature is $338 \times 17 + 325 \times 89 = 34671$. We use 0, 0 as the segmentation between each feature to fill it into a feature matrix of 187×187 , and null values after the last feature in the matrix are filled with 0. It is defined as follows:

$$\text{MF}_{c_x} \leftarrow \left[\overrightarrow{F_{1_{c_x}}}, t0, 0n, q\overrightarrow{F_{2_{c_x}}}\, h_{,0}, 0x, 7 \ldots CF\overrightarrow{_{106_{c_x}}}; , 0, \ldots, 0\right]. \tag{4}$$

By observing the feature vectors, we find that there are no elements with a value of 1 in the vector. We intended to split the feature vectors by "1, 1″, but considering that the maximum pooling is used in the subsequent network design, this will cause the network to train the separator we set as an important parameter, so, finally, we use "0, 0″ as the separator.

After obtaining the feature matrix of the cover image MF_{c_x} and the stego image MF_{s_x}, our goal is to use DRHNet to train a mapping Map$(\cdot)$ based on the difference between them, so that the mapping satisfies the following equation:

$$\begin{cases} \text{Map}\left(\text{MF}_{s_x}\right) = 1, \\ \text{Map}\left(\text{MF}_{c_x}\right) = 0, \end{cases} \tag{5}$$
$$\text{Map}(\cdot) \leftarrow \text{DRHNet}\left(\text{MF}_{c_x}, \text{MF}_{s_x}\right).$$

As discussed in Section 2, the feature extraction procedure of JRM is similar to SRM. The difference is that the features of JRM consist of the second-order cooccurrence matrices of the block coefficients of JPEG and their residual. JRM will double the feature dimension through Cartesian calibration, which produces 22510 features. Finally, a 151×151 feature matrix is generated.

3.2. Dual Residual Heterogeneous Network Architecture

3.2.1. Deep Residual Network. The structure used in this paper is similar to the 34-layer structure of ResNet [14]. We also adopt batch normalization (BN) [19] after each convolution and before ReLU [20]. The difference is that we added the SRM-Extract-Merge (SRMEM) layer between the image and the first convolutional layer of DRHNet for steganographic analysis of the spatial domain; we added the JRM-Extract-Merge (JRMEM) layer between the image and the first convolutional layer of DRHNet for steganographic analysis of the frequency domain. This reduces the data dimension that the network actually handles from 256×256 to 187×187 or 151×151. And the image represented by the feature matrix is no longer the content of the image. On the contrary, it is the statistical feature of the image residual, so it is more abstract. In addition, since Adamx [21] can reach convergence faster than stochastic gradient descent (SGD), we use Adamx as the optimizer to replace SGD.

The structure of the network is shown in Figure 2. The layered structure in the figure is not just a layer of convolution, but a convolution block containing two layers of convolution. The DRHNet network parameter settings are shown in Table 1. Please note that the DRHNet used for the steganographic analysis of the spatial domain is known as S-DRHNet in the following sections and the DRHNet used for the steganographic analysis of the frequency domain is known as J-DRHNet. The S-DRHNet and J-DRHNet share similar network architecture. The only difference is that they own different feature extraction layers. The S-DRHNet adopts the SRMEM as the feature extraction layer and the J-DRHNet uses the JRMEM as the feature extraction layer.

3.2.2. Heterogeneous Kernel. Another difference of DRHNet compared to ResNet is that the HetConv [15] is used as the convolution kernel, instead of the conventional convolution kernel. The channel is filled in the order of a 3×3 and three

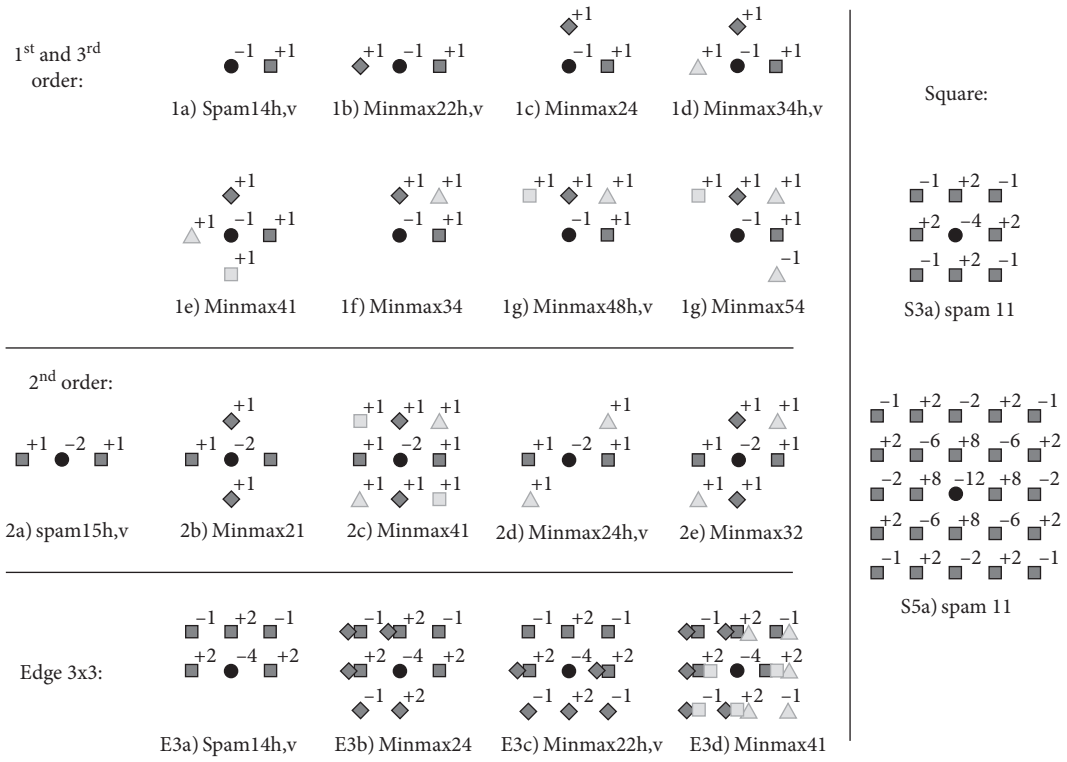


Figure 1: The commonly used filters of SRM.

1×1 convolution kernels. The convolution kernel of the next convolutional layer remains in this order, but the overall arrangement is shifted to the right by a convolution kernel. The structure of the DRHNet convolution kernel is shown in Figure 3.

It can be seen that there are two convolutional layers in the convolution block, each layer consisting of 64 convolution kernels of 3×3 and 1×1 sizes, arranged and offset in the above order. Using HetConv instead of a conventional convolution kernel can reduce network parameters and reduce computational complexity.

4. Experimental Results and Analysis

All experiments in this paper were evaluated and contrasted on BOSSbase 1.01, which contains 10,000 grayscale images with a size of 512×512. The experimental environment for this article is a host with an NVIDIA GeForce 1080 Ti graphics card and an Intel i7-9700 CPU. The more pixels image has, the more information it can embed, as well as the higher the computational complexity in the steganographic analysis process. This does not affect the performance of DRHNet, regardless of which size of the image will be extracted in the preprocessing into a feature matrix of 187×187 or 151×151. To facilitate comparison with the other steganalysis methods, we resized all the images into 256×256. In the setting of the spatial domain, WOW [22], S-UNIWARD [11], and MiPOD [23] are used as steganographic algorithms to embed ciphertext in the image. The SCA-TLU-CNN [9] and the SRNet [10] are utilized as the competing steganographic analysis method to be compared with S-DRHNet. In the setting of the frequency domain, the UED [24] and J-UNIWARD [11] are chosen as the steganographic algorithms. Because SRNet can also be used in the frequency domain, we compare J-DRHNet with SRNet to evaluate the effectiveness of DRHNet in the setting of the frequency domain. The payload of each steganographic algorithm is set from 0.2 to 0.4 bits per pixel (bpp), respectively. For the dataset, 5000 cover-stego image pairs, 10000 images, were randomly selected as the training set. For clarity of expression, the following are all counted in cover-stego image pairs. As the same, 2500 were selected as the validation set, and the remaining 2500 combined with the 2500 randomly selected unembedded image pairs were used as the test set. The experimental epoch of this paper is 100 times, the minibatch size of the training set is 20 cover-stego image pairs, and the validation set is 10. The first 80 epochs of the experiment trained the network at a learning rate of 0.001 and trained it at 0.0001 for the last 20 epochs.

4.1. DRHNet Steganalysis Level Experiment. We adopt the detection error P_{error} as the evaluation criteria. The definition of P_{error} is as follows:

$$P_{\text{error}} = \frac{n_{\text{FP}} + n_{\text{FN}}}{n}, \tag{6}$$

where n is the total number of images in the test set, and n_{FP} and n_{FN} are the number of false positive and false negative errors in the machine learning concept.

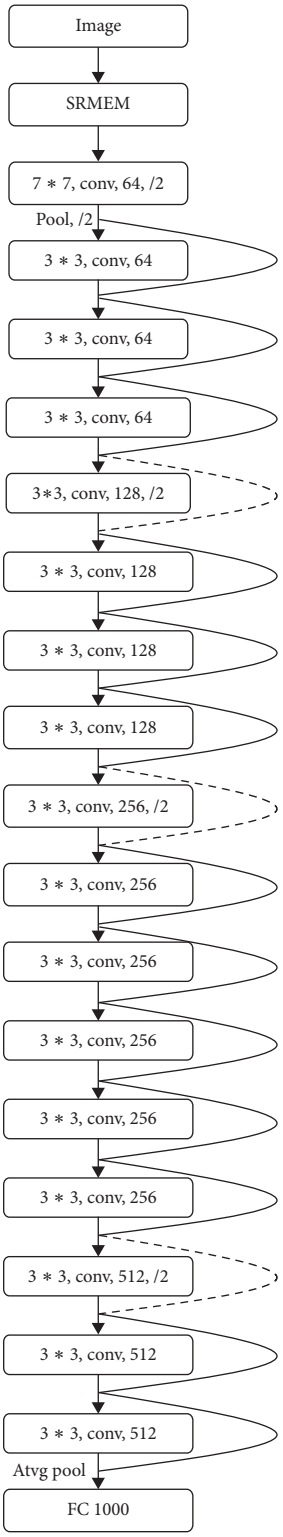


Figure 2: Network architectures for DRHNet.

4.1.1. The Performance of S-DRHNet in the Spatial Domain. The performance of S-DRHNet in the spatial domain is shown in Figures 4–6. It is observed in Figures 4–6 that SRNet's [10] P_{error} is less than 1% lower than our proposed structure when applying the WOW steganography algorithm and with a payload of 0.4 bpp; SCA-TLU-CNN [9] has a detection error rate of 6% lower than the DRHNet when applying the S-UNIWARD steganography algorithm and with a payload of 0.2 bpp. Apart from the above two situations, DRHNet generally has better performance than the other two steganographic analysis networks. And it can be seen that as the payload increases, P_{error} of DRHNet decreases faster than the other two networks.

The ROC curves of SCA-TLU-CNN, SRNet, and S-DRHNet against S-UNIWARD at 0.4 bpp are shown in Figure 7. The AUC of S-DRHNet, SRNet, and SCA-TLU-CNN are 0.97, 0.94, and 0.92. The accuracy of S-DRHNet against S-UNIWARD is higher than SRNet and SCA-TLU-CNN at the high payload.

The training of DRHNet was iterated 100 times in total. The learning rate was set to 0.001 in the first 80 iterations and 0.0001 in the last 20 iterations. Figure 8 shows the changes in detection error during training and validation of S-DRHNet. The data are obtained on S-UNIWARD at payload 0.4 bpp.

Table 1: Parameter s for DRHNet.

Layer name	Output size	Convolution kernel
Conv_1	94×94	$7 \times 7, 64$
Max pool		
Conv_2	47×47	$\left[\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix} \times 3\right]$
Conv_3	24×24	$\left[\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix} \times 4\right]$
Conv_4	12×12	$\left[\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix} \times 6\right]$
Conv_5	6×6	$\left[\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{pmatrix} \times 3\right]$
Avg pool		
Fc 1000		
Softmax		

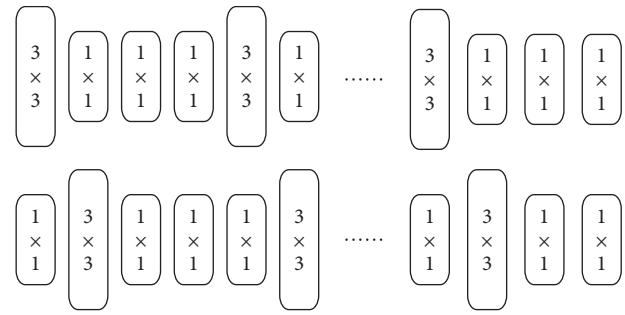


Figure 3: $3 \times 3 \times 64$ convolutional block structure in DRHNet.

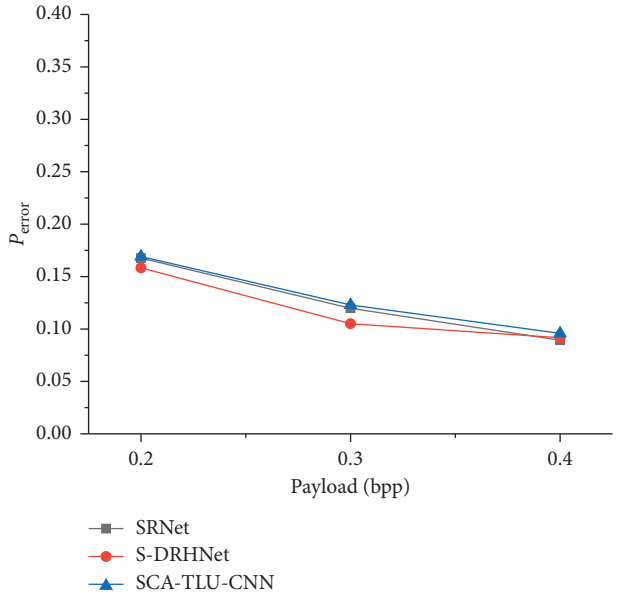


Figure 4: Detection error P_{error} of SCA-TLU-CNN, SRNet, and S-DRHNet against WOW.

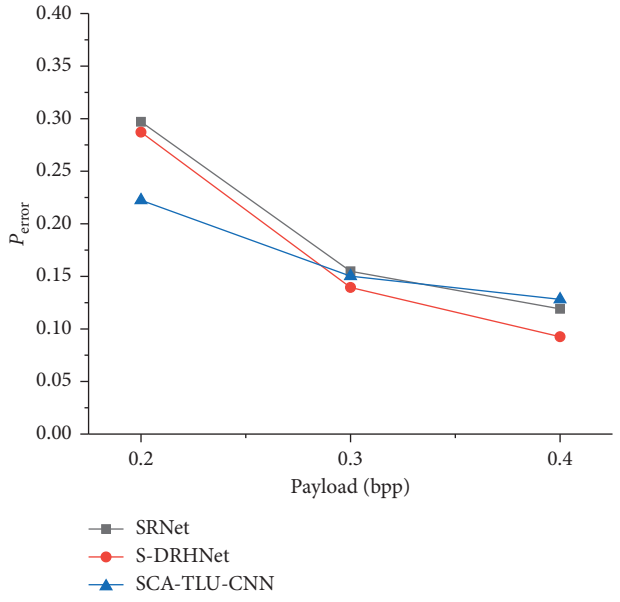


Figure 5: Detection error P_{error} of SCA-TLU-CNN, SRNet, and S-DRHNet against S-UNIWARD.

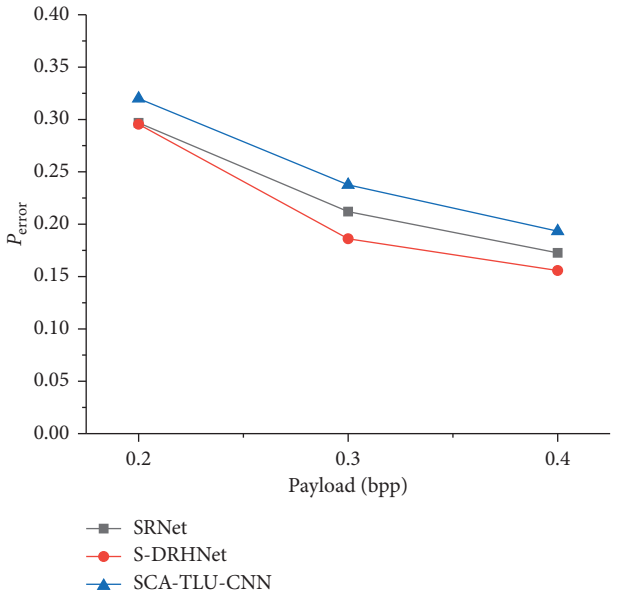


Figure 6: Detection error P_{error} of SCA-TLU-CNN, SRNet, and S-DRHNet against MiPOD.

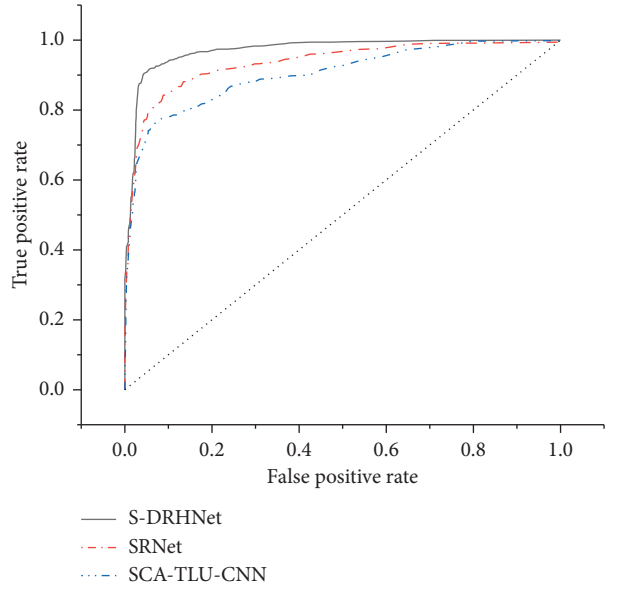


Figure 7: ROC curves of SCA-TLU-CNN, SRNet, and S-DRHNet for S-UNIWARD at 0.4 bpp.

Figure 9 shows the progression of the training and validation loss when training S-DRHNet in the same circumstance. Because the curves of detection error and loss of J-DRHNet during training and validation are similar to S-DRHNet, we only show the corresponding curves of S-DRHNet here.

4.1.2. The Performance of J-DRHNet in the Frequency Domain. The performance of J-DRHNet in the frequency domain is shown in Figures 10 and 11. The J-DRHNet shows better performance than SRNet against J-UNIWARD. When the payload is high, the J-DRHNet shows better performance than SRNet against UED.

The ROC curves of SRNet and J-DRHNet against UED at 0.4 bpp are shown in Figure 10. We can observe from Figure 12 that the accuracy of J-DRHNet against UED is close to that of SRNet.

4.2. DRHNet Steganalysis Generality Experiment. Using the feature matrix extracted by SRM or JRM as the input of the network makes DRHNet have good versatility. To evaluate the versatility of DRHNet, we adopt one steganographic

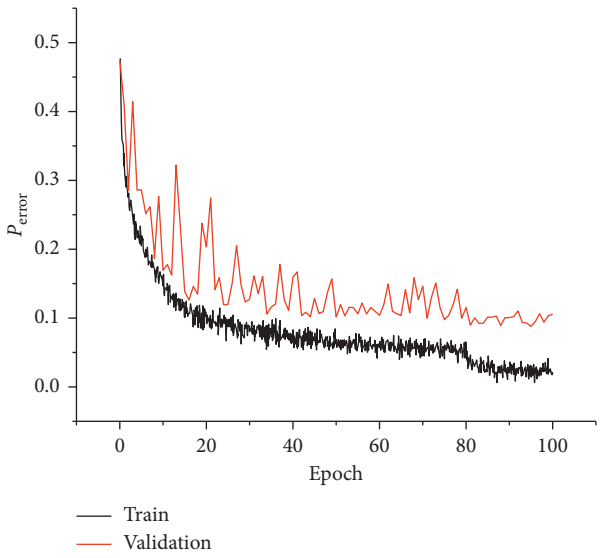


Figure 8: S-DRHNet's training and validation detection error P_{error} for S-UNIWARD at 0.4 bpp.

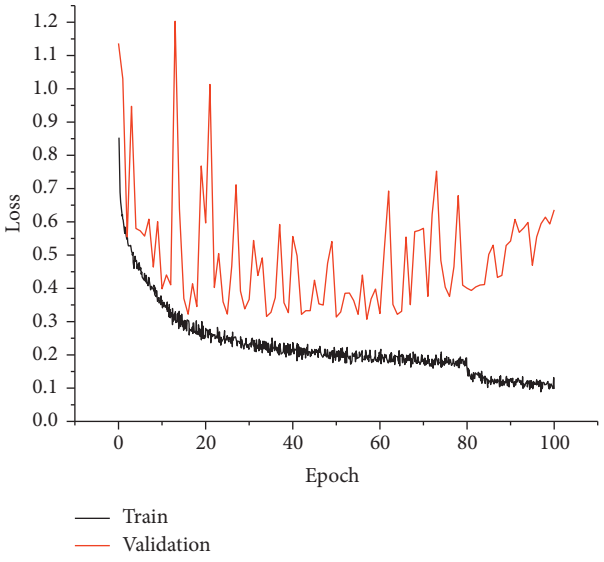


Figure 9: S-DRHNet's training and validation loss for S-UNIWARD at 0.4 bpp.

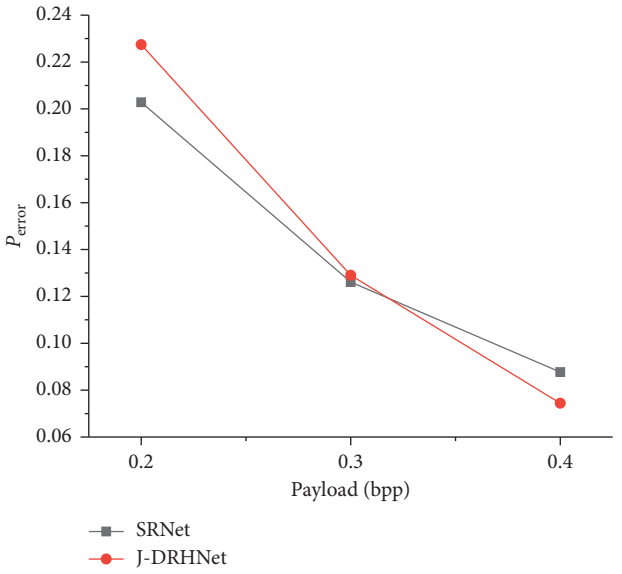


Figure 10: Detection error P_{error} of SRNet and J-DRHNet against UED.

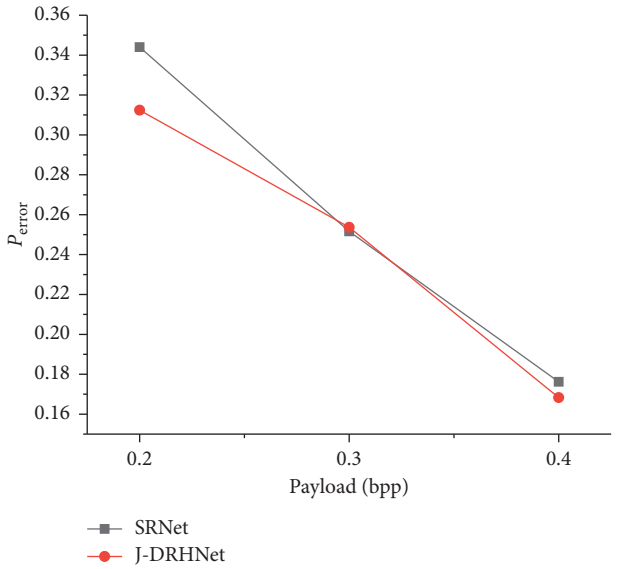


Figure 11: Detection error P_{error} of SRNet and J-DRHNet against J-UNIWARD.

algorithm to generate steganographic images as the training set and validation set. Then, the steganographic images generated by another steganographic algorithm are used as the test set. We observe the detection error of DRHNet in this situation. Because the J-DRHNet and S-DRHNet own similar architecture except for the feature extraction modules, we only show the results of the cross test of S-DRHNet here. The detection error of the cross test of S-DRHNet is shown in Table 2. The detection error obtained by using the same steganography algorithm for the test set and training set is shown in bold in Table 2, and the error rate of the cross test is shown in normal font. The detection error obtained by adopting different steganographic algorithms is generally 0.01–0.03 higher than that of using the same steganography algorithm. From the results in Table 2, we can conclude that the S-DRHNet shows good versatility against different steganographic methods.

4.3. The Time Consumption and Computational Complexity of DRHNet. DRHNet also reduces time consumption while improving accuracy. Table 3 shows the parameters, computational complexity, and time consumption of the above four types of steganalysis networks. The network structure of

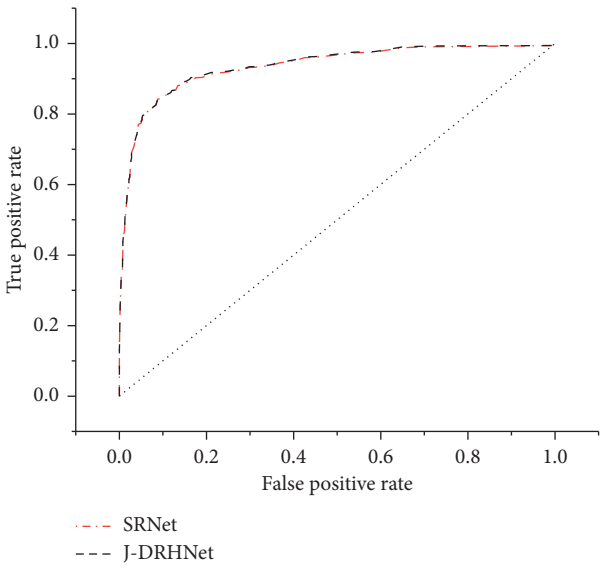


Figure 12: ROC curves of SRNet and J-DRHNet against UED at 0.4 bpp.

Table 2: The detection error of the cross test of S-DRHNet.

Train\test	Payload (bpp)	WOW	S-UNIWARD	MiPOD
WOW	0.2	**0.1584**	0.1807	0.1802
	0.3	**0.1050**	0.1268	0.1341
	0.4	**0.0917**	0.1099	0.1125
S-UNIWARD	0.2	0.2943	**0.2872**	0.2960
	0.3	0.1496	**0.1395**	0.1561
	0.4	0.1083	**0.0926**	0.1069
MiPOD	0.2	0.3182	0.3297	**0.2954**
	0.3	0.2049	0.2088	**0.1861**
	0.4	0.1609	0.1653	**0.1557**

Table 3: The computational complexity of the three networks.

Network	Input resolution	Parameters (M)	FLOPs (G)	Hours
SCA-LTU-CNN	256×256	10.52	1.74	44
SRNet	256×256	12.18	2.09	53
S-DRHNet	$\mathbf{256 \times 256}$	**9.88**	**1.67**	**41**
J-DRHNet	$\mathbf{256 \times 256}$	**9.88**	**1.59**	**37**

SCA-TLU-CNN has ten layers, and the SRNet has twelve layers, so ResNet is higher than SCA-TLU-CNN in the three metrics mentioned in Table 3. Although the DRHNet designed in this paper is a 34-layer network structure, the application of HetCov as a convolution kernel greatly reduces the parameters of the network, resulting in shortened computational complexity and time consumption than the other two networks, while still ensuring considerable accuracy. Because the feature matrix of J-DRHNet is smaller than S-DRHNet, the time consumption of J-DRHNet is lower than that of S-DRHNet.

5. Conclusion

In this paper, a deep neural network with high accuracy and low time consumption is proposed for steganalysis. The SRMEM and JRMEM layers are used to extract features from the original images and combine them into a feature matrix, which provides versatility for the steganographic analysis method while reducing the network dimension. Furthermore, we select HetConv as the convolution kernel of the DRHNet network, which greatly reduces the computational complexity while ensuring accuracy. By combining different feature preprocessing modules, that is, SRMEM and JRMEM, DRHNet can be flexibly applied in both spatial domain and frequency domain. The preliminary experimental results show that the DRHNet shows excellent steganalysis performance in both the spatial domain and frequency domain. The DRHNet outperforms the existing state-of-the-art steganographic analysis algorithms such as SCA-TLU-CNN and SRNet and shows excellent performance against the state-of-the-art steganographic methods such as S-UNIWARD, J-UNIWARD, and WOW. The image embedded in ciphertext may be compressed during transmission.

The cross test of SRNet and J-DRHNet will be further studied and verified in the following research. How to extract the embedded image after compression will be our next research direction.

Data Availability

The software code used to support the findings of this study is available from the corresponding author upon request. The data used to support the findings of this study are available at http://agents.fel.cvut.cz/stegodata/.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Science Foundation of China under Grant no. U1831131, the Special Funds of Central Government of China for Guiding Local Science and Technology Development under Grant no. [2018]4008, and the Science and Technology Planned Project of Guizhou Province, China, under Grant no. [2020]2Y013.

References

[1] Z. Xia, X. Wang, X. Sun, and B. Wang, "Steganalysis of least significant bit matching using multi-order differences," *Security and Communication Networks*, vol. 7, no. 8, pp. 1283–1291, 2014.

[2] S. Gupta, A. Goyal, and B. Bhushan, "Information hiding using least significant bit steganography and cryptogrammmphy," *International Journal of Modern Education and Computer Science*, vol. 4, no. 6, p. 27, 2012.

[3] C. Chen and Y. Q. Shi, "JPEG image steganalysis utilizing both intrablock and interblock correlations," in *Proceedings of the*

2008 IEEE International Symposium on Circuits and Systems, pp. 3029–3032, IEEE, Washington, DC, USA, June 2008.

[4] Q. Liu, "Steganalysis of DCT-embedding based adaptive steganography and YASS," in *Proceedings of the Thirteenth ACM Multimedia Workshop On Multimedia and Security*, pp. 77–86, ACM, New York, USA, September 2011.

[5] H. Li, Z. Lin, X. Shen et al., "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, Boston, MA, USA, June 2015.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097–1105, 2012.

[7] M. Rastegari, V. Ordonez, J. Redmon et al., "XNOR-Net: Imagenet Classification Using Binary Convolutional Neural networks," in *Proceedings of the European Conference on Computer Vision*, pp. 525–542, Springer, Amsterdam, The Netherlands, October 2016.

[8] Y. Qian, J. Dong, W. Wang et al., "Deep learning for steganalysis via convolutional neural networks," *International Society for Optics and Photonics*, vol. 9409, Article ID 94090J, 2015.

[9] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.

[10] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2018.

[11] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 1, no. 1, 2014.

[12] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[13] Y. Ma, X. Luo, X. Li, Z. Bao, and Y. Zhang, "Selection of rich model steganalysis features based on decision rough set α-positive region reduction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 336–350, 2019.

[14] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[15] P. Singh, V. K. Verma, P. Rai et al., "HetConv: heterogeneous kernel-based convolutions for deep CNNs," 2019, http://arxiv.org/abs/1903.04120.

[16] D. Hu, S. Zhou, Q. Shen, S. Zheng, Z. Zhao, and Y. Fan, "Digital image steganalysis based on visual attention and deep reinforcement learning," *IEEE Access*, vol. 7, pp. 25924–25935, 2019.

[17] T. Pevny, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215–224, 2010.

[18] K. Jan and J. Fridrich, "Steganalysis in high dimensions: fusing classifiers built on random subspaces," *Proceedings of SPIE-The International Society for Optical Engineering*, vol. 7880, no. 1, pp. 181–97, 2011.

[19] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, http://arxiv.org/abs/1502.03167.

[20] B. Xu, N. Wang, T. Chen et al., "Empirical evaluation of rectified activations in convolutional network," 2015, http://arxiv.org/abs/1505.00853.

[21] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[22] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *Proceedings 2012 IEEE International workshop on information forensics and security (WIFS)*, pp. 234–239, IEEE, Costa Adeje, Spain, December 2012.

[23] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 221–234, 2015.

[24] L. Guo, J. Ni, and Y. Q. Shi, "Uniform embedding for efficient JPEG steganography," *IEEE Transactions on In-Formation Forensics and Security*, vol. 9, no. 5, pp. 814–825, 2014.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8889122, 15 pages
https://doi.org/10.1155/2020/8889122

Research Article

Exposing Spoofing Attack on Flocking-Based Unmanned Aerial Vehicle Cluster: A Threat to Swarm Intelligence

Xinyu Huang[1], **Yunzhe Tian**[1], **Yifei He**[1], **Endong Tong**[1], **Wenjia Niu**[1], **Chenyang Li**[1], **Jiqiang Liu**[1] and **Liang Chang**[2]

[1]*Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuan Village, Haidian District, Beijing 100044, China*
[2]*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*

Correspondence should be addressed to Endong Tong; edtong@bjtu.edu.cn and Wenjia Niu; niuwj@bjtu.edu.cn

Received 10 August 2020; Revised 4 November 2020; Accepted 29 November 2020; Published 10 December 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Xinyu Huang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid development of wireless communication technology and intelligent mobile devices, unmanned aerial vehicle (UAV) cluster is becoming increasingly popular in both civilian and military applications. Recently, a swarm intelligence-based UAV cluster study, aiming to enable efficient and autonomous collaboration, has drawn lots of interest. However, new security problems may be introduced with such swarm intelligence. In this work, we perform the first detailed security analysis to a kind of flocking-based UAV cluster with 5 policies, an upgrade version of the well-known Boids model. Targeting a realistic threat in a source-to-destination flying task, we design a data spoofing strategy and further perform complete vulnerability analysis. We reveal that such design and implementation are highly vulnerable. After breaking through the authentication of ad hoc on-demand distance vector (AODV) routing protocol by rushing attack, an attacker can masquerade as the first-arrival UAV within a specific scope of destination and generate data spoofing of arrival status to the following UAVs, so as to interfere with their normal flying paths of destination arrival and cause unexpected arrival delays amid urgent tasks. Experiments with detailed analysis from the 5-UAV cluster to the 10-UAV cluster are conducted to show specific feature composition-based attack effect and corresponding average delay. We also discuss promising defense suggestions leveraging the insights from our analysis.

1. Introduction

UAVs have been widely used in military and civilian fields due to their low cost and high flexibility. With the rapid development of wireless communication technology and mobile devices, the unmanned aerial vehicle cluster is becoming increasingly popular in tasks of monitoring, search, and rescue in a dangerous environment. Recently, swarm intelligence in the field of AI has been employed in the UAV cluster to provide an efficient and effective collaboration of UAVs that does not require any external remote guidance or any central-node UAV control, amid complex tasks difficult for a single UAV. Thus, the multi-UAV cluster aims to realize an autonomous mechanism by imitating different kinds of swarm including a flock of birds and a swarm of bees. Although such swarm intelligence has shown its potential to provide an efficient and effective UAV collaboration, few studies focus on the security issues brought by swarm intelligence. Thus, it is highly important to deep assess the security of swarm intelligence-based UAV cluster and discover a possible vulnerability that can be maliciously exploited.

In this paper, we focus on the swarm intelligence of flocking inspired by birds. There is a classical model named Boids model [1] proposed by Reynolds in 1986, of which there is no central node and each UAV is autonomous through perceiving its neighbors' information within a certain range nearby [2, 3]. Each UAV's decision-making is based on three simple policies: dispersion, alignment, and cohesion. Such a collaboration mechanism of policies derives from the birds' flocking; thus, it is also called flocking in short. In this work, we choose the latest upgraded version of

the Boids model including five policies: alignment, homing, cohesion, dispersion, and following, which is more complete for flocking [4]. In the rest of this paper, the flocking is always referred to as this 5-policy flocking. Also, we target a source-to-destination flying task for cluster flying and perform the first detailed security analysis to the flocking-based UAV cluster flying. To the best of our knowledge, no similar work has focused on it.

Facing cyberattacks, it is highly important to firstly understand potential security vulnerabilities so that they can be actively resolved before real large-scale deployment. As the first step, we implement the 5-policy flocking algorithm. Through tuning a set of parameters, we successfully realize the flocking to maintain a relatively stable formation for UAV cluster flying. Then we identify basic security challenges, involved authentication, data spoofing, and so on, aiming to understand whether the current design or implementation of the flocking algorithm for UAV cluster is vulnerable and why it is vulnerable and hope to provide insights on how to fundamentally protect it before large-scale UAV cluster deployment.

For simplicity, we assume that the UAVs in the cluster move at a constant speed and use the mobile ad hoc network (MANET) [5] to exchange data with other UAVs within the specific range of wireless sensing. The data includes GPS location, speed direction, and arrival status. The only attack requirement that we limit is that there is just one UAV masqueraded to send spoofed data to other UAVs. This can be realized by a rushing attack to break through the authentication of the ad hoc on-demand distance vector (AODV) routing protocol [6, 7]. As reported in former work, such compromise can be performed physically [8], wirelessly [9], or through malware [10, 11]. Thus, only rushing attack one first-arrival UAV for data spoofing of arrival status is closer to the attack reality and ensures that our analysis has highly practical implications.

In this work, we find that data spoofing of arrival status is very effective for autonomous flocking-based UAV cluster: through masqueraded UAV sending spoofed data to the following UAVs, the maximum percentage of delay can even reach up to nearly 50%, which completely subverts the advantage of the flocking, as a highly efficient algorithm. Figure 1 shows an attack snapshot of the simulation for 8-UAV flocking-based flying from the left sources to the right destinations (denoted as red circles, respectively) in farmland. The axes represent the flying coordinates in meters. Figure 1(a)) shows the attack place occurring to the first-arrival UAV. As shown in Figure 1(b)), during the attack, some UAVs' trajectories in the cluster have been affected (seen in the green circle) and caused an obvious delay to their destinations. We find that this is due to a vulnerability in the trade-off between security and formation stability: in the flocking algorithm, to maintain the stability of the cluster formation, once a UAV has reached its destination, its information will not be used in the other UAVs' decision-making anymore.

In our experiment, we find the following: (1) fixing the arrival determining threshold with 20 meters, the delay percentage of smaller UAV cluster is higher than that of a larger cluster, which appears from cluster speed 10 m/s to 20 m/s; (2) lower arrival determining threshold will cause an opposite result, that is, larger cluster having higher delay percentage; (3) fixing the cluster's speed, whatever the cluster size is, lower arrival determining threshold will cause a higher delay percentage; (4) the maximum delay percentage of 47.84% is obtained in the occasion of cluster size 9, cluster speed 16 m/s, and arrival determining threshold 12 meters.

According to our analysis, the current flocking algorithm design is highly vulnerable to data spoofing, causing the whole UAV cluster to delay its arrival to a large extent. We also discuss promising defense directions leveraging the insights from our analysis.

We summarize our contributions as follows:

(i) We perform the first security analysis of flocking-based UAV cluster flying. We formulate the problem with a highly realistic threat model, involved AODV link authentication, rushing attack, and data spoofing to UAV cluster and analyze the algorithm design to identify the data spoofing strategy.

(ii) Targeting the goal of causing unexpected arrival delays in flying task, we first perform vulnerability analysis to understand the attack effectiveness. We find that the flocking algorithm design and implementation are highly vulnerable, causing nearly 50% arrival delay in some cases.

(iii) Through massive experiments, we obtain detailed attack results under different cluster features, which helps to provide promising defense directions from our analysis and experimental results.

2. Background

In this section, we introduce the necessary background about the swarm intelligence in the UAV cluster and the flocking algorithm that we target.

2.1. UAV Swarm Intelligence. Swarming behavior is a common phenomenon in nature. Typical examples include flocks of birds migrating in formation, schools of fish parading in groups, colonies of ants working together, and colonies of bacteria that grow together. The common feature of these phenomena is that a certain number of autonomous individuals, through mutual collaboration and self-organization, present an orderly coordinated movement and behavior on the collective level [12]. In the early stage of research in this area, a lot of work focused on modeling and simulation of natural biological populations. Scholars use a large amount of experimental data to explore the influence of individual behavior and the relationship between individuals on the overall behavior of the group [13, 14].

In the field of UAV, the Boids model is naturally applied as the earliest swarm intelligence model, establishing and maintaining collision-free cohesive flocking which requires only three simple policies between idealistic agents: (1)

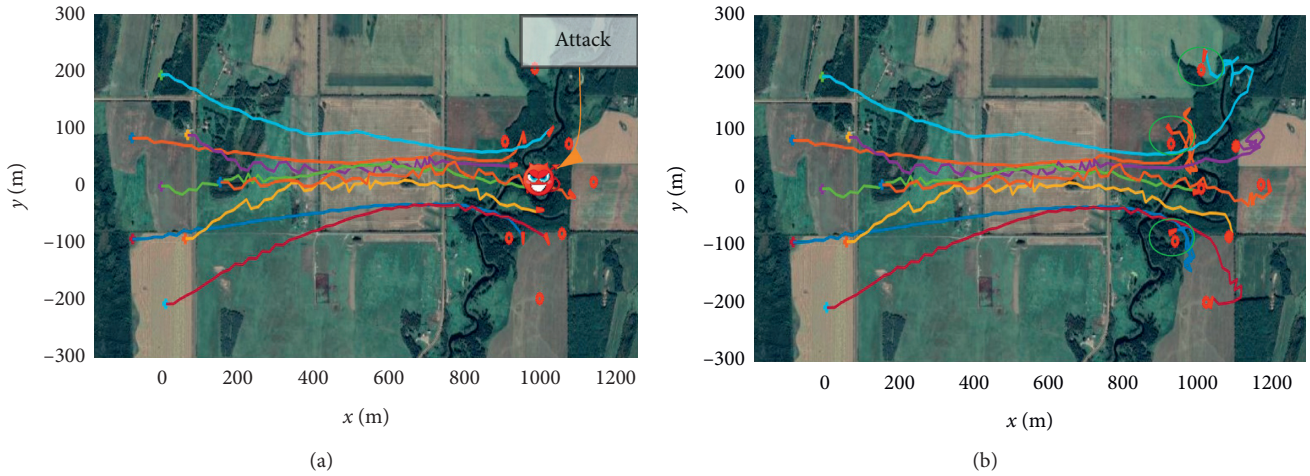


Figure 1: A snapshot of the simulation of 8-UAV flocking-based cluster flying. (a) The attack place to first-arrival UAV. (b) Arrival delays with affected trajectories in green circles.

gathering: make the agents in the entire group closely adjacent; (2) keeping distance: adjacent agents keep a safe distance; (3) motion matching: neighboring agents have the same motion state. This model roughly describes the movement characteristics of swarms in nature. Lately, Sharma and Ghose extended the Boids and proposed a swarm intelligence algorithm to avoid cluster collisions [4]. Our work is based on Sharma et al.'s algorithm to build a flocking-based UAV cluster as our target of security analysis.

2.2. Flocking Algorithm. In a UAV cluster, the equation of motion in the two-dimensional plane for a UAV can be represented as

$$\begin{cases} \dot{x}_i = \mathbf{v}\cos\theta_i, \\ \dot{y}_i = \mathbf{v}\sin\theta_i, \\ \dot{\theta}_i = \dfrac{\eta}{V}, \end{cases} \tag{1}$$

where $\mathbf{x}_i$, $\mathbf{y}_i$, θ_i are the x-axis coordinate, y-axis coordinate, and heading angle of the i-th UAV, respectively. $V = |\mathbf{v}|$ means UAV's speed and η is the acceleration. The detailed calculation is shown as follows:

$$\begin{aligned} \eta &= k\Delta\theta_i, \\ \Delta\theta_i &= w_1\left(\theta_{\text{req1}} - \theta_i\right) + w_2\left(\theta_{\text{req2}} - \theta_i\right) + \cdots + w_n\left(\theta_{\text{req}\,m} - \theta_i\right), \end{aligned} \tag{2}$$

where k is the acceleration constant and $\theta_{\text{req1}}, \theta_{\text{req2}}, \ldots, \theta_{\text{req}\,m}$ are the desired heading angles corresponding to different policies. We can vary the policy weights $w_1, w_2, \ldots, w_m$ to obtain different composite policies.

We target the flocking with five basic policies including cohesion, dispersion, following, homing, and alignment as shown in Table 1.

We use w_h, w_a, w_c, w_d, w_f to represent the weight of homing, alignment, cohesion, dispersion, and following policies, respectively.

Figure 2 shows the heading angle computation of a UAV in the cluster. The above basic policies can ensure that the self-organized UAV swarm remains stable and collision-free during the flight. However, when the UAV swarm arrives near the destination, if the arrived UAV is still involved in the angle calculations for other unreached UAVs, the directions of the unreached UAVs will be affected not to reach their destinations. Thus, once there is a UAV in the cluster that has reached its destination, it has to be timely excluded from the calculations.

Also, in general, to determine whether a UAV has arrived, we calculate the distance between the current location of the UAV and the destination, in which the specified parameter R is defined as an arrival determining the threshold. When the distance between the current location of the UAV and the destination is less than R, the UAV is determined to have reached its destination.

3. Threat Model

3.1. Communication. In the UAV cluster, MANET is a common-used communication network to support UAVs' communication (see Figure 3) [15]. There are two channels: data channel and protocol channel. In our experiment, we limit three data to transmit including heading angle, coordinate, and arrival status.

As we can see, MANET is an ad hoc decentralized type of wireless network that does not rely on a preexisting infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks [5]. Instead, each node participates in routing by forwarding data to other nodes, so the determination of which nodes forward data is made dynamically based on network connectivity and the routing algorithm in use [16]. Such wireless networks lack the complexities of infrastructure setup and administration

Table 1: Desired heading angles based on different policies in the flocking algorithm.

Policies	Desired heading angles	Notation
Cohesion	$\theta_{Ci} = \arctan((Y_{C_{\rho,i}} - y_i)/(X_{C_{\rho,i}} - x_i))$	$(X_{C_{\rho,i}}, Y_{C_{\rho,i}})$: the centroid of all UAVs in sensor range ρ
Dispersion	$\theta_{Di} = \arctan((y_i - Y_{C_{d,i}})/(X_{C_{d,i}} - x_i))$	$(X_{C_{d,i}}, Y_{C_{d,i}})$: the centroid of all UAVs within the $d_{\min}$ range
Following	$\theta_{Fi} = (\theta_{Ni} + \theta_{Ri})/2$ $\theta_{Ri} = \arctan((Y_{Ri} - y_i)/(X_{Ri} - x_i))$ $\theta_{Ni} = \arctan((Y_{Ni} - y_i)/(X_{Ni} - x_i))$	(X_{Ri}, Y_{Ri}): the coordinate of the randomly selected UAV (X_{Ni}, Y_{Ni}): the coordinate of the nearest UAV
Homing	$\theta_{Hi} = \arctan((Y_{Hi} - y_i)/(X_{Hi} - x_i))$	(X_{Hi}, Y_{Hi}): the destination coordinate
Alignment	$\theta_{Ai} = (1/n_i)\sum_{j=1}^{n_i} \theta_j$	θ_j: the heading angle of j-th UAV
Combined	$\theta_i' = \theta_i + \Delta\theta_i$ $\Delta\theta_i = w_h(\theta_{Hi} - \theta_i) + w_a(\theta_{Ai} - \theta_i) + w_c(\theta_{Ci} - \theta_i) + w_d(\theta_{Di} - \theta_i) + w_f(\theta_{Fi} - \theta_i)$	w_h, w_a, w_c, w_d, w_f: weights

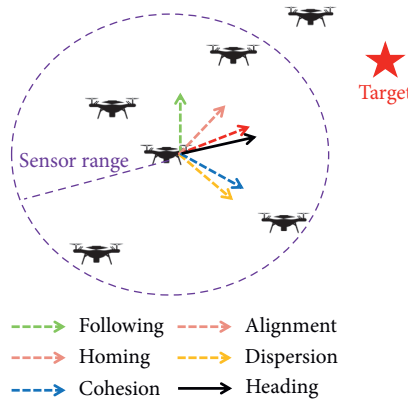


Figure 2: The heading angle of a UAV in its sensor range. By combining the 5 policies' weight, the heading angle is calculated according to vector addition.

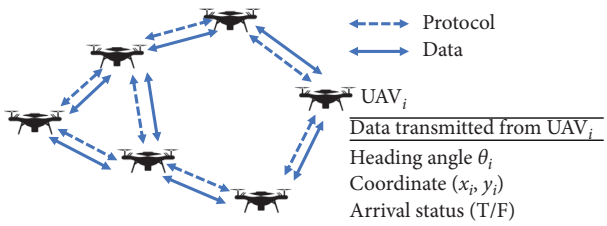


Figure 3: The MANET-based UAV communication. The transmitted data is limited to the heading angle, coordinate, and arrival status.

while enabling UAVs to create and join networks anytime efficiently.

Due to common attacks in MANET, there is also a communication-related threat to the flocking-based cluster [17], such as wormhole attack, rushing attack, joint attack, Sybil attack, denial of service attacks, and eavesdropping attacks. In this work, we only utilize one communication attack: the rushing attack.

Through the rushing attack, data spoofing can be performed. Previous work has demonstrated sensor spoofing attacks to single UAV, such as GPS spoofing [18, 19] to misguide UAV's trajectory and optical spoofing [20, 21] to gain an implicit control channel. Thus, it is confident to assume that the attacker can perform data spoofing in any type, physically [8], wirelessly [9], or through malware [10, 11].

3.2. AODV Link Authentication. MANET is inherently vulnerable to attack due to its fundamental characteristics, such as open medium, distributed nodes, the autonomy of nodes participation in a network (nodes can join and leave the network on its will), lack of centralized authority which can enforce security on the network, distributed coordination, and cooperation [6].

Many of the routing protocols devised for use in MANET have their individual character and rules. The most widely used routing protocol is AODV, which relies on the individual node's cooperation in establishing a valid routing table. However, it only enables a weak link authentication based on all nodes trusted in the network. In the AODV protocol, each node first establishes a valid route to the destination before transmitting its data. Sender node broadcasts an RREQ (route request) message to neighbors and valid route replies with RREP (route reply) with proper route information. The AODV protocol uses a duplicate suppression mechanism to limit the route request and reply chatter in the network (see Figure 4). In Figure 4(a)), there are three RREQ messages.

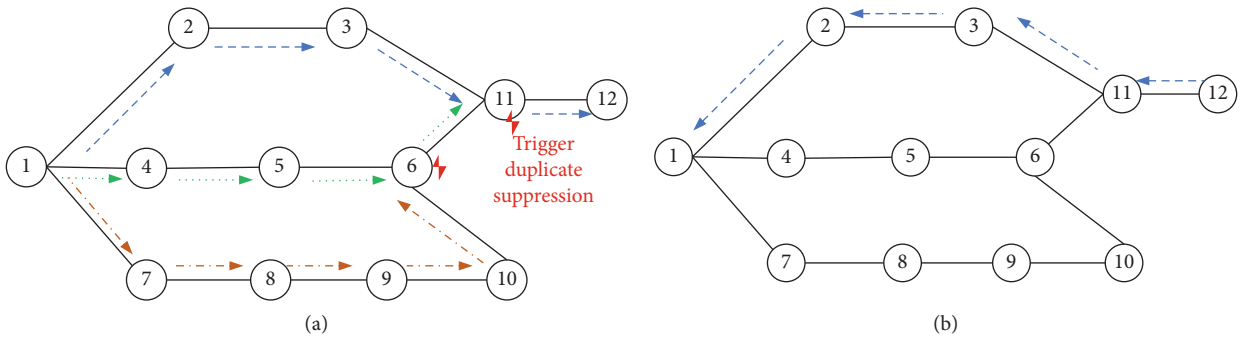


FIGURE 4: The duplicate suppression mechanism in AODV protocol, only with a weak link authentication. (a) RREQ process. (b) RREP process.

Due to the first arrival of the blue path, the duplicate suppression is triggered to ensure that the blue path is a valid link-authenticated one as the right subfigure shows.

Although AODV has high efficiency and complexity, AODV does not have heavy node authentication. Thus, once a malicious node can join and disrupt the network by hijacking the routing tables or bypassing valid routes, then it has a chance to eavesdrop on the network if the node can establish the shortest route to any destination by exploiting the unsecured routing protocols [6].

As shown in Figure 5, the attacker joins the network easily using a rushing attack. The attacker quickly forwards a malicious RREP. Due to duplicate suppression, an actual valid RREP message from the valid node will be discarded and consequently, the attacking node becomes part of the route. After that, the attacker can change the data from the initiator node. To maximize the realism of our threat model, in this paper we assume that only one heading UAV is attacked.

4. Attack Construction

In this section, we describe the attack goal and present the data spoofing attack to affect the flocking.

4.1. Attack Goal: Arrival Delay. As the first security study on the flocking-based UAV cluster, our analysis in this paper focuses on interfering with the UAV cluster's normal flying paths to the destination, causing unexpected delays. More specifically, the attacker aims to pretend to be the first arrived UAV to further send spoofed data to the following UAVs and interferes with their heading angles (see Figure 6).

Such an attack can cause economic losses and affect production work. For example, one of the most potential applications of UAV cluster is for monitoring, search, and rescue in a dangerous environment. Using UAVs to do such a task can reduce unnecessary casualties, and thus, it is highly important to ensure that such flying is well protected and functions correctly and efficiently. Of course, the attacker can also attack multiple heading UAVs at the same time.

4.2. Security Analysis and Attack Flow. Our security analysis consists of the following key steps:

(1) Data spoofing strategy identification: before analyzing the vulnerability of the flocking algorithm, we first need to identify meaningful data spoofing strategies. We analyze the parameter data flow to understand how the spoofed data can potentially influence the UAV cluster.

(2) Vulnerability analysis: with the data spoofing strategy identified, we then further perform vulnerability analysis to reveal the attack consequences. To ensure the generality of this analysis, we choose the most potential application of UAV cluster in flying tasks for monitoring, search, and rescue in a dangerous environment and compare the flying trajectory and total time of the task with and without attack.

(3) Cause analysis and exploit construction: with the attack effectiveness for data spoofing quantified, we perform cause analysis for the successful attacks to understand why the current flocking algorithm is vulnerable. Leveraging the insights, we construct corresponding exploits.

Figure 7 shows the attacked data flow. An attacker can masquerade as the first-arrival UAV within a specific scope of destination and generate data spoofing of arrival status to the following UAVs. Note that GPS and gyroscope are not spoofed. Due to the affected heading angle computed based on weights set of policies, the attacker can interfere with normal flying paths of destination arrival, causing unexpected arrival delays.

The detailed data spoofing attack process is shown in Algorithm 1.

5. Attack Evaluation

5.1. Setup. We perform a real-world source-to-destination flying task in a simulation environment. The swarm consists of five to ten UAVs and uses the flocking algorithm. This

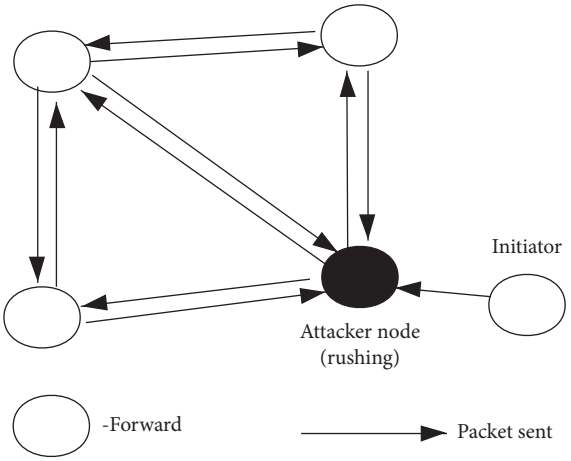


Figure 5: Rushing attack launched by the attacker node.

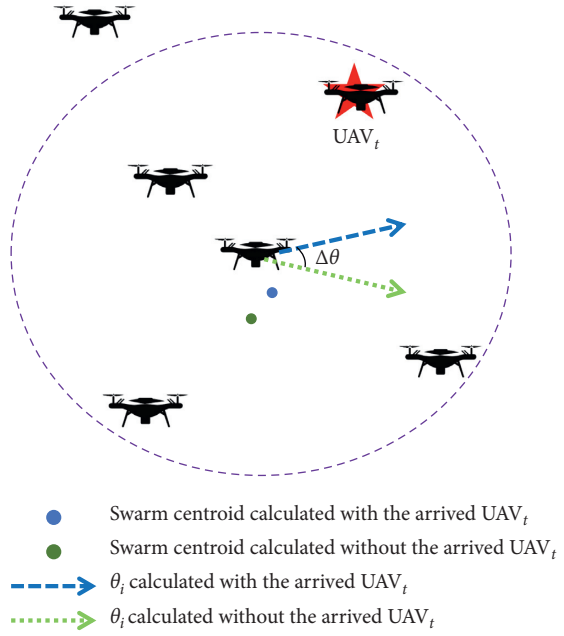


Figure 6: The illustration of attack for interfering heading angle by comparing different centroids.

paper uses a laptop computer as the simulation experiment platform to match the real attack scene, and the software platform is MATLAB R2019b. The experimental environment configuration is shown in Table 2.

5.2. Feature Computation. We hope to bring a delay of the UAV swarm arriving at the target points. When the cluster is about to reach the target, we launch a rushing attack to the arrived UAV in the MANET, by changing the arrival status data from T (arrived) to F (not arrived), and send the arrival status data to other UAVs. In this way, the following UAVs will calculate to get a wrong heading angle which will increase the length of their trajectories to the target points and cause arrival delay. In the analysis, we use the increased delay time under attack as a percentage of the original time of the flying task to quantify the effectiveness of our attack.

5.3. Influence from Speed. We first evaluate the impact of UAV speed on our attack. We take six different UAV speeds from 10 m/s to 20 m/s to study the effect of speed changes on the delay under a fixed arrival determining threshold $R = 20$ m. In order to have universal applicability, we conduct experiments in the case of 5–10 UAV clusters. In the experiments, the initial coordinates of UAVs are random points within five to ten ranges (to ensure the initial safety distance). The target points are 1000 m away from the initial coordinates and the structure of target points is consistent with the initial points' structure. The velocity of UAVs is

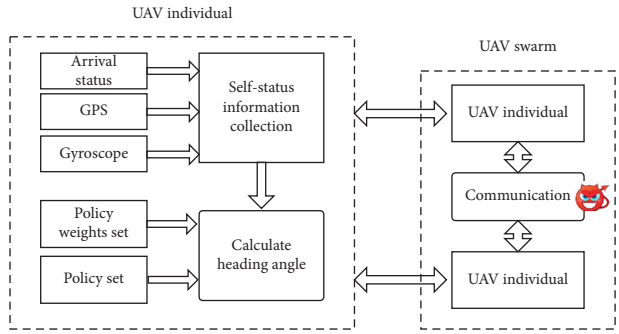


Figure 7: The attacked data flow in the UAV swarm.

Input: R (arrival determining threshold), UAV_i(i-th UAV)
Output: $\theta_i^{,\text{attack}}$ (desired combined heading angle of the i-th UAV after the attack)
(1) *//normal*
(2) **when** the first-arrival UAV_t is within R scope of its destination **then**:
(3) UAV_t sends arrival status (T) to the following UAVs;
(4) UAV_i flies along its desired combined heading angle θ_i' calculated by flocking algorithm without UAV_t;
(5) *//with attack*
(6) **when** the first-arrival UAV_t is within R scope of its destination **then**:
(7) Attacker masquerades as UAV_t exploiting rushing attack;
(8) The attacker generates data spoofing of arrival status and sends the malicious arrival status (F) to following UAVs;
(9) UAV_i flies along its desired combined heading angle $\theta_i^{',\text{attack}}$ calculated by flocking algorithm including UAV_t;
(10) **return** $\theta_i^{',\text{attack}}$

Algorithm 1: Data spoofing attack algorithm.

Table 2: Experimental environment configuration.

Experimental environment	Environmental configuration
Operating system	macOS
CPU	2.4 GHz Intel Core i5
Memory	16 GB
Hardware	500 G
Main tools	MATLAB R2019b

from $V = 10$ m/s to $V = 20$ m/s, the maximum communication range among UAVs is $\rho = 150$ m, the threshold for the application of the cohesion policy is $d_{\max} = 120$ m, and the threshold for the application of the dispersion policy is $d_{\min} = 30$ m. The threshold for judging whether the UAV has reached the target point is $R = 20$ m; that is, when the distance between the location of the UAV and its target point is less than 20 m, it can be regarded as having arrived. The weight of 5 policies is $w_h = 0.9, w_a = 0.5, w_c = 0.2, w_d = 0.4$, and $w_f = 0.1$. This set of weights can ensure that the UAV cluster can fly smoothly to the destination when it is not attacked.

The experiment results are shown in Figure 8. Figures 8(a)–8(f)) are corresponding to different UAV numbers of the cluster from 5 to 10. We can see that the medium numbers in each experiment are over 10% and tend to increase first and then decrease with the speed of the cluster. The corresponding average percentage of delays increment is summarized in Table 3. We use the increased delay time as a percentage of the original time of the task to quantify the effectiveness of our attack. In these columns, each value is the average delay increment under attack as a percentage of the average task completion time without attack.

In this experiment, we fix the arrival determining the threshold at 20 meters. We find the following: (1) from the general trend, the delay increment percentage of a smaller UAV cluster is higher than that of a larger cluster; (2) for a given cluster size, the average percentage of delay increment first increases with cluster speed and then decreases; (3) the maximum delay increment percentage of 27.76% is obtained in the occasion of cluster size 8 and cluster speed 16 m/s.

5.4. Influence from Threshold R. Then, we evaluate the impact of arrival determining threshold R on our attack. We took five different arrival determining threshold from 12 m to 20 m for experiments to study the effect of arrival determining threshold R on the delay under different UAV speeds. In order to have universal applicability, we conducted experiments in the case of 5–10 UAV clusters. In the experiment, the initial coordinates of UAVs are random points within five to ten ranges (to ensure the initial safety distance). The target points are 1000 m away from the initial coordinates and the structure of target points is consistent with the initial points' structure. The velocity of UAVs is from $V = 10$ m/s to $V = 20$ m/s, the maximum communication range between UAVs is $\rho = 150$ m, the threshold for

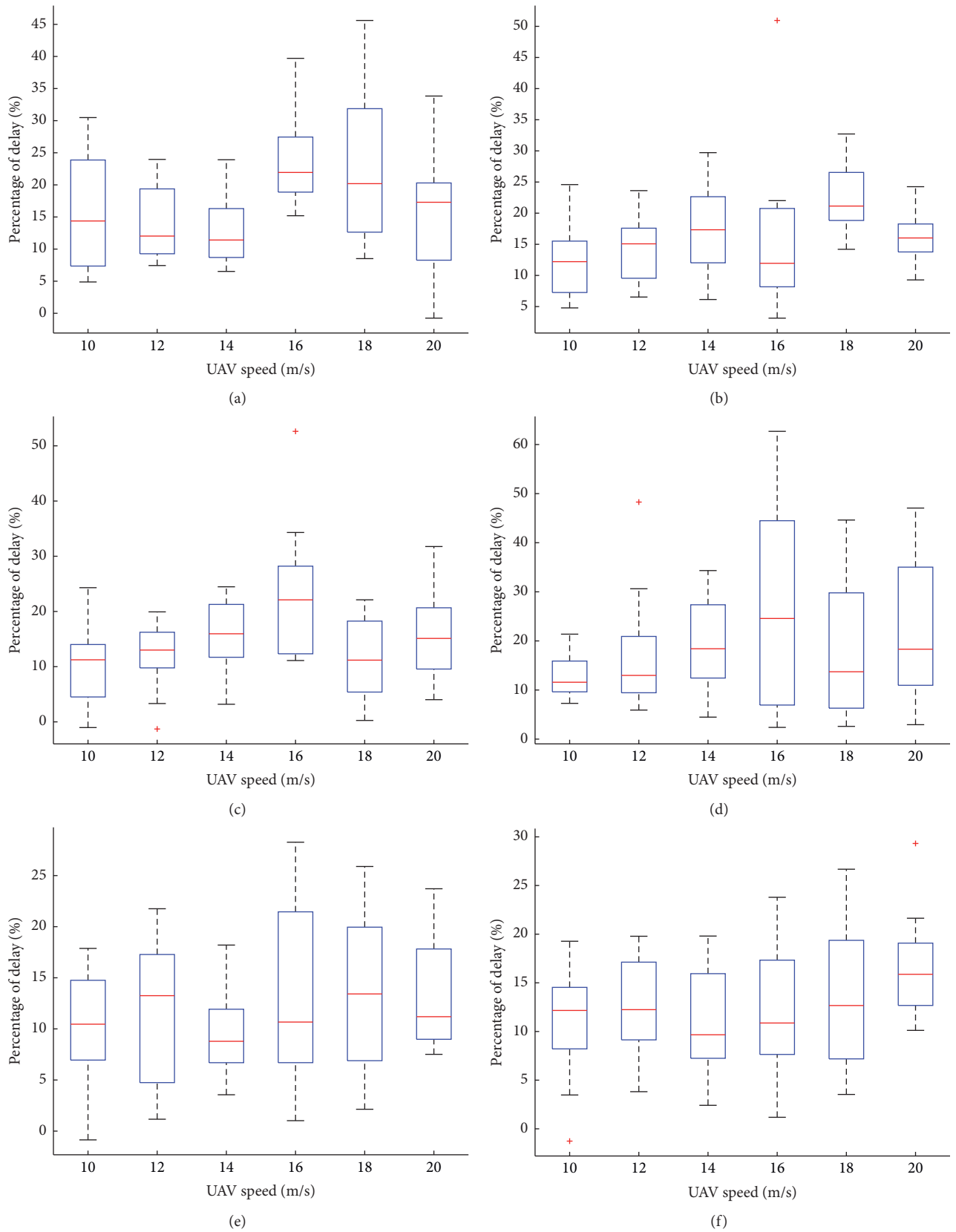


FIGURE 8: Delay increment for multi-UAV clusters of different sizes at a speed from 10 to 20 m/s and with $R = 20$. (a) Boxplot for 5-UAV clusters. (b) Boxplot for 6-UAV clusters. (c) Boxplot for 7-UAV clusters. (d) Boxplot for 8-UAV clusters. (e) Boxplot for 9-UAV clusters. (f) Boxplot for 10-UAV clusters.

Table 3: Average delay increment comparison with $R = 20$.

Speed (m/s)	5-UAV cluster (%)	6-UAV cluster (%)	7-UAV cluster (%)	8-UAV cluster (%)	9-UAV cluster (%)	10-UAV cluster (%)
10	15.70	12.79	10.37	13.08	9.76	10.90
12	13.77	14.47	11.72	17.74	11.55	12.42
14	12.83	17.33	15.53	19.10	9.52	11.01
16	25.00	15.97	23.69	**27.76**	12.83	11.52
18	22.39	22.22	11.17	18.29	13.66	13.76
20	15.64	16.17	15.67	21.66	13.70	17.03

the application of the cohesion policy is $d_{\max} = 120$ m, and the threshold for the application of the dispersion policy is $d_{\min} = 30$ m. The arrival determining threshold R is between 12 m and 20 m. The weight of 5 policies is $w_h = 0.9, w_a = 0.5, w_c = 0.2, w_d = 0.4$, and $w_f = 0.1$.

The experiment results are shown in Figure 9. The corresponding average percentage of delays increment is summarized in Tables 4–9. In this experiment, we find that when fixing the cluster's speed, lower arrival determining threshold R will cause a higher delay increment percentage no matter what the cluster size; the maximum delay increment percentage of 47.84% is obtained in the occasion of cluster size 9, cluster speed 16 m/s, and arrival determining threshold 12 meters.

6. Defense Suggestion

As our research shows, even though the swarm intelligence algorithm shows high effectiveness under a benign set of weights, the current algorithm design is still very vulnerable to data spoofing attacks. In order to proactively solve these problems before large-scale deployment, this section discusses the defense direction based on the insights derived from our analysis.

From our experimental analysis, compared with the higher arrival determining threshold R, a lower R will cause a higher delay percentage because it is difficult to fly very close to the destination especially at a highly fast speed and under the data spoofing attack. Thus, we suggest increasing R in a reasonable practical range to match the speed. But it is a trade-off between accuracy and safety because if we use a too high value of R, the cluster will lose its accuracy and stop too far away from the destination.

Similarly, UAVs at a constant higher speed will not steadily fly to the destination when it approaches the destination and under the attack. To ensure the efficiency of performing tasks, the speed setting can be divided into two stages. In the beginning, UAVs fly at a higher speed. When they approach their destinations, a lower speed should be adopted for a steady fly towards the destination.

Another possible defense is increasing the weight of homing policy and decreasing the weight of cohesion, following, and alignment policies especially when the swarm approaches the destination. In this case, although the first-arrival UAV is attacked, the following UAVs will still fly towards their destination due to the enhanced homing force and thus can decrease the influence of attack. Thus, it is highly important to adjust policy weights and speed dynamically and timely according to the current state to achieve a robust, conflict-free, and efficient UAV swarm.

The data spoofing attack against UAV clusters is rooted from the attacker breaking through the vulnerable authentication of AODV exploiting a rushing attack. Several robust defenses against rushing attack can be introduced in the multi-UAV cluster communication network [7], such as secure neighbor detection which allowed the sender and the receiver to verify that the other party is within the normal direct wireless communication range because the attacker often forwards an RREQ beyond the normal radio transmission range to achieve faster transit; randomized RREQ forwarding allowed a node first to collect a number of RREQs and select one at random to forward to replace traditional duplicate suppression mechanism in AODV which could be exploited by rushing attackers and blacklist mechanisms using the property of nonrepudiation to spread information about identified malicious nodes. Also, there is always a trade-off issue between strong authentication and flocking efficiency, requiring further light authentication study.

7. Related Works

7.1. Data Spoofing in UAV. In [8, 9], the authors demonstrated the feasibility of infiltrating internal networks to launch data spoofing through the network node. Lately, a lot of studies show the sensor's attack to launch data spoofing. More specifically, the GPS spoofing [18, 22] by sending interfering signals is very common. In addition to GPS sensors, other sensors including optical sensors [20, 21] and context-aware services [23–26] can also be threatened to cause data spoofing physically [8, 19, 20, 27–29], wirelessly [9, 30, 31], or through malware [10, 11]. Similar to previous work, our data spoofing is launched through a node in the ad hoc network. Some attacks have been revealed involving wormhole attack, rushing attack, joint attack, Sybil attack, denial of service attacks, and eavesdropping attacks [17]. Prior to our work, there are some related works to attack UAV. The detailed comparison is given in Table 10. We can see that our work focuses on attacking the swarm algorithm as our directed target, compared to those physical sensors or software modules. In addition, compared to the Partial Differential Equation (PDE) algorithm [31], the flocking algorithm enables more control policy that is more practical in real applications. As to the attack effect, general effects include position error, crash, unstable flight, path deviation, and time delay. Although without a crash, our method discovers the attack to cause a heavy time delay of swarm

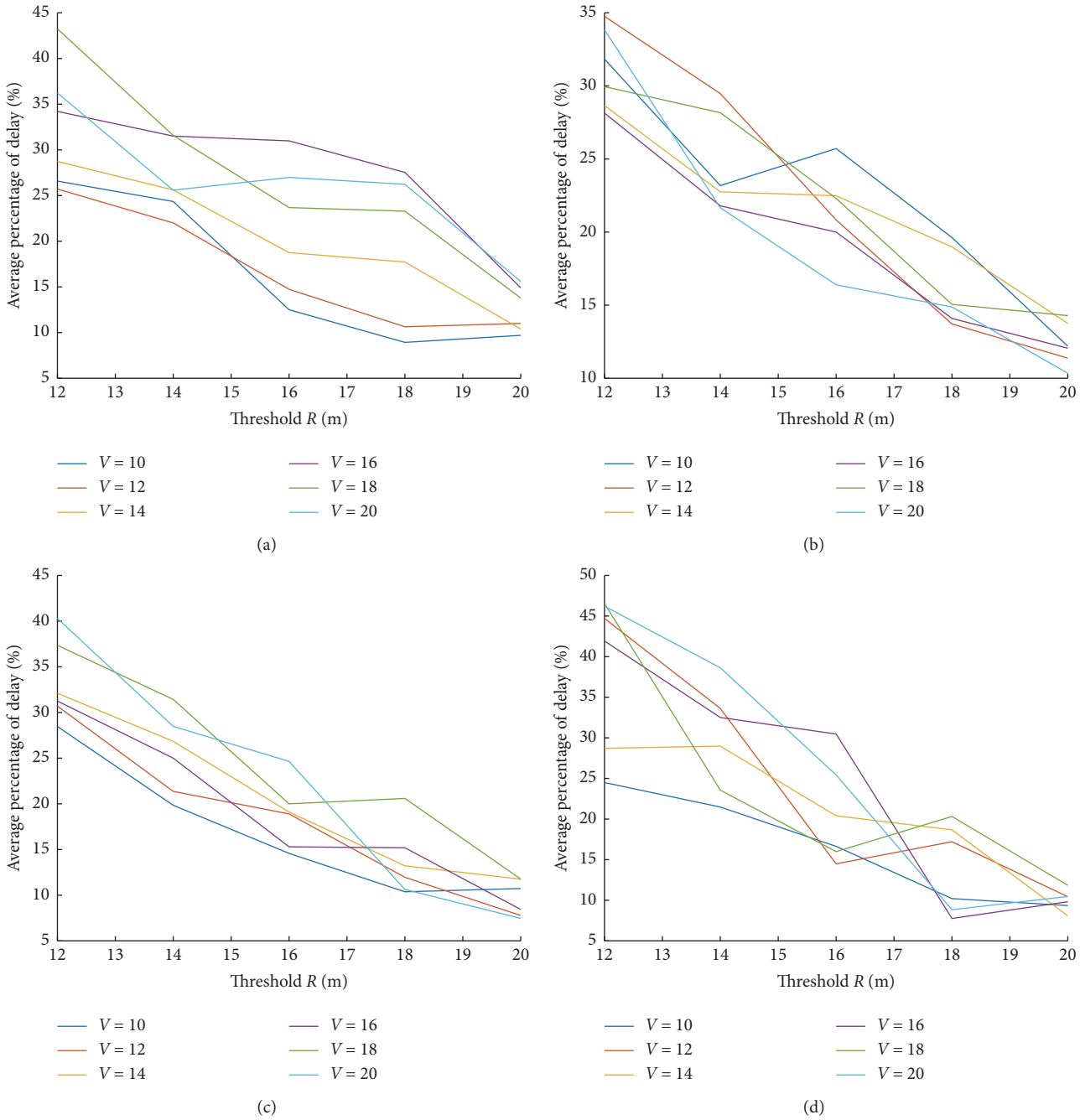


Figure 9: Continued.

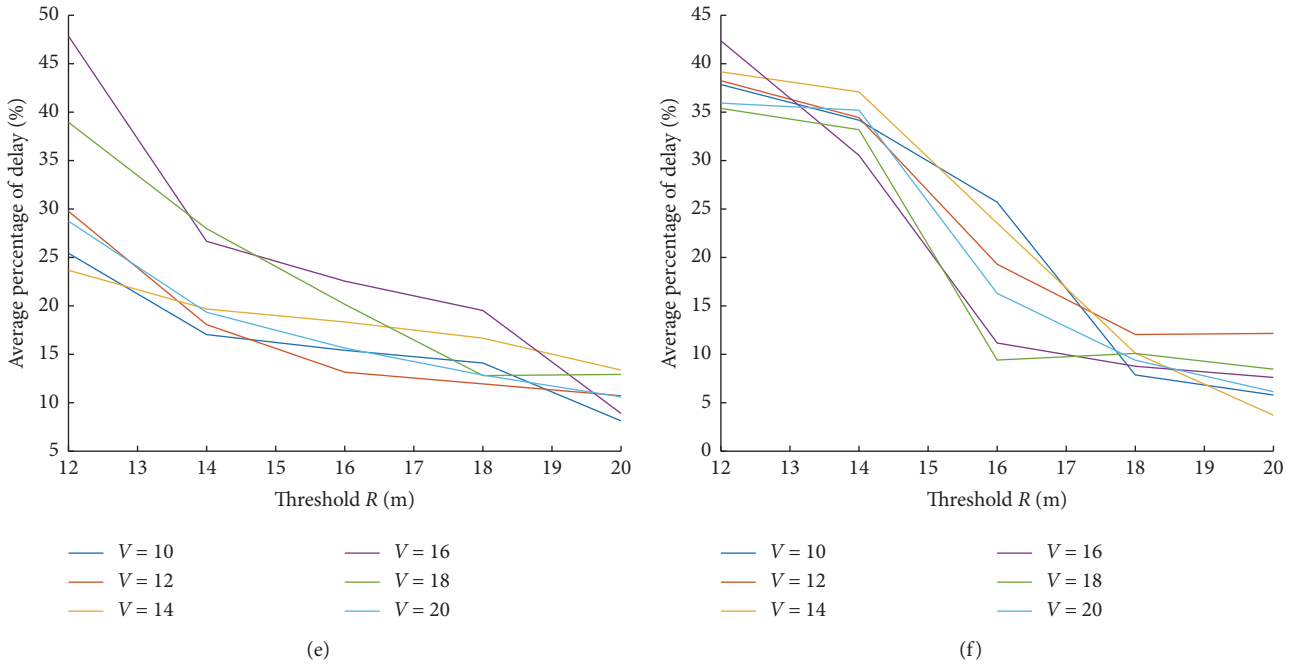


Figure 9: Average delay increment comparison for multi-UAV clusters of different sizes at a speed from 10 to 20 m/s and with R from 12 to 20 m. (a) Result for 5-UAV clusters. (b) Result for 6-UAV clusters. (c) Result for 7-UAV clusters. (d) Result for 8-UAV clusters. (e) Result for 9-UAV clusters. (f) Result for 10-UAV clusters.

Table 4: Average delay increment for 5-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	26.58	25.71	28.74	34.21	43.24	36.23
14	24.35	22.00	25.61	31.51	31.58	25.59
16	12.15	14.74	18.75	30.99	23.68	26.98
18	8.93	10.64	17.72	27.54	23.29	26.23
20	9.70	11.03	10.39	14.90	13.78	15.60

Table 5: Average delay increment for 6-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	31.84	34.76	28.65	28.14	29.95	33.85
14	23.18	29.47	22.75	21.79	28.17	21.67
16	25.71	20.83	22.49	20.01	22.31	16.39
18	19.63	13.71	18.99	14.09	15.05	14.88
20	12.20	11.38	13.75	12.06	14.29	10.34

Table 6: Average delay increment for 7-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	28.46	30.70	32.11	31.25	37.35	40.30
14	19.84	21.36	26.84	25.00	31.43	28.48
16	14.57	18.90	19.10	15.29	20.00	24.65
18	10.37	11.96	13.22	15.19	20.59	10.63
20	10.73	7.78	11.75	8.44	11.76	7.46

Table 7: Average delay increment for 8-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	24.50	44.71	28.72	41.93	46.56	46.22
14	21.48	33.62	28.98	32.51	23.56	38.63
16	16.64	14.48	20.41	30.47	15.99	25.43
18	10.20	17.21	18.67	7.76	20.32	8.83
20	9.35	10.44	8.09	9.82	11.84	10.47

Table 8: Average delay increment for 9-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	25.44	29.77	23.68	47.84	38.94	28.74
14	17.04	18.07	19.67	26.66	27.97	19.34
16	15.41	13.16	18.34	22.55	20.16	15.64
18	14.10	11.95	16.67	19.52	12.80	12.85
20	8.13	10.73	13.37	8.90	12.92	10.60

Table 9: Average delay increment for 10-UAV clusters at a speed of 10–20 m/s and with R of 12–20 m.

Threshold R (m)	$V = 10$ m/s (%)	$V = 12$ m/s (%)	$V = 14$ m/s (%)	$V = 16$ m/s (%)	$V = 18$ m/s (%)	$V = 20$ m/s (%)
12	37.84	38.24	39.15	42.36	35.38	35.93
14	34.16	34.43	37.08	30.53	33.18	35.19
16	25.70	19.29	23.54	11.17	9.40	16.29
18	7.87	12.04	10.12	8.76	10.09	9.39
20	5.79	12.15	3.71	7.61	8.46	6.14

Table 10: Comparison of different attack methods.

Method		Target	Attack effect
Physically	Son et al. [27]	Gyroscopic sensor	Crash
	Choi et al. [28]	GPS sensor	Path deviation
	Trippel et al. [29]	Accelerometers	Unstable flight
	Davidson et al. [20]	Optical flow sensor	Position error
	Tippenhauer et al. [19]	GPS sensor	Position error
Through malware	Dash et al. [11]	Software stack	Unstable flight
	Mazloom et al. [10]	Software stack	Unstable flight
Wirelessly	Highnam et al. [30]	Wireless channel	Disrupted communications
	Ghanavati et al. [31]	PDE algorithm	Time delay
	Our method	Flocking algorithm	Time delay

flight, which is still unacceptable in emerging tasks of monitoring, search, and rescue.

7.2. Autonomous UAV Cluster and Security. Privacy leak and collision are two common security issues for the UAV cluster. In the works [32–35], researchers studied privacy issues in UAV clusters, such as privacy refers to preventing the inference of the leader's identity in leader-follower structure swarms. Our work belongs to the latter security issue of flocking-based cluster and collision. Reynolds proposed the Boids model in 1986, which is the earliest flocking model [1]. Zaera et al. intended to develop neural network-based controllers for schooling behavior in three dimensions, using realistic Newtonian kinematics, such as inertia and drag [36]. Vicsek et al. proposed a particle swarm model [37], in which each particle moves at the same unit speed, and the direction is the average of the direction of its neighbor particles. Although this model only achieves the overall direction consistency of the particle swarm and ignores the collision avoidance of each particle, it still makes an important contribution to the modeling of swarm agents. Sharma and Ghose extended the Boids model and proposed a swarm intelligence algorithm to avoid cluster collisions [4].

One of the recent applications of the self-organized swarm intelligence algorithm is collective UAVs [38], where decentralized control algorithms for groups of autonomous

UAVs can be developed on the basis of interactions as a prerequisite for safe operation. In comparison, our attack target is based on the 5-policy flocking algorithm, which has more widespread applications.

The largest UAV cluster so far was developed by Ehang with more than 1000 UAVs. However, these UAVs were individually programmed for predefined trajectories or were centrally controlled and did not satisfy the autonomy with swarm intelligence [39]. The US military had an experiment with fixed-wing drone swarms called Perdix [40, 41]. Unfortunately, there is no public information about its control mechanisms, communication schemes, or possible collision avoidance behaviors to reliably evaluate. In comparison, we target a lab-level UAV cluster with 10 UAVs.

A previous related work is the intrusion detection-based multi-UAV mission execution [42]. In their method, a subflight area is demarcated by the coordinates of the waypoints in the assigned tasks of the UAV. According to the UAV's GPS coordinates, if the current UAV exceeds this area, it is considered suspicious and will cut off the connection with other UAVs, and the others reconnect. In comparison, our method focuses on flocking-based UAV collaboration, and we do not perform GPS spoofing but arrival status spoofing, which is a special parameter in the swarm intelligence of flocking.

8. Conclusions

In this work, we perform the first security analysis of the flocking algorithm that is most widely used in UAV clusters. Targeting a highly realistic threat model in AODV link authentication through rushing attack-based data spoofing, we perform vulnerability analysis and find that the current flocking algorithm design is highly vulnerable to data spoofing attacks. The evaluation results in the simulation environment validate the effectiveness of the attack and show that the attack can even cause nearly 50% arrival delay. Defense directions are then discussed leveraging the insights.

This work serves as a first step to understand the new security problems and challenges in the flocking, a main kind of swarm intelligence algorithm of UAVs. It is expected to inspire a series of follow-up studies, including but not limited to (1) more extensive evaluation with UAV clusters with different formation structures, (2) more extensive analysis considering other swarm intelligence algorithms, such as the one imitating bees or ants, and (3) more concrete defense approach and evaluation.

Data Availability

All data generated or analyzed during this study are owned by all the authors and will be used for further research. The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61972025, 61802389, 61672092, U1811264, and 61966009), the Fundamental Research Funds for the Central Universities of China (2018JBZ103 and 2019RC008), Science and Technology on Information Assurance Laboratory (614200103011711), and Guangxi Key Laboratory of Trusted Software (KX201902).

Supplementary Materials

We provide a video file to show a simulated attack in MATLAB. In this attack scenario, there is a cluster of 8 UAVs with an applied flocking algorithm, keeping a formation from the left source to the right destination. A data spoofing through masquerading as the first-arrival UAV is performed at a location of 20-meter distance away from the destination, and then a heavy delay occurred among the following UAVs. Through the video supplementary material, we reveal the whole flying and attacking process. (*Supplementary Materials*)

References

[1] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 25–34, Anaheim, CA, USA, August 1987.

[2] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[3] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 916–923, 2018.

[4] R. K. Sharma and D. Ghose, "Collision avoidance between UAV clusters using swarm intelligence techniques," *International Journal of Systems Science*, vol. 40, no. 5, pp. 521–538, 2009.

[5] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for WSNs," in *Proceedings of the 81st IEEE Vehicular Technology Conference (VTC Spring)*, pp. 1–6, Glasgow, UK, May 2015.

[6] A. Bhattacharyya, A. Banerjee, D. Bose et al., *Different Types of Attacks in Mobile ADHOC Network: Prevention and Mitigation Techniques*, Department of Computer Science & Engineering, Institute of Engineering & Management, Saltlake, UT, USA, 2011.

[7] Y. C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proceedings of the 2nd ACM Workshop on Wireless Security*, pp. 30–40, San Diego CA USA, September 2003.

[8] K. Koscher, A. Czeskis, F. Roesner et al., "Experimental security analysis of a modern automobile," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pp. 447–462, IEEE, Oakland, CA, USA, May 2010.

[9] S. Checkoway, D. McCoy, B. Kantor et al., "Comprehensive experimental analyses of automotive attack surfaces," *USENIX Security Symposium*, vol. 4, pp. 447–462, 2011.

[10] S. Mazloom, M. Rezaeirad, A. Hunter et al., "A security analysis of an in-vehicle infotainment and app platform," in

Proceedings of the 10th USENIX Conference on Offensive Technologies, pp. 232–243, Austin, TX, USA, August 2016.

[11] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Out of control: stealthy attacks against robotic vehicles protected by control-based techniques," in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 660–672, San Juan, PR, USA, December 2019.

[12] H. Hildenbrandt, C. Carere, and C. K. Hemelrijk, "Self-organized aerial displays of thousands of starlings: a model," *Behavioral Ecology*, vol. 21, no. 6, pp. 1349–1359, 2010.

[13] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[14] A. A. Paranjape, S.-J. Chung, K. Kim, and D. H. Shim, "Robotic herding of a flock of birds using an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 901–915, 2018.

[15] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.

[16] F. Van Breugel, K. Morgansen, and M. H. Dickinson, "Monocular distance estimation from optic flow during active landing maneuvers," *Bioinspiration & Biomimetics*, vol. 9, no. 2, Article ID 025002, 2014.

[17] H. Deng, W. Li, and D. P. Agrawal, "Routing security in wireless ad hoc networks," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 70–75, 2002.

[18] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki et al., "Assessing the spoofing threat: development of a portable GPS civilian spoofer," in *Proceedings of the Institute of Navigation GNSS*, pp. 1198–1209, Savannah, GA, USA, September 2008.

[19] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen et al., "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 75–86, Chicago, IL, USA, October 2011.

[20] D. Davidson, H. Wu, R. Jellinek et al., "Controlling UAVs with sensor input spoofing attacks," in *Proceedings of the 10th USENIX Conference on Offensive Technologies*, pp. 221–231, Austin, TX, USA, August 2016.

[21] S. McLaughlin and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in *Proceedings of the 2014 IEEE international Conference on smart Grid communications (SmartGridComm)*, pp. 848–853, IEEE, Venice, Italy, November 2014.

[22] J. S. Warner and R. G. Johnston, "A simple demonstration that the global positioning system (GPS) is vulnerable to spoofing," *Journal of Security Administration*, vol. 25, no. 2, pp. 19–27, 2002.

[23] W. Niu, J. Lei, E. Tong et al., "Context-aware service ranking in wireless sensor networks," *Journal of Network and Systems Management*, vol. 22, no. 1, pp. 50–74, 2014.

[24] W. Li, G. Li, Z. Zhao, H. Tang, and Z. Shi, "Multi-granularity context model for dynamic Web service composition," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 312–326, 2011.

[25] W. Niu, G. Li, H. Tang, X. Zhou, and Z. Shi, "CARSA: a context-aware reasoning-based service agent model for AI planning of web service composition," *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1757–1770, 2011.

[26] E. Tong, W. Niu, G. Li et al., "Bloom filter-based workflow management to enable QoS guarantee in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 39, pp. 38–51, 2014.

[27] Y. Son, H. Shin, D. Kim et al., "Rocking drones with intentional sound noise on gyroscopic sensors," in *Proceedings of the 24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 881–896, Washington, DC, USA, August 2015.

[28] H. Choi, W. C. Lee, Y. Aafer et al., "Detecting attacks against robotic vehicles: a control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 801–816, Toronto, Canada, October 2018.

[29] T. Trippel, O. Weisse, W. Xu et al., "WALNUT: waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks," in *Proceedings of the 2017 IEEE European symposium on security and privacy (EuroS&P)*, pp. 3–18, IEEE, Paris, France, April 2017.

[30] K. Highnam, K. Angstadt, K. Leach et al., "An uncrewed aerial vehicle attack scenario and trustworthy repair architecture," in *Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 222–225, IEEE, Toulouse, France, June 2016.

[31] M. Ghanavati, A. Chakravarthy, and P. Menon, "PDE-based analysis of cyber-attacks in vehicle swarms," in *Proceedings of the 2018 IEEE Conference on Decision and Control (CDC)*, pp. 1329–1334, IEEE, Miami Beach, FL, USA, December 2018.

[32] H. Zheng, J. Panerati, G. Beltrame, and A. Prorok, "An adversarial approach to private flocking in mobile robot teams," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1009–1016, 2020.

[33] A. Prorok and V. Kumar, "Privacy-preserving vehicle assignment for mobility-on-demand systems," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1869–1876, Vancouver, Canada, September 2017.

[34] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, "Coordinated multi-robot planning while preserving individual privacy," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2188–2194, Montreal, Canada, May, 2019.

[35] Y. Zhang and D. A. Shell, "Complete characterization of a class of privacy-preserving tracking problems," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 299–315, 2019.

[36] N. Zaera, D. Cliff, and J. Bruten, "(Not) Evolving collective behaviours in synthetic fish," in *Proceedings of the International Conference on the Simulation of Adaptive Behavior*, pp. 635–642, MIT Press, Cambridge, MA, USA, August 1996.

[37] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995.

[38] M. Cohen, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[39] F. Y. Hadaegh, S. J. Chung, and H. M. Manohara, "On development of 100-gram-class spacecraft for swarm applications," *IEEE Systems Journal*, vol. 10, no. 2, pp. 673–684, 2014.

[40] G. Vásárhelyi, C. Virágh, G. Somorjai et al., "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, pp. 1–13, 2018.

[41] R. Mitchell and I. R. Chen, "Specification based intrusion detection for unmanned aircraft systems," in *Proceedings of the First ACM MobiHoc Workshop on Airborne Networks and Communications*, pp. 31–36, Hilton Head, SC, USA, June 2012.

[42] Z. Fu, Y. Mao, D. He, J. Yu, and G. Xie, "Secure multi-UAV collaborative task allocation," *IEEE Access*, vol. 7, pp. 35579–35587, 2019.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8847487, 12 pages
https://doi.org/10.1155/2020/8847487

Research Article

An Efficient Outsourced Oblivious Transfer Extension Protocol and Its Applications

Shengnan Zhao,[1] **Xiangfu Song**,[1] **Han Jiang**,[1] **Ming Ma**,[1] **Zhihua Zheng**,[2] **and Qiuliang Xu**[1]

[1]*School of Software, Shandong University, Jinan 250100, China*
[2]*School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China*

Correspondence should be addressed to Han Jiang; jianghan@sdu.edu.cn and Qiuliang Xu; xql@sdu.edu.cn

Received 20 August 2020; Revised 2 November 2020; Accepted 18 November 2020; Published 5 December 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Shengnan Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Oblivious transfer (OT) is a cryptographic primitive originally used to transfer a collection of messages from the sender to the receiver in an oblivious manner. OT extension protocol reduces expensive asymmetric operations by running a small number of OT instances first and then cheap symmetric operations. While most earlier works discussed security model or communication and computation complexity of OT in general case, we focus on concrete application scenarios, especially where the sender in the OT protocol is a database with less computation and limited interaction capability. In this paper, we propose a generic outsourced OT extension protocol ($\mathbb{O}Tex$) that outsources all the asymmetric operations of the sender to a semihonest server so as to adapt to specific scenarios above. We give $\mathbb{O}Tex$ a standard security definition, and the proposed protocol is proven secure in the semihonest model. In $\mathbb{O}Tex$, the sender works on the fly and performs only symmetric operations locally. Whatever the number of rounds OT to be executed and the length of messages in OT to be sent, our protocol realizes optimal complexity. Besides, $\mathbb{O}Tex$ can be used to construct high-level protocols, such as private membership test (PMT) and private set intersection (PSI). We believe our $\mathbb{O}Tex$ construction may be a building block in other applications as well.

1. Introduction

Oblivious transfer (OT) is one of the most important primitives in secure computation. It is wildly used in Yao's protocol [1], GMW construction [2], and preprocessing phase of SPDZ-like [3] protocols. With the development of big data, cloud computing, and mobile computing, the demand for joint computation grows rapidly between different organizations and individuals. In order to ensure the security of such computing tasks against complicated external environment, cryptographic components have to be used to design protocols, in which OT plays a pivotal role together with homomorphic encryption, secret sharing, and garbled circuit.

However, OT is public-key primitive centered, which makes it computational expensive for secure computation. Many privacy-preserving protocols, such as private membership test (PMT) and private set intersection (PSI), rely heavily on huge number of OT instances for secure computation to get the trade-off between computation and communication. The most efficient way to produce many OT instances is through OT extension protocol [4, 5]. In such protocol, two participants collectively run few "base" OT instances and then perform some cheap symmetric operations to produce many OT instances.

1.1. Motivation. In an OT extension protocol, the sender $\mathcal{S}$ needs to interact with the receiver $\mathcal{R}$ for each step during the protocol, which involves exponential calculation and intensive interaction. In some application scenarios, however, $\mathcal{S}$ could be a mobile device with less computation power or a database holder with limited interaction capability. When invoking OT extension as a subprotocol in some more complex computation tasks, $\mathcal{S}$ needs to respond requests from $\mathcal{R}$ as fast as possible.

Nowadays, many applications are rapidly transferred to cloud-based service, and it would be desired to seek some server-aided OT extension protocol to relief the burden of $\mathcal{S}$ under reasonable security assumption. A considerable literature [6–12] has grown up around the theme of fertilizing functionality of OT or optimizing communication cost of the receiver $\mathcal{R}$. However, far too little attention has been paid to investigate sender side of OT adapting to specific scenarios.

To this end, we propose a generic outsourced oblivious transfer extension protocol ($\mathbb{O}Tex$) in the semihonest model. In $\mathbb{O}Tex$, the sender $\mathcal{S}$ first outsources all expensive asymmetric operations to a third party who runs a subprotocol called "base" OT instances with the receiver $\mathcal{R}$. Based on the corresponding outputs of the subprotocol and other auxiliary information, $\mathcal{S}$ generates symmetric keys used to encrypt sending messages in OT. As a result, the sender $\mathcal{S}$ works on the fly and sends its inputs encrypted by symmetric key generated from $\mathbb{O}Tex$ to the receiver $\mathcal{R}$, and thus, it enables two parties to complete the whole OT extension protocol.

Recent trends in OT extension have led to a proliferation of studies showing how to design an efficient PSI [13–17] or PSI-based protocols [18–21] in different secure models since OT extension protocol is an important component in secure computation and plays a key role in set operations. Take PSI as an example, without violating individuals' privacy, and the use of PSI in contact tracing [21] can help prevent the further spread of COVID-19. Therefore, $\mathbb{O}Tex$ framework has a wide range of applications in outsourced scenarios, and as a building block, we think $\mathbb{O}Tex$ can be applied conveniently to high-level protocols.

1.2. Related Work. Rabin [22] first introduced the notion of OT that the receiver receives a message sent from the sender with probability 1/2 and the sender does not know whether the receiver has received the message or not. Then, a line of works seek to enrich functionality of OT, and they mainly consist of 1-out-of-2 OT [6], 1-out-of-n OT [7, 8], and k-out-of-n OT [9, 10], in which the first two functionalities are considered in this paper. Some researchers [23–25] proposed OT protocols in different security models although we focus on semihonest model that is sufficient enough to deploy our framework in future among including malicious model, covert adversary model, and universally composable model.

Because OT is public-key primitive centered, the issue of efficiency has received considerable critical attention after Rabin's work [22]. Beaver [26] showed that OT can be precomputed using only prior transfers. Studies over the past two decades have proved that extending a small number of OT to huge number of OT instances can be achieved by one-way function [27–29]. Until 2003, Ishai et al. [4] proposed an efficient method to extend 1-out-of-2 OTs by running few "base" OT instances, which is also known as the IKNP OT extension protocol. Kolesnikov and Kumaresan [5] generalized IKNP from a coding understanding and proposed an improved OT extension protocol allowing to do 1-of-out-n OTs with less communication and computation cost. Lindell et al. [11] studied input-size hiding two-party computation based on fully homomorphic encryption (FHE) and proposed a secure OT extension protocol to reduce the communication cost of both the sender and receiver. Cho et al. [12] focused on the receivers' communication cost in OT and proposed laconic OT protocol based on the decisional Diffie–Hellman (DDH) assumption. Carter et al. [30] proposed outsourced OT protocol specifically for the mobile use-case where the cloud receives outputs of OT. Recently, Mansy and Rindal proposed [31] noninteractive OTs from noninteractive key exchange. However, the resulting OT extension would require 3 rounds.

The research to date has tended to focus more on the cost of the receiver and less on the sender in OT, and there are few studies that have investigated computation and communication complexity on sender side. The aim of this work is to explore the cost of sender in OT and construct efficient OT extension framework assisted by a third party. In addition, OT extension provides a brief but useful account of the construction of oblivious pseudorandom function (PRF). Also, oblivious PRF has been attracting a lot of interest in very recent years, such as multiparty PSI [14], PSI cardinality [21], and private set union [32]. This indicates a need to adapt OT extension to outsourcing scenarios due to practical constraints.

1.3. Our Contribution. In this paper, we focus on server-aided OT to reduce the sender's public-key computation and rounds of interaction with the receiver. Main contributions of our work go as follows:

(i) We propose a generic outsourced OT extension protocol ($\mathbb{O}Tex$). In $\mathbb{O}Tex$, the server and the receiver cooperatively run a small number of OT instances, and at this moment, the sender can be offline. After the first phase, the sender can fetch necessary corelated randomness from the server whenever needed; then, the sender can send their inputs encrypted only by a symmetric key to the receiver, which completes an OT instance. We design a novel mechanism for this purpose and formally prove its security under semihonest model.

(ii) We analyze the complexity of our construction and perform implementation, and the experiment shows that our construction is practical and efficient.

(iii) Our $\mathbb{O}Tex$ construction can be applied to improve the efficiency of OT-based privacy-preserving primitives in server-aided setting, such as oblivious pseudorandom function, and high-level protocols, such as PMT and PSI, which is of independent interest.

2. Preliminaries

2.1. Notation. Unless otherwise stated, we use OT to denote 1-out-of-2 OT and OT_l^k to denote k instances of 1-out-of-2 OT of l-bit strings. For simplicity, let $H(\cdot)$ denote *random*

oracles with a suitable secure output length, which will be well defined in an actual protocol. Matrices are denoted by capital letters, and vectors are denoted by small bold letters; that is, $\mathbf{t}_i$ denotes the i-th row of a matrix T, and $\mathbf{t}^j$ denotes the j-th column of T. Small light front with subscript s_i denotes the i-th bit of a string $\mathbf{s}$. Notably, we regard strings as vectors and do not distinguish the difference between strings and vectors. If $s = s_1 \| \cdots \| s_n$ and $t = t_1 \| \cdots \| t_n$ are two strings, then the notation $s \oplus t$ denotes $(s_1 \oplus t_1) \| \cdots \| (s_n \oplus t_n)$. Similarly, the notation $s \odot t$ denotes the vector $(s_1 \cdot t_1) \| \cdots \| (s_n \cdot t_n)$. Specially, when $c \in \{0,1\}$, $c \cdot \mathbf{s}$ denotes the vector $(c \cdot s_1) \| \cdots \| (c \cdot s_n)$.

2.2. Secure Computation and Security Model. The formal definition of security of a secure multiparty protocol [33] is based on comparing two output distributions coming from an ideal world and a real world, respectively. The functionality of three parties P_1, P_2, P_3 is denoted as f: $\{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2, f_3)$ and P_i gets f_i as output.

Ideal/reality Simulation Paradigm. In an ideal world, participants send the input to an external trusted party who computes the functionality and sends each participant the corresponding output. Suppose there exists an adversary who has the inputs and outputs of a protocol in the ideal world and executes attack against a real protocol, then there always be an adversary executing the same attack in the ideal world. In a real protocol, if no adversary can do more harm than the execution of the protocol in the ideal world, the protocol in the real world is said to secure.

Computationally Indistinguishability. Two distribution probability ensembles $X = \{X(a,n)\}_{a \in \{0,1\}^*; n \in \mathbb{N}}$ and $Y = \{Y(a,n)\}_{a \in \{0,1\}^*; n \in \mathbb{N}}$ are said to be *computationally indistinguishable*, denoted by $X^c \equiv Y$, if for every nonuniform polynomial-time algorithm D, there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and every $n \in \mathbb{N}$:

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n). \tag{1}$$

Semihonest Adversary Model. In the semihonest adversary model, corrupted participant must execute the protocol correctly. However, the adversary can comprehensively obtain the internal status of the corrupted party, *e.g.,* the transcripts of all received messages, and then tries to obtain additional information that should be kept confidential. Semihonest model is sufficient and captures many scenarios in practice although it is a very weak adversary model.

In this paper, we focus on semihonest model and honest majority case where an adversary can corrupt at most one participant and any two participants will not get colluded. In the following, the formal security definition is proposed.

Definition 1. Let $f = (f_1, f_2, f_3)$ be a deterministic functionality and π be a three-party protocol for computing f. Given the security parameter κ and triple inputs (x, y, z), where x is from P_1, y is from P_2, and z is from P_3, the view of P_i $(i = 1, 2, 3)$ in the protocol π is denoted as $view_i^\pi(x, y, z, \kappa) = (w, r_i, m_i^1, \ldots, m_i^t)$, where $w \in \{x, y, z\}$, r_i is the randomness used by P_i, and m_i^j is the j-th message received by P_i; the output of P_i is denoted as $output_i^\pi(x, y, z, \kappa)$, and the joint output of the parties is $output^\pi(x, y, z, \kappa) = (output_1^\pi(x, y, z, \kappa), output_2^\pi(x, y, z, \kappa), output_3^\pi(x, y, z, \kappa))$. We say that π securely computes f in the semihonest model if the following holds:

(i) The correctness holds:

$$\text{output}^\pi(x, y, z, \kappa)^c \equiv \{f(x, y, z)\}_{x,y,z,\kappa}. \tag{2}$$

(ii) There exist probabilistic polynomial-time simulators $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ such that

$$\begin{aligned} \{\mathcal{S}_1(1^\kappa, x, f_1(x, y, z))\}_{x,y,z,\kappa}{}^c &\equiv \{view_1^\pi(x, y, z, \kappa)\}_{x,y,z,\kappa}, \\ \{\mathcal{S}_2(1^\kappa, y, f_2(x, y, z))\}_{x,y,z,\kappa}{}^c &\equiv \{view_2^\pi(x, y, z, \kappa)\}_{x,y,z,\kappa}, \\ \{\mathcal{S}_3(1^\kappa, z, f_3(x, y, z))\}_{x,y,z,\kappa}{}^c &\equiv \{view_3^\pi(x, y, z, \kappa)\}_{x,y,z,\kappa}. \end{aligned} \tag{3}$$

2.2.1. $\mathbb{O}Tex$ Security Model. In $\mathbb{O}Tex$, a simpler definition can be used since two of three parties output nothing (see $\mathcal{F}_{\mathbb{O}Tex}$ in Figure 1). Specifically, three parties P_1, P_2, P_3 stand for server $\mathbb{S}$, sender $\mathcal{S}$, and receiver $\mathcal{R}$, respectively. The functionality of $\mathbb{O}Tex$ can be given by simply writing f: $(\{0,1\}^*, \{(x_i^0, x_i^1)\}, \mathbf{c}) \mapsto (\lambda, \lambda, x_i^{c_i})$, where λ denotes the empty string. We still require *correctness* described above, and security meaning that there exist three simulators $Sim_{\mathbb{S}}$, $Sim_{\mathbb{S}}$, and $Sim_{\mathcal{R}}$ such that

$$\begin{aligned} \{\text{Sim}_{\mathbb{S}}(1^\kappa, \mathbf{s}, \perp)\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}{}^c &\equiv \{\text{view}_{\mathbb{S}}^\pi(\mathbf{x}, \mathbf{s}, c, \kappa)\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}, \\ \{\text{Sim}_{\mathcal{S}}(1^\kappa, \mathbf{x}, \perp)\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}{}^c &\equiv \{\text{view}_{\mathcal{S}}^\pi(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}, \\ \{\text{Sim}_{\mathcal{R}}(1^\kappa, \mathbf{c}, \tilde{x})\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}{}^c &\equiv \{\text{view}_{\mathcal{R}}^\pi(\mathbf{x}, \mathbf{s}, \mathbf{c}, \kappa)\}_{\mathbf{x},\mathbf{s},\mathbf{c},\kappa}, \end{aligned} \tag{4}$$

where $\mathbf{x}$ denotes input set $\{(x_i^0, x_i^1)\}$ from $\mathcal{S}$, $\tilde{x}$ denotes output $\{x_i^{c_i}\}$ of $\mathcal{R}$, and $\perp$ denotes null value.

2.3. OT Extension. We start by introducing the definition of standard 1-out-of-2 OT, where a sender holding two messages (m_0, m_1) interacts with a receiver holding a choice bit b. The 1-out-of-2 OT protocol guarantees that the receiver obtains m_b without knowing anything about m_{1-b}, while the sender knows nothing about b. The ideal functionality for 1-out-of-2 OT, denoted as $\mathcal{F}_{\text{OT}}$, is described in Figure 2.

In most settings, it is necessary to run a large number of OT instances at the same time. The multiexecution of OT is

Functionality $F_{\mathbb{O}\mathrm{Tex}}$

Let k be security parameter.
Three parties: sender S, receiver R, server $\mathbb{S}$.

(i) Input:
  (1) S inputs m pairs of messages (x_i^0, x_i^1), each of length l, $1 \le i \le m$.
  (2) $\mathbb{S}$ inputs random string $r \leftarrow \{0,1\}^{\kappa}$.
  (3) R inputs m-bit selection vector $c = c_1 || \ldots || c_m$.
(ii) Output:
  (1) S outputs nothing.
  (2) $\mathbb{S}$ outputs nothing.
  (3) R outputs $\{x_i^{c_i} \mid i \in [m]\}$ and learns nothing about $\{x_i^{1-c_i} \mid i \in [m]\}$.

Figure 1: Outsourced oblivious transfer functionality $\mathcal{F}_{\mathbb{O}Tex}$.

Functionality F_{OT}

Let $l \in$ N.Two parties: sender S and receiver R.

(i) Input:
  (1) S inputs two messages $x_0, x_1 \in \{0,1\}^l$.
  (2) R inputs a selection bit $\sigma \in \{0,1\}$.
(ii) Output:
  (1) S has no outputs.
  (2) R outputs x_σ and learns nothing about $x_{1-\sigma}$.

Figure 2: 1-out-of-2 oblivious transfer functionality $\mathcal{F}_{\mathrm{OT}}$.

Functionality $F_{\mathrm{OT}_l^m}$

Let l, $m \in N$. Two parties: sender S and receiver R.

(i) Input:
  (1) S inputs m pairs of messages $\{(x_i^0, x_i^1) \mid i \in [m], |x_i^0| = |x_i^1| = l\}$.
  (2) R inputs m-bit selection string $c = c_1 || \ldots || c_m$.
(ii) Output:
  (1) S has no outputs.
  (2) R inputs $\{x_i^{c_i} \mid i \in [m]\}$ and learns nothing about $\{x_i^{1-c_i} \mid i \in [m]\}$.

Figure 3: Batch oblivious transfer functionality $\mathcal{F}_{\mathrm{OT}_l^m}$.

called batch OT (see Figure 3) denoted as $\mathcal{F}_{\mathrm{OT}_l^m}$. The IKNP OT extension protocol [4] is a real milestone in the development of OT research computing $\mathcal{F}_{\mathrm{OT}_l^m}$ efficiently. It is trivial to compute $\mathcal{F}_{\mathrm{OT}_l^m}$ by simultaneously running $\mathcal{F}_{\mathrm{OT}} m$ times although it leads to large costs on computation and communication. Therefore, OT extension is the most efficient way for executing $\mathcal{F}_{\mathrm{OT}_l^m}$ instead of running m instances of OT in parallel.

As shown in Figure 3, the functionality of IKNP is that $\mathcal{R}$ receives messages $\{x_i^{c_i} | it \in n[m]\}$ without knowing anything about $\{x_i^{1-c_i} | it \in n[m]\}$, while $\mathcal{S}$ knows nothing about $\mathbf{c}$, where c_i denotes the i-th bit of $\mathbf{c}$. In the IKNP protocol, after acting as the *receiver* in $\mathcal{F}_{\mathrm{OT}}$ and running it k times, $\mathcal{S}$ computes m pairs of symmetric keys denoted as $\{(\mathcal{K}_1^0, \mathcal{K}_1^1), \ldots, (\mathcal{K}_m^0, \mathcal{K}_m^1)\}$, which are used to encrypt each pair of messages $\{(x_i^0, x_i^1)\}$, where $i \in [m]$. For each pair of symmetric keys $(\mathcal{K}_i^0, \mathcal{K}_i^1)$, $\mathcal{R}$ only knows the exact one according to his selection string, *i.e.*, $\{\mathcal{K}_i^{c_i} | t\, nc_i q \in hc, x\, 7iC \in ; [m]\}$. The key insight of IKNP OT extension is to execute such an OT-based key agreement between $\mathcal{S}$ and $\mathcal{R}$ in the following way:

$\mathcal{R}$ forms $m \times k$ matrix T at random and then computes matrix U such that $\mathbf{t}_i \oplus \mathbf{u}_i = (c_i \| \cdots \| c_i)$. $\mathcal{S}$ chooses $\mathbf{s} \leftarrow \{0,1\}^k$ at random. For each $j \in k$, $\mathcal{R}$ invokes $\mathcal{F}_{\mathrm{OT}}$ with input $(\mathbf{t}^j, \mathbf{u}^j)$, and $\mathcal{S}$ acts as the *receiver* with selection bit s_j. $\mathcal{S}$ receives $\mathbf{q}^j$ after each computation of $\mathcal{F}_{\mathrm{OT}}$ and then forms matrix Q. For each column of matrix Q, it implies $\mathbf{q}^j = [(s_j \oplus 1) \cdot \mathbf{t}^j] \oplus (s_j \cdot \mathbf{u}^j) = \mathbf{t}^j \oplus (s_j \cdot \mathbf{c})$. The essential observation is that, for each row of matrix Q, it holds

$$\mathbf{q}_i = \mathbf{t}_i \oplus (c_i \cdot \mathbf{s}). \tag{5}$$

$\mathcal{S}$ prepares key pairs by $\mathcal{K}_i^0 = H(\mathbf{q}_i)$ and $\mathcal{K}_i^1 = H(\mathbf{q}_i \oplus \mathbf{s})$ and $\mathcal{R}$ exactly knows $\mathcal{K}_i^{c_i} = H(\mathbf{t}_i)$, where $\mathbf{t}_i$ is generated by $\mathcal{R}$ locally. In addition, due to the randomness of $\mathbf{s}$ chosen by $\mathcal{S}$, $\mathcal{R}$ learns $\mathcal{K}_i^{1-c_i} = H(\mathbf{t}_i \oplus \mathbf{s})$ with no more than a negligible probability $(1/2^k)$. Then, $\mathcal{S}$ can execute just symmetric operations so as to encrypt each pair of messages using key pairs $(\mathcal{K}_i^0, \mathcal{K}_i^1)$ and sends m pairs of encrypted messages to $\mathcal{R}$. Finally, $\mathcal{R}$ can decrypt the corresponding message for each message pairs under $\mathcal{K}_i^{c_i} = H(\mathbf{t}_i)$.

As described above, the IKNP protocol begins with running $\mathcal{F}_{\mathrm{OT}}$ k times and then executes lots of symmetric operations to "extend" these OT instances, which are also called as "base" OTs.

2.4. Oblivious PRF. Freedman et al. [34] proposed oblivious evaluation of pseudorandom function (OPRF) and gave a general construction of OPRF from OT. An OPRF is a two-party protocol where a *sender* P_1 inputting a random seed s obtains nothing while a *receiver* P_2 inputting an evaluation point x obtains $f_s(x)$ for some pseudorandom function family f_s. The functionality of OPRF (see Figure 4) can be defined by $(s, x) \mapsto (\lambda, f_s(x))$.

A general definition of OPRF is that the receiver P_2 inputs an evaluation set $X = \{x_i\}$ and obtains the evaluations on a PRF in an oblivious manner, *i.e.*, $\{f_s(x_i)\}$. The functionality in Figure 4 can be regarded as a special and simplified case although it is sufficient to construct such an

Functionality F_{OPRF}

Let f be a pseudorandom function. Two parties: *sender* S and *receiver* R.

(i) Input:
  (1) S inputs a random seed s.
  (2) R inputs a value r.

(ii) Output:
  (1) S has no outputs.
  (2) R outputs $f_s(r)$.

Figure 4: Oblivious PRF functionality $\mathcal{F}_{\text{OPRF}}$.

efficient OPRF for some scenarios. In this paper, we consider only the definition where *receiver* P_2 evaluates the PRF on a single point x, which can be constructed massively in an efficient manner.

2.4.1. Batched OPRF Based on OT Extension. Kolesnikov and Kumaresan [5] generalized IKNP and proposed KK protocol realizing $1-\text{out}-\text{of}-2^l$ OTs in an efficient way. In IKNP, the equation $\mathbf{t}_i \oplus \mathbf{u}_i = (c_i \| \cdots \| c_i)$ means that each row of matrix $T \oplus U$ is either all zeros or all ones. This feature was interpreted as a *repetition code* in KK, and they improved it using a *linear error correcting code* denoted by $\mathscr{C}$. Now for each row in T and U, it holds that $\mathbf{t}_i \oplus \mathbf{u}_i = \mathscr{C}(c_i)$, where $\mathscr{C}$ is a public linear error correcting code of dimension l and codeword length k. Therefore, in KK, equation (5) becomes

$$\mathbf{q}_i = \mathbf{t}_i \oplus [\mathscr{C}(c_i) \odot \mathbf{s}]. \tag{6}$$

Notably, c_i stands for the i-th element in vector $\mathbf{c}$, and now, it is **no longer** a binary bit in equation (5) but an l-bit string. The codeword length of $\mathscr{C}(c_i)$ determines the length of $\mathbf{s}$ and the number of columns of matrices T and U, instead in IKNP of k (k is relative to a security parameter κ). In KK, to reach the same security requirement, the codeword length k' is about twice as much as k. That is to say, $k_{\text{KK}} \approx 2.5k_{\text{IKNP}}$. As a consequence, the number of "base" OT in KK is doubled than that in IKNP.

Based on 1-out-of-n OT extension, Kolesnikov et al. [13] proposed a variant of OPRF and described an efficient protocol to generate batched OPRF instances (known as *BaRK-OPRF*). From the point of adaption of OT extension, the variant OPRF functionality based on equation (6) is that

$$((\mathbf{q}_i, \mathbf{s}), c_i) \longmapsto (\lambda, H(i, \mathbf{q}_i \oplus [\mathscr{C}(c_i) \odot \mathbf{s}])), \tag{7}$$

where $\mathscr{C}$ now is a *pseudorandom code* instead of a linear error correcting code.

Notably, the random seed s consists of $(\mathbf{q}_i, \mathbf{s})$ in *BaRK-OPRF* and $i \in [m]$, indicating that equation (6) essentially instantiates m different OPRFs in total, and this is why *BaRK-OPRF* is called bathed and key-related OPRF. In addition, the codeword length of pseudorandom code in *BaRK-OPRF* is approximately 2 times longer than the output length of linear error correcting code in KK, which is necessary to reach security requirement. Based on OPRF, it is trivial to construct the private membership test, which will be illustrated later.

3. Overview of Our Construction

The functionality of $\mathbb{O}Tex$ is described in Figure 1. While the essential difference between $\mathcal{F}_{\text{OT}_l^m}$ in Figure 3 and $\mathcal{F}_{\mathbb{O}Tex}$ in Figure 1 may appear to be unimpressive and unnecessary, the distinction becomes more pronounced in the case of practical OT extension applications and even more so in outsourced OT scenarios.

The codeword length of code schemes in equations (6) and (7), *i.e.,* pseudorandom code and linear error correcting code, determines the number of "base" OT instances to be evaluated. This gives us the intuition that we could use a specific number of OTs to extend any large amount of OT instances we need. Both Ishai et al. [4] and Kolesnikov et al. [5, 13] show n OTs of long strings that can be reduced to k "base" OTs of shorter strings. That is, given pseudo-random generator and a small number of OTs, we can implement any $\mathcal{F}_{\text{OT}_{l'}^{m'}}$ we want. Therefore, we make OT execute in a program-iteration-like way (see Figure 5) to reach the final functionality $\mathcal{F}_{\text{OT}_l^m}$. Here, for clarity, we denote selection string in $\mathcal{F}_{\text{OT}_{k_i}^{k_{i+1}}}$ by bold letter $\mathbf{r}_i$ instead of $\mathbf{c}$, and j-th element of $\mathbf{r}_i$ is denoted by $\mathbf{r}_i[j]$.

To better understand our work, let us review IKNP OT extension where $\mathcal{S}$ holds m pairs of l-length messages $\{(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)\}$, and the receiver $\mathcal{R}$ holds m-bit selection vector $\mathbf{c}$. According to Figure 5, we now focus on $\text{OT}_{k_2}^{k_1}$ and $\text{OT}_{k_3}^{k_2}$ only. For $i \in [m]$, each row of matrices T_2 and T_2' consists of x_i^0 and x_i^1, respectively; that is, both T_2 and T_2' are $m \times l$ matrices and $\mathbf{r}_2 = \mathbf{c}$. Now, $\text{OT}_{k_2}^{k_1}$ serves as "base" OT in IKNP, while the final functionality is to realize $\text{OT}_{k_3}^{k_2}$, where k_1 is the security parameter, $k_2 = m$, and $k_3 = l$. After $\text{OT}_{k_2}^{k_1}$, $\mathcal{S}$ computes equation (5) and sends encrypted T_2 and T_2' to $\mathcal{R}$. Using symmetric-key operations only, $\mathcal{R}$ receives V_2 from T_2 and T_2' in an oblivious way. For $i \in [m]$, each row of matrix V_2 is $x_i^{c_i}$. We have reviewed the whole framework of IKNP protocol so far, and it does matter the relationships among T_1, T_1', and $\mathbf{r}_2$. If we write the algorithm implementing $\text{OT}_{k_{i+1}}^{k_i}$ as recursive function $F_{\text{OT}(k_i,k_{i+1})}$ in programming language, then the key step in IKNP is that $F_{\text{OT}(m,l)}$ invokes $F_{\text{OT}(\kappa,m)}$; *i.e.,* OT_l^m is reduced to OT_m^κ, where κ is the security parameter in IKNP. Thus, we implement $F_{\text{OT}(m,l)}$ by invoking $F_{\text{OT}(\kappa,m)}$ first and then executing some symmetric-key encryption operations.

In more general case, two parties P_1 and P_2 prepare to run protocol $\text{OT}_{k_{i+1}}^{k_i}$, where P_1 acts as sender $\mathcal{S}$ holding messages matrices T_i and T_i' while P_2 holds chosen vector $\mathbf{r}_i$. After $\text{OT}_{k_{i+1}}^{k_i}$, P_2 gets outputs V_i consisting of those messages he/she chooses to receive. According to OT extension protocol, P_1 and P_2 first run $OT_{k_i}^{k_{i-1}}$, where P_1 acts as a *receiver* and P_2 acts as a *sender*. For each $\text{OT}_{k_{i+1}}^{k_i}$ and $\text{OT}_{k_i}^{k_{i-1}}$, it holds

$$T_{i-1}[j] \oplus T_{i-1}'[j] = f(\mathbf{r}_i[j]). \tag{8}$$

$T_{i-1}[j]$ denotes the j-th row of matrix $T_{i-1}[j]$, $\mathbf{r}_i[j]$ denotes j-th element of $\mathbf{r}_i$, and $f(\cdot)$ for some encoding scheme, such as repetition code in equation (5) and linear error correcting code in equation (6). We present detailed

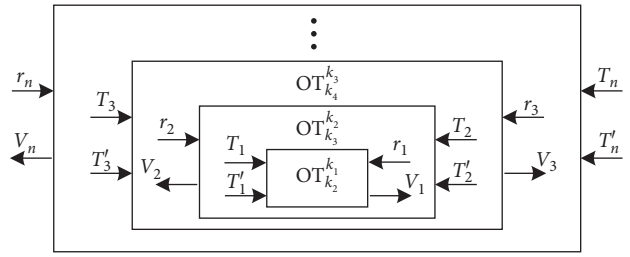


Figure 5: Overview of OT iteration.

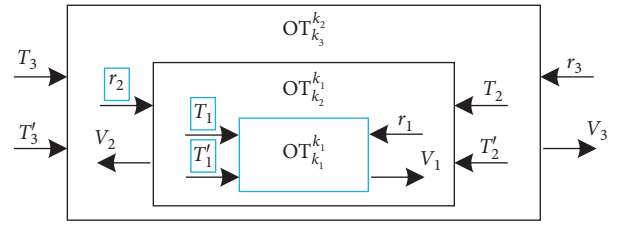


Figure 6: Construction of $\mathbb{O}Tex$.

$f(\cdot)$ in concrete protocol and just emphasize the connections between T_{i-1} and $\mathbf{r}_i$ here.

$\mathrm{OT}_{k_{i+1}}^{k_i}$ and $\mathrm{OT}_{k_i}^{k_{i-1}}$ are named as *outer* and *inner* OT, respectively. Specially, $\mathrm{OT}_{k_2}^{k_1}$ is the innermost one. Then, we find that the two input vectors T_{i-1} and $T_{i-1}{}'$ of the inner $\mathrm{OT}_{k_i}^{k_{i-1}}$ are determined by the input $\mathbf{r}_i$ of the outer $\mathrm{OT}_{k_{i+1}}^{k_i}$. The role of P_1 and P_2 get reversed each time $F_{\mathrm{OT}(k_i,k_{i+1})}$ invokes $F_{\mathrm{OT}(k_{i-1},k_i)}$; that is to say, $\mathcal{S}$ in outer OT becomes the *receiver* in inner OT, so as to $\mathcal{R}$. Furthermore, $\mathrm{OT}_{k_i}^{k_{i+1}}$ (if exists) is the *outer* OT related to $\mathrm{OT}_{k_{i-1}}^{k_i}$; therefore, the input vectors of T_i and T_i' are determined by $\mathbf{r}_{i+1}$.

The transformation among $\mathrm{OT}_{k_{i+2}}^{k_{i+1}}$, $\mathrm{OT}_{k_{i+1}}^{k_i}$, and $\mathrm{OT}_{k_i}^{k_{i-1}}$ is the main part of our work. The key observation is that the inputs of both two parties in $\mathrm{OT}_{k_i}^{k_{i-1}}$ and $\mathrm{OT}_{k_{i+2}}^{k_{i+1}}$ are independent. That is, T_{i+1} and T_{i-1}, $T_{i+1}{}'$ and $T_{i-1}{}'$, and $\mathbf{r}_{i+1}$ and $\mathbf{r}_{i-1}$ are independent. If $\mathrm{OT}_{k_{i+2}}^{k_{i+1}}$ is a necessary component in protocol, we can precompute $\mathrm{OT}_{k_i}^{k_{i-1}}$ assisted with a semihonest server. We focus on a special-but-sufficient case where $\mathcal{S}$ and $\mathcal{R}$ prepare to execute $\mathrm{OT}_{k_3}^{k_2}$; notably, the innermost is $\mathrm{OT}_{k_1}^{k_1}$ (see Figure 6) instead of $\mathrm{OT}_{k_2}^{k_1}$. In such a scenario, k_1 should not be less than security parameter. If $\mathcal{R}$ is a normal user acting as the receiver in $\mathrm{OT}_{k_3}^{k_2}$ and $\mathcal{S}$ is a database with limited computation power and interaction capability, the framework becomes more useful and efficient. Therefore, we consider a server-aided oblivious transfer reducing the sender's all public-key computation and numbers of interaction with other parties. As shown in Figure 6, $\mathcal{S}$ outsources the computation marked with blue rectangle to a server.

Our $\mathbb{O}Tex$ protocol consists of two major phases. First, in the **outsourced phase**, $\mathcal{R}$ and *server* $\mathbb{S}$ run $\mathrm{OT}_{k_1}^{k_1}$ as follows. $\mathcal{R}$ inputs random selection string $\mathbf{r}_1$, while $\mathbb{S}$ inputs two random $k_1 \times k_1$ matrices T_1 and T_1'. After $\mathrm{OT}_{k_1}^{k_1}$, $\mathcal{R}$ gets output V_1 and prepares T_2 (and T_2') for $\mathrm{OT}_{k_2}^{k_1}$. In the second phase, $\mathcal{S}$ aims to **respond to the request** for $\mathrm{OT}_l^{k_2}$. This phase can be concluded as $F_{\mathrm{OT}(k_1,k_2)}$ invoking $F_{\mathrm{OT}(k_1,k_1)}$ and $\mathcal{S}$ sending pairs of messages (T_3 and T_3' encrypted by V_2) to $\mathcal{R}$. In the last phase, $\mathcal{R}$ gets outputs from $OT_l^{k_2}$, which can be seen as $F_{\mathrm{OT}(k_2,l)}$ invoking $F_{\mathrm{OT}(k_1,k_2)}$.

4. Outsourced Oblivious Transfer Extension

In this section, we show how to construct an outsourced oblivious transfer extension protocol $\mathbb{O}Tex$, where three-party functionality $\mathcal{F}_{\mathbb{O}Tex}$ can be securely computed in the presence of semihonest adversaries.

Our $\mathbb{O}Tex$ protocol consists of two major phases among three parties, *sender* $\mathcal{S}$, *receiver* $\mathcal{R}$, and *server* $\mathbb{S}$. Figure 6 shows $\mathbb{O}Tex$ construction in a program-iteration-like way, where a small number of OT instances, the innermost $\mathrm{OT}_{k_1}^{k_1}$, are first to be executed cooperatively. In fact, we let $k_1 = \kappa$, and it becomes $\mathrm{OT}_{\kappa}^{\kappa}$, where κ is the security parameter in real protocol. From the global perspective, we give procedure of $\mathbb{O}Tex$ in Figure 7 so as to have a better understanding of roles three parties play in every phase.

First, in the **outsourced phase**, $\mathcal{R}$ and $\mathbb{S}$ run $\mathrm{OT}_{\kappa}^{\kappa}$ as follows. $\mathcal{R}$ inputs random selection string $\mathbf{r}$, while $\mathbb{S}$ inputs two random $\kappa \times \kappa$ matrices T and T'. After $\mathrm{OT}_{\kappa}^{\kappa}$, $\mathcal{R}$ gets outputs D and prepares V (and V') for OT_m^{κ}. In the second phase, $\mathcal{S}$ aims to **respond to the request** for OT_l^m. This phase can be concluded as $F_{\mathrm{OT}(\kappa,m)}$ invoking $F_{\mathrm{OT}(\kappa,\kappa)}$ and $\mathcal{S}$ sending pairs of messages (T and T' encrypted by E) to $\mathcal{R}$. In the last phase, $\mathcal{R}$ gets outputs from OT_l^m, which can be seen as $F_{\mathrm{OT}(m,l)}$ invoking $F_{\mathrm{OT}(\kappa,m)}$.

We describe $\mathbb{O}Tex$ in Figure 8 which realizes functionality $F_{\mathbb{O}Tex}$ in Figure 4. The invoking procedure in $\mathbb{O}Tex$ can be written as $F_{\mathrm{OT}(m,l)} \gg F_{\mathrm{OT}(\kappa,m)} \gg F_{\mathrm{OT}(\kappa,\kappa)}$.

According to equation (8), each time $F_{\mathrm{OT}(k_i,k_{i+1})}$ invokes $F_{\mathrm{OT}(k_{i-1},k_i)}$, and it holds $T_{i-1}[j] \oplus T_{i-1}{}'[j] = f(\mathbf{r}_i[j])$. Let $f(\cdot)$ represent different encoding schemes each time iteration occurs. In $\mathbb{O}Tex$, we have $T[i] \oplus T'[i] = f_1(\mathbf{s})$ and $V[j] \oplus V'[j] = f_2(\mathbf{c})$, where both $f_1(\cdot)$ and $f_2(\cdot)$ are repetition code with output length κ, *i.e.*,

$$\begin{aligned} f_1(\mathbf{s}) &= s_i \| \cdots \| s_i, \\ f_2(\mathbf{c}) &= c_i \| \cdots \| c_i. \end{aligned} \tag{9}$$

Notably, in more general case, the outputs of $f_1(\mathbf{r}_i[j])$ and $f_1(\mathbf{r}_{i+1}[j])$ may be not equal length, which is determined by $\mathrm{OT}_{k_i}^{k_{i-1}}$ and $\mathrm{OT}_{k_{i+1}}^{k_i}$, respectively. Figures 7 and 8 illustrate $\mathbb{O}Tex$ framework and procedure, where the symbols are consistent and the output length of repetition code is embodied in the number of columns in the matrices T and V.

Theorem 1. *The $\mathbb{O}Tex$ protocol in Figure 8 securely computes the functionality $F_{\mathbb{O}Tex}$ (Figure 4) in semihonest setting, as described in Definition 1, given random oracle and functionality $\mathcal{F}_{OT}$(Figure 1).*

Proof. We begin by proving the correctness. After $\mathbb{O}Tex$, we prove that $\mathcal{R}$ only receives $x_i^{c_i}$ and knows nothing about $x_i^{1-c_i}$ when computing $\tilde{x}_i = \tilde{z}_i^{c_i} \oplus H(i, \mathbf{v}_i)$. In the *outsourced phase*, it holds that

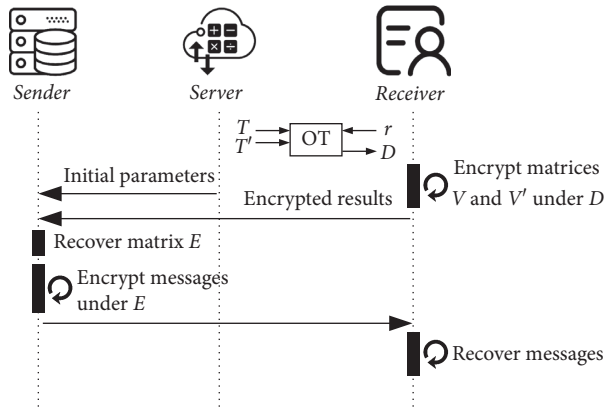


Figure 7: Procedure of outsourced oblivious transfer.

INPUT OF *sender* S: m pairs (x_i^0, x_i^1) of l-bit binary string.
INPUT OF *server* $\mathbb{S}$: random string $s \leftarrow \{0,1\}^\kappa$.
INPUT OF *receiver* R: m selection bit $c = c_1 || \ldots || c_m$.
COMMON INPUT: a security parameter κ.
ORACLE: *random oracles* H: $[m] \times \{0,1\}^\kappa \to \{0,1\}^l$, $H_1 : [\kappa] \times \{0,1\}^\kappa \to \{0,1\}^m$.
CRYPTOGRAPHIC PRIMITIVE : an ideal $\mathrm{OT}_\kappa^\kappa$ primitive.

(i) Outsourced phase:
(1) R chooses $r \leftarrow \{0,1\}^\kappa$ at random.
(2) $\mathbb{S}$ computes $\kappa \times \kappa$ matrix T at random and sets matrix T' such that $t_i \oplus t_i' = (s_i || \ldots || s_i)$.
(3) For each $j \in [\kappa]$, $\mathbb{S}$ and R invoke $F_{\mathrm{OT}_\kappa^\kappa}$ in the following way:
(a) R acts as *recevier* with input r_j.
(b) S acts as *sender* with input (t^j, t'^j).
(c) R recevies output d^j.
R forms $\kappa \times \kappa$ matrix D such that the j-th column of D is the vector d^j.

(ii) Responding phase:
(4) $\mathbb{S}$ sends string s and matrix T to S.
(5) R forms $m \times \kappa$ matrix V at random and sets matrix T' such that $v_i \oplus v_i' = (c_i || \ldots || c_i)$, where $i \in [m]$. For $j \in [\kappa]$, R computes $y_j^0 = v^j \oplus H_1(j, d_j)$ and $y_j^1 = v'^j \oplus H_1(i, d_j \oplus r)$ and then sends messages (y_j^0, y_j^1) to S.
(6) S recovers $m \times \kappa$ matrix E by computing $e^j = y_j^{s_j} \oplus H_1(j, t_j)$, $j \in [\kappa]$. For $i \in [m]$, S computes $z_i^0 = x_i^0 \oplus H(i, e_i)$ and $z_i^1 = x_i^1 \oplus H(i, e_i \oplus s)$ and then sends messages (z_i^0, z_i^1) to R.

(iii) Outputting phase:
(7) For $i \in [m]$, after receiving $(\tilde{z}_i^0, \tilde{z}_i^1)$ to R outputs $\tilde{x}_i = \tilde{z}_i^{c_i} \oplus H(i, v_i)$.

Figure 8: Outsourced oblivious transfer extension protocol.

$$\begin{aligned} \mathbf{d}^j &= \mathbf{t}^j \oplus \left[r_j \cdot \mathbf{s}\right], \\ \mathbf{d}_i &= \mathbf{t}_i \oplus \left[s_i \cdot \mathbf{r}\right], \quad \text{where } i, j \in [\kappa]. \end{aligned} \tag{10}$$

Then, in the responding phase, Step 5, $\mathscr{R}$ essentially computes

$$\begin{aligned} y_j^0 &= \mathbf{v}^j \oplus H_1\left(j, \mathbf{t}_j \oplus \left[s_j \cdot \mathbf{r}\right]\right), \\ y_j^1 &= \mathbf{v}'^j \oplus H_1\left(j, \mathbf{t}_j \oplus \left[\left(s_j \oplus 1\right) \cdot \mathbf{r}\right]\right), \quad \text{where } j \in [\kappa]. \end{aligned} \tag{11}$$

Therefore, in Step 6, for $j \in [\kappa]$, by computing $\mathbf{e}^j = y_j^{s_j} \oplus H_1(j, \mathbf{t}_j)$, $\mathscr{S}$ gets $\mathbf{e}^j = \mathbf{v}^j$ when $s_j = 0$ and gets $\mathbf{e}^j = \mathbf{v}'^j$ when $s_j = 1$. Due to $\mathbf{v}_i \oplus \mathbf{v}_i' = (c_i \| \cdots \| c_i)$, we have $\mathbf{e}^j = \mathbf{v}^j \oplus [s_j \cdot \mathbf{c}]$ and

$$\mathbf{e}_i = \mathbf{v}_i \oplus \left[c_i \cdot \mathbf{s}\right], \quad \text{where } i \in [m], j \in [\kappa]. \tag{12}$$

For $\sigma \in \{0,1\}$, $\mathscr{S}$ computes $z_i^\sigma = x_i^\sigma \oplus H(i, \mathbf{e}_i \oplus [\sigma \cdot \mathbf{s}])$ and essentially sends to $\mathscr{S}$ the following messages:

$$z_i^\sigma = x_i^\sigma \oplus H\left(i, \mathbf{v}_i \oplus \left[c_i \oplus \sigma\right] \cdot \mathbf{s}\right). \tag{13}$$

It holds that $H(i, \mathbf{v}_i \oplus [c_i \oplus \sigma] \cdot \mathbf{s}) = H(i, \mathbf{v}_i)$ if and only if $c_i = \sigma$. Concretely, when $c_i = 0$, $\mathscr{R}$ computes

$$\widetilde{x}_i = \widetilde{z}_i^0 \oplus H(i, \mathbf{v}_i) = x_i^0 \oplus H(i, \mathbf{v}_i \oplus [0 \oplus 0] \cdot \mathbf{s}) \oplus H(i, \mathbf{v}_i) = x_i^0. \tag{14}$$

When $c_i = 1$, $\mathcal{R}$ computes

$$\widetilde{x}_i = \widetilde{z}_i^1 \oplus H(i, \mathbf{v}_i) = x_i^1 \oplus H(i, \mathbf{v}_i \oplus [1 \oplus 1] \cdot \mathbf{s}) \oplus H(i, \mathbf{v}_i) = x_i^1. \tag{15}$$

In summary, $\mathcal{R}$ only receives $x_i^{c_i}$ in $\mathbb{O}Tex$ and cannot recover $x_i^{1-c_i}$ by computing $\widetilde{z}_i^{1-c_i} \oplus H(i, \mathbf{v}_i)$ because $\mathcal{R}$ knows nothing about $H(i, \mathbf{v}_i \oplus [0 \oplus 1] \cdot \mathbf{s})$, *i.e.,* $H(i, \mathbf{v}_i \oplus [c_i \oplus (1 - c_i)] \cdot \mathbf{s})$.

This concludes the correctness of $\mathbb{O}Tex$.

We now construct three simulators $Sim_{\mathbb{S}}$, $Sim_{\mathcal{R}}$, and $Sim_{\mathcal{R}}$ for simulating corrupt $\mathcal{S}$, $\mathbb{S}$, and $\mathcal{R}$ such that the produced transcript and the view of real execution are computationally indistinguishable. That is,

$$\begin{aligned} &\{Sim_{\mathbb{S}}(1^\kappa, \mathbf{s}, \perp)\}_{x,\mathbf{s},\mathbf{c},\kappa}{}^c \equiv \{view_{\mathbb{S}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{x,\mathbf{s},\mathbf{c},\kappa}, \\ &\{Sim_{\mathcal{S}}(1^\kappa, x, \perp)\}_{x,\mathbf{s},\mathbf{c},\kappa}{}^c \equiv \{view_{\mathcal{S}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{x,\mathbf{s},\mathbf{c},\kappa}, \\ &\{Sim_{\mathcal{R}}(1^\kappa, \mathbf{c}, \widetilde{x})\}_{x,\mathbf{s},\mathbf{c},\kappa}{}^c \equiv \{view_{\mathcal{R}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{x,\mathbf{s},\mathbf{c},\kappa}, \end{aligned} \tag{16}$$

where x denotes input set $\{(x_i^0, x_i^1)\}$ from $\mathcal{S}$, $\widetilde{x}$ denotes output $\{\widetilde{x}_i\}$ of $\mathcal{R}$, and $\perp$ denotes the null value.

Corrupt $\mathbb{S}$. Given $\mathcal{F}_{\mathrm{OT}}$, it is easy to perfectly simulate the view of the *Server* $\mathbb{S}$ because $\mathbb{S}$ only has input, neither receives messages nor outputs during the execution of the protocol. That is, $\{Sim_{\mathbb{S}}(1^\kappa, \mathbf{s}, \perp)\}_{x,\mathbf{s},\mathbf{c},\kappa}$ is computationally indistinguishable with $\{view_{\mathbb{S}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{x,\mathbf{s},\mathbf{c},\kappa}$.

Corrupt $\mathcal{S}$. In protocol, *Sender* $\mathcal{S}$ works only in the *responding phase*, when $\mathcal{S}$ receives messages from $\mathbb{S}$ and $\mathcal{R}$ then sends encrypted inputs to $\mathcal{R}$. The messages obtained by $\mathcal{S}$ are $\{\mathbf{s}, T, (y_j^0, y_j^1)\}$. That is to say, $view_{\mathcal{S}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)$ consists of input x and messages $\{\mathbf{s}, T, (y_j^0, y_j^1)\}$, where $j \in [\kappa]$. It is trivial for $Sim_{\mathcal{S}}$ to generate a simulation of $\{\mathbf{s}, T\}$ since both are random values from the point of view of $\mathcal{S}$.

Now, $Sim_{\mathcal{S}}$ simulates a simulation of $\{(y_j^0, y_j^1) | j \in [\kappa]\}$ by choosing a κ-length string $\widehat{\mathbf{s}}$, m-length strings $\widehat{\mathbf{r}}_j$, $\kappa \times \kappa$ matrix $\widehat{T}$, and $m \times \kappa$ matrix $\widehat{E}$ at random and computing

$$\begin{aligned} \widehat{y}_j^{\widehat{\mathbf{s}}_j} &= \widehat{\mathbf{e}}^j \oplus H_1\left(j, \widehat{\mathbf{t}}_j\right), \\ \widehat{y}_j^{1-\widehat{\mathbf{s}}_j} &= \widehat{\mathbf{r}}_j. \end{aligned} \tag{17}$$

Let $\{\widehat{\mathbf{s}}, t\widehat{T}n, q(\widehat{y}_j^0, \widehat{y}_j^1)\}$ be the output of $Sim_{\mathcal{S}}(1^\kappa, x, \perp)$. Therefore, we claim that the view generated by $Sim_{\mathcal{S}}$ and the view of corrupt $\mathcal{S}$ in a real protocol are computationally indistinguishable.

Corrupt $\mathcal{R}$. We construct a simulator $Sim_{\mathcal{R}}$ that simulates the view of corrupt $\mathcal{R}$ in the real protocol execution. We first analyze $\mathcal{R}$'s view $view_{\mathcal{R}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)$ in $\mathbb{O}Tex$. $\mathcal{R}$ obtains matrix D in the outsourced phase and $(\widetilde{z}_i^0, \widetilde{z}_i^1)$ in the outputting phase. Therefore, $view_{\mathcal{R}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)$ consists of $\mathcal{R}$'s input $\mathbf{c}$ and messages $\{D, (\widetilde{z}_i^0, \widetilde{z}_i^1)\}$. To construct the simulation of the view, $Sim_{\mathcal{R}}$ works as follows.

(i) In the outsourced phase, given the security parameter κ, $\mathcal{R}$'s input $\mathbf{c}$, and the randomness $\mathbf{r}$, $Sim_{\mathcal{R}}$ calls simulation $Sim_{\mathrm{OT}_\kappa^\kappa}$ with input $(\widehat{T}, \widehat{T}', r)$ and gets output $\widehat{D}$. Then, $Sim_{\mathcal{R}}$ appends the output of $Sim_{\mathrm{OT}_\kappa^\kappa}$, *i.e.,* matrix $\widehat{D}$, to its own output.

(ii) In the outputting phase, given the security parameter κ, $\mathcal{R}$'s input $\mathbf{c}$, the randomness V, and output $\widetilde{x}$, $Sim_{\mathcal{R}}$ simulates a simulation of $\{(\widetilde{z}_i^0, \widetilde{z}_i^1) | it \in n[m]\}$ by choosing m-length strings $\widehat{\mathbf{r}}_j$ and computing

$$\begin{aligned} \widehat{z}_i^{\mathbf{c}_i} &= \widetilde{x}_i \oplus H(i, \mathbf{v}_i), \\ \widehat{z}_i^{1-\mathbf{c}_i} &= \widehat{\mathbf{r}}_j. \end{aligned} \tag{18}$$

Then, $Sim_{\mathcal{R}}$ appends $\{(\widehat{z}_i^0, \widehat{z}_i^1) | it \in n(m\}$ to its output.

Combining two phases described above in sequence, we finally claim that the output of $Sim_{\mathcal{R}}$ is computationally indistinguishable from the real execution, *i.e.,* $\{Sim_{\mathcal{R}}(1^\kappa, \mathbf{c}, \widetilde{x})\}_{x,\mathbf{s},\mathbf{c},\kappa}{}^c \equiv \{view_{\mathcal{R}}^{\pi}(x, \mathbf{s}, \mathbf{c}, \kappa)\}_{x,\mathbf{s},\mathbf{c},\kappa}$.

This completes the construction of three simulators: $Sim_{\mathbb{S}}$, $Sim_{\mathcal{R}}$, and $Sim_{\mathcal{R}}$. In summary, $\mathbb{O}Tex$ is secure under semihonest model. □

4.1. Performance Analysis. We now analyze the performance of $\mathbb{O}Tex$ in semihonest model. The complexities of $\mathbb{O}Tex$ are presented in Table 1.

(i) *Computation complexity*. The number of symmetric and asymmetric operation to be executed in $\mathbb{O}Tex$ essentially depends on the size of matrices T, V, and E, respectively. In the outsourced phase, $\mathcal{F}_{\mathrm{OT}_\kappa^\kappa}$ consists of $O(\kappa)$ public-key and $O(\kappa)$ private-key operations, for both $\mathbb{S}$ (acting as sender) and $\mathcal{R}$ (acting as receiver). In the responding phase, $\mathcal{R}$ invokes random oracle and performs XOR operation for $O(\kappa)$ times. Then, $\mathcal{S}$ recovers matrix E and encrypts messages (x_i^0, x_i^1), which only consists of $O(m + \kappa)$ symmetric operations. In the outputting phase, $\mathcal{R}$ performs symmetric operations $O(m)$ times to compute its outputs.

(ii) *Communication complexity*. In the outsourced phase, the number of bits transferred between $\mathbb{S}$ and $\mathcal{R}$ is bounded by $O(\kappa^2)$. In the responding phase, $\mathcal{S}$ receives messages from $\mathbb{S}$ and $\mathcal{R}$ and sends encrypted messages to $\mathcal{R}$, which consist of $O(ml + \kappa^2)$. In the outputting phase, $\mathcal{R}$ receives $2ml$-bit messages in total from $\mathcal{S}$.

(iii) *Round complexity*. The only interaction in $\mathbb{O}Tex$ exists in the outsourced phase between $\mathbb{S}$ and $\mathcal{R}$, *i.e.,* $\mathcal{F}_{\mathrm{OT}_\kappa^\kappa}$. The number of interaction round in $\mathcal{F}_{\mathrm{OT}_\kappa^\kappa}$ is bounded by 1.

Table 1: Complexity of $\mathbb{O}Tex$.

Party	Computation		Communication	Round
	Asymmetric	Symmetric		
Server $\mathbb{S}$	$O(\kappa)$	$O(\kappa)$	$O(\kappa^2)$	1
Sender $\mathcal{S}$	–	$O(m+\kappa)$	$O(ml+\kappa^2)$	–
Receiver $\mathcal{R}$	$O(\kappa)$	$O(m+\kappa)$	$O(ml+\kappa^2)$	1

Table 2: Efficiency comparison.

Protocol	Round	Communication	Asymmetric computation	Security model
[4]	2	$O(ml+m\kappa)$	$O(\kappa)$	Semihonest and malicious
[5]	2	$O((mk/\log n)+\ln\log n)$	$O(k)$	Semihonest
[16]	2	$O(ml+\kappa^2)$	$O(\kappa+\ln)$	Semihonest and one-sided malicious
[30]	2	$O(ml+m\kappa)$	$O(\kappa)$	Semihonest
[31]	3	$O(ml+\log\lvert\mathbb{G}\rvert+\kappa)$	$O(\kappa)$	Malicious
Ours	–	$O(ml+\kappa^2)$	–	Semihonest

Note: the efficiency of *sender* in OT is presented here, who inputs m pairs of l-bit length messages in protocol. Specially, κ is security parameter, $\mathbb{G}$ denotes a group, and $k \approx 2.5\kappa$ in 1-out-of-n OT.

4.2. Efficiency Comparison. Since we focus on the efficiency of the sender $\mathcal{S}$ in OT, we provide comparisons to prior classical protocols from the $\mathcal{S}$'s point of view in Table 2. In $\mathbb{O}Tex$, all asymmetric operations are operated between the receiver $\mathcal{R}$ and the cloud server $\mathbb{S}$, and $\mathcal{S}$ works on the fly and conducts symmetric computations locally.

5. Performance

In this section, we test the performance of $\mathbb{O}Tex$. The experiments were performed on a Linux machine equipped with 4 cores 3.40 GHz Intel Core i5-7500 CPU and 8 GB RAM.

Our tests refer to the implement on GitHub: https://github.com/emp-toolkit/emp-ot. We simulated main asymmetric operations in $\mathbb{O}Tex$ on a single machine. We compared $\mathbb{O}Tex$ with the OT protocol used in recent work [32], where they compute at least 450 OT instances locally from 128 "base" OTs. Therefore, we regarded $n = 2^8$ as a reference and simulated the outsourced phase. After setting the length of security parameter equal to 128, we simulated $\mathbb{O}Tex$ for $n = 2^7$, 2^9, and 2^{10}, respectively. We repeated each simulation 20 times, and the result is shown in Figure 9. Although $\mathbb{O}Tex$ shows an almost identical performance to Vladimir's, the running time tends to be steady sooner when $n = 2^9$. In addition, in Figure 9, the sender $\mathcal{S}$ of $\mathbb{O}Tex$ works offline for most of the time during simulation, and this is one of the advantages of $\mathbb{O}Tex$.

The other advantage of $\mathbb{O}Tex$ found in performance tests is the reduction of communication bottleneck caused by the sender $\mathcal{S}$. The main idea of OT extension protocol is to extend a small number of base OTs to perform many OTs. As a consequence, we summed from the time of base OTon setup to the time of protocol finish and then computed average time for each original OT.

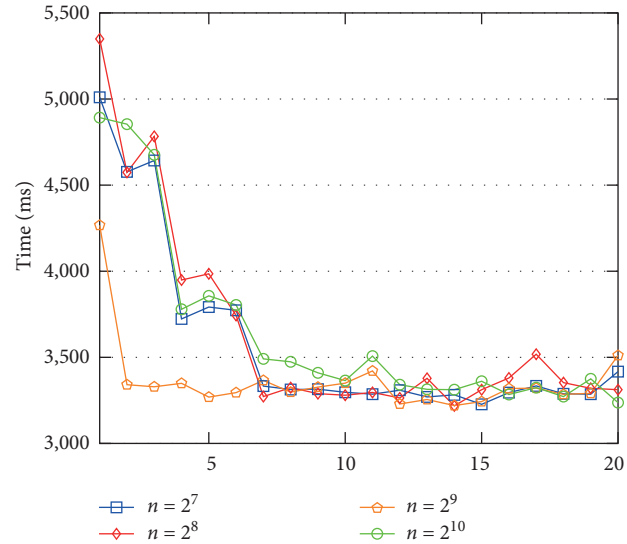


Figure 9: Comparison of running time.

The averaged time is illustrated in Figure 10. We tested $n' = 2^{8+x}$ OT instances for different x values and computed the average running time for one single OT, *i.e.*, OT_n^1. From Figure 10, we can see that the average time keeps steady when x is less than 8. If we extend base OT to an adequate number of OTs, for example, $n' = 2^{18}$, the average time has a sharp increase. It does not matter in $\mathbb{O}Tex$, however, since the extension process is conducted between $\mathcal{R}$ and $\mathbb{S}$ in the outsourced phase.

6. Applications of $\mathbb{O}Tex$

The $\mathbb{O}Tex$ framework has a wide range of applications in outsourced scenarios. In this section, we take private set intersection (PSI) as a case study and demonstrate that, as a

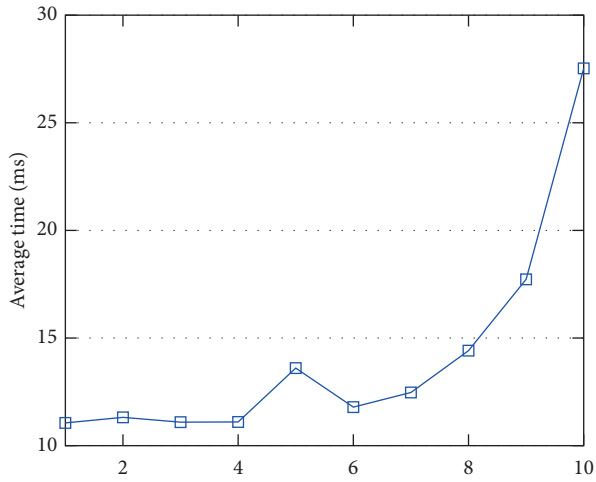


Figure 10: Average time for OT instances in $\mathbb{O}Tex$.

building block, $\mathbb{O}Tex$ can be applied conveniently to high-level protocol.

We first introduce a private membership test (PMT) protocol to estimate whether an element x belongs to a set $Y = \{y_1, \ldots, y_m\}$ or not and then describe how to efficiently implement it via $\mathbb{O}Tex$. The resulting protocol can then be simply extended to compute functionality $\mathcal{F}_{\text{PSI}}$ in Figure 11 by applying the $\mathbb{O}Tex$-based PMT protocol.

6.1. OPRF-Based PMT Protocol. PMT protocol involves two parties: *Alice* who holds an element x and *Bob* who holds a set Y. After PMT, *Bob* knows nothing about x, and *Alice* only knows whether her element x belongs to the set Y or not.

Based on OPRF, we can construct PMT protocol as follows. First, during the OPRF phase, *Bob* acts as *sender* with a random seed s. And *Alice*, acting as *receiver*, inputs her element x. After OPRF, *Alice* obtains $f_s(x)$ and *Bob* has $f_s(\cdot)$. Then, *Bob* sends all $\{f_s(y_i)\}$ to *Alice* who compares $f_s(x)$ with $\{f_s(y_i)\}$ one by one. In conclusion, *Alice* outputs 1 if and only if there exits an i such that $f_s(y_i) = f_s(x)$ and else outputs 0.

The security of PMT via OPRF relies on the fact that an OPRF protocol is secure. In a secure OPRF, it guarantees that $f_s(\cdot)$ is a one-way pseudorandom function and the length of s equals to secure parameter. Therefore, *Alice* receives $\{f_s(y_i)\}$ and then learns whether there exists an i such that $f_s(y_i) = f_s(x)$, but she will not learn the exact values $\{y_i | y_i t \in nYq, hf_{s(}xy_i 7)C \neq ; f_s(x)\}$ with non-negligible probability. We omit concrete security proof here.

6.2. $\mathbb{O}Tex$-Based PMT Protocol. Given functionality $\mathcal{F}_{\text{OPRF}}$, we can construct PMT protocol in a simple manner. In this section, we introduce how to design OPRF protocol based on $\mathbb{O}Tex$. Equation (7) indicates that 1-out-of-n OT extension implies *BaRK-OPRF*; therefore, we focus on the transformation from 1-out-of-n OT extension to $\mathbb{O}Tex$. As a result, $\mathbb{O}Tex$ can be easily used to construct PMT protocol.

We can apply pseudorandom code to equation (8) for defining new relationship among T_2 and T_2' in Figure 6,

Functionality F_{PSI}

Two parties: sender S and receiver R.

(i) Input:
  (a) S inputs $Y = \{y_1, \ldots, y_m\}$.
  (b) R inputs $X = \{x_1, \ldots, x_m\}$.

(ii) Output:
  (a) S has no outputs.
  (b) R outputs $X \cap Y$.

Figure 11: Private set intersection functionality $\mathcal{F}_{\text{PSI}}$.

which implements functionality $\mathcal{F}_{\text{OPRF}}$ in Figure 3. In $\mathbb{O}Tex$, however, we stress that it *always* holds $T_1[i] \oplus T_1'[i] = f_1(\mathbf{r}_2[i])$ and $T_2[j] \oplus T_2'[j] = f_1(\mathbf{r}_3[j])$, where $f_1(\cdot)$ is the repetition code. This means that we implement INKP protocol in an outsourced manner since we use the same repetition code $f_1(\cdot)$ in $F_{\text{OT}(k_1,k_2)}$ and $F_{\text{OT}(k_2,k_3)}$. That is to say, three parties, $\mathbb{S}$, $\mathcal{S}$, and $\mathcal{R}$, evaluate corporately m instances of 1-out-of-2 OT of l-bit messages where repetition code is used twice in total, each by $\mathbb{S}$ and $\mathcal{R}$, respectively. Now, suppose *Alice* and *Bob* would like to execute m instances of 1-out-of-n OT where *Bob* inputs $\{(y_1^j, y_2^j, \ldots, y_n^j) | j \in [m], |y_i^j| = l\}$ and *Alice* inputs her selection vector $\mathbf{r}_3$, and they need make the following adjustments:

(i) *Bob* organizes the inputs in $\mathbb{O}Tex$ with $m \times l$ matrices $T_3^{(1)}, T_3^{(2)}, \ldots, T_3^{(n)}$, where the j-th row of matrix $T_3^{(i)}$ is y_i^j.

(ii) *Alice* makes her selection vector in $\mathbb{O}Tex$ be $\{\mathbf{r}_3 | jt \in n[m]q, h\mathbf{r}_{3[}j]x \in 7[n]\}$.

Then, the only adaption they need take in $\mathbb{O}Tex$ is that $T_2[j] \oplus T_2'[j] = f_2(\mathbf{r}_3[j])$, where $f_2(\cdot)$ is the linear error correcting code $\mathscr{C}$. To reach the security requirement defined in [13], in $\mathbb{O}Tex$, we need to reset security parameter k rather than κ—concretely $3\kappa < k < 4\kappa$.

Given 1-out-of-n OT protocol designed from $\mathbb{O}Tex$, it is trivial to design $\mathbb{O}Tex$-based OPRF so is to $\mathbb{O}Tex$-based PMT. Again, *Alice* prepares her element $x = \mathbf{r}_3[1]$, while *Bob* holds set $Y = \{y_1^1, y_2^1, \ldots, y_n^1\}$. In brief, if it holds in $\mathbb{O}Tex$ that

$$\begin{aligned} T_1[i] \oplus T_1'[i] &= f_1(\mathbf{r}_2[i]), \\ T_2[j] \oplus T_2'[j] &= \mathscr{C}(\mathbf{r}_3[j]), \end{aligned} \tag{19}$$

where $f_1(\cdot)$ is the repetition code and $\mathscr{C}$ is the pseudorandom code; $\mathbb{O}Tex$ implies the functionality $\mathcal{F}_{\text{OPRF}}$ where the seed s consists of $(V_2, \mathbf{r}_2)$ generated by *Bob* and *Alice* gets $f_s(x)$. Meanwhile, this essentially completes the main construction of $\mathbb{O}Tex$-based PMT protocol. All needed to do next is that *Bob* sends $\{f_s(y_i^1) | i \in [n]\}$ to *Alice* who compares $f_s(x)$ with $\{f_s(y_i^1) | i \in [n]\}$ locally.

6.3. $\mathbb{O}Tex$-Based PSI Protocol. To obtain the final PSI protocol that computes $X \cap Y$, *Alice* simply invokes the PMT protocol for each $x_i \in X$. The protocol in Figure 12 computes $X \cap Y$ in an outsourced manner. The intuition is that

INPUT OF *Alice*: $\{x_1, \ldots, x_m\}$, $|x_i| = l$.

INPUT OF *Bob*: $\{y_1, \ldots, y_m\}$, $|y_i| = l$.

COMMON INPUT: a security parameter k, pseudorandom code C.

(i) Precomputing phase:

(1) For $i \in m$, *Alice* computes $r_3[i] = C(x_i)$.

(2) For $i, j \in m$, *Bob* computes $T_3^{(i)}[j] = C(y_i)$.

(ii) $\mathbb{O}$Tex phase:

(3) Aided by a sever $\mathbb{S}$, *Alice* and *Bob* execute $\mathbb{O}$Tex with inputs r_3 and $T_3^{(i)}$, respectively.

(4) For $j \in [m]$, *Bob* gets $f_{s_j}(\cdot)$ and *Alice* gets $f_{s_j}(x_i)$ from $\mathbb{O}$Tex.

(iii) Outputting phase:

(5) For each x_i and Y, *Bob* sends $\{f_{s_i}(y_j) \mid i, j \in [m]\}$ to *Alice* who outputs m-length string z, where
- $z_i = 1$, if there exists a j such that $f_{s_i}(x_i) = f_{s_i}(y_j)$,
- $z_i = 0$, otherwise.

(6) *Alice* outputs $\{x_i \mid i \in [m], z_i = 1\}$.

Figure 12: Outsourced private set intersection protocol.

Alice holds a set X, while *Bob* holds Y, and after PMT, *Alice* knows whether her element x_i belongs to Y or not, *i.e.,* the intersection of two sets.

Some details on method of preprocessing items in set are simply omitted in Figure 12. We can hash the item to bins and then operate the $\mathbb{O}Tex$-based PSI protocol on each bin separately. Specifically, we use Cuckoo hashing [35] for PSI in the following way. First, *Alice* and *Bob* agree on 3 random hash functions h_1, h_2, h_3: $\{0,1\}^* \longrightarrow [m']$ suitable for 3-way Cuckoo hashing. *Alice* places her items into m in either $h_1(x)$, $h_2(x)$, or $h_3(x)$, and each bin contains at most one item. *Bob* places each of his items y in locations $h_1(y)$, $h_2(y)$, and $h_3(y)$. At this point, *Alice* pads her input with dummy items so that each bin contains exactly 1 item. Note that when we use cuckoo hashing, then there will be some items which cannot be placed into the table and have to be moved to a stash, which does not matter because of the usage of stash-less cuckoo hashing [20]. Finally, *Alice* and *Bob* perform a PSI in each bin.

7. Conclusions and Future Work

In this paper, we proposed a generic outsourced OT extension protocol ($\mathbb{O}Tex$) which can outsource all the "base" OT from the sender to a semihonest server. The proposed protocol realizes optimal computational and communication complexity relative to security parameter. In addition, we showed that $\mathbb{O}Tex$ can be efficiently used in private membership test and private set intersection. In the future, we will consider malicious model and construct efficient outsourced OT extension protocols secure against malicious adversary.

Data Availability

The performance test data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61572294 and 61632020), Major Innovation Project of Science and Technology of Shandong Province (Grant no. 2018CXGC0702), Natural Science Foundation of Shandong Province (Grant no. ZR2017MF021), and Fundamental Research Funds of Shandong University (Grant no. 2017JC019).

References

[1] C.-C. Y. Andrew, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pp. 162–167, Washington, DC, USA, October 1986.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp. 307–328, Weizmann Institute of Science, Rehovot, Israel, 2019.

[3] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proceedings of the Annual Cryptology Conference*, pp. 643–662, Santa Barbara, CA, USA, August 2012.

[4] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Proceedings of the Annual International Cryptology Conference*, pp. 145–161, Santa Barbara, CA, USA, August 2003.

[5] V. Kolesnikov and R. Kumaresan, "Improved ot extension for transferring short secrets," in *Proceedings of the Annual Cryptology Conference*, pp. 54–70, Santa Barbara, CA, USA, 2013.

[6] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[7] G. Brassard, C. Claude, and J.-M. Robert, "All-or-nothing disclosure of secrets," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*, pp. 234–238, Santa Barbara, CA, USA, 1986.

[8] W.-G. Tzeng, "Efficient 1-out-n oblivious transfer schemes," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 159–171, Paris, France, February 2002.

[9] Y. Mu, J. Zhang, and V. Varadharajan, "m out of n oblivious transfer," in *Proceedings of the Australasian Conference on Information Security and Privacy*, pp. 395–405, Melbourne, Australia, 2002.

[10] C.-K. Chu and W.-G. Tzeng, "Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 172–183, New York, NY, USA, March 2005.

[11] Y. Lindell, K. Nissim, and C. Orlandi, "Hiding the input-size in secure two-party computation," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 421–440, Bengaluru, India, December 2013.

[12] C. Cho, N. Döttling, S. Garg, D. Gupta, P. Miao, and A. Polychroniadou, "Laconic oblivious transfer and its applications," in *Proceedings of the Annual International Cryptology Conference*, pp. 33–65, Santa Barbara, CA, USA, 2017.

[13] V. Kolesnikov, R. Kumaresan, M. Rosulek, and T. Ni, "Efficient batched oblivious PRF with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 818–829, Vienna, Austria, October 2016.

[14] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and T. Ni, "Practical multi-party private set intersection from symmetric-key techniques," *IACR Cryptology ePrint Archive*, vol. 799, p. 2017, 2017.

[15] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Transactions on Privacy and Security*, vol. 21, no. 2, p. 7, 2018.

[16] B. Pinkas, M. Rosulek, T. Ni, and A. Yanai, "Spot-light: lightweight private set intersection from sparse ot extension," in *Proceedings of the Annual International Cryptology Conference*, pp. 401–431, Santa Barbara, CA, USA, August 2019.

[17] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious PRF," in *Proceedings of the Annual International Cryptology Conference*, pp. 34–63, Santa Barbara, CA, USA, August 2020.

[18] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based psi via cuckoo hashing," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 125–157, Darmstadt, Germany, May 2018.

[19] M. Ciampi and C. Orlandi, "Combining private set-intersection with secure two-party computation," in *Proceedings of the International Conference on Security and Cryptography for Networks*, pp. 464–482, Amalfi, Italy, September 2018.

[20] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based psi with linear communication," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 122–153, Darmstadt, Germany, May 2019.

[21] T. Duong, D. H. Phan, and T. Ni, *Catalic: Delegated PSI Cardinality with Applications to Contact Tracing*, IACR Cryptology ePrint Archive, vol. 1105, p. 2020, 2020.

[22] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptology ePrint Archive*, vol. 187, p. 2005, 2005.

[23] A. Y. Lindell, "Efficient fully-simulatable oblivious transfer," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 52–70, San Francisco, CA, USA, April 2008.

[24] B. Zeng, C. Tartary, P. Xu, J. Jing, and X. Tang, "A Practical Framework for t-out-of-n oblivious transfer with security against covert adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 465–479, 2012.

[25] M. Green and S. Hohenberger, "Universally composable adaptive oblivious transfer," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 179–197, Melbourne, Australia, December 2008.

[26] D. Beaver, "Precomputing oblivious transfer," in *Proceedings of the Annual International Cryptology Conference*, pp. 97–109, Santa Barbara, CA, USA, August 1995.

[27] D. Beaver, "Correlated pseudorandomness and the complexity of private computations," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 479–488, Philadelphia, PA, USA, May 1996.

[28] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan, "The relationship between public key encryption and oblivious transfer," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 325–335, Redondo Beach, CA, USA, November 2000.

[29] Y. Gertner, T. Malkin, and O. Reingold, "On the impossibility of basing trapdoor functions on trapdoor predicates," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 126–135, Heraklion, Greece, 2001.

[30] H. Carter, B. Mood, P. Traynor, and K. Butler, "Secure outsourced garbled circuit evaluation for mobile devices," *Journal of Computer Security*, vol. 24, no. 2, pp. 137–180, 2016.

[31] D. Mansy and P. Rindal, "Endemic oblivious transfer," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 309–326, London, UK, November 2019.

[32] V. Kolesnikov, M. Rosulek, T. Ni, and X. Wang, "Scalable private set union from symmetric-key techniques," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 636–666, Kobe, Japan, December 2019.

[33] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, Cambridge, UK, 2009.

[34] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Proceedings of the Theory of Cryptography Conference*, pp. 303–324, Cambridge, MA, USA, February 2005.

[35] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8819958, 22 pages
https://doi.org/10.1155/2020/8819958

Research Article

Modelling the Mimic Defence Technology for Multimedia Cloud Servers

Feng Feng,[1] **Xiabing Zhou**,[2] **Bin Li**,[1] **and Qinglei Zhou**[1]

[1]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[2]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*

Correspondence should be addressed to Xiabing Zhou; zhouxiabing@suda.edu.cn and Bin Li; iebinli@zzu.edu.cn

Received 15 August 2020; Revised 27 September 2020; Accepted 26 October 2020; Published 28 November 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Feng Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A current research trend is to combine multimedia data with artificial intelligence and process them on cloud servers. In this context, ensuring the security of multimedia cloud servers is critical, and the cyber mimic defence (CMD) technology is a promising approach to this end. CMD, which is an innovative active defence technology developed in China, can be applied in many scenarios. However, although the mathematical model is a key component of CMD, a universally acceptable mathematical model for theoretical CMD has not been established yet. In this work, the attack problems and modelling difficulties were extensively examined, and a comprehensive modelling theory and concepts were clarified. By decoupling the model from the input and output of the specific system scene, the modelling difficulties were effectively avoided, and the mathematical expression of the CMD mechanism was enhanced. Furthermore, the process characteristics of the attack behaviour were identified by using a specific mathematical mapping method. Finally, based on the decomposition problem of large prime factors and convolution operations, an intuitive and exclusive CMD mathematical model was proposed. The proposed model could clearly express the CMD mechanism and transform the problems of attack and defence in the CMD domain into corresponding mathematical problems. These aspects were considered to qualitatively assess the CMD security, and it was noted that a high level of security can be realized. Furthermore, the overhead of CMD was analyzed. Moreover, the proposed model can be directly programmed.

1. Introduction

With the development of information technology, multimedia has been widely applied in the human society. Moreover, the emergence of artificial intelligence (AI) technologies has considerably enhanced the ability to analyse, process, and utilise multimedia. Because multimedia processing and AI technologies are computation- and storage-intensive, they are realized through the relatively mature distributed computing [1] and cloud computing [2] techniques. Cloud computing is based on server clusters. To enable the efficient processing of multimedia data, ensuring the security of multimedia cloud servers is essential. Multimedia cloud servers face nearly the same issues as those of cyberspace, such as network attacks and multimedia data protection. The multiple reports of network security incidents indicate that attackers mostly exploit the software and hardware vulnerabilities, and backdoor attacks constitute the majority of network security incidents. Nevertheless, at the current level of science and technology, the unknown vulnerabilities and backdoors and the associated security threats cannot be eliminated [3]. Traditional defence technology tends to rely on the attack technology as a priori knowledge, which involves hysteresis and passivity, and thus, the traditional approaches cannot block unknown vulnerabilities and backdoors. Therefore, the development of innovative network security theories and techniques has become a key research direction in the field of network security. Moving target defence (MTD) is a new type of active defence technology developed in the United States to achieve a considerable advantage in the field of network attack and defence [4]. At present, the research and application of this technology are mainly concentrated in developed countries, with the United States as the main hub.

Cyber mimic defence (CMD) is an innovative active defence technology independently developed in China, whose emergence can offset the unbalanced situation of the network attack and defence techniques [5]. The CMD mechanism facilitates its application in the multimedia cloud server cluster environment. The redundancy characteristics in the cluster environment conform with the CMD aspects. In particular, the computing power of the cluster environment can satisfy the performance overhead of the CMD to realize computationally intensive operations such as redundant encryption or decryption. Furthermore, the already mature virtual technology [6, 7] can provide support for the CMD and has been widely used in server cloud environments. In summary, the CMD technology can be applied to multimedia cloud servers. In this study, two cases were considered to enable the protection of multimedia cloud servers through CMD. One approach involves constructing a mimic multimedia cloud server architecture, and the other approach involves the use of the mimic encryption to protect the multimedia data. For the first approach, heterogeneous redundancy is achieved with the granularity at software or hardware on the multimedia cloud servers. It can be heterogeneous at the hardware level, operating system level, database level, server software level, background application level, and can also be heterogeneous at several levels at the same time, which will form a rich heterogeneous redundant server pool. Scheduling these heterogeneous servers through the CMD mechanism can greatly protect software and hardware on them. For the second approach, the main purpose is to protect multimedia data. The use of heterogeneous redundant encryption to protect multimedia data and the use of hash fingerprint comparison to detect and shield threats are in line with the principle of CMD and can greatly improve the security of multimedia data. China has actively promoted the research on the theory and technology of CMD, and a theoretical system of CMD has been established [8, 9] and is considerably different from the static and fixed traditional systems. In this context, it is necessary to clarify the technical aspects of the mechanism of the mimic defence at the theoretical level [10] and to establish an intuitive and exclusive mathematical model according to the CMD mechanism to formulate a mapping relationship with the mathematical aspects. In this manner, the mechanism and protection capabilities of CMD can be clarified, and the research on CMD can be further promoted. Nevertheless, a universally accepted mathematical model for CMD has not been established yet.

In this study, by extensively analysing the attack problems and modelling difficulties, a clear modelling concept was established. By decoupling the model from the specific system input and output scenes, a clear mathematical expression was formulated, while avoiding the modelling difficulties. Furthermore, the process characteristics of the attack were highlighted by establishing a specific mathematical mapping method. The excellent security capacities of CMD were demonstrated using the proposed model. Finally, an intuitive and exclusive mathematical model for CMD was established, which could express the CMD mechanism mathematically and transform the problems of the attack and defence game of the CMD into corresponding mathematical subproblems, thereby enabling the qualitative assessment of the CMD safety capacities.

This paper first presents the research background followed by the main research content. Section 2 introduces the CMD concepts and mathematical knowledge required for modelling. Section 3 describes the prerequisite knowledge for the modelling. Section 4 describes the modelling process and model mechanisms. Section 5 clarifies the mathematical problems of the attack and defence game of the CMD examined through a simulation experiment, analysis, and evaluation, describes the qualitative assessment of the CMD safety aspects, and analyzes the overhead of CMD. Section 6 presents the concluding remarks.

2. Background Knowledge

2.1. Cyber Mimic Defence. Cyber mimic defence [11] is a revolutionary defence technology of "game-changing" initiated by China. The development of CMD was inspired by the mimicry phenomenon and biological immune system in the biological world. The dynamic heterogeneous redundancy (DHR) architecture was used as the core architecture of the CMD. Finally, the CMD theory was formulated, with "structure determines security" as the core idea. CMD is a nonpoint type defence technology with a dynamic [12] and closed-loop mechanism. The high security and high robustness [13] in the core architecture (DHR) of CMD are endogenous and coexisting. The unknown vulnerabilities and backdoors cannot be easily exploited under the CMD framework. In contrast to the static and single characteristics, the uncertainty may induce the attacker's cognitive dilemma, thereby making it nearly impossible for the attacker to form an attack chain.

Here is a brief introduction to the principle of CMD [14]. For specific vulnerabilities or backdoors, defenders must first be able to identify them before they can accurately defend. However, the current level of technology cannot grasp all unknown vulnerabilities and backdoors in advance. When the same target function is implemented in heterogeneous forms, the probability of them having the same vulnerabilities or backdoors will be greatly reduced. CMD puts these heterogeneous redundant executors to work in parallel without communicating with each other. In other words, they have no cooperative relationship and do not know each other's existence. It is a very rare event that the same vulnerabilities or backdoors are existent and are triggered at the same time in heterogeneous executors. Therefore, if the unknown vulnerabilities or backdoors in the CMD system are triggered, the output results of heterogeneous executors will be inconsistent. At this time, CMD uses the "relatively correct" principle to not only detect abnormal situations and perceive threats but also locate abnormal executors based on certain strategies and then take corresponding measures against them. For the aforementioned parallel heterogeneous executors, CMD then introduces a dynamic scheduling mechanism to make the system dynamic.

Next, through the DHR architecture visually shows the principle of CMD. The DHR architecture is shown in Figure 1.

The DHR architecture introduces a dynamic scheduling mechanism and feedback control mechanism [15] based on the executor heterogeneous redundancy and multimode adjudication, respectively. The operating mechanism of the DHR architecture is as follows:

(1) A functionally equivalent heterogeneous executor pool is constructed for a target business function.

(2) Through the dynamic scheduling strategy, several "online" heterogeneous executors are selected from the functionally equivalent heterogeneous executor pool.

(3) When the system input arrives, it is distributed to each "online" heterogeneous executor through the input distributor to ensure that each executor can be executed separately without coordination and communication.

(4) The output vectors of all the "online" heterogeneous executors produce the final output result through the multimode adjudication strategy. At this time, if an abnormal "online" heterogeneous executor is "perceived," the feedback control mechanism is activated according to the multimode adjudication strategy.

(5) If the feedback control is activated, the abnormal "online" heterogeneous executors are replaced by "offline" executors through the feedback control strategy, and the subsequent processes such as self-cleaning, self-reconstruction, self-reorganisation, log recording, and log analysis are performed in the background.

The DHR architecture, as the core architecture of CMD, illustrates the premise of CMD. CMD can be applied to a target object having the form of "Input-Process-Output," and it is represented by the I [P] O model. In [11], the input distributor and output arbiter were termed as "mimic brackets" (MB), and the scope of protection limited by the MB was defined as the "mimic defence boundary" (MDB). The MDB is usually a heterogeneous execution environment with unknown vulnerabilities, backdoors, viruses, and Trojan horses.

In recent years, research pertaining to the theory and mechanism of CMD has progressed rapidly. At present, the mimic domain name server has been implemented online, and the principle prototypes of the mimic web server [16, 17] and mimic router [18, 19] have been developed. In addition, the CMD technology has been applied in several fields to ensure multimedia security, 5G security [20], SDN network security [21, 22], software diversification [23], mimic storage system realization, mimic encryption, and mimic cloud, mimic firewall, and mimic gateway realization.

2.2. Mathematical Knowledge Required for Modelling

2.2.1. Decomposition Problem of Large Prime Factors. The decomposition problem of large prime factors can be described as follows: if there exist several large prime factors, they can be easily multiplied to obtain a large composite number. However, it is extremely difficult to obtain these large prime factors by factorising the large composite number. This problem has been studied by mathematicians for hundreds of years, and no rapid algorithm is available to solve this problem. The research on the decomposition problem of large prime factors is challenging, but has theoretical and application value [24]. For example, the popular RSA [25] encryption algorithm relies on the decomposition problem of large prime factors.

In addition to the early violent trial division method, certain algorithms to solve the decomposition problem of large prime factors have been proposed by researchers, such as the ρ-method [26], P − 1 method [27], elliptic curve method [28], random square method, quadratic sieve method [29], and number field sieve method [30]. Among these algorithms, the number field sieve method is considered to be the best at present.

2.2.2. Convolution Operation. Convolution is a key operation in analytical mathematics and is performed as follows: first, two independent functions and the definition domain of their parameters are specified. Subsequently, the function values are calculated, and the corresponding function values are multiplied. Finally, all the products are summed. Specifically, convolution involves rolling a binary function into a univariate function in a process commonly known as "dimension reduction."

The convolution operation can be divided into continuous and discrete convolutions:

(1) To perform convolution, a certain relationship must exist among variables x, y, and n. Assume that the relation $x + y = n$ holds. For a particular n, this relation can represent a straight line with a slope of 1 in the Cartesian coordinate system.

(2) The variable τ is defined, and the range of τ is expressed as R. According to the aforementioned relation, $x = \tau$, and $y = n - \tau$.

(3) If the functions of x and y are $f(x)$ and $g(y)$, respectively, they can be written as $f(\tau)$ and $g(n - \tau)$, respectively.

In this case, the discrete and continuous convolution can be represented as in formulas (1) and (2), respectively:

$$(f * g)(n) = \sum_{\tau \in R} f(\tau) g(n - \tau), \tag{1}$$

$$(f * g)(n) = \int_{\tau \in R} f(\tau) g(n - \tau)\, \mathrm{d}\tau. \tag{2}$$

Here, $(f * g)(n)$ is termed as the convolution of functions f and g.

A convolution operation can reflect valuable physical meanings in an engineering system and can be used to calculate the output of such a system.

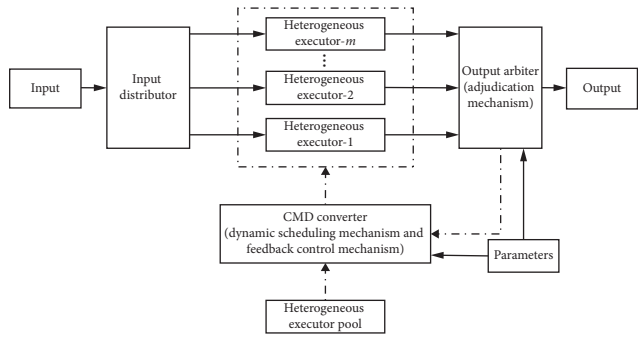


Figure 1: DHR diagram.

2.2.3. Martingale. The martingale concept originated from the mathematical description of the fair gambling process. Specifically, the martingale system is a concept in probability theory, which describes a special class of stochastic processes. Herein, the definitions and meanings of discrete and continuous martingale are presented.

(1) Discrete Martingale. If a discrete-time stochastic process X_n satisfies the following conditions,

(a) $E(|X_n|) < \infty$ and

(b) $E(X_n + 1 \mid X_1,\ldots,X_n) = X_n$,

the stochastic process X_n is a discrete martingale. Specifically, for the stochastic process X_n, if all the values at the present moment and all the previous moments are known, the conditional expectation value at the next moment is equal to the value of the present moment.

If the discrete-time stochastic processes X_n and Y_n satisfy the following conditions,

(a) $E(|Y_n|) < \infty$ and

(b) $E(Y_n + 1 \mid X_1,\ldots,X_n) = Y_n$,

the stochastic process Y_n is a discrete martingale on X_n. Specifically, for the stochastic processes X_n and Y_n, if all the values of the former process at the present moment and previous moments are known, the conditional expectation value of the latter process at the next moment is equal to the value of the latter process at the present moment.

(2) Continuous Martingale. If a continuous-time stochastic process X_t satisfies the following conditions,

(a) $E(|X_t|) < \infty$ and

(b) $E(X_t \mid \{X_m, m \le s\}) = X_s$,

the stochastic process X_t is a continuous martingale. Specifically, for the stochastic process X_t, if all the values up to time s are known, the conditional expectation value at time t $(t > s)$ is equal to the value at time s.

If the continuous-time stochastic processes X_t and Y_t satisfy the following conditions,

(a) $E(|Y_t|) < \infty$ and

(b) $E(Y_t \mid \{X_m,\ m \le s\}) = Y_s$,

the stochastic process Y_t is a continuous martingale on X_t. Specifically, for the stochastic processes X_t and Y_t, if all the values of the former process up to time s are known, the conditional expectation value of the latter process at time t $(t > s)$ is equal to the value of the latter process at time s.

3. Modelling Concept

3.1. Network Attack Analysis. The network attack technology, in combination with computer technology, is undergoing constant development. The attack behaviour has the characteristics of uncertainty, complexity, and diversity and is developing towards a large-scale, collaborative, and multilevel framework. Therefore, the research [31–34] and formal description of network attacks are of considerable significance to both sides. The classical network attack modelling methods involve the use of the attack tree [35], attack graph [36], and attack network [37]. In addition, the attack surface (AS) [38, 39] and mobile attack surface (MAS) [40, 41] theories have emerged in recent years to analyse and examine the law of network attack behaviour.

3.1.1. Attack on the Traditional Static and Single System. The following is described in the context of the traditional static and single system.

Although there are differences in the description of the network attack process by the above methods or theories, in conclusion, a successful network attack is a process comprising several stages, and there may be repeated backtracking subprocesses, as shown in Figure 2.

To summarize, a successful network attack can be simply described as a critical path from the beginning of the attack to the success of the attack based on each stage. In this paper, this critical path is referred to as the successful attack vector (SAV). By further analysing the SAV, the following two characteristics can be defined:

(1) *Target Characteristics.* The SAV has target characteristics similar to a vector, and the target and direction from the beginning to the end point are clear and unique. Consequently, a successful network attack can be easily described as a chain because of the targeting characteristics of the SAV.

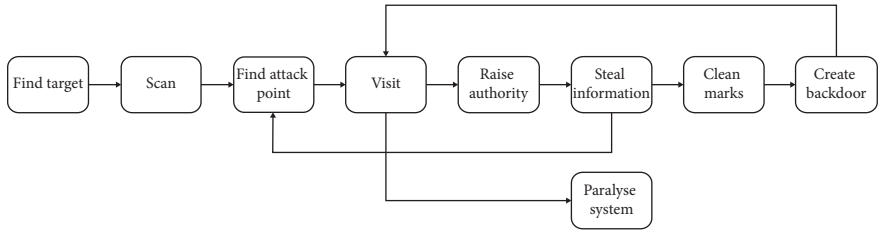


Figure 2: Example of the attack process.

(2) *Process Characteristics.* The SAV can clearly reflect the process characteristics of the network attack. Specifically, the negative influence of a successful attack follows the attack behaviour. Because the final result is not immediately visible, it is latent to the attacked object. Thus, the attacker must move sequentially toward the final goal with considerable patience, which leads to a typical "delayed satisfaction."

3.1.2. Attack on the CMD System. Because the technology of the CMD is not widespread at present, the data of only a few typical attack cases are available. Nevertheless, it is possible to analyse and estimate the attacks against the CMD system.

This paper emphasises that the premise of the CMD is to realize scenarios that satisfy the I [P] O model, which is critical to activate the defence effect of the CMD. In this paper, we divide the attacks against the CMD system into type-ρ and type-φ attacks.

(1) Type-ρ Attack. A type-ρ attack is a kind of attack that deviates from the premise of CMD. The CMD cannot always resist this kind of attack, but it can be applied to supplement the existing security defence technology or means [42]. In this paper, certain simple examples are presented to illustrate the type-ρ attacks:

(a) The attacker successfully intrudes an "online" heterogeneous execution of a certain CMD system and deletes an important file in it. The successful attack does not produce any output that must pass through the CMD system arbiter.

(b) The attacker successfully intrudes an "online" heterogeneous execution of a certain CMD system, cracks the key of an important encrypted file on it, and browses the valuable data in the file. Moreover, the successful attack does not produce any output that must pass through the CMD system arbiter.

(c) A certain CMD system is subjected to a DDoS attack.

(2) Type-φ Attack. In contrast to a type-ρ attack, a type-φ attack is a kind of attack that satisfies the premise of the CMD. In this case, after the attacker successfully attacks the "online" heterogeneous executor in the CMD system, an output that must pass through the CMD system arbiter is produced. This output is the desired attack result for the attacker and the so-called abnormal output for the CMD system. From the perspective of the CMD system, inconsistencies among the abnormal outputs may exist in theory. However, in terms of the rationality, because the CMD adjudication mechanism is implemented, for a type-φ attack to succeed, a "joint escape" in which a consistent abnormal output is generated must be realized, and it is meaningless for the attackers to create an inconsistent abnormal output. Therefore, all the abnormal outputs mentioned in this paper are consistent, which negates the problem of the type-φ attack on the CMD system and facilitates follow-up research. Considering this background, a successful and failed type-φ attack is that in which a "joint escape" can and cannot be realized, respectively.

Considering these aspects and the categorization of the attacks against the CMD system, it can be noted that only the φ-attack problem has practical research significance for both sides of the CMD attack and defence.

3.2. Problem Description. A mathematical model is key to support the continuous improvement of a theoretical system and to promote the related research work. In this context, developing a mathematical model can help describe and analyse the related mechanism of the CMD.

At present, two kinds of CMD modelling are implemented, specifically, modelling the overall mechanism of the CMD and modelling only part of the mechanism of the CMD [43, 44]. This paper focuses on the former approach, and thus, the following problems as well as the existing research results correspond to this approach. Until now, a universally accepted mathematical model for the theory of CMD has not been established, and the mainstream modelling method is mainly based on the "Markov chain." [45, 46]

(1) Realistic mapping of the overall mechanism modelling of CMD:

This problem is defined according to the aforementioned SAV characteristics.

(a) In contrast from that in the traditional static and single system, the target protection object in the CMD system is heterogeneously redundant. The heterogeneous redundant executors are completely independent and do not communicate with one another. If an attacker wants to successfully realize the φ-attack, they must complete several independent SAVs simultaneously, and this requirement is different from that for attacking a traditional static and single system. However, because several independent SAVs are

present, the relevant stages may not be completely consistent or have a one-to-one correspondence. In addition, backtracking and repeated subprocesses may be present in the attack process. Therefore, although the target characteristics of a single SAV facilitate the description of a successful network attack in the chain mode, the description of several independent SAVs in the chain mode simultaneously is complex and challenging. In this case, a single chain node must represent the stages or states from several independent SAVs simultaneously. If these nodes are combined as in the actual situation, the number of chain nodes required may be considerably higher than that in a "traditional chain," as discussed in the subsequent sections. This type of chain structure is easy to network; however, the target characteristic of each independent SAV becomes difficult to identify, which makes the analysis and research highly challenging.

(b) In the core architecture (DHR) of the CMD, one input corresponds to one output. If a type-φ attack fails and an abnormal output is present, it will be detected in the adjudication stage of the DHR. As mentioned previously, an SAV has a process characteristic, and when an attack is conducted, an abnormal output is not necessarily produced. In other words, the attack cannot be represented by only one input-output pair. At present, the DHR architecture cannot directly reflect this process characteristic. Therefore, it is necessary to model the overall mechanism of CMD with a "buffer" that can reflect the delay in the process characteristic. As a simple and intuitive example, we consider the case of uploading files on a server. In the absence of any other security protection measures, three steps can be defined: request, upload, and access. In malicious attacks, the three steps may be as follows: request, upload the virus, and access and activate the virus. Each of these three steps corresponds to a system input. The attack behaviour can be considered to start in Step 1 or 2, although the negative impact of the successful attack occurs in Step 3.

(2) Problem of modelling the overall mechanism of CMD based on the Markov chain:

(a) A Markov chain is a state chain that describes a process of state dynamic transfer. At present, the Markov chain is applied to the CMD theory to describe the process of an attack and defence game. Therefore, the Markov chain modelling is focused on describing the transfer path of the attack and defence game state, owing to which the CMD mechanism appears to be highly abstract. This phenomenon occurs because the state transfer path occupies the main body, and only the state nodes in the chain are used to represent the state of the whole CMD system at a certain time. The CMD mechanism cannot be intuitively reflected in this scenario, which hinders the mechanism description and analysis.

(b) The Markov chain is a generic modelling tool and not specifically applied to CMD modelling. To apply the Markov chain to model the overall mechanism of the CMD, a custom set of state parameters must be used to describe the state of the CMD system. The complexity of this set of custom state parameters directly affects the complexity of the Markov Chain, and the complexity of the real scene directly affects the complexity of the custom state parameter set. Owing to this strong binding with the real scene, the CMD Markov chain can only be used to model simple system scenes, for instance, to model the type of system input that would lead to a certain type of system state and system output.

Considering these problems, in this work, the modelling method based on the Markov chain was not employed, and the objective was to develop an intuitive and exclusive mathematical model for the CMD.

3.3. Core Concept. From the mathematical viewpoint, this paper performs a formal analysis of the heterogeneous redundant executors of the CMD framework:

(1) For a given CMD system, the heterogeneous executor pool can be represented by the mathematical set E, where $E = \{e_1, e_2, e_3, \ldots, e_n\}$ with e_i $(1 \le i \le n)$, and the set elements represent the heterogeneous executors. The set of "online" heterogeneous executors is represented as the mathematical set E_O, where $E_O = \{e_{o1}, e_{o2}, e_{o3}, \ldots, e_{om}\}$ with e_{oj} $(1 \le j \le m)$, and the set elements represent the "online" heterogeneous executors. The set E_O is a subset of E.

(2) In the CMD mechanism, the set of "online" heterogeneous executors is modified according to the strategy. Therefore, the mechanism can be formalised as the process of selecting a series of E_O on set E in "1" according to the strategy, which represents a mathematical process of selecting combinations.

(3) A combination in "2" is termed as a "sample," represented by S. The sample space comprising all possible S values is represented as S_S. S_S is similar to a sample warehouse, termed as the "mimic warehouse" in this paper. Therefore, the essence of the CMD mechanism is to schedule and operate the "samples" on the so-called "mimic warehouse."

(4) The elements of sample S in "3" are formalised as large prime factors, represented as f_j $(1 \le j \le m)$, where m represents the number of large prime factors corresponding to the number of elements of combination S and is distinguished by label j.

Subsequently, the sample S can be formalised as the product of these large prime factors, represented as the following formula:

$$S = \prod_{j=1}^{m} f_j, \quad 1 \le j \le m. \tag{3}$$

The objective of this formal analysis process is to associate the "online" heterogeneous executor set of the CMD system at a certain time with a product of the large prime factors. Analysing the product of the large prime factors corresponds to the analysis of the corresponding "online" heterogeneous executor set.

4. Large-Number Convolutional Mimic Defence Mathematical Model

The proposed mathematical model of the CMD is based on the convolution operation and depends on the decomposition problem of large prime factors. The model is termed as the large-number convolutional mimic defence (LNCMD). The LNCMD model is an intuitive and exclusive mathematical model of CMD. This section describes the modelling process and mechanism of the LNCMD.

4.1. Modelling

4.1.1. Assumptions

(1) The LNCMD model has the same black box characteristics as the CMD system.

(2) Based on the relationship described in the "core concept" section, the input and output of the LNCMD model are not the input and output of the information system in the actual sense, respectively. Therefore, the LNCMD model does not involve any specific input and output values of the actual CMD system. The "problem description" section highlighted the problems faced by the CMD Markov chain owing to its feature of strong binding with the real scene. The aforementioned configuration helps the LNCMD model avoid these problems.

(3) Each strategy in the CMD system, namely, the dynamic scheduling strategy, adjudication strategy, and feedback control strategy, is configured by the defender.

(4) The attack considered in the LNCMD model is a type-φ attack. The abnormal output, successful type-φ attack, and failed type-φ attack are defined according to the aforementioned descriptions.

(5) All the heterogeneous executors in the CMD system have a comprehensive evaluation value. A larger evaluation value corresponds to a more safe executor [47]. It is considered that the comprehensive evaluation value is a large prime number. It is assumed that the development tasks of all the heterogeneous executors in a CMD system are not completed by the defender, and the comprehensive evaluation values of all the heterogeneous executors are known.

4.1.2. Model Framework.

Based on the model components, the LNCMD model logically divides the CMD system into three layers, namely, the extraction, convolution, and judgement layers. The mechanism of the LNCMD model corresponds to the cooperation of these three layers. The logic of the extraction, convolution, and judgement layers is represented by a mathematical function, and the corresponding functions are termed as extraction, convolutional, and judgement functions, respectively.

4.1.3. Model Components and Rules

(1) Private components:

According to the assumptions, the LNCMD model exhibits black box features to the external environment. The private components are visible only inside the LNCMD model and can be read and written only inside the LNCMD model.

(a) Large prime factor pool: this pool stores the comprehensive evaluation values of all the heterogeneous executors in the CMD system. In a practical sense, this pool represents the heterogeneous executor pool of the CMD system.

(b) Convolution kernel vector: this vector stores the comprehensive evaluation values of the "online" heterogeneous executors. The length of the vector is equal to the number n of the "online" heterogeneous executors, and the i^{th} element on the vector corresponds to "online" heterogeneous executor i $(1 \le i \le n)$.

(c) Product variable: this variable stores the product of all the elements of the convolution kernel vector. Although the variable is in the form of a product, the order of each factor of the product strictly follows the order of the original element position on the convolution kernel vector.

(d) Hidden matrix: this matrix is an abstract matrix with the dimensions $n \times m$, where n is the number of "online" heterogeneous executors, and row i corresponds to "online" heterogeneous executor i $(1 \le i \le n)$. Here, m is an uncertain value when a column of the hidden matrix corresponds to a model input, and it increases dynamically with the number of inputs. The element values of the hidden matrix are generated through the model input as the excitation. The j^{th} excitation can only generate the elements of column j, and the elements after column j are not visible. The value range of the hidden matrix elements is [0, 1], which represents the attack progress of 0–100%. In the LNCMD model, this range represents the progress of the attacker decomposing the specified large prime factor from the value of the product variable. In terms

of the actual meaning, this range represents the amount of the SAV covered by the attacker attacking a specific "online" heterogeneous executor.

(e) Layer signal: the values can be 0, 1, or 2, which correspond to the operation of the extraction, convolutional, and judgement functions, respectively. When the layer signal changes from one value to another, it is considered that the extraction, convolutional, and judgement functions are strictly mutually exclusive even if one function is operating, that is, only one layer is allowed to operate at a certain time. The extraction, convolutional, and judgement functions can influence the signal actively. However, due to the dynamic scheduling strategy, the signal may be set passively through to dynamic scheduling. For example, the online time of the "online" executor may reach the limit. The logical turbulence caused by the passive setting above that of the LNCMD model depends on the rationality of the dynamic scheduling strategy and is not related to the LNCMD model.

(2) Nonprivate components:

Nonprivate components are those components of the LNCMD model that communicate with the outside.

(a) *Model Input*. This input is generated by the customer. In terms of the influence, the external environment influences the LNCMD model through the input of the convolution layer function. In terms of the attack and defence game, the attacker tries to decompose the large prime factor from the product as the game action.

(b) *System Strategies*. These strategies are configured by the defender. In terms of the influence, the system strategies, as the system level configuration, play a key role in the LNCMD model, and they are not used as the input of any layer function of the LNCMD model. In terms of the attack and defence game, the defender adopts the game actions by changing various system strategies.

(c) *Model Output*. This output is used to indicate whether the LNCMD model has been attacked according to the adjudication strategy. The output is a Boolean type value, with true-1 and false-0 indicating an attacked and not attacked state, respectively.

4.1.4. Model Symbol Set. The LNCMD model comprises tuples as follows: LNCMD = {*C*, eL, cL, jL}.

(1) *C* represents the components and is denoted as *C* = {Pri, Pub}, where Pri and Pub represent the private components and nonprivate components, respectively.

(a) Pri = {pool, vector, product, matrix, signal}
Here, "pool" is the large prime factor pool, "vector" is the convolution kernel vector, "product" is the product variable, "matrix" is the hidden matrix, and "signal" is the layer signal.

(b) Pub = {input, strategy, output}
Here, "input" is the model input, "strategy" represents the system strategies, and "output" is the model output.

(2) eL represents the extraction layer, with the logic corresponding to the extraction function.

(3) cL represents the convolution layer, with the logic corresponding to the convolutional function.

(4) jL represents the judgement layer, with the logic corresponding to the judgement function.

The LNCMD model is shown in Figure 3.

4.2. Mechanism

4.2.1. Extraction Layer. The logic of the extraction function in the extraction layer is to refresh the convolution kernel vector, set the product variable, refresh the hidden matrix according to the scheduling strategies on the heterogeneous executors—including the dynamic scheduling strategy and feedback control strategy—and, finally, set the layer signal. The extraction function is described in Table 1.

The detailed functions are as follows:

(1) Refresh the convolution kernel vector:

The extraction function first determines the "online" heterogeneous executors 1,. . .,*n* according to the scheduling strategies of the heterogeneous executors. Subsequently, the extraction function extracts the comprehensive evaluation values of these "online" heterogeneous executors from the large prime factor pool. $F_1, \ldots, F_n$ are the comprehensive evaluation values of the extracted heterogeneous executors, where F_i corresponds to "online" heterogeneous executor *i*. Finally, the extraction function places $F_{1,}. \ldots, F_n$ into the convolution kernel vector, where F_i is placed into the position of the i^{th} element of the convolution kernel vector.

The redundancy scale of three is considered as an example, as shown in Figure 4.

(2) Set the product variable:

Calculate the product of all the elements of the convolution kernel vector, and place this product into the product variable.

(3) Refresh the hidden matrix:

Considering the scheduling strategies of the heterogeneous executors, the "online" heterogeneous executors $1, \ldots, n$ are redetermined. Therefore, the extraction function must refresh the hidden matrix at this time by reinitialising the hidden matrix in an abstract sense.

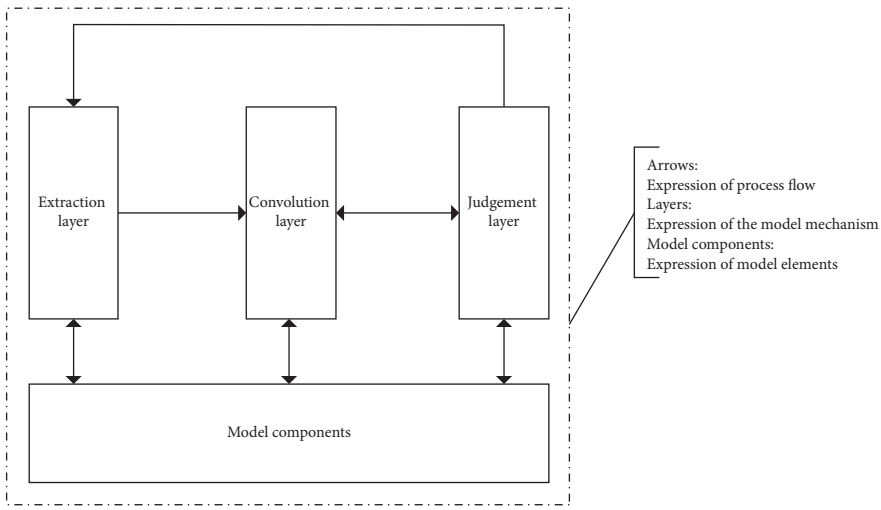


Figure 3: LNCMD diagram.

Table 1: Explanation of the extraction function.

Category	Explanation
Function input	Not applicable
Function output	Not applicable
Process summary	The extraction function operates internally on the private components of the LNCMD model
Function shape	Pseudo-code shape: void extract ();

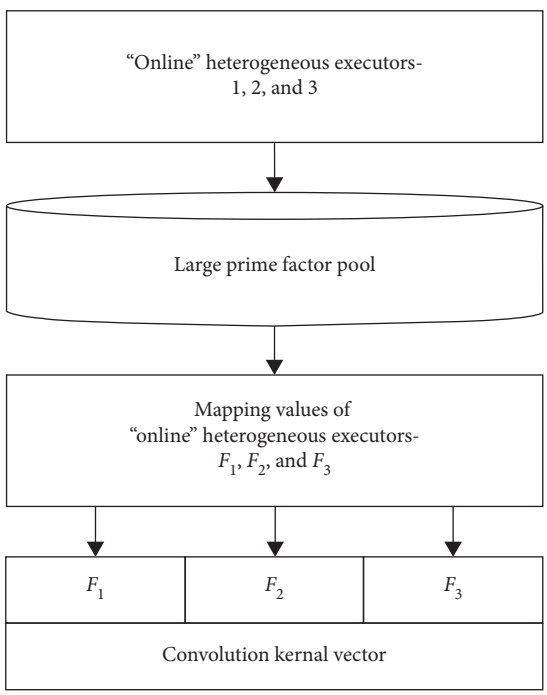


Figure 4: Refresh the convolution kernel vector.

Considering the redundancy scale of three as an example, the action of refreshing the hidden matrix is illustrated in Figure 5.

The significance of refreshing the hidden matrix is as follows: because the combination of the "online" heterogeneous executors changes, the statistical number of the model input times should be recounted from zero; consequently, the exploration or expansion of the hidden matrix columns excited by the model inputs should also be restarted. In a logical sense, this action represents a milestone start.

The overall significance of the two actions of refreshing the convolution kernel vector and hidden matrix is to reflect the scheduling mechanism of the CMD for "online" heterogeneous executors, and this mechanism is reflected by these two actions on the LNCMD model.

(4) Set the layer signal:

The extraction function sets the layer signal as 1, indicating the commencement of the convolutional function operation.

4.2.2. Convolution Layer. The logic of the convolutional function in the convolution layer is to wait for the j^{th} model input, which excites the convolutional function to produce the j^{th} convolution output. The convolutional function is described in Table 2.

The detailed functions are as follows:

(1) Convolution operation:

The convolution operation of the convolutional function is performed between the vector and the matrix, and thus, it is a type of discrete convolution. The discrete convolution formula of the convolutional function is as follows:

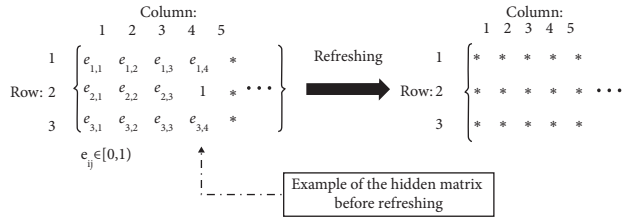


Figure 5: Refresh the hidden matrix.

Table 2: Explanation of the convolutional function.

Category	Explanation
Function input	j^{th} model input
Function output	Arithmetic formula of the j^{th} convolution result
Process summary	The output of the convolutional function is generated by a special convolution operation with the parameter input as the excitation
Function shape	Pseudo-code shape: outputType convolutional (inputType inputJ)

$$\text{convolutional}(\text{input } J) = (f^* g)(\text{input } J) = \sum_{i=1}^{n} f(i) g(\text{try}F_i), \quad n \geq 1. \tag{4}$$

Assuming that the value of the product variable is P, the formula can be explained as follows:

(a) Parameters:

There exist three parameters, i, $tryF_i$, and $inputJ$. Parameter i is the label of the i^{th} "online" heterogeneous executor. The parameter $tryF_i$ is abstract. In an arithmetic sense, this parameter is a trial value of F_i, and F_i is the i^{th} large prime factor of P. In a logical sense, this parameter represents an attempt of the attacker to attack the i^{th} "online" heterogeneous executor. The parameter $inputJ$ is the j^{th} model input.

The aforementioned three parameters satisfy the following relationship: $i + tryF_i = inputJ$. According to this relationship, in the arithmetic sense, $inputJ$ corresponds to a set of points $(i, tryF_i)$ on the "logical straight line" $(i + tryF_i = inputJ)$. The two functions $f(i)$ and $g(tryF_i)$ are combined into one convolutional function ($inputJ$) along the direction of the "logical straight line" $(i + tryF_i = inputJ)$. In a logical sense, this parameter indicates that a model input corresponds to a set of the attackers' attempts to attack each large prime factor (ordered) in P. In a practical sense, this parameter is the input distribution mechanism of the CMD.

(b) Range:

The value range of Σ summation is 1 to n, where n is the number of current "online" heterogeneous executors. This value is the length of the convolution kernel vector, which is the number of large prime factors of P, equal to the number of rows of the hidden matrix.

(c) Logic of subfunctions:

After the $f(i)$ function obtains parameter i, it searches for the i-th element F_i in the convolution kernel vector, and the function returns the reciprocal of F_i.

After the $g(tryF_i)$ function obtains the parameter $tryF_i$, the element value e of the corresponding position of the hidden matrix is calculated, and the return value of the $g(tryF_i)$ function is $1 - e$. The $g(tryF_i)$ function calculates e as follows:

The $g(tryF_i)$ function which calculates e requires a kind of carrier function, which needs to satisfy the following properties:

(1) The function is a one-variable continuous function, which can be expressed as $y = f(x)$
(2) In the domain $(0, +\infty)$, the function values are always greater than 0 and have a unique absolute maximum value
(3) Assuming that the function value of $f(x)$ at x_0 is the unique absolute maximum value, then x_0 should be a large prime number

This paper gives a typical example, selecting the Gaussian distribution probability density function as the carrier function of the $g(tryF_i)$ function. The characteristic of the Gaussian distribution probability density function is that a value closer to the

mathematical expectation (average) μ corresponds to a greater probability density. The $g(tryF_i)$ function uses this property for the mapping; therefore, the closer $tryF_i$ is to F_i, the closer the value e is to 1, and the attack progress is closer to 100%.

The formula of the Gaussian distribution probability density function is as follows:

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad -\infty < x < +\infty. \tag{5}$$

The formula of the Gaussian distribution probability density function in the $g(tryF_i)$ function is as follows (5):

$$\begin{cases} \dfrac{1}{\sqrt{2\pi}}\exp\left(-\dfrac{(tryF_i - F_i)^2}{2}\right), & 0 < tryF_i < +\infty, \\ 0, & tryF_i = 0, \\ 0, & tryF_i = +\infty. \end{cases} \tag{6}$$

In formula (5), when $tryF_i$ is equal to F_i, the calculated value is $(2\pi)^{-0.5}$. Finally, the formula to calculate the value of e in the $g(tryF_i)$ function is as follows:

$$\frac{1/\sqrt{2\pi}\exp\left(-(tryF_i - F_i)^2/2\right)}{\left(2\pi^{-0.5}\right)}. \tag{7}$$

(d) Calculation result:

The result of the convolution operation of the convolutional function corresponds to the multiplication of the return values of the $f(i)$ and $g(tryF_i)$ functions. Subsequently, a summation formula in the range $i = 1$ to n is defined, which is used to determine the output of the convolutional function as

$$\sum_{i=1}^{n}\frac{1-e_{ij}}{F_i}. \tag{8}$$

(2) Set the layer signal:

The convolutional function sets the layer signal as 2, indicating the operation of the judgement function.

4.2.3. Judgement Layer. The logic of the judgement function in the judgement layer is to receive the output of the convolutional function, perform an adjudication according to the adjudication strategy, and generate the model output. The judgement function is described in Table 3.

The detailed functions are as follows:

Considering the redundancy scale of three as an example, the logic of the judge function can be described as follows:

(1) Generic Pretreatment. The generic preprocessing consists of four steps. Let the value of the product variable be P.

Step 1: define P to formulate a score with a real value of 1 as follows:

$$\frac{P}{P} = \frac{F_1 \times F_2 \times F_3}{F_1 \times F_2 \times F_3} = 1. \tag{9}$$

Step 2: set the function input as convolutionJ. Multiply convolutionJ by the fraction in Step 1 to merge it into a single fractional form. In the arithmetic sense, this step involves merging all the subfractions in convolutionJ with P as the denominator as follows:

$$\begin{aligned} &\text{convolution } J\colon \frac{1-e_{1j}}{F_1} + \frac{1-e_{2j}}{F_2} + \frac{1-e_{3j}}{F_3} \Rightarrow \\ &\text{convolution } J \times \frac{P}{P} \Rightarrow \frac{\left(1-e_{1j}\right)\times F_2 \times F_3 + \left(1-e_{2j}\right)\times F_1 \times F_3 + \left(1-e_{3j}\right)\times F_1 \times F_2}{F_1 \times F_2 \times F_3}. \end{aligned} \tag{10}$$

Table 3: Explanation of the judgement function.

Category	Explanation
Function input	Output of the convolutional function
Function output	Output of the LNCMD model
Process summary	The process of generating a function output with the parameter input in the judgement function is highly dependent on the adjudication strategy
Function shape	Pseudo-code shape: bool judge (inputType convolutionJ)

Step 3: for the numerator of the result fraction in Step 2, extract the common factor based on the factor in the denominator. The following example illustrates the successful extraction of the common factor:

$$\begin{aligned}\text{convolution } J\text{: } \frac{1-e_{1j}}{F_1}+\frac{1-e_{3j}}{F_3} &\Longrightarrow \text{convolution } J\\ &\times\frac{P}{P}\Longrightarrow\frac{\left(1-e_{1j}\right)\times F_2\times F_3+\left(1-e_{3j}\right)\times F_1\times F_2}{F_1\times F_2\times F_3}\\ &\Longrightarrow\frac{F_2\times\left(\left(1-e_{1j}\right)\times F_3+\left(1-e_{3j}\right)\times F_1\right)}{F_1\times F_2\times F_3}.\end{aligned} \tag{11}$$

In the arithmetic sense, the extraction of certain common factors indicates that, in the convolution process of the convolutional function, there exist subfractions of the following form, owing to which certain F_i are extracted as common factors:

$$f(i)g\left(tryF_i\right)=\frac{1}{F_i}\times 0. \tag{12}$$

In other words, certain outputs of the $g(tryF_i)$ function are 0. In the logical sense, the extracted common factor directly corresponds to the large prime factor decomposed by the attacker from P. In the practical sense, the extracted common factor reflects the "online" heterogeneous executor whose SAV is covered by the attacker.

Step 4: define the result formula in Step 3 as the "result."

(2) Process Highly Dependent on the Adjudication Strategy. Because this process can affect the logical flow direction of the LNCMD model according to different adjudication strategies, this process is highly dependent on the adjudication strategy. We assume that the adjudication strategy is the majority voting strategy, described as follows:

(1) Output: the most consistent results are considered as the final result.

(2) Adjudication: when the results are not completely consistent, the system is considered to be attacked, and feedback control is launched. At this time, the information of the executors whose outputs are inconsistent with the final result is sent to the feedback control strategy.

The process strongly dependent on the adjudication strategy includes three sequential subprocesses. The sequence is $a\rightarrow b\&c$, in which the b and c subprocesses can be performed simultaneously.

(a) Judgement process:
According to the adjudication strategy, the attack status of a system can be determined. The judgement process is shown in Figure 6.
When the logic passes through the left branch, the number of common factors extracted in the "result" is 0 or 3, which represents the situation in which the output vectors of the "online" heterogeneous executors are completely consistent. In particular, when the number of common factors extracted in the "result" is 3, the situation is an extreme one in which the attacker has covered the SAVs of all the "online" heterogeneous executors. When the logic passes through the right branch, the number of common factors extracted in the "result" is neither 0 nor 3, which represents the inconsistency of the output vectors of the "online" heterogeneous executors. Furthermore, in this case, two situations may occur, corresponding to the abnormal output vectors being in the majority or minority.
Let the judgement result of this subprocess be "judge."

(b) Output process:
The judgement result "judge" of subprocess a is considered as the output of the model.

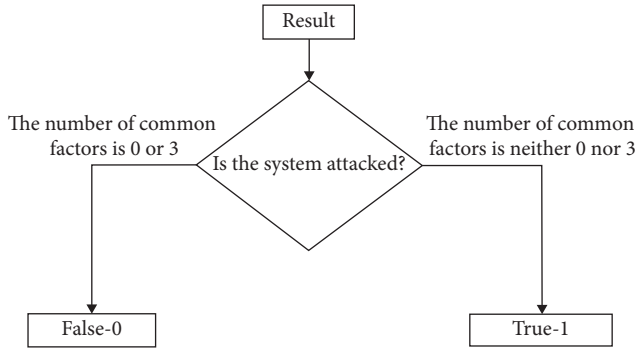


Figure 6: Judgement process flow.

(c) Process of setting the layer signal:
The layer signal settings are defined according to the adjudication strategy. The judge function sets the layer signal to 0, indicating the operation commencement of the extraction function. The judge function sets the layer signal to 1, indicating the operation commencement of the convolutional function. The practical significance is the determination of whether the feedback control must be started, according to the adjudication strategy. This subprocess can be regarded as a snapshot for the partial action of the DHR architecture, which pertains to the launch of the feedback control, as shown in Figure 7.

The process strongly dependent on the adjudication strategy ($a \rightarrow b\&c$) is illustrated in Figure 8.

Because the processing of the judgement function in the judge layer depends strongly on the adjudication strategy, the following description holds.

The aforementioned modelling helped define the role and rules for each strategy in the CMD system in the LNCMD model. The LNCMD model does not directly reflect the logic encapsulated by each strategy. As shown in the figure, the logical process in the dashed box is actually the action of the LNCMD model. Moreover, although the left branch of the "result" variable is the logic encapsulated by the adjudication strategy, it does not belong to the process of the LNCMD model.

4.3. Summary. The foregoing content establishes the LNCMD model and introduces the composition of the model and the mechanism of the model in detail. For the mathematical knowledge introduced in Section 2, it has been integrated into the LNCMD model. The decomposition problem of large prime factors is regarded as a core mathematical problem throughout the entire model mechanism. How to solve this problem is imposed on the attacker, and the defender creates this problem through the LNCMD model. Both sides carry out game behaviour around this mathematical problem. Convolution operation supports the mechanism of the entire convolution layer, and it is an important bridge to carry out the game behaviour around the decomposition problem of large prime factors. Martingale is a special stochastic process. When the LNCMD model is not started to run, the martingale cannot be reflected in it. When the LNCMD model is started to run, it has actual procedural properties. At this time, the martingale can be used to evaluate the safety status of the LNCMD model. This will appear and be described in detail in Section 5.

5. Simulation Experiment and Evaluation

5.1. Simulation Experiment

5.1.1. Simulation Environment Design

(1) Physical Background. Considering the assumed web service programme as the physical background for the simulation experiment and to apply the CMD technology, the redundancy scale of the "online" heterogeneous service programmes was set as three. The DHR architecture to construct this physical background is shown in Figure 9.

(2) LNCMD Model Population. In the modelling, a symbol set for the LNCMD model was established but not populated. This symbol set of the LNCMD model was populated according to the physical background. The specific population details for the LNCMD model are presented in Table 4. The populated LNCMD model is termed as lncmdDemo in this paper.

(3) Experiment Configuration. In the simulation experiment, a feature string backdoor is defined. Let this feature string backdoor be "door" and the corresponding feature string be "flag." In the simulation experiment, it is assumed that the only way for the attacker to complete the SAV is to trigger the "door" by trying the correct "flag." In lncmdDemo, the "flag" that can trigger the "door" is the large prime factors (ordered) of P.

The simulation experiment is performed under 5 configurations. In the simulation experiment, a simulation run with lncmdDemo is performed under each configuration. The 5 configurations are as follows:

Configuration 1: all the three "online" heterogeneous executors have no door, that is, $a\&b\&c = +\infty$

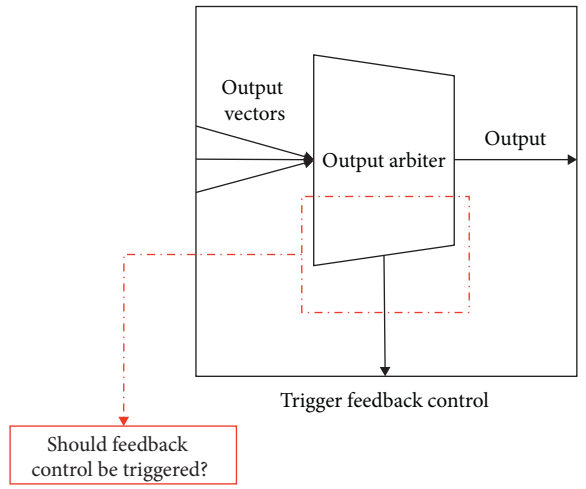


Figure 7: Local snapshot to determine whether the DHR starts the feedback control.

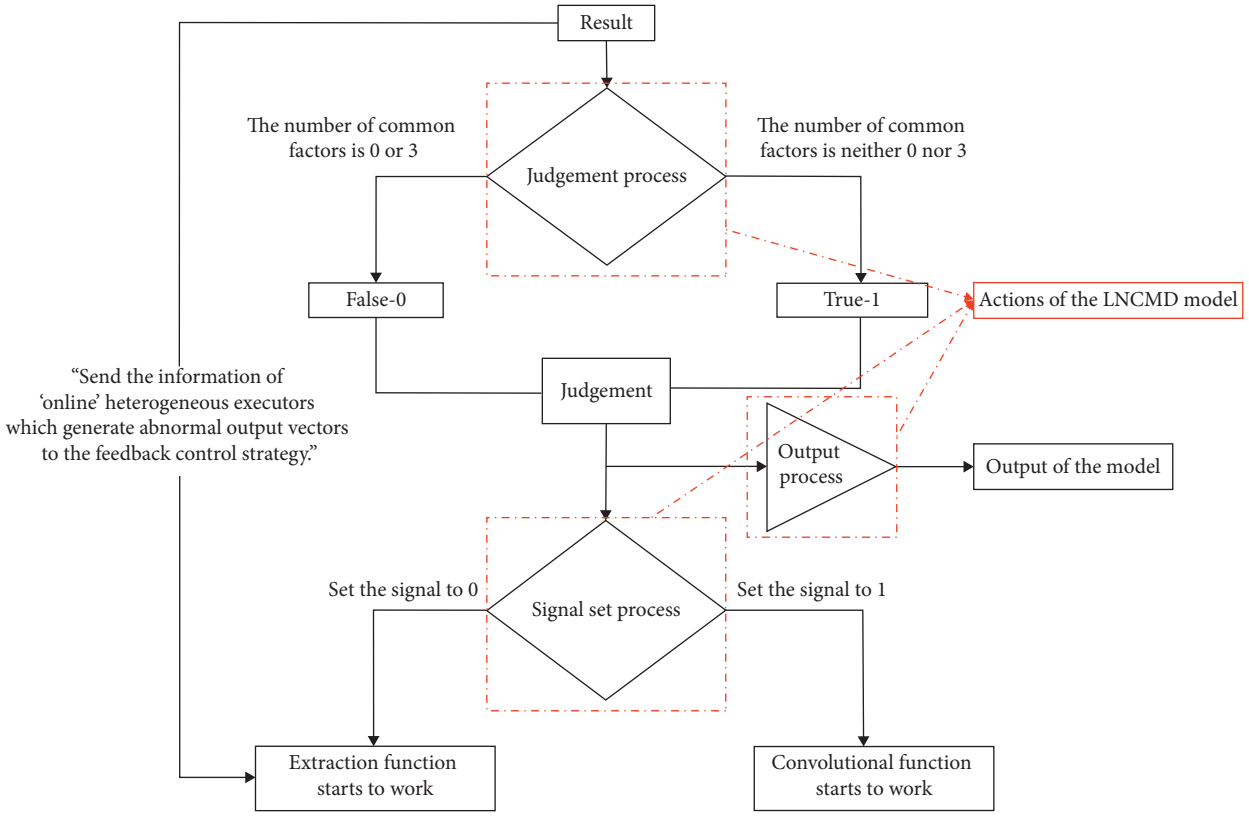


Figure 8: Flow of process highly dependent on the adjudication strategy.

Configuration 2: only the first "online" heterogeneous executor has a door, that is, $a < +\infty$ and $b\&c = +\infty$

Configuration 3: only the first "online" heterogeneous executor does not have a door, and the doors of the other two "online" heterogeneous executors are the same, which means that the flags that trigger the two doors are the same, that is, $a = +\infty$ and $(b = c) < +\infty$

Configuration 4: all the three "online" heterogeneous executors have the same doors, which means that the flags that trigger these doors are the same, that is, $(a = b = c) < +\infty$

Configuration 5: all the three "online" heterogeneous executors have different doors, which means that the flags that trigger these doors are unique, that is,

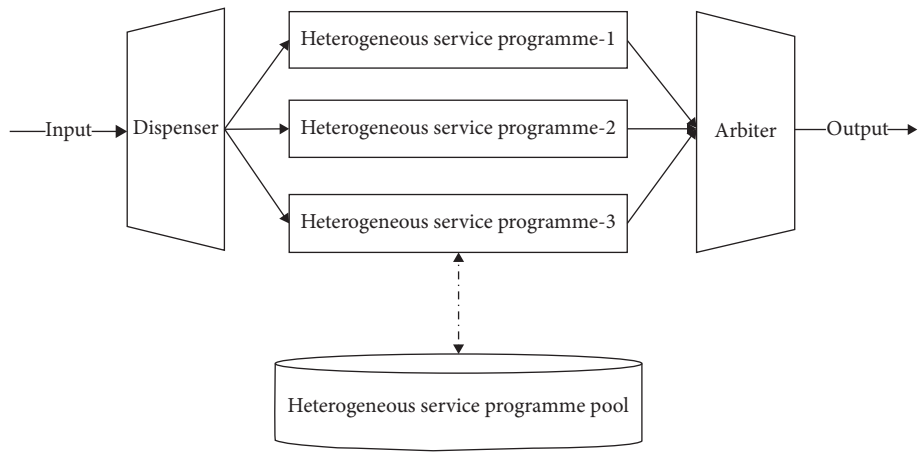


Figure 9: Physical background for the simulation experiment.

Table 4: Population of the LNCMD model.

Populated items	Content
Pool	The mapping values of the heterogeneous executors in the pool are $\{a, b, c, d, e, f, g\}$
Vector	The values in the vector are $[a, b, c]$
Product	The product P is $P = a \times b \times c$
Matrix	The matrix is as follows: The number of rows is 3, and no elements are visible
Signal	The value of the signal is 1
Input	The current status of the input is as follows: Waiting for the customer to write
Strategy	The system strategies are as follows: (1) Dynamic scheduling strategy: all the "online" heterogeneous executors are replaced randomly from the heterogeneous executor pool at fixed intervals (2) Feedback control strategy: the abnormal "online" heterogeneous executors are replaced randomly from the heterogeneous executor pool (3) Adjudication strategy: the aforementioned majority vote strategy is selected
Output	The current status of the output is not applicable

The populated LNCMD model is termed as lncmdDemo in this paper.

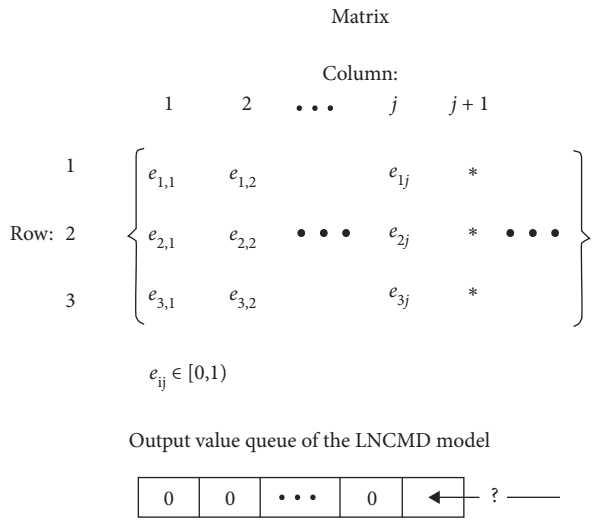


Figure 10: Simulation to observe lncmdDemo under configuration 1.

$$(a \neq b \neq c) < +\infty$$

5.1.2. Simulation Result. Let the output value queue of the LNCMD model be "Q." In this case, the simulation experiment of the LNCMD model can be experimentally observed through the "matrix" and "Q."

The matrix and Q corresponding to lncmdDemo in configuration 1 are shown in Figure 10.

In configuration 1, it is impossible for the attacker to cover the SAV on any "online" heterogeneous executor; that is, the attacker cannot hit any large prime factor (ordered) in P. The model output shows that lncmdDemo was never attacked.

The matrix and Q corresponding to lncmdDemo in configuration 2 are shown in Figure 11.

In configuration 2, the attacker can cover the SAV on only the first "online" heterogeneous executor; that is, the attacker may hit the first large prime factor in P. Once the hit

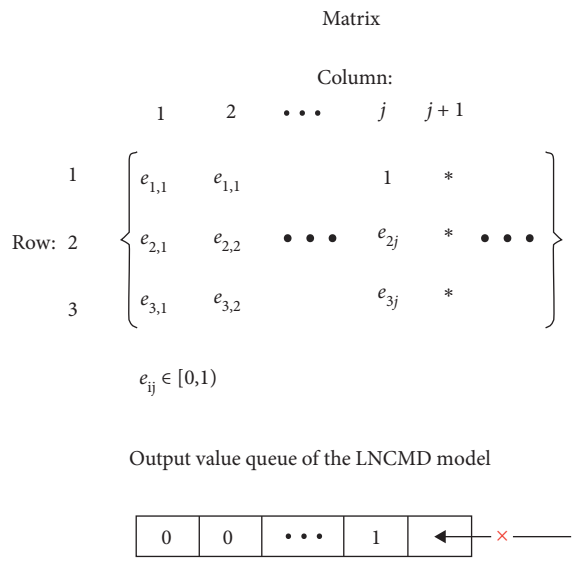


Figure 11: Simulation to observe lncmdDemo under configuration 2.

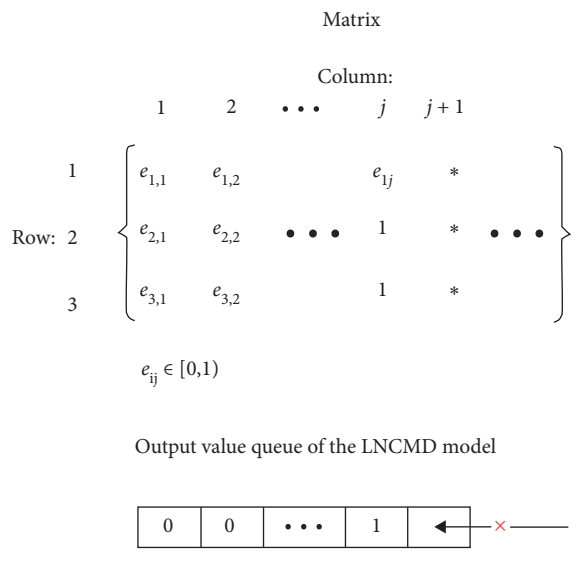


Figure 12: Simulation to observe lncmdDemo under configuration 3.

is successful, the model output will show that lncmdDemo is under attack and cause a "reset" operation.

The matrix and Q corresponding to lncmdDemo in configuration 3 are shown in Figure 12.

In configuration 3, the attacker cannot cover the SAV on the first "online" heterogeneous executor; that is, the attacker may hit the second and third large prime factors in P simultaneously. Once the hit is successful, the model output will show that lncmdDemo is under attack and cause a "reset" operation.

The matrix and Q corresponding to lncmdDemo in configuration 4 are shown in Figure 13.

In configuration 4, the attacker can cover the SAVs on all the "online" heterogeneous executors simultaneously; that is, the attacker may hit all the large prime factors (ordered) in P simultaneously. When the hit is successful, the model output will show that lncmdDemo was not attacked.

The matrix and Q corresponding to lncmdDemo in configuration 5 are shown in Figure 14.

The observation result of lncmdDemo under configuration 5 may be the same as that of any of the previous four configurations. If the attacker cannot cover the SAV on any "online" heterogeneous executor, lncmdDemo will behave as in configuration 1. If the attacker can only cover the SAV on one "online" heterogeneous executor, lncmdDemo will behave as in configuration 2. If the attacker can only cover the SAV on two "online" heterogeneous executors,

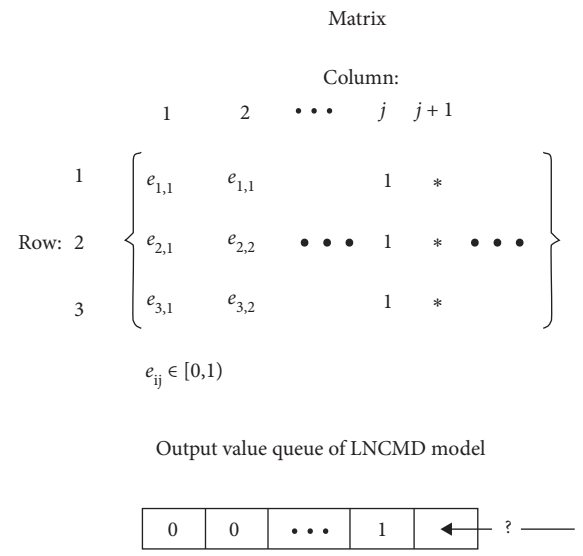


Figure 13: Simulation to observe lncmdDemo under configuration 4.

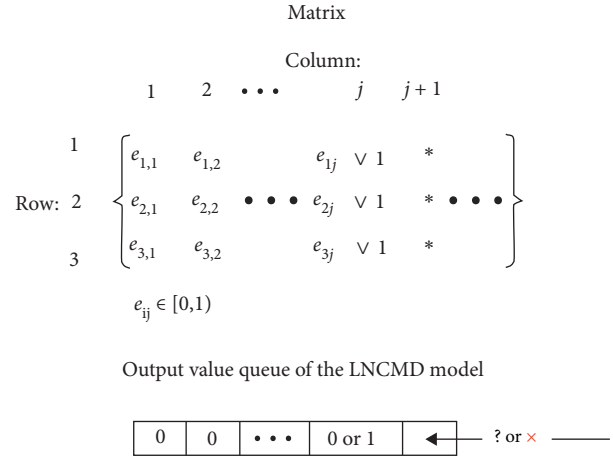


Figure 14: Simulation to observe lncmdDemo under configuration 5.

lncmdDemo will behave as in configuration 3. If the attacker can cover the SAVs on all the "online" heterogeneous executors, lncmdDemo will behave as in configuration 4.

5.1.3. Experimental Analysis and Evaluation. A part of the observation results of lncmdDemo under the five configurations is expected by the defender, and the remaining results are expected by the attacker. In this paper, the expected LNCMD model observation result for one side is termed as the "solution" of this side under the LNCMD model.

(1) For the defender:

(a) Analysis:
The simulation experiment indicates that the observation results of lncmdDemo under configurations 1, 2, and 5 are solutions for the defender.
The observation result of lncmdDemo in configuration 1 reflects that the CMD system always maintains a normal system output. Moreover, the feedback control is not triggered, and thus, no system turbulence is caused by the feedback control. The observation result of lncmdDemo in configuration 1 can be considered as the "optimal solution" for the defender under lncmdDemo.
The observation result of lncmdDemo in configuration 2 reflects that the CMD system always maintains a normal system output, although the feedback control may be triggered, which may induce the system turbulence caused by the feedback control. The observation result of lncmdDemo in configuration 2 can be considered as the "ordinary solution" for the defender.
The observation result of lncmdDemo in configuration 5 is unstable, that is, it may or may not be what the defender expects, and a luck component is involved. The observation result of

lncmdDemo in configuration 5 can be considered as the "worst solution" for the defender.

The observation results of lncmdDemo in configurations 3 and 4 do not correspond to the "solution" for the defender. At this time, the CMD system already exhibits a "joint escape" phenomenon.

(b) Evaluation:

Through the simulation experiment, it can be concluded that chasing the "optimal solution" under the LNCMD model is the ultimate goal of the defender. The defender's "optimal solution" under the LNCMD model can be attained by ensuring that the LNCMD model exhibits the following martingale characteristic.

Hypothesis:

Let the judgement of the attack by the judgement function excited by the j^{th} model input be a stochastic process Y_j.

In the j^{th} model input, let the probability that the attacker cannot decompose any large prime factor (ordered) from the value of the product variable correspond to a stochastic process X_j.

When a stochastic process X_j is used as a condition, it means that an event with X_j as the probability has occurred, so $0 < X_j \leq 1$.

Then, this martingale characteristic is

$$E\left(Y_{(j+1)} \mid X_1, \ldots, X_j\right) = Y_j = 0. \tag{13}$$

At this time, the stochastic process Y_j is a discrete martingale on X_j. Its meaning is that the attacker can never decompose any large prime factor (ordered) from the value of the product variable, which makes the judgement function always judge that the LNCMD model was not attacked. On this basis, for the future $(j+1)$-th model input, the conditional expectation of the event $Y_{(j+1)}$ is that the LNCMD model will not be attacked. In a practical sense, this scenario means that, in the CMD system, the attacker cannot cover the SAV on any "online" heterogeneous executor, and no system turbulence is caused by triggering the feedback control. When the LNCMD model has the above martingale characteristic, it reflects the defender's "optimal solution" under the LNCMD model.

Therefore, the problem of how the defender chases the "optimal solution" under the LNCMD model can be transformed into the problem of ensuring that the LNCMD model exhibits the aforementioned martingale characteristic. The defender considers how to adjust various system strategies to ensure that the LNCMD model exhibits the aforementioned martingale characteristic, which is the mathematical nature of the problem faced by the defender in the CMD system in the practical sense, as clarified by the LNCMD model.

(2) For the attacker:

(a) Analysis:

The simulation experiment indicates that the observation results of lncmdDemo under configurations 3, 4, and 5 are solutions for the attacker.

The observation result of lncmdDemo in configuration 4 reflects that the CMD system has a "joint escape" phenomenon; however, it is impossible to perceive the occurrence of the attack. The observation result of lncmdDemo in configuration 4 can be considered as the "optimal solution" for the attacker.

The observation result of lncmdDemo in configuration 3 reflects that the CMD system exhibits a "joint escape" phenomenon, and the occurrence of the attack can be perceived. The observation result of lncmdDemo in configuration 3 can be considered as the "ordinary solution" for the attacker.

The observation result of lncmdDemo in configuration 5 is unstable, that is, it may or may not be what the attacker expects, and a luck component is involved. The observation result of lncmdDemo in configuration 5 can be considered as the "worst solution" for the attacker.

The observation results of lncmdDemo in configurations 1 and 2 do not correspond to the attacker's "solution." In this scenario, the CMD system always maintains a normal system output.

(b) Evaluation:

The simulation experiment indicated that chasing the "optimal solution" under the LNCMD model is the ultimate goal of the attacker. The attacker wants to obtain the "optimal solution" under the LNCMD model depending on the composition of the large prime factors in the value of the product variable and the hit method for each large prime factor (ordered).

The following simulation experiment is considered as an example:

The large prime factors contained in P for lncmdDemo are a, b, and c. Under configurations 1, 2, and 3, the compositions of the large prime factors in P are $(a\&b\&c = +\infty)$, $(a < +\infty$ and $b\&c = +\infty)$, and $(a = +\infty$ and $(b = c) < +\infty)$, respectively. Under these compositions of the large prime factors, the attacker cannot obtain the "optimal solution" under lncmdDemo, regardless of the hit method employed by the attacker. Under configuration 4, the composition of the large prime factors in P is $(a = b = c) < +\infty$. Under this composition of the large prime factors, the attacker can obtain the "optimal solution" under lncmdDemo. At this time, the hit method used by the attacker determines the rate at

which the "optimal solution" is attained under lncmdDemo. Under configuration 5, the composition of the large prime factors in P is $(a \neq b \neq c) < +\infty$. Under this composition of the large prime factors, the attacker can attain the "optimal solution" under lncmdDemo. At this time, the hit method used by the attacker determines whether the attacker can obtain the "optimal solution" under lncmdDemo.

5.2. Security of CMD. The evaluation of the network security is different from that of the information system performance. The former evaluation is more difficult to describe quantitatively compared to the latter evaluation [48]. Therefore, by modelling the CMD mechanism and using a mathematical model to express the CMD mechanism, this paper establishes a connection between the CMD mechanism and mathematics to ensure that the safety of the CMD can be qualitatively evaluated based on the LNCMD model. The following section describes the qualitative evaluation of the safety of the CMD based on the LNCMD model.

The LNCMD model uses a large prime factor product to represent the "online" heterogeneous executor set of the CMD system at a certain time, and it uses the large prime factor decomposition problem to map an attacker's attack on the CMD system. The concept of the LNCMD model is applied, and the traditional static and single system is represented as one large prime number. For the convenience of the subsequent description, the aforementioned large prime factor product and large prime factor are, respectively, represented as "composite" and "pfactor$_i$" ($1 \leq i \leq n$, n is the number of factors), and the aforementioned large prime number is represented as "prime." At the same time, the "ray" that starts from 0 and grows to $+\infty$ is used to represent a dimension. Considering these aspects, "composite" and "pfactor$_i$" are shown in the upper part of Figure 15, and "prime" is shown in the lower part of Figure 15.

When the "composite" is determined, the attacker must analyse "pfactor$_i$s" that the "composite" comprises. The "composite" is only on a one-dimension "ray," and each "pfactor$_i$" that composes the "composite" is also on a one-dimension "ray." Therefore, a mapping relationship from a multidimensional "ray" to a one-dimension "ray" is formed between the "composite" and "pfactor$_i$." However, the attacker must also analyse this mapping from the multidimensional "ray" to the one-dimension "ray." Furthermore, for "prime," the attacker only needs to analyse the specific value of the "prime." Because the "prime" does not have a multidimensional mapping relationship, the attacker only needs to analyse on a one-dimension "ray."

To impede the attacker from performing this analysis, the complexity can be considered as a general approach. The complexity for a "composite" can be improved by increasing the number or value of "pfactor$_i$," and these two aspects can be combined. For "prime," the complexity can only be improved by increasing the value of "prime." However, according to the CMD mechanism, the "composite" is not immutable. Through the multidimensional dynamic driving one-dimensional dynamic, the "composite" can implement active changes based on the above two aspects or passive changes caused by the attacker's analysis. Regardless of the active or passive changes in the "composite," all the previous efforts of the attacker may be wasted, thereby greatly increasing the complexity on the original basis and rendering the analysis to be conducted by the attacker more difficult.

Next, an arithmetic analysis is carried out to intuitively reflect this complexity relationship. We use time to measure the above complexity, denoted as T. Assuming that the redundancy scale of the "online" heterogeneous executors in the CMD system is three, then there are "pfactor$_1$," "pfactor$_2$," and "pfactor$_3$." At the same time, "pfactor$_i$s " ($1 \leq i \leq 3$) are different from each other; then, their complexity is t_1, t_2, and t_3, respectively. We take the dynamic characteristic of CMD as a weight, denoted as w, w tends to $+\infty$. For the traditional static and single system, let the complexity of "prime" be t. The following formula gives this complexity relationship:

$$\begin{aligned} T(\text{Traditional}) &= t, \\ T(\text{CMD\_Static}) &= t_1 + t_2 + t_3, \\ T(\text{CMD}) &= t_1 \times w + t_2 \times w + t_3 \times w \\ &= (t_1 + t_2 + t_3) \times w \approx +\infty. \end{aligned} \tag{14}$$

For the above complexity, T(Traditional) refers to the traditional static and single system, T(CMD_Static) refers to the CMD system without the dynamic characteristic, and T(CMD) refers to the CMD system. For t, t_1, t_2, and t_3, they should be in the same order of magnitude, but there are differences in size. It is not difficult to see that the complexity of CMD far exceeds the traditional static and single system.

Therefore, based on the analysis of the aforementioned arithmetic significance, this paper performs a qualitative assessment of the safety of the CMD. In contrast to the traditional static and single system, the CMD system raises the difficulty level of guessing a single large prime number to the difficulty level of solving the decomposition problem of the dynamic large prime factor product. Therefore, the CMD is highly secure.

5.3. Overhead of CMD. While the use of CMD technology brings high security and high robustness, additional system overhead is inevitable because the use of any technology has to pay a certain price, but it is entirely possible to control these additional overheads within an acceptable range by certain means. The additional overheads brought by CMD can be analyzed from the following two aspects:

(1) Complexity Overhead. We use space complexity to measure the complexity overhead and use the CMD system to compare with the traditional static and single system. Here, the space complexity is expressed in terms of quantity, and a unit quantity is $O(1)$.

The complexity overhead of the CMD system is

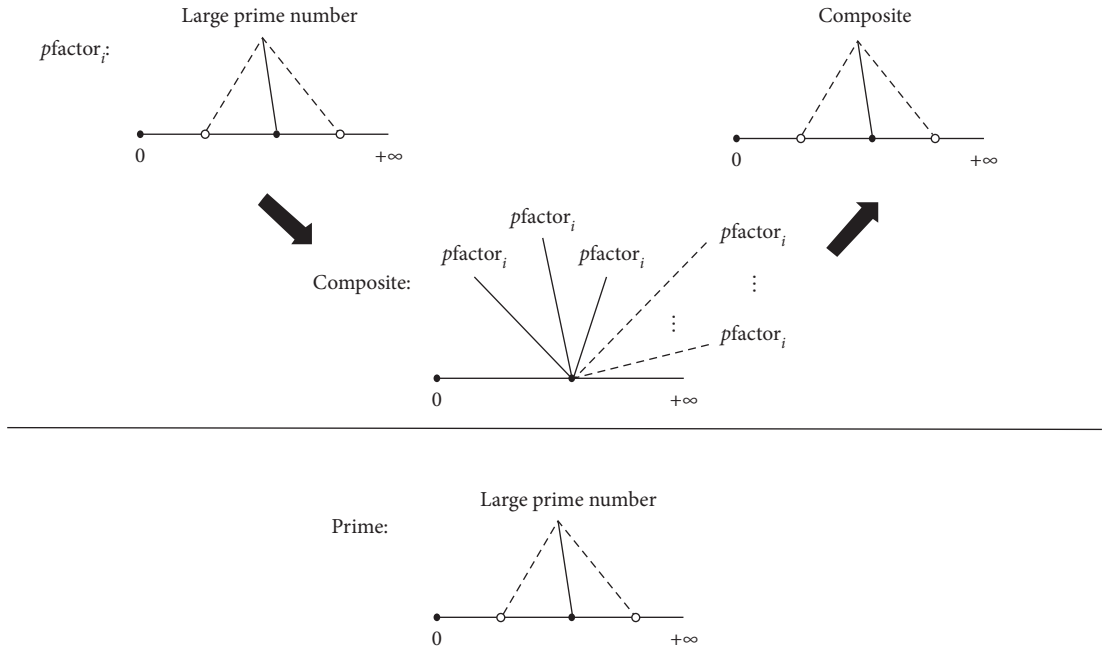


Figure 15: Diagram of "composite & $p\text{factor}_i$ and prime."

$$\begin{cases} O(E) = O(e_1) + O(e_2) + \cdots + O(e_n), \\ S(\text{CMD}) = O(d) + O(E) + O(a) + O(c) + \Delta. \end{cases} \tag{15}$$

$O(E)$ is the space complexity of the heterogeneous executor pool. $O(e_i)$ is the space complexity of the heterogeneous executor. Assuming that there are a total of n heterogeneous executors in the heterogeneous executor pool, then $O(E)$ is the sum of each $O(e_i)$ $(1 \le i \le n)$. $O(d)$ is the space complexity of the input distributor. $O(a)$ is the space complexity of the output arbiter. $O(c)$ is the space complexity of the CMD converter. Δ is the extra space complexity. For example, when the CMD system includes the "cleaning" function for abnormal executors, this part of the space complexity belongs to Δ. Ultimately, the complexity overhead of the CMD system is $S(\text{CMD})$.

The complexity overhead of the traditional static and single system is

$$S(\text{Traditional}) = O(e). \tag{16}$$

$O(e)$ is the space complexity of the executor that achieves the target function; then, the complexity overhead of the traditional static and single system is $S(\text{Traditional})$.

So far, it can be concluded that the additional complexity overhead of the CMD system compared to the traditional static and single system is $S(\text{CMD}) - S(\text{Traditional})$. For the executors in these two types of systems, their space complexity is similar, that is, $O(e) \approx O(e_i)$ $(1 \le i \le n)$.

(2) Performance Overhead. We use the time complexity to measure the performance overhead and use the CMD system to compare with the traditional static and single system. Here, the time complexity is expressed in terms of quantity, and a unit quantity is $O(1)$.

The performance overhead of the CMD system is

$$\begin{cases} O(E) = \max(O(e_1), O(e_2), \ldots, O(e_n)), \\ T(\text{CMD}) = O(d) + O(E) + O(a) + O(c) + \Delta. \end{cases} \tag{17}$$

$O(E)$ is the time complexity of the "online" heterogeneous executor set. $O(e_i)$ is the time complexity of the "online" heterogeneous executor. Assuming that there are a total of n "online" heterogeneous executors in the "online" heterogeneous executor set, because the "online" heterogeneous executors are executed in parallel, $O(E)$ is the maximum value of all $O(e_i)$ $(1 \le i \le n)$. $O(d)$ is the time complexity of the input distributor. $O(a)$ is the time complexity of the output arbiter. $O(c)$ is the time complexity of the CMD converter. Δ is the extra time complexity. For example, when the CMD system includes the "cleaning" function for abnormal executors, this part of the time complexity belongs to Δ. Ultimately, the performance overhead of the CMD system is $T(\text{CMD})$.

The performance overhead of the traditional static and single system is

$$T(\text{Traditional}) = O(e). \tag{18}$$

$O(e)$ is the time complexity of the executor that achieves the target function; then, the performance overhead of the traditional static and single system is $T(\text{Traditional})$.

So far, it can be concluded that the additional performance overhead of the CMD system compared to the traditional static and single system is $T(\text{CMD}) - T(\text{Traditional})$. For the executors in these two types of systems, their time complexity is similar, that is, $O(e) \approx O(e_i)$ $(1 \le i \le n)$.

For the additional overheads of using CMD technology, it is necessary to reduce them to an acceptable range. Referencing the aforementioned various complexities to

optimize the system implementation is the first method, and focusing on the rationality of using CMD technology is the second method. In terms of rationality, assuming that CMD technology is used to protect data, the overheads of only using CMD technology to protect a small amount of critical data are far less than using CMD technology to protect ordinary mass data, but the security will not differ too much. For example, using CMD technology to protect the access control list (ACL) in the firewall, the overhead caused by encryption operation is acceptable. At the same time, because ACL is the critical data, the overall security of the system will also be greatly improved.

6. Conclusion and Future Work

This paper proposes a large-number convolutional mimic defence mathematical model. The LNCMD model is an intuitive and exclusive mathematical model of the CMD. The LNCMD model transforms the problems of the attack and defence game of the CMD into corresponding mathematical problems. For the defender, the LNCMD model transforms the problem of how the defender uses the CMD for security protection into the problem of how the defender adjusts various system strategies to ensure that the LNCMD model has a specific martingale characteristic. For the attacker, the LNCMD model innovatively transforms the problem of the attacker attacking the CMD system into the problem of the attacker factorising the large prime factor product. Therefore, based on the LNCMD model, this paper performs a qualitative assessment that indicates that the CMD is highly secure. The proposed LNCMD model can be implemented directly through programming, and the subsequent step is to programme the LNCMD model to further examine the key technologies of the CMD framework.

Data Availability

The simulation data used to support this study are included within this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (no. 2016YFB0800100).

References

[1] T. D. Braun, H. J. Siegel, N. Beck et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[2] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] R. C. Newman, "Cybercrime, identity theft, and fraud: practicing safe internet - network security threats and vulnerabilities," in *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development*, pp. 68–78, Kennesaw, Georgia, 2006.

[4] G. Cai, B. Wang, T. Wang, Y. Luo, and X. Cui, "Research and development of moving target defense technology," *Journal of Computer Research & Development*, vol. 53, no. 5, pp. 968–987, 2016.

[5] X. Luo, Q. Tong, Z. Zhang, and J. Wu, "Mimic defense technology," *Strategic Study of Chinese Academy of Engineering*, vol. 18, no. 6, pp. 69–73, 2016.

[6] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, and W. Karl, "Scientific cloud computing: early definition and experience," in *Proceedings of the the 10th IEEE International Conference on High Performance Computing and Communications*, pp. 825–830, Dalian, China, September 2008.

[7] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 239, 2014.

[8] J. Wu, *Introduction to Cyberspace Mimic Defense*, Science Press, Beijing,China, 2017.

[9] J. Wu, *Principle of Cyberspace Mimic Defense*, Science Press, Beijing, China, 2018.

[10] X. Si, W. Wang, J. Zeng et al., "A review of the basic theory of mimic defense," *Engineering Sciences*, vol. 18, no. 6, pp. 62–68, 2016.

[11] J. Wu, "Research on cyber mimic defense," *Journal of Cyber Security*, vol. 1, no. 4, pp. 1–10, 2016.

[12] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, p. 417, 2009.

[13] K. Zhou, C. John, and Doyle, *Essentials of Robust Control*, p. 38, Prentice Hall, Upper Saddle River, NJ, USA, 1998.

[14] J. Wu, *Cyberspace Mimic Defense*, Springer, Switzerland Cham, Switzerland, 2020.

[15] G. F. Franklin, J. D. Powell, and E. Abbas, *Feedback Control of Dynamic Systems*, China Machine Press, Beijing, China, 2016.

[16] Q. Tong, Z. Zhang, W. Zhang, and J. Wu, "Design and implementation of mimic defense web server," *Journal of Software*, vol. 28, no. 4, pp. 883–897, 2017.

[17] Z. Zhang, B. Ma, and J. Wu, "The test and analysis of prototype of mimic defense in web servers," *Journal of Cyber Security*, vol. 2, no. 1, pp. 13–28, 2016.

[18] H. Ma, P. Yi, Y. Jiang, and L. He, "Dynamic heterogeneous redundancy based router architecture with mimic defenses," *Journal of Cyber Security*, vol. 2, no. 1, pp. 29–42, 2016.

[19] H. Ma, Y. Jiang, B. Bai, and J. Zhang, "Tests and analyses for mimic defense ability of routers," *Journal of Cyber Security*, vol. 2, no. 1, pp. 43–53, 2017.

[20] X. Ji, K. Huang, L. Jin et al., "Overview on 5G security technology," *Mobile Communications*, vol. 43, no. 1, pp. 34–39+45, 2019.

[21] Z. Wang, H. Hu, and G. Cheng, "Design and implementation of mimic network operating system," *Journal of Computer Research and Development*, vol. 54, no. 10, pp. 2321–2333, 2017.

[22] Z. Gu, X. Zhang, and S. Lin, "Research on security mechanism for SDN control layer based on mimic defense theory," *Application Research of Computers*, vol. 35, no. 7, pp. 2148–2152, 2018.

[23] J. Pang, Y. Zhang, Z. Zhang, and J. Wu, "Applying a combination of mimic defense and software diversity in the

software security industry," *Engineering Sciences*, vol. 18, no. 6, pp. 74–78, 2016.
[24] X. Liu, X. Zou, J. Tan, and J. Wu, "Survey of large integer factorization algorithms," *Application Research of Computers*, vol. 31, no. 11, pp. 3201–3207, 2014.
[25] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[26] J. M. Pollard, "A Monte Carlo method for factorization," *Bit*, vol. 15, no. 3, pp. 331–334, 1975.
[27] J. M. Pollard, "Theorems on factorization and primality testing," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 76, no. 3, pp. 521–528, 1974.
[28] H. W. Lenstra, "Factoring integers with elliptic curves," *The Annals of Mathematics*, vol. 126, no. 3, pp. 649–673, 1987.
[29] C. Pomerance, "The quadratic sieve factoring algorithm," in *Proceedings of the of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory & Application of Cryptographic Techniques*, Springer, Paris, France, April 1984.
[30] A. K. Lenstra and H. W. Lenstra, *The Development of the Number Field Sieve*, Springer-Verlag, Berlin, Germany, 1956.
[31] K. Pipyros, L. Mitrou, D. Gritzalis et al., "A cyber attack evaluation methodology," in *Proceedings of the 13th European Conference on Cyber Warfare and Security*, pp. 264–270, Piraeus, Greece, 2014.
[32] Z. Shi, G. Zhao, and J. Liu, "The effect evaluation of the network attack based on the fuzzy comprehensive evaluation method," in *Proceedings of the International Conference on Systems & Informatics*, November 2016.
[33] G. Tuvell, C. Jiang, and S. Bhardwaj, "Off-line mms malware scanning system and method," 2008.
[34] A. Orebaugh, *Ethereal Packet Sniffing*, Syngress Publishing, Amsterdam, Netherlands, 2004.
[35] B. Schneier, "Attack trees: modeling security threats," *Dobb's Journal*, vol. 24, no. 12, pp. 4–6, 1999.
[36] L. P. Swiler and C. Phillips, "A graph-based system for network-vulnerability analysis," in *Proceedings of the Workshop on New Security Paradigms*, pp. 71–79, New York, NY, USA, 1998.
[37] J. P. Mcdermott, "Attack net penetration testing," in *Proceedings of the Workshop on new security paradigms*, ACM, New York, NY, USA, 2001.
[38] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, 2011.
[39] M. Howard, J. Pincus, and J. M. Wing, "Measuring relative attack surfaces," in *Computer Security in the 21st Century*, pp. 109–137, Springer, Berlin, Germany, 2005.
[40] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," *Moving Target Defense II*, Springer, New York, NY, USA, 2013.
[41] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, Springer, Berlin, Germany, 2011.
[42] W. Stallings, *Network and Internetwork Security: Principles and Practice. Network and Internetwork Security: Principles and Practice*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.
[43] S. Lin, Q. Liu, and X. Wang, "Competitive arbitration model for mimic defense system," *Computer Engineering*, vol. 44, no. 4, pp. 193–198, 2018.
[44] W. Li, Z. Zhang, L. Wang, and J. Wu, "The modeling and risk assessment on redundancy adjudication of mimic defense," *Journal of Cyber Security*, vol. 3, no. 5, pp. 64–74, 2018.
[45] X. Zhang, Z. Gu, S. Wei, and J. Shen, "Markov game modeling of mimic defense and defense strategy determination," *Journal on Communications*, vol. 39, no. 10, pp. 143–154, 2018.
[46] Q. Ren, L. He, and J. Wu, "Analysis of different anti-interference system models based on discrete time markov chain," *Chinese Journal of Network and Information Security*, vol. 4, no. 4, pp. 30–37, 2018.
[47] S. E. Chang and C. B. Ho, "Organizational factors to the effectiveness of implementing information security management," *Industrial Management & Data Systems*, vol. 106, no. 3, pp. 345–361, 2006.
[48] J. Zhang, J. Pang, and Z. Zhang, "Quantification method for heterogeneity on web server with mimic construction," *Journal of Software*, vol. 31, no. 2, pp. 564–577, 2020.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8882494, 11 pages
https://doi.org/10.1155/2020/8882494

WILEY | Hindawi

Research Article

Challenging the Adversarial Robustness of DNNs Based on Error-Correcting Output Codes

Bowen Zhang,[1] **Benedetta Tondi**,[2] **Xixiang Lv**,[1] **and Mauro Barni**[2]

[1]*School of Cyber Engineering, Xidian University, Xi'an 710126, China*
[2]*Department of Information Engineering and Mathematics, University of Siena, Siena 53100, Italy*

Correspondence should be addressed to Xixiang Lv; xxlv@mail.xidian.edu.cn

Received 21 August 2020; Revised 21 September 2020; Accepted 7 October 2020; Published 16 November 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Bowen Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The existence of adversarial examples and the easiness with which they can be generated raise several security concerns with regard to deep learning systems, pushing researchers to develop suitable defence mechanisms. The use of networks adopting error-correcting output codes (ECOC) has recently been proposed to counter the creation of adversarial examples in a white-box setting. In this paper, we carry out an in-depth investigation of the adversarial robustness achieved by the ECOC approach. We do so by proposing a new adversarial attack specifically designed for multilabel classification architectures, like the ECOC-based one, and by applying two existing attacks. In contrast to previous findings, our analysis reveals that ECOC-based networks can be attacked quite easily by introducing a small adversarial perturbation. Moreover, the adversarial examples can be generated in such a way to achieve high probabilities for the predicted target class, hence making it difficult to use the prediction confidence to detect them. Our findings are proven by means of experimental results obtained on MNIST, CIFAR-10, and GTSRB classification tasks.

1. Introduction

Deep neural networks can solve complicated computer vision tasks with unprecedented high accuracies. However, they have been shown to be vulnerable to *adversarial examples*, namely, properly crafted inputs introducing small (often imperceptible) perturbations, inducing a classification error [1–3]. The possibility of crafting both nontargeted and targeted attacks has been demonstrated, the goal of the former being to induce any kind of classification error [4, 5], while the latter aims at making the network decide for a target class chosen a priori [1, 6]. It goes without saying that, in general, targeted attacks are more difficult to build.

As a reaction to the threats posed by adversarial examples, many defence mechanisms have been proposed to increase the adversarial robustness of deep neural networks [7–12]. However, in a white-box setting wherein the attacker has a full knowledge of the attacked network, including full knowledge of the defence mechanism, more powerful attacks can be developed, thus tipping again the scale in favour of the attacker [4, 13].

In this race of arms, a novel defence strategy based on *error-correcting output coding* (ECOC) [14] has been proposed recently in [15], to counter adversarial attacks in a white-box setting. More specifically, given a general multiclass classification problem, error-correcting output codes are used to encode the various classes and represent the network's outputs. To explain how, let us refer to the output of the last layer of the network, prior to the final activation layer, as logit values or simply logits. In general, the final activation layer consists of the application of an activation function, which maps the logits into a prescribed range, and a normalization layer, which maps the output of the activation functions into a probability vector, associating a probability value to each class. In the common case of one-hot-encoding, a softmax layer is used, in which case these two steps are performed simultaneously. During training, the network learns to output a large logit value for the true class and small values for all the others. With the ECOC approach, instead, the network is trained in such a way to produce normalized logit values that correlate well with the codeword used to encode the class the input sample belongs

to. In general, ECOC codewords have many nonzero values, thus marking a significant difference with respect to the one-hot-encoding case.

The rationale behind the use of the ECOC architecture to counter the construction of adversarial examples [15] is that while with classifiers based on standard one-hot-encoding the attacker can induce an error by modifying one single logit (reducing the one associated to the ground-truth class or increasing the one associated to the target class), the final decision of the ECOC classifier depends on multiple logits in a complicated manner, and hence it is supposedly more difficult to attack (especially when longer codewords are used).

In [15], the authors considered nontargeted attacks in their experiments and showed with the popular white-box C&W attack that the attack success rate on CIFAR-10 [16] passes from 100%, for one-hot-encoding, to 29%, for an ECOC-based classifier.

Another alleged advantage of the ECOC architecture proposed in [15] is linked to the way the probabilities associated with each class are computed. Rather than using a softmax function as commonly done with one-hot-encoding, first the correlation between the activated outputs and the codeword is computed, and then a linear normalization procedure is applied (see equation (2) in the following). In this way, the probability assigned to the class chosen by the classifier grows more smoothly, and samples close to the decision region boundary (like adversarial examples are likely to be) are classified with a low confidence. Results presented in [15], in fact, show that the ECOC model tends to provide sharp results for clean images, while it is often uncertain about the (incorrect) prediction made on adversarial examples. This behavior could be exploited to, at least, distinguish between adversarial examples and benign inputs.

The goal of this paper is to further verify if and to which extent the use of error correction codes to encode the output of deep neural networks allows to increase the robustness against targeted adversarial examples. We do so by introducing a new white-box attack, inspired to C&W attack, explicitly thought to work not only against ECOC but also other multilabel classifiers. In fact, the original C&W is naturally designed to deceive networks adopting the one-hot-encoding strategy, and it loses some of its advantages when used against ECOC systems. We stress that, in contrast to previous works (see, for instance, [15] and Section 10 in [17]), we aim at developing a targeted attack, which is a more difficult task than crafting nontargeted adversarial examples. This is a reasonable choice for at least two reasons. First, targeted attacks are more flexible than nontargeted ones since they can be used in a wider variety of applications, wherein the ultimate goal of the attack may vary considerably. Secondly, being able to attack a defence under most stringent attacking constraints illustrates better the weakness of the defence itself.

We ran extensive experiments to evaluate the ability of ECOC-based classifiers to resist the new attack and compared the results we got with those obtained by applying a fine-tuned version of C&W attack and the LOTS attack introduced in [18]. The experiments were carried out by considering three different classification tasks, namely, traffic sign classification (GTSRB) [19], CIFAR-10 classification [16], and MNIST [20]. As a result, we found that the ECOC classifiers can be successfully attacked with a high success rate. In particular, the new attack outperforms the other two especially when long codewords are used by ECOC. We also verified that, by increasing the confidence of the attack, adversarial examples can achieve high probabilities for the predicted target class, similar to those of benign samples, hence making it difficult to use the prediction confidence to detect adversarial samples. Overall, our analysis reveals that the security gain achieved by the ECOC scheme is a minor one, thus calling for more powerful defences.

The rest of this paper is organised as follows: we first briefly review the ECOC scheme presented in [15], and then we describe the proposed attack. The setup considered for the experiments, and the results we got are reported and discussed in Section 4. Eventually, we review the related work at the end of the paper.

2. ECOC-Based Classification

Let us first introduce the notation for a general multiclass CNN. Let x be the input of the network and k the class label, $k = 1, 2, \cdots, M$, where M denotes the number of classes. Let $f(x)$ indicates the decision function of the network. We denote by $\mathbf{z} = (z_1, z_2, \cdots,)$, the vector with the logit values, that is, the network values before the final activations and the mapping to class probabilities. For one-hot-encoding schemes, $\mathbf{z}$ has length M, and the logits are directly mapped into probability values through the softmax function ψ as follows:

$$p_\psi(k) = \psi_k(\mathbf{z}) = \frac{\exp(z_k)}{\sum_{i=1}^{M} \exp(z_i)}, \tag{1}$$

for $k = 1, \cdots, M$. Then, the final prediction is made by letting $f(x) = \arg\max_k p_\psi(k)$.

The error-correction-output-coding (ECOC) scheme proposed in [15] assigns a codeword $\mathbf{C}_k$ of length N ($N \geq M$) to every output class ($k = 1, \cdots, M$). $\mathbf{C}$ denotes the $M \times N$ codeword matrix. Each element of $\mathbf{C}$ can take values in $\{-1, 1\}$. In this way, the length of the logit vector $\mathbf{z}$ is N. The logits are first mapped into the $[-1, 1]$ range by means of an activation function $\sigma(\cdot)$ (e.g., the tanh function that $\sigma(x) = (e^x - e^{-x})/(e^x + e^{-x})$). Then, the probability of class k is computed by looking at the correlation with $\mathbf{C}_k$, according to the following formula:

$$p_\sigma(k) = \frac{\max(\sigma(\mathbf{z}) \cdot \mathbf{C}_k, 0)}{\sum_{i=1}^{M} \max(\sigma(\mathbf{z}) \cdot \mathbf{C}_i, 0)}, \tag{2}$$

where $\cdot$ denotes the inner product and $\sigma(\cdot)$ is a sigmoid activation function applied element-wise to the logits. Since C_{ij}s take values in $\{-1, 1\}$, the max is necessary to avoid negative probabilities. According to [15], the common softmax rule (equation (1)) is able to express uncertainty between two classes only when the logits are roughly equal

(i.e., $z_1 \approx z_2$ and the two probabilities are close $p_\psi(i) \approx p_\psi(j)$). In a two dimensional case, this corresponds to a very narrow stripe, approximate to a line, across the boundary of the decision region, while in high dimensional spaces, the region $z_i \approx z_j$ approximates a hyperplan, an $M-1$ dimensional subspace of $\mathbb{R}^M$ with negligible volume, and hence the classifier outputs high probabilities almost everywhere. This makes it very easy for the attacker to find an adversarial input that is predicted (wrongly) with high confidence. With ECOC (equation (2)), instead, it is sufficient that two approximate correlations express low uncertainty ($\sigma(\mathbf{z}) \cdot \mathbf{C}_i \approx \sigma(\mathbf{z}) \cdot \mathbf{C}_j$), and then a non-trival volume is allocated to low-confidence region in the logit space, thus limiting the freedom of the attacker to craft high-confidence adversarial examples. An overall sketch of the ECOC scheme is depicted in Figure 1. The logits $\mathbf{z}$ are first mapped into correlation values, $\rho := \sigma(\mathbf{z}) \cdot \mathbf{C}$ (mapping step 1), and then the vector with the correlations is normalized so to form a probability distribution (mapping step 2) via the normalization function in (2). The model's final predicted label is $\arg\max_k p_\sigma(k)$. Equation (2) is a generalization of the standard softmax activation in equation (1) and reduces to it for the case of one-hot-encoding, that is, when $\mathbf{C} = \mathbf{I}_M$, with $N = M$, and where $\mathbf{I}_M$ is the identity $M \times M$ matrix.

The purpose of the ECOC architecture is to design a classifier which is robust to changes of multiple logits and then, expectedly, more difficult to attack (with standard one-hot-encoding the adversary can succeed by altering a single logit). For the scheme to be effective, codewords characterised by a large minimum Hamming distance must be chosen. For simplicity, in [15], the ECOC classifier is built by using Hadamard codes taking values in $\{-1, 1\}$ (when $\mathbf{C}$ is a Hadamard matrix, the Hamming distance for large M approaches the limit value $N/2$). An advantage with this choice is that, since $\mathbf{C}$ is orthogonal, whenever the network outputs a codeword exactly (that is when $\sigma(\mathbf{z}) = \mathbf{C}_k$), then $p_\sigma(k) = 1$. The tanh function is selected as the activation function $\sigma(\cdot)$.

The authors also found that, rather than considering a single network with N outputs, a classifier consisting of an ensemble of several smaller networks, each one outputting a few codeword elements, permits to achieve a larger robustness against attacks. By training separate networks, in fact, the correlation between errors affecting different bits of the codewords is reduced, thus forcing the attacker to attack all the bits independently. In the scheme in Figure 1, every network outputs one codeword bit only, resulting in N ensemble branches.

3. Attacking ECOC

We start by considering the basic C&W attack introduced in [6]. We notice that some of the good properties of C&W do not hold longer when the attack is applied against the ECOC scheme since it has been originally designed to work against networks adopting the one-hot-encoding strategy. Then, we propose a new more effective attack, which is specially tailored to multilabel structures like ECOC.

In general, constructing an adversarial example corresponds to finding a small perturbation δ (under some distance metric) that once added to image x will change its classification. Such a problem is usually formalised as

$$\begin{aligned} &\min \mathrm{D}(x, x+\delta), \\ &\text{s.t. } f(x+\delta) = t, \end{aligned} \tag{3}$$

where D is some distance metric (e.g. the L_2 metric) and t is a chosen target class. As this problem is difficult to solve, C&W attack aims at solving its Lagrangian approximation defined as

$$\min \|\delta\|_2 + \lambda \cdot \max\left(\max_{i \neq t} \left(z_i(x+\delta)\right) - z_t(x+\delta), c \right), \tag{4}$$

where the second term is any function such that $f(x+\delta) = t$ if and only if this term $\leq c$. $\|\cdot\|_2$ denotes the L_2-norm, λ and c are constant parameters ruling, respectively, the tradeoff between the two terms of the optimization problem and the confidence margin of the attack (In [6], $\delta = 1/2(\tanh(w) + 1) - x$, and the minimization is carried out over w to have box constraints on δ when optimizing equation (4) with a common optimizer like Adam.). Equation (4) is designed for the common one-hot encoding case. In fact, it is easy to see that for ECOC the motivation of such a design does not hold anymore and ensure that the second term is less than c and does not guarantee that $f(x+\delta) = t$. Therefore, the C&W attack must be adjusted to fit the ECOC framework. By noting that, in ECOC, correlations are proportional to probabilities (instead of the logits as with one-hot encoding), and C&W shall be modified as

$$\underset{\delta}{\text{minimize}} \|\delta\|_2 + \lambda \cdot \max\left(\max_{i \neq t} \left(\rho_i(x+\delta)\right) - \rho_t(x+\delta), c \right), \tag{5}$$

where $\rho_t(x+\delta) = \sigma(\mathbf{z}(x+\delta)) \cdot \mathbf{C}_t$.

A key advantage of C&W attack against one-hot-encoding networks is that it works directly at the logits level. In fact, logits are more sensitive to modifications of the input than the probability distribution obtained after the softmax activation (most adversarial attacks work directly on the probability values obtained after the softmax, which makes them less effective than C&W and prone to gradient-vanishing problems).

When C&W attack is applied against ECOC (by means of (5)), it does not work at the output logit level, but after that, the correlations are computed (mapping step 1) since this is the layer that precedes the application of the softmax-like function. The correlations between the activations of the logits and the codewords will likely have a reduced sensitivity to input modifications, and this may decrease the effectiveness of the attack. We also found that during the attack, it is possible to change only one bit of the output while the others are almost unchanged. This can be explained by observing that ECOC trains each output bit separately, so that each bit can be treated as an individual label. In this way, the correlation between the output bits is significantly reduced compared to classifiers adopting the one-hot encoding approach. We exploit this fact to design our attack in such a way as to make it modify a single bit at a

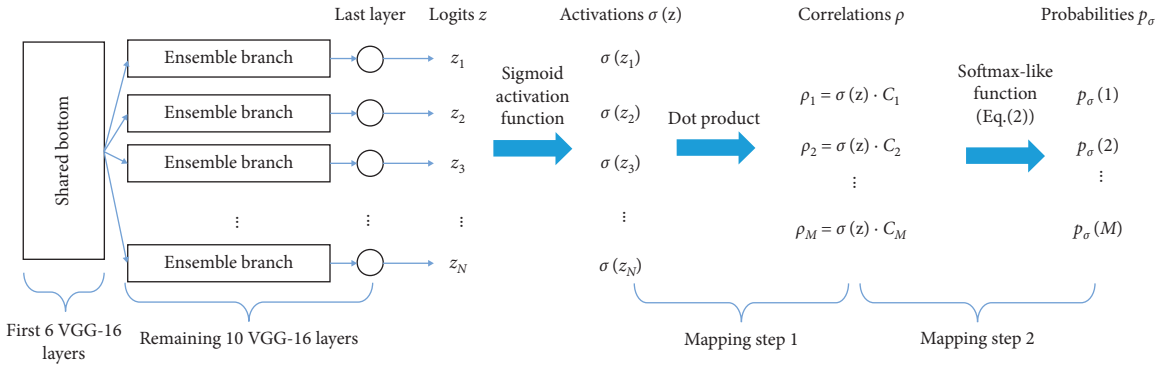


Figure 1: Block diagram of ECOC architecture.

time and iteratively repeat this process to eventually change multiple bits.

With the above ideas in mind, the new attack is formulated as follows:

$$\underset{\delta}{\text{minimize}} \|\delta\|_2 - \lambda \cdot \min_i (2t_i \cdot z_i(x+\delta), c), \tag{6}$$

where $(t_1, t_2, \cdots, t_N) = \mathbf{C}_t$ is the desired target codeword ($t_i \in \{-1, 1\}$), λ is a parameter controlling the tradeoff between the two terms of the objective function, and c is a constant parameter used to set a confidence threshold for the attack. Specifically, the attack seeks to minimize (6) until the product between t_i and z_i reaches this threshold; thus, a higher c will result in adversarial examples exhibiting a higher correlation with the target codeword, that is, adversarial examples that are (wrongly) classified with a higher confidence.

The choice of λ also plays an important role in the attack, given that a very small λ would lead to a vanishing perturbation. On the contrary, using larger λ results in a more effective attack at the cost of a larger perturbation. To optimize the value of λ, we use a binary search similar to the one used in [6] to determine the optimum value of λ in C&W attack. By doing so, the parameters of the proposed attack have the same meaning of those in C&W attack; thus, the two methods can be compared on a fair basis under the same parameter setting. An overall description of our attack is given in Algorithm 1, whose goal is to find a valid adversarial example, with the desired confidence level c and with the smallest perturbation. As a result of the optimization in Algorithm 1, all logit values z_i of the resulting adversarial image will tend to be highly correlated with t_i.

It is worth observing that, even if we designed the new attack explicitly targeting the ECOC classifier, the algorithm in (6) is generally applicable to any multilabel classification network since it manipulates the output bits of the network, regardless of the adopted coding strategy. This point can be evidenced by considering two limit cases of ECOC. In the first case, we avoid using error correction to encode the output classes. This is equivalent to multilabel classification problems with N labels [21, 22], and the proposed attack can still be applied. In the second case, we may consider one-hot-encoding as a particular way of encoding the output class. This perspective, also been considered in [15], would degrade the ECOC system to a common network that uses one-hot-encoding and softmax to solve a multiclass classification task. Since our attack does not involve the decoding part of the network, it can still be applied to such networks.

4. Experiments

4.1. Methodology. In [15], the authors tested the robustness of the ECOC architecture for various combinations of codeword matrices $\mathbf{C}$, activation functions $\sigma(\cdot)$, and network structures. In particular, they considered the MNIST [20] and CIFAR-10 [16] classification tasks ($M = 10$ in both cases). In the end, the best performing system was obtained by considering a Hadamard code with $N = 16$ and the tanh activation function. An ensemble of 4 ($N/4$) networks each one outputting 4 bits was considered. The authors argue that using a large number of ensembles increases the performance of the system against attacks (by decreasing the dependency among output bits). Then, in our experiments, we used N ensembles, with only one output bit each. The authors also indicate that the robustness of ECOC scheme can be improved by using longer codewords. Then, in our experiments, in addition to MNIST and CIFAR-10 already considered in [15], we also considered traffic sign classification (GTSRB dataset) [19], to test the robustness of ECOC on a larger number of classes and with codewords of a larger size, which potentially means higher robustness. To be specific, for traffic sign classification, we set $M = 32$, by selecting the classes with more examples among the total number of 44 classes in GTSRB, and chose a Hadamard code with $N = M = 32$, which is twice the size of the code used for MNIST and CIFAR-10. A diagram of the ECOC scheme with the N ensemble structure is shown in Figure 1. We used a standard VGG-16 network [23] as the base block of our implementation. Following the ECOC design scheme, the first 6 layers form the so-called *shared bottom* part, that is, the layers shared by all the networks of the ensemble. Then, the remaining 10 layers (the last 8 convolutional layers and the 2 fully connected layers) are trained separately for each ensemble branch.

For each task, we first trained one M-class classification network, and then we fine-tuned the weights to get the N

ensemble networks. The error rates of the trained models on clean images are equal to 2.14% for MNIST, 13.9% for CIFAR-10, and 1.28% for traffic sign (GTSRB) classification.

In addition to the extended C&W attack described in Section 3, we also considered a new attack named layerwise origin-target synthesis (LOTS) introduced in [18]. In a few words, LOTS aims at modifying the deep representation at a chosen attack layer, by minimizing the Euclidian distance between the deep features of the to-be-attacked input and a target deep representation chosen by the attacker. In our tests, we applied LOTS to the logits level, and we obtained the target deep representation (logits) by randomly choosing 50 images belonging to the target class.

4.2. Results. We attacked 300 images randomly chosen from the test set of each task. For each attack, we carried out a targeted attack with the target class chosen at random among the remaining $M - 1$ classes (i.e., all the M classes except the original class of the unperturbed image). The label t of the target class was used to run the C&W attack in equation (5) and LOTS, while the codeword $\mathbf{C}_t$ associated to t is considered in (6) for the new attack. We use the attack success rate (ASR) to measure the effectiveness of the attack, i.e., the percentage of generated adversarial examples that are assigned to the target class, and the peak signal-to-noise ratio (PSNR) to measure the distortion introduced by the attack, which is defined as $\text{PSNR} = 20^* \log_{10}(255^* \sqrt{N}/\|\delta\|_2)$, where $\|\delta\|_2$ is the L_2-norm of the perturbation and N is the size of the image.

As the parameters of the new attack have the same meaning as those of C&W attack, we first compare the C&W and the new attack with several settings of the input parameters. The results we got are shown in Tables 1–3, for GTSRB, CIFAR-10, and MNIST, respectively. In all the cases, c was set to 0. The results obtained by using the C&W attack against the standard one-hot-encoding VGG-16 network with M classes are also reported in the last column. By looking at the different rows, we can first see that when the strength of the attack is increased, e.g., by using a larger number of iterations or a larger number of steps during the binary search, the ASR of both attacks increases, at the price of a slightly larger distortion. For instance, for CIFAR-10, the ASR of the proposed attack increases from 69.3% to 98.6%, with a decrease in the PSNR of less than 1 dB, and the ASR of the C&W attack increases from 53.6% to 92.6% with an extra distortion of 3 dB. Then, by comparing different columns, we see a clear advantage of the proposed attack over C&W attack since the former achieves a higher ASR for the same parameter settings.

By comparing the different tables, we see that the advantage of the new attack is more evident with GTSRB than with CIFAR-10. The use of longer codewords in GTSRB, in fact, makes it harder to attack this classifier; however, the new attack can still achieve an ASR = 93.3% with a PSNR equal to 39 dB.

For MNIST dataset, the ASR is lower compared to the CIFAR-10 and GTSRB. This result agrees with the results reported in [15]. One possible explanation of this fact is advanced in [10] where the peculiarities of the MNIST dataset are highlighted and used to argue that high robustness can be easily reached on MNIST.

The comparison with LOTS must be carried on a different ground since such an attack is designed in a different way, and the only parameter shared with the new attack is the maximum number of iterations allowed in the gradient descent. For this reason, we applied LOTS by allowing a maximum number of iterations equal to 2000, which is the same number we have used for the other two attacks. We have verified experimentally that LOTS converges within 1000 iterations 92% of the times (The convergence is determinate by checking whether the new loss value is close enough to the average loss value of the last 10 iterations.), thus validating the adequacy of our choice. Then, we measured the ASR for a given maximum PSNR, thus allowing us to plot the ASR as a function of PSNR. The results we got are shown in Figure 2. Upon inspection of the figure, we observe a behavior similar to Tables 1–3. The proposed attack greatly outperforms LOTS and C&W on GTSRB when longer codewords are used. The ASR of the new attack, in fact, achieves nearly 100% for smaller PSNR's, while LOTS and C&W stop at 42% and 42.3%, respectively. For the other two datasets, the gap between the different attacks is smaller than in the GTSRB case. Specifically, the proposed attack and LOTS perform almost the same on CIFAR-10, while LOTS provides slightly better results on MNIST. This observation can also be verified in Figure 3, where we show some images that are successfully attacked by all the attacks. We can see from the figure that the proposed attack requires less distortion to attack the selected examples, producing images that look visually better than the others. The advantage is particularly evident for the GTSRB case, but is still visible for the CIFAR-10 and MNIST images.

As for time complexity, we observe that though our attack aims at modifying fewer bits each time, its complexity is very similar to that of C&W attack. Specifically, if we allow 2000 iterations (10 binary searches) for each attack, for CIFAR-10, the new attack and C&W attack require about 800 seconds and 1000 seconds to attack an image, respectively (The times are measured using one NVIDIA RTX2080 GPU without paralleling.). On the other hand, LOTS is considerably faster since it needs about 80 seconds to attack an image. The reason behind the high computational complexity of C&W and our new attack is the binary search carried out at each step. In fact, we verified that, by reducing the number of steps the binary search consists of, the speed of both attacks improves greatly. However, since our main purpose is to test the robustness of the ECOC system, we did not pay much effort to optimize our attack from a computational point of view, all the more that its complexity is already similar to that of C&W attack.

As an overall conclusion, the experimental analysis reveals that, in the white-box scenario, the security gain that can be achieved through the ECOC scheme is quite limited since by properly applying existing attacks and especially by using the newly proposed attack, the ECOC classifiers could be attacked quite easily.

Another expected advantage of ECOC is that adversarial examples tend to be classified with a lower probability than

Input:
The start point and number of binary search λ_1, n;
The step size and max iteration of gradient descent, ε, m;
To be attacked image, x;
Output:
Adversarial perturbation δ

upper bound $\longleftarrow \infty$
lower bound $\longleftarrow 0$
for $i \in [1, n]$ **do**
$\delta_1 \longleftarrow 0$
found adv $\longleftarrow$ False
  for $j \in [1, m]$ **do**
    if $x + \delta_j$ is adversarial and $\|\delta_j\| < \|\delta\|$ **then**
      found adv $\longleftarrow$ True
    end if
    $\delta_j \longleftarrow \delta_j - \varepsilon \times (\nabla_i / \|\nabla_i\|)$, where ∇_i is the gradient of equation (6) with current λ_i w.r.t. the perturbation δ_j
  end for
  if found adv = True **then**
    upper bound $\longleftarrow \lambda_i$
  else
      lower bound $\longleftarrow \lambda_i$
  end if
  if upper bound = ∞ **then**
      $\lambda_i \longleftarrow 10 \times \lambda_i$
  else
    $\lambda_i \longleftarrow$ (upperbound + lowerbound)/2
  end if
end for

Algorithm 1: Solving optimization in (6).

Table 1: Results of the attack against ECOC for GTSRB classification.

Parameters	Proposed (ECOC)		C&W (ECOC)		C&W (one-hot)	
	ASR (%)	PSNR	ASR (%)	PSNR	ASR (%)	PSNR
(1$e-4$, 5, 100, 0)	40.3	41.43	7.0	48.80	25.5	46.82
(1$e-4$, 5, 200, 0)	45.6	41.58	8.6	48.48	37.5	45.73
(1$e-4$, 5, 500, 0)	53.3	41.65	11.3	48.03	47.5	46.07
(1$e-2$, 5, 500, 0)	70.6	39.85	20.0	43.94	61	43.41
(1$e-1$, 10, 2000, 0)	93.3	38.97	42.3	39.20	81.5	42.51

Reported parameters indicate, respectively (start point, number of steps of binary search, max iterations, and confidence).

clean images. Here, we show that such a behavior can be inhibited, at the price of a slightly larger distortion, by increasing the confidence of the attack. If a larger confidence margin c is used, in fact, the model becomes more certain about the wrongly predicted class. To back such a claim, in Table 4, we report the results of the new attack for different confidence values c for the CIFAR-10 case (To clearly show the effect of confidence, we did not consider adversarial examples that do not reach the chosen confidence margin c, which leads to a slight drop of the ASR.). The table shows the average probability assigned by the ECOC model to the original class (Prob. true class) and to the target class of the attack (Prob. targ class), before and after the attack.

From the table, we see that, by increasing c, the adversarial examples are assigned higher and higher probabilities for the target class, getting closer to those of the benign samples. In particular, the average probability for the target class passes from 0.546 (with $c = 0$) to 0.993 (with $c = 5$), which is even higher than the average probability of the clean images before the attack (0.908), and the probability of the original (true) class decreases from 0.194 (with $c = 0$) to a value lower than 0.001 (with $c = 5$). A similar behavior can be observed for the C&W attack when c is raised from 0 to 15.

Figure 4 shows the distribution of the probabilities assigned to the most probable class for clean and adversarial images generated by the proposed attack. The plot confirms that the ECOC classifier assigns low probabilities only in the presence of adversarial examples obtained with a low confidence value c. When c grows, in fact, the probability

Table 2: Results of the attack against ECOC for CIFAR-10 classification.

Parameters	Proposed (ECOC)		C&W (ECOC)		C&W (one-hot)	
	ASR (%)	PSNR	ASR (%)	PSNR	ASR (%)	PSNR
$(1e-4, 5, 100, 0)$	69.3	38.59	53.6	39.71	92.5	40.03
$(1e-4, 5, 500, 0)$	88.0	38.52	62.3	39.94	100	40.27
$(1e-4, 10, 200, 0)$	90.6	37.84	79	37.32	100	40.18
$(1e-4, 10, 500, 0)$	95.0	38.39	82.6	37.55	100	40.30
$(1e-1, 10, 2000, 0)$	98.6	38.41	92.6	36.97	100	39.99

Reported parameters indicate, respectively (start point, number of steps of binary search, max iterations, confidence).

Table 3: Results of the attack against ECOC for MNIST classification.

Parameters	Proposed (ECOC)		C&W (ECOC)		C&W (one-hot)	
	ASR (%)	PSNR	ASR (%)	PSNR	ASR (%)	PSNR
$(1e-3, 10, 100, 0)$	29.3	21.26	26	21.19	1.5	32.48
$(1e-3, 10, 200, 0)$	43.6	21.49	35.6	20.73	8	27.69
$(1e-3, 10, 500, 0)$	55.6	21.91	43.6	20.37	40.5	24.29
$(1e-3, 10, 1000, 0)$	64.6	22.23	49	20.56	66.5	24.97
$(1e-1, 10, 2000, 0)$	72.3	22.35	57.6	20.35	78	25.29

Reported parameters indicate, respectively (start point, number of steps of binary search, max iterations, confidence).

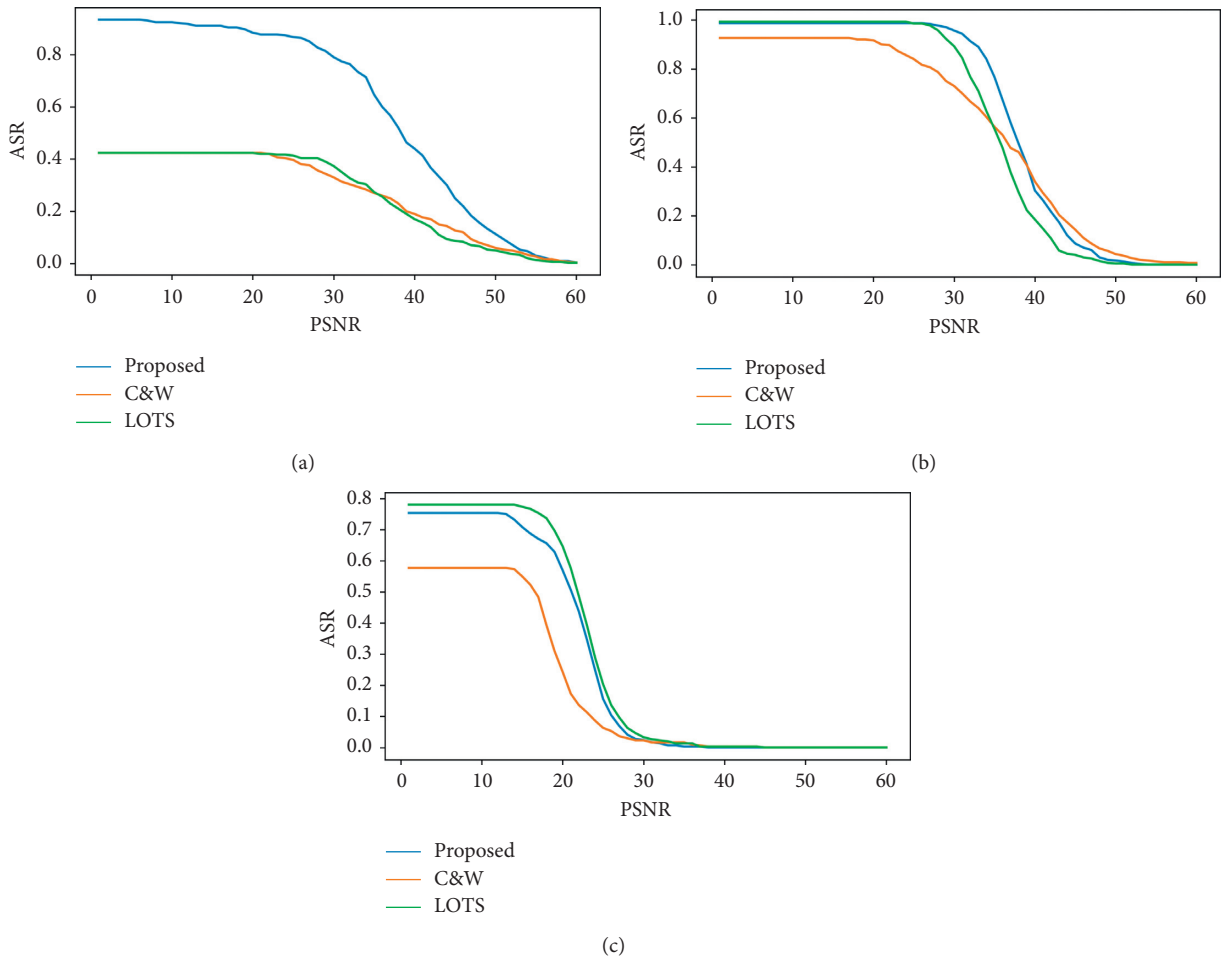


Figure 2: Performance of different attacks against the ECOC system. The x-axis indicates the PSNR(db) and y-axis indicates the ASR. (a) GTSRB. (b) CIFAR. (c) MNIST.

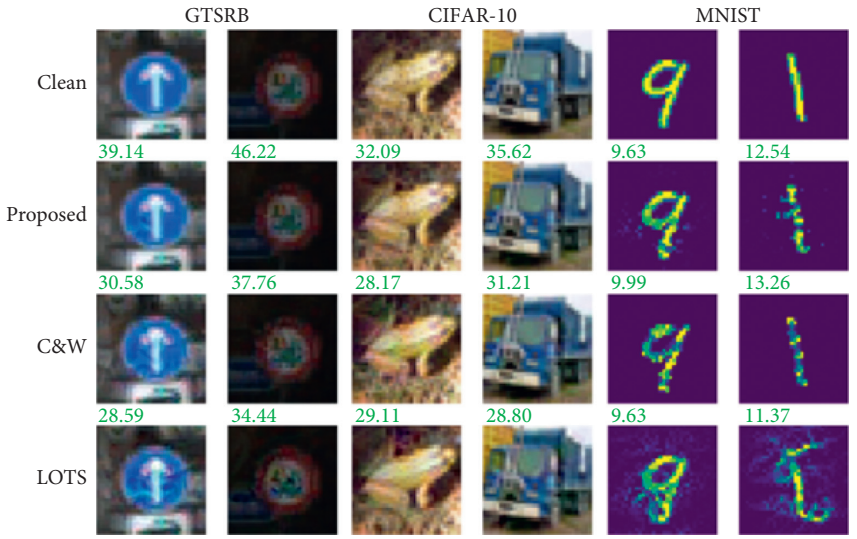


Figure 3: Examples of attacked images for different attacks. Besides the first row of clean images, the green number locates at the top left of each attacked image indicates its PSNR value.

Table 4: Output of probability values by the ECOC classifier on CIFAR-10 for different confidence margins of the attack.

C&W attack	$(1e-4, 5, 500, 0)$	$(1e-4, 5, 500, 8)$	$(1e-4, 5, 500, 12)$	$(1e-4, 5, 500, 14)$	$(1e-4, 5, 500, 15)$
ASR	62.3%	44.6%	44.6%	42.6%	42.3%
PSNR (dB)	39.94	40.78	39.57	39.00	38.40
Prob. true class	(B) 0.881 (A) 0.251	(B) 0.881 (A) 0.153	(B) 0.881 (A) 0.084	(B) 0.881 (A) 0.043	(B) 0.881 (A) 0.021
Prob. target class	(B) 0.013 (A) 0.328	(B) 0.013 (A) 0.534	(B) 0.013 (A) 0.721	(B) 0.013 (A) 0.843	(B) 0.013 (A) 0.914
Proposed attack	$(1e-4, 5, 500, 0)$	$(1e-4, 5, 500, 1.5)$	$(1e-4, 5, 500, 2.5)$	$(1e-4, 5, 500, 4.0)$	$(1e-4, 5, 500, 5.0)$
ASR	88.0%	87.6%	86.3%	85.1%	82.7%
PSNR (dB)	38.53	37.48	37.02	36.07	35.40
Prob. true class	(B) 0.908 (A) 0.194	(B) 0.908 (A) 0.063	(B) 0.908 (A) 0.024	(B) 0.908 (A) 0.005	(B) 0.908 (A) 0.001
Prob. target class	(B) 0.009 (A) 0.546	(B) 0.009 (A) 0.824	(B) 0.009 (A) 0.923	(B) 0.009 (A) 0.981	(B) 0.009 (A) 0.993

The parameters of the attacks are indicated according to the following format: (starting point, number of steps of binary search, max iterations, confidence). Prob. true and target class indicate the probabilities of the original (true) and target classes, before (B) and after (A) the attack.

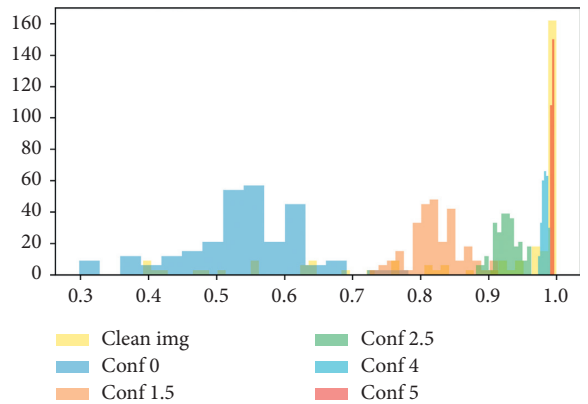


Figure 4: Distribution of probabilities assigned to the most probable class of the attacked examples with the proposed attack on CIFAR-10. The x-axis indicates the probability, and the y-axis indicates the number of examples classified with such a probability.

distribution of adversarial examples get closer and closer to that of clean images, and when $c = 5$, it becomes impossible to distinguish clean images and adversarial examples by setting a threshold on the probability assigned to the most probable class.

5. Related Works

Adversarial examples, i.e., small, often imperceptible, ad hoc perturbations of the input images, have been shown to be able to easily fool neural networks [1, 2, 5, 24] and have received great attention in the last years.

Different attacks have been proposed to obtain adversarial examples in various ways. Some works focus on diminishing the computational cost necessary to build the adversarial examples [2, 25], while others aim at lowering the perturbation introduced to generate the adversarial examples [1, 6, 26]. There are also some works whose goal is to find adversarial examples that modify only one pixel [27] and adversarial perturbations that can either fool several models at the same time [28], or can be applied to several clean images at the same time [4].

As a response to the threats posed by the existence of adversarial examples and by the ease with which they can be created, many defence mechanisms have also been proposed. According to [29, 30], defences can be roughly categorized into two branches, which either work in a reactive or proactive manner. The first class of defences is applied after the DNNs have been built. This class includes approaches exploiting randomization, like, for instance, stochastic activation pruning, in which node activations at each (or some) layers are randomly dropped out during the forward propagation pass [31], and, more recently, model switching [32], where random selection is performed between several trained submodels. Other approaches attempt to intentionally modify the network input to mitigate the adversarial perturbation, e.g., by projecting the input into a different space [33] or by applying some input transformations [34]. Other approaches attempt to reject input samples that exhibit an outlying behavior with respect to the unperturbed training data [35]. The second branch of defences aims at building more robust DNNs. One simple approach to improve the robustness against adversarial example is adversarial training, which consists in augmenting the training set with adversarial examples [10–12, 36]. More recently, as more attention has been paid to hidden layers with respect to the robustness of DNNs [37], it has been proven that rather than augmenting the training set, the robustness of DNNs can be improved by directly injecting adversarial noise into the hidden nodes, thus improving the robustness of single neurons [38, 39].

The ECOC scheme considered in this paper, belongs to the second class of defences and is derived from similar attempts made in the general machine learning literature to improve the robustness of multiclass classification problems [14, 40, 41]. The robustness of ECOC against adversarial examples is assessed in [15] by considering conventional adversarial attacks like [6, 42], which have not been explicitly designed for multilabel classification. As suggested in [17], however, in order to properly assess the effectiveness of a defence mechanism, the case of an informed attacker should be considered, and then the robustness should be evaluated against attacks targeting the specific defence mechanism. Following the spirit of [17], in this paper, we developed a targeted attack against the ECOC system, which exploits the multiclass and multilabel nature of such a system. We observe that the capability of ECOC to hinder the generation of adversarial examples has already been challenged in [17] (Section 10); however, the analysis in [17] is carried out under the more favourable (for the attacker) assumption of a nontargeted attack, thus marking a significant difference with respect to the current work [4, 43].

6. Conclusion

In order to investigate the effectiveness of ECOC-based deep learning architectures to hinder the generation of adversarial examples, we have proposed a new targeted attack explicitly thought to work with such architectures. We measured the validity of the new attack experimentally on three common classification tasks, namely, GTSRB, CIFAR-10, and MNIST. The results we have got show the effectiveness of the new attack and, most importantly, demonstrate that the use of error correction to code the output of a CNN classifier does not increase significantly the robustness against adversarial examples, even in the more challenging case of a targeted attack. In fact, the ECOC scheme can be fooled by introducing a small perturbation into the images, both with the new attack and, to a lesser extent, by applying C&W and LOTS attacks with a proper setting. No significant advantage in terms of decision confidence is observed as well, given that, by properly setting the parameters of the attack, adversarial examples are assigned to the wrong class with a high probability.

Data Availability

The data used to support the findings of this study are available from the first author (bowenzhang.psnl@outlook.com) upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the China Scholarship Council (CSC) (file no. 201806960079).

References

[1] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[3] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: a survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[4] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi et al., "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765–1773, Honolulu, HI, USA, July 2017.

[5] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, Las Vegas, NV, USA, June 2016.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, San Jose, CA, USA, May 2017.

[7] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, San Jose, CA, USA, May 2016.

[8] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," 2014, https://arxiv.org/abs/1412.5068.

[9] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195–204, 2018.

[10] F. Tramèr, A. Kurakin, N. Papernot et al., "Ensemble adversarial training: attacks and defenses," 2017, https://arxiv.org/abs/1705.07204.

[11] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016, https://arxiv.org/abs/1605.07725.

[12] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robust- ness of deep neural networks via stability training," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4480–4488, Las Vegas, NV, USA, June 2016.

[13] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, Dallas, TX, USA, November 2017.

[14] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1994.

[15] G. Verma and A. Swami, "Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks," *Advances in Neural Information Processing Systems*, pp. 8643–8653, 2019, http://papers.nips.cc/paper/9070-error-correcting-output-codes-improve-probability-estimation-and-adversarial-robustness-of-deep-neural-networks.

[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[17] F. Tramer, "On adaptive attacks to adversarial example defenses," 2020, https://arxiv.org/abs/2002.08347.

[18] A. Rozsa, M. Günther, and T. E. Boult, "LOTS about attacking deep features," in *Proceedings of the 2017 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 168–176, Denver, CO, USA, October 2017.

[19] 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.

[20] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits," 1998, https://yann.lecun.com/exdb/mnist/.

[21] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Multi-label vs. combined single-label sound event detection with deep neural networks," in *Proceedings of the 2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 2551–2555, Nice, France, 2015.

[22] M. L. Zhang and Z. H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[24] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, Boston, MA, USA, June 2015.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[26] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, Saarbrucken, Germany, March 2016.

[27] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[28] C. Xie, Z. Zhang, Y. Zhou et al., "Improving transferability of adversarial examples with input diversity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2730–2739, Long Beach, CA, USA, June 2019.

[29] B. Biggio and F. Roli, "Wild patterns: ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[30] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[31] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton et al., "Stochastic activation pruning for robust adversarial defense," 2017, https://arxiv.org/abs/1803.01442.

[32] X. Wang, "Block switching: a stochastic approach for deep learning security," 2020, https://arxiv.org/abs/2002.07920.

[33] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: adversarial training using generative models," 2017, https://arxiv.org/abs/1712.09196.

[34] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: detecting adversarial examples in deep neural networks," 2017, https://arxiv.org/abs/1704.01155.

[35] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: attribute-steered detection of adversarial samples," *Advances in Neural Information Processing Systems*, pp. 7717–7728, 2018, http://papers.nips.cc/paper/7998-attacks-meet-interpretability-attribute-steered-detection-of-adversarial-samples.

[36] H. Yu, A. Liu, X. Liu et al., "PDA: Progressive data augmentation for general robustness of deep neural networks," https://arxiv.org/abs/1909.04839v3.

[37] M. Cisse, P. Bojanowski, E. Grave et al., "Parseval networks: improving robustness to adversarial examples," *Proceedings of*

the 34th International Conference on Machine Learning, vol. 70, pp. 854–863, 2017.

[38] A. Liu, X. Liu, C. Zhang et al., "Training robust deep neural networks via adversarial noise propagation," https://arxiv.org/abs/1909.09034.

[39] C. Zhang, A. Liu, X. Liu et al., "Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity," https://arxiv.org/abs/1909.06978.

[40] N. Eghbali and G. A. Montazer, "Improving multiclass classification using neighborhood search in error correcting output codes," *Pattern Recognition Letters*, vol. 100, pp. 74–82, 2017.

[41] M. Lachaize, S. L. Hégarat-Mascle, E. Aldea, A. Maitrot, and R. Reynaud, "Evidential framework for error correcting output code classification," *Engineering Applications of Artificial Intelligence*, vol. 73, pp. 10–21, 2018.

[42] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[43] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples," 2018, https://arxiv.org/abs/1802.00420.

[44] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples," 2018, https://arxiv.org/abs/1802.00420.

[45] F. Tramer, "On adaptive attacks to adversarial example defenses," 2020, https://arxiv.org/abs/2002.08347.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8868672, 24 pages
https://doi.org/10.1155/2020/8868672

Research Article

Mimic Encryption Box for Network Multimedia Data Security

Xiabing Zhou,[1] Bin Li,[2] Yanrong Qi,[2] and Wanying Dong[2]

[1]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*

Correspondence should be addressed to Bin Li; cctvlibin@163.com

Received 19 August 2020; Revised 25 September 2020; Accepted 13 October 2020; Published 29 October 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Xiabing Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid development of the Internet, the security of network multimedia data has attracted increasingly more attention. The moving target defense (MTD) and cyber mimic defense (CMD) approaches provide a new way to solve this problem. To enhance the security of network multimedia data, this paper proposes a mimic encryption box for network multimedia data security. The mimic encryption box can directly access the network where the multimedia device is located, automatically complete the negotiation, provide safe and convenient encryption services, and effectively prevent network attacks. According to the principles of dynamization, diversification, and randomization, the mimic encryption box uses a reconfigurable encryption algorithm to encrypt network data and uses IP address hopping, port number hopping, protocol camouflage, and network channel change to increase the attack threshold. Second, the mimic encryption box has a built-in pseudorandom number generator and key management system, which can generate an initial random key and update the key with the hash value of the data packet to achieve "one packet, one key." Finally, through the cooperation of the ARM and the FPGA, an access control list can be used to filter illegal data and monitor the working status of the system in real time. If an abnormality is found, the feedback reconstruction mechanism is used to "clean" the FPGA to make it work normally again. The experimental results and analysis show that the mimic encryption box designed in this paper has high network encryption performance and can effectively prevent data leakage. At the same time, it provides a mimic security defense mechanism at multiple levels, which can effectively resist a variety of network attacks and has high security.

1. Introduction

With the rapid development of information, network, communication, image processing, and other technologies, the security requirements of network multimedia are becoming increasingly more urgent [1, 2]. To obtain economic benefits, occupy commercial competitive advantages, and achieve certain military needs, attackers usually use techniques such as reverse analysis, vulnerability attacks, and network sniffing to steal sensitive data, thereby triggering data security protection problems.

The life cycle of network multimedia data can be divided into four main parts: collection, transmission, storage, and processing. Regarding these functions, in order to protect the confidentiality of data transmission and storage and prevent the leakage of private data, it is necessary to encrypt the data [3]. However, the current network equipment and components have a large number of security vulnerabilities, allowing attackers to directly bypass the encryption link and steal the original data [4]. Second, security protection for massive amounts of multimedia data, including images, voices, and video, entails higher requirements on the efficiency of cryptographic algorithms. In the process of high-speed data transmission, traditional CPU encryption algorithms make it difficult to meet computing requirements [5]. Thus, a more efficient implementation of cryptographic algorithms is required in order to improve encryption speed. Finally, due to the staticity and similarity of traditional network components [6], such as the same hardware equipment, network protocol, fixed IP address, and port number, it is easy for an attacker to study the network operating rules, dig out security flaws, and conduct detection and continuous intrusion.

To improve the security of the multimedia network, the academic community has proposed a moving target defense (MTD) [7] technology and a cyber mimic defense (CMD) technology [8]. On the basis of MTD, CMD technology provides a dynamic heterogeneous redundancy (DHR) architecture, which uses the active transformation of functionally equivalent heterogeneous executive bodies to change the components of the information system, therefore realizing network, platform, environment, software, structure, and data dynamic changes or migrations. For attackers, it is difficult to observe and predict the target changes; therefore, system security risks are greatly reduced. At the same time, CMD introduces a negative feedback mechanism. According to the configured negative feedback control strategy, if the current system is found to be abnormal, the system will be cleaned by self-reorganization and reconstruction. In addition, new functionally equivalent heterogeneous executors will be randomly selected "online." Obviously, CMD not only greatly increases the difficulty and cost of attacks but also can detect the attack behavior of successful intrusions in real time.

Among various network security defense measures, the security of the hardware structure and the operating system is the foundation, and cryptography is the key technology. In this paper, combining the concepts of MTD and CMD, using the FPGA as the hardware platform, a mimic encryption box is designed. This device provides dynamic encryption, network structure transformation, and a feedback reconstruction mechanism in order to realize the security protection of multimedia data.

Our main contributions in this paper can be outlined as follows:

(1) A mimic encryption box with dynamics, diversity, and randomness is designed and will be able to directly connect to the network where the multimedia device is located and provide secure encrypted transmission of data.

(2) The TCP/IP layer data encryption and variable network configuration are realized on an FPGA, and system management and a feedback mechanism are realized on an ARM. In this system, software and hardware work cooperatively.

(3) The reconfigurable underlying cryptographic algorithm, pseudorandom number generator, packet filtering, and storage functions are optimized on the FPGA, and the programmability and high anti-interference of the FPGA make it difficult for attackers to establish a continuous and reliable attack chain.

(4) The hash value of the packet is used as the key to achieve "one packet, one key," and autonegotiation is used to change the IP address, port number, and protocol type in order to achieve dynamic network transformation. Through cooperation between the ARM and the FPGA, abnormal condition-related "cleaning" and the reconfiguration of FPGA are achieved.

(5) The encryption and decryption performance, network performance, security, and antiattack of the mimic encryption box are evaluated from several aspects.

The remainder of this paper is organized as follows: Section 2 introduces the existing network multimedia protection technologies. Section 3 describes the design of the mimic encryption box in detail. Section 4 analyzes and evaluates the mimic encryption box from multiple perspectives. Section 5 discusses the applicability and limitations of this method. Finally, Section 6 concludes this paper.

2. Related Work

Regarding MTD research, Aydeger et al. [9] analyzed crossfire attack planning and utilized the analyzed results to develop the defense mechanism that in turn reorganizes the routes in such a way that the congested links are avoided during packet forwarding. In addition, for use when implementing MTD mechanisms via route mutation, Aydeger et al. [10] proposed various virtual shadow networks created through network functions virtualization (NFV), which can dynamically change the routes for specific reconnaissance packets so that attackers will not be able to easily identify the actual network topologies. Zeitz et al. [11] explored the uses of a micromoving target IPv6 defense (μMT6D). The μMT6D is designed to work on low-power and low-resource devices and can prevent targeted attacks through rotating the IPv6 address. Wang et al. [12] proposed a network defense method based on random domain name and address mutation (RDAM). This method increases the scanning space of the attacker through a dynamic domain name method and reduces the probability that a host will be hit by an attacker scanning IP addresses. Zheng and Namin [13] presented the results of an analysis performed on simulating a simplified attack scenario against hosts on a network. They investigated the influence of the host IP address change rate and host complexity on the success rate of attacks. Hong and Kim [14] incorporated MTD techniques into a security model, namely, a hierarchical attack representation model (HARM), in order to assess the effectiveness of the techniques. Zhao et al. [15] proposed an SDN-based double hopping communication (DHC) approach. The DHC approach is able to increase the overhead of a sniffer attack, as well as the difficulty of communication data recovery. Jafarian et al. [16] proposed random host address mutation (RHM), which uses hierarchical fast transitions based on the address space and the IP address in order to distort attacker reconnaissance and deter attacks. On this basis, Jafarian et al. [17] proposed the proactive-adaptive defense technique, which monitors an attacker's behavior in real time and performs active address hopping, thus significantly raising the bar against stealthy scanning.

In terms of CMD, Hu et al. [18] designed and implemented a mimic network operating system (MNOS), an active defense architecture based on mimic security defense, in order to ensure SDN control plane security. This

architecture can effectively reduce the probability of successful attack and has good fault tolerance. Then, Hu et al. [19] introduced the mimic defense (MD) framework and detailed the "dynamic, heterogeneity, and redundancy" core mechanism. Their results showed that MD can significantly increase the difficulty faced by attackers and enhance the security of cyber systems. Ma and Zhang [20] described the mimic defense system formally and, through results from Monte Carlo simulations, analyzed the security effects of redundancy in mimic defense systems. Qi et al. [21] proposed Mcad-SA, an aware decision-making security architecture with multiple controllers, which exploits heterogeneity and redundancy from different controllers to prevent an attack proactively. Based on the mimic defense theory and technology, Liu et al. [22] proposed a framework against zero-day attacks. To protect the security of distributed storage systems, Li et al. [23] presented a storage architecture for mimic defense (SAMD). This architecture adopts a heterogeneous multirandom coding defense mechanism to actively and dynamically defend against indeterminate attacks.

In terms of network encryption, Abusukhon et al. [24] focused on data encryption techniques and proposed a new method for data encryption based on encrypting the plain text into a white page image. In addition, Abusukhon et al. [25] proposed a Diffie–Hellman text-to-image encryption algorithm (DHTTIE), adding a new security level to the TTIE algorithm. Tang et al. [26] proposed a dynamic three-layer encryption scheme based on DES and network coding, with a low-complexity partial key update mechanism, which increases its adaptability to various cyber conditions. Khan et al. [27] further reduced the cost of a network coding mechanism by reducing the size of data used for permutation. They also proposed an algorithm for key generation and random permutation confusion key calculation. Jiang et al. [28] presented a compromising method to take both the security level and the speed of data transmission into account by means of mixing the RSA and DES algorithms. In addition, the security interceptor of Spring Security was extended, and a series of security filters were added to keep Web attackers away. Li et al. [29] proposed a new attribute-based data-sharing scheme suitable for resource-limited mobile users in cloud computing. For the sake of data security, a Chameleon hash function was used against adaptive chosen-ciphertext attacks. Jia et al. [30] designed an identity-based anonymous authenticated key agreement (AAKA) protocol for the mobile edge computing environment. This protocol achieves mutual authentication in only a single message exchange round and assures both user anonymity and untraceability. Indira et al. [31] proposed a standard two-phase implementation of round key- and random key-based cryptosecurity encryption standard (R2R-CSES) for improving security system in a cloud environment.

In many security applications, a variety of defense technologies were realized based on FPGA. Maciel et al. [32] proposed a high-performance and energy-efficient reconfigurable FPGA-based K-means/K-modes architecture for network intrusion detection. Joseph et al. [33] studied the efficiency of an intrusion detection system by using an FPGA with a string-matching system design and a predecoder finite state machine for use in the high-speed network intrusion detection system. Lin et al. [34] introduced the design of a Gigabit Ethernet firewall based on FPGA. The FPGA functions were implemented to achieve legitimacy in network packet inspection and for internal network protection. Keni and Mande [35] used the highly parallelized structure of FPGA to form a rule set for allowing incoming and outgoing IP addresses to filter the IPv4 protocol. Ricart-Sanchez et al. [36] proposed a fully functional, FPGA-based 5G firewall that is capable of effectively detecting cyber-attacks in 5G multitenant scenarios with user mobility support.

In summary, although the use of IP address, port number hopping, and other mechanisms increases the difficulty of the attack to some extent, if plaintext transmission is used, the attacker can still obtain useful information. Second, simple network data encryption cannot resist key exhaustive attacks and ciphertext-only attacks. Third, FPGA is a suitable and popular hardware platform for many network security applications. In addition to being used in firewalls and packet inspection, it can also be used in MTD and CMD. The combination of ARM and FPGA can be used to expand the attack surface and improve security. Therefore, this paper combines MTD, CMD, network encryption, and hardware protection technologies and proposes a dynamically reconfigurable mimic encryption box to solve the abovementioned problems.

3. Design Method of the Mimic Encryption Box

3.1. Overall Structure. The mimic encryption box is located between the network multimedia device and the user. It provides secure communication services to the user by binding the terminal multimedia device. The system architecture is shown in Figure 1. In the architecture, there are management and feedback modules running on an ARM. These modules mainly communicate with the key management center to complete the following: the initialization of the parameters, key distribution, the generation of FPGA status statistics, self-reconstruction cleaning, and the dynamic loading of bitstreams. In addition, an FPGA is mainly used for the realization of core cryptographic algorithms, including hash algorithms, symmetric encryption algorithms, and asymmetric encryption algorithms. By using the reconfigurable ability of the FPGA, various keys are generated pseudorandomly, and cryptographic algorithms are dynamically invoked to complete efficient data encryption and decryption processing. Second, through the access control list and memory management, the rule filtering and storage of data are realized. Finally, multiple 10G and 1G communication network ports are integrated on the FPGA, and the network ports and channels are dynamically switched according to the configuration of the ARM, making full use of the flexibility and scalability of the FPGA to confuse attackers and prevent attacks such as network sniffing.

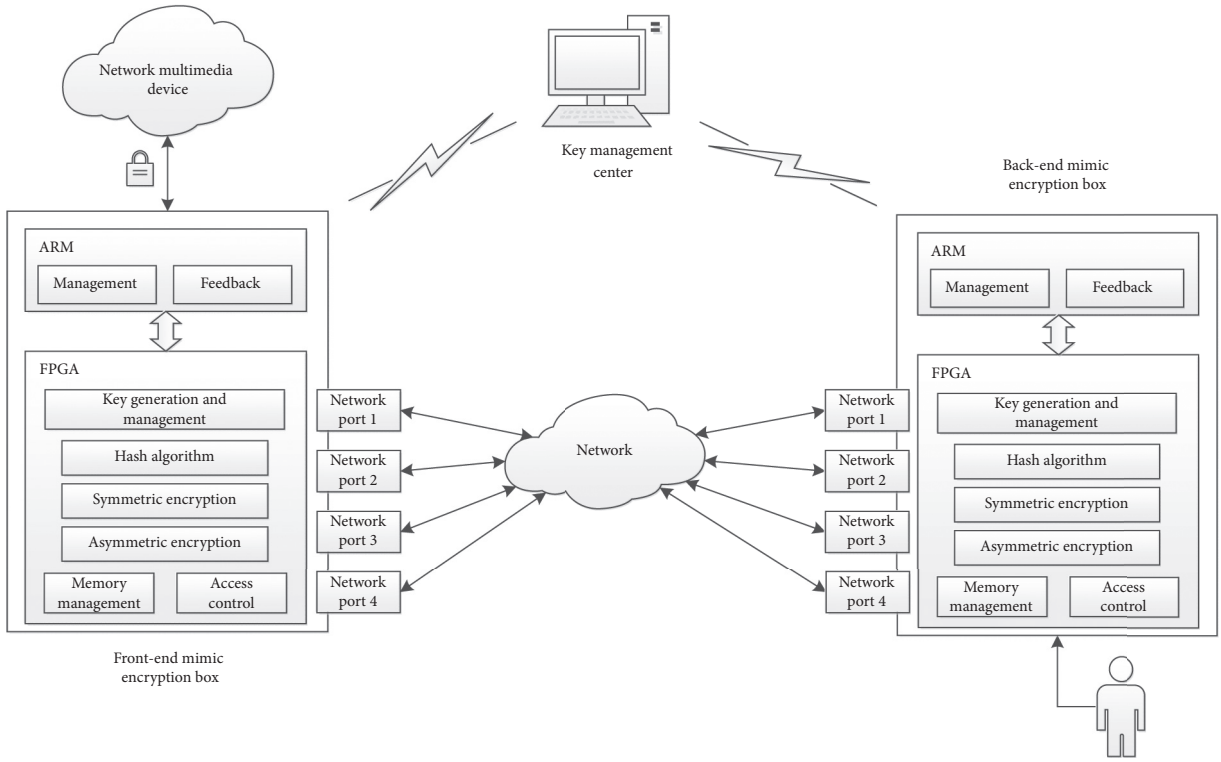


Figure 1: Mimic encryption box system architecture.

3.2. Work Flow. Currently, network multimedia devices mostly work at the TCP/IP layer in order to provide stable and reliable data transmission. Therefore, in this paper, mainly TCP/IP data encryption and camouflage is completed. As shown in Figure 2, the comparison before and after TCP/IP packet encryption is given. Here, the IP address, protocol, and port number are hopped and disguised. Then, a 1-byte PID field and a 4-byte CRC32 field are added: the PID is used to address the key, and the CRC32 is used to calculate the TCP checksum. Finally, the other TCP fields, the data, the pad, and the CRC32 are encrypted.

The mimic encryption box completes the packet encryption based on the principles of randomization, dynamization, and diversification. As shown in Figure 3, it is mainly composed of the MAC layer interface, packet parsing, access control, encryption and decryption algorithms, key generation management, packet encapsulation, memory, arbitration, state information collection, parameter initialization, and the ARM system. Among the components, through filtering rules, access control manages the plaintext path, the ciphertext path, discarding, and other processing and forwarding to the ARM. At the same time, in the front-end mimic encryption box, network port F is used to connect to the network multimedia device, and network port B is connected to the external network.

The encryption process of the entire system is as follows:

(1) The key management center establishes a secure tunnel with the ARM of both communicating parties to obtain the mimic encryption box information and distribute the initial information and working key.

(2) The communication parties negotiate to complete the dynamic configuration of the IP address, port number, protocol type, encryption algorithm, hash algorithm, and network interface, and the system parameters are initiated by the ARM. Then, the initial encryption key is generated by the pseudo-random number generator.

(3) After network port F receives the packet, it parses the packet and forwards it to the corresponding processing channel by the filtering rules.

(4) In the encryption processing, the PID and CRC32 fields are added to the TCP/IP packet, and the corresponding initial key for the encryption is selected. Then, a hash operation on the first 64 bytes of the encrypted packet is performed to generate a new key, and the key corresponding to the serial number is updated.

(5) The negotiated IP address, port number, and protocol type are used to disguise and encapsulate the original packet.

(6) The encrypted packets, plaintext packets, and ARM packets are cached in memory and, then, sent out from network port B.

(7) Network port B receives the packet from the external network, parses it, and forwards it to the corresponding processing channel according to the filtering rules.

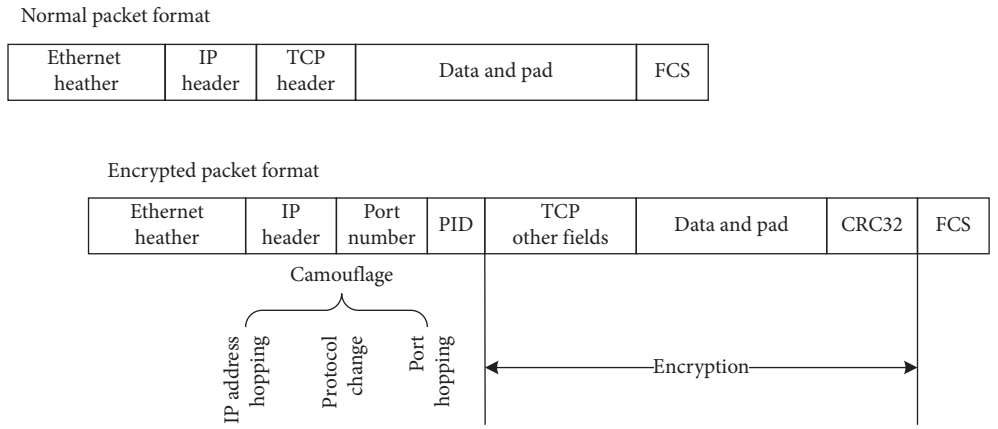


Figure 2: Comparison of normal and encrypted packet formats.

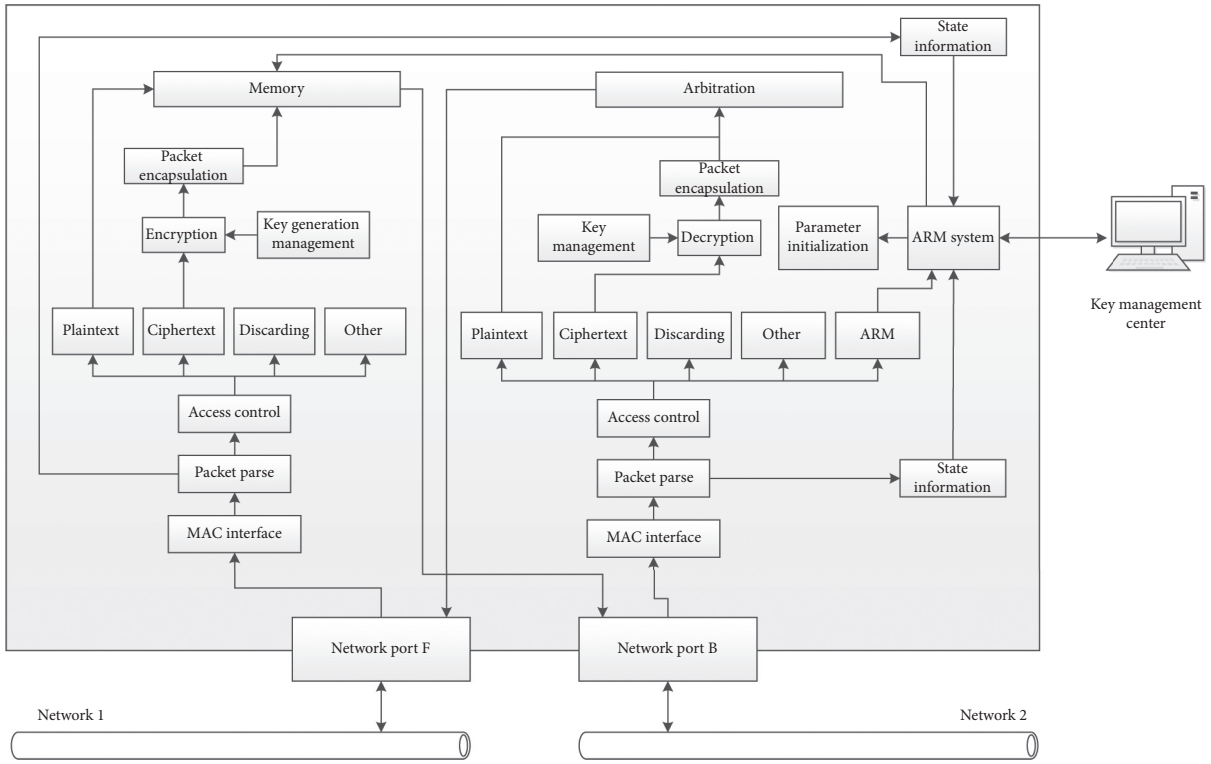


Figure 3: Front-end mimic encryption box structure.

(8) In the decryption processing, the key corresponding to the PID field is selected in order to decrypt and check the CRC32, and at the same time, the corresponding key from the hash value of the first 64 bytes of the current encrypted packet is directly updated.

(9) The camouflage is removed, the original IP address, port number, protocol type is restored, and the packet is encapsulated.

(10) In the ARM path, the data are transmitted to the ARM system, and the ARM system hands them over to the key management center to complete the corresponding processing.

(11) Then, the arbitration module is used to directly send the decrypted and encapsulated packet to the network multimedia device.

In the process of communication between the two sides, the packet information of network port F and B is obtained in real time, and after being collected by the ARM system, it is sent to the key management center to realize the real-time monitoring of both sides.

For the back-end mimic encryption box, the functional modules are similar to the front-end, and the workflow is basically the same, but the network connection method is slightly different. Specifically, network port F is connected to the external network to provide the decryption channel and

ARM management; network port B is connected to the user and provides an encrypted channel.

3.3. Optimization of Core Encryption Algorithm

3.3.1. Hash Algorithm. A hash algorithm is an irreversible one-way function that can output any length of data. At present, the commonly used hash algorithms are MD5, SHA1, SHA256, SHA512, RIPEMD160, and SHA3. These hash functions are all based on logical operations. The data are filled, grouped, and then, iteratively compressed by the round function, and the result is generated after n rounds of calculation. Since the structure of each round is similar, it can be implemented in a full-pipeline parallel manner.

Here, according to the number of iterations of the hash algorithm, all the loops are expanded [37, 38] to form a full-pipeline structure. When it is working at full capacity, in the overall pipeline, each clock cycle can calculate a set of hash values. Then, precalculation and carry-save adders (CSA) are used to optimize and reconstruct the round function to reduce the critical path delay.

We take SHA1 as an example. The SHA1 algorithm fills the initial information into 512 bits and initializes it to 16 32-bit groups $w[15:0]$. Let $A = 0x67452301$, $B = 0xEFCDAB89$, $C = 0x98BADCFE$, $D = 0x10325476$, and $E = 0xC3D2E1F0$ be the initial link variables, and let a, b, c, d, and e be 5 intermediate variables used to perform 80 rounds of iterative computations. Each iteration is performed as follows:

$$\begin{aligned} a &= a\_\text{next};\\ b &= a;\\ c &= c\_\text{next};\\ d &= c;\\ e &= d,\\ a\_\text{next} &= \{a[26:0], a[31:27]\} + f_t + e + k_t + w_t;\\ c\_\text{next} &= \{b[1:0], b[31:2]\}, \end{aligned} \tag{1}$$

where f_t is a nonlinear function, k_t is a constant, and w_t is a grouped data block.

Finally, the 160-bit hash value cascaded output is given by $a = a + A$; $b = b + B$; $c = c + C$; $d = d + D$; and $e = e + E$.

Clearly, b, c, d, and e can be obtained directly via value passing, while a requires complex operations, and the delay consumption is concentrated along the critical path of a. For the FPGA, the delay of addition is much larger than the bit operation. Therefore, to reduce the use of adders, we define the CSA as follows:

$$\begin{aligned} \text{CSA}(x, y, z) = {} & \left(x \hat{} y \hat{} z\right) + (((x\&y)|(x\&z)|(y\&z)) \\ & \ll 1) = x + y + z. \end{aligned} \tag{2}$$

At the same time, a variable g is introduced for precalculation, as follows: $g = \text{CSA}(d, k_{t+1}, w_{t+1})$. That is, we use the result d of the next round e and k_{t+1}, w_{t+1} to calculate g in advance. Then, the calculation of a can be simplified as

$$a\_\text{next} = \text{CSA}\left(\{a[26:0], a[31:27]\}, f_t(b, c, d), g\right). \tag{3}$$

Figure 4 shows an optimized round function structure.

Then, a unit kernel is built with round functions, the kernels are interconnected by registers, and all kernels are executed in parallel. The overall structure of the hash algorithm is shown in Figure 5.

Comprising FPGA storage resources, the registers are widely distributed. BRAM is located in a fixed area, and mixed storage is adopted, enabling the full utilization of FPGA resources and effectively shortening the critical path delay. Therefore, in Figure 5, the initialization variables are stored by using Shift RAM. Due to its small scale and scattered values, the constant list uses a direct assignment strategy. For grouped data blocks, a two-dimensional register array is used, values are assigned through circular shifts in one-dimensional space, and values are transferred through a register copy in two-dimensional space, thereby realizing data multiplexing and reducing data overlap. For the output of the results, to achieve a balance between resources and performance, data cascading is used, as it is more conducive to constraining the data concentration in a logical area.

In addition, by reconstructing the round functions of different hash algorithms and using units and hybrid storage methods, a hash algorithm with higher performance and better expansibility can be formed to meet the needs of various encryption computing.

3.3.2. Symmetric Encryption Algorithm. The current mainstream encryption algorithms are 3DES, AES128, AES256, Twofish, and Serpent. The encryption scheme based on the FPGA chip level is fast, safe, and of low cost. As the FPGA technology is reconfigurable, through user programming to change the circuit structure on the chip, different encryption and decryption algorithms can be realized. In this paper, AES128 is taken as an example. Based on the idea of reconfigurability, the cryptographic algorithm is modularized, providing the encryption and decryption architecture of AES.

The main modules of AES include KeyExpansion, AddRoundKey, SubBytes, ShiftRows, and MixColumns. The SubBytes module enables the realization of the obfuscation principle, and ShiftRows and MixColumns modules mainly allow realization of the diffusion principle [39, 40]. The AES decryption algorithm key selection sequence is exactly the opposite of encryption, and the key expansion process is irreversible. Therefore, to ensure that encryption and decryption have the same execution cycle, all round keys need to be generated at once. The AES structure is shown in Figure 6.

The AES single-round encryption process is shown in Figure 7. Each round of operations is compressed to 1 clock cycle in a cascaded manner. The entire encryption process requires 23 clocks, of which the key expansion is 11 clocks and encryption and decryption is 12 clocks. In addition, compared with the LUT method, the method in which the BRAM embedded in the FPGA is used to realize the storage

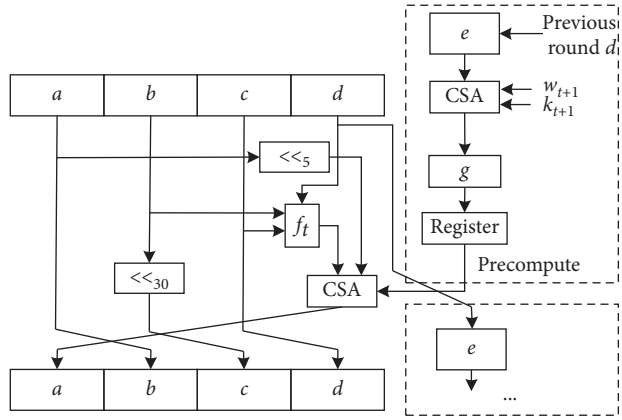


Figure 4: SHA1 single iteration structure.

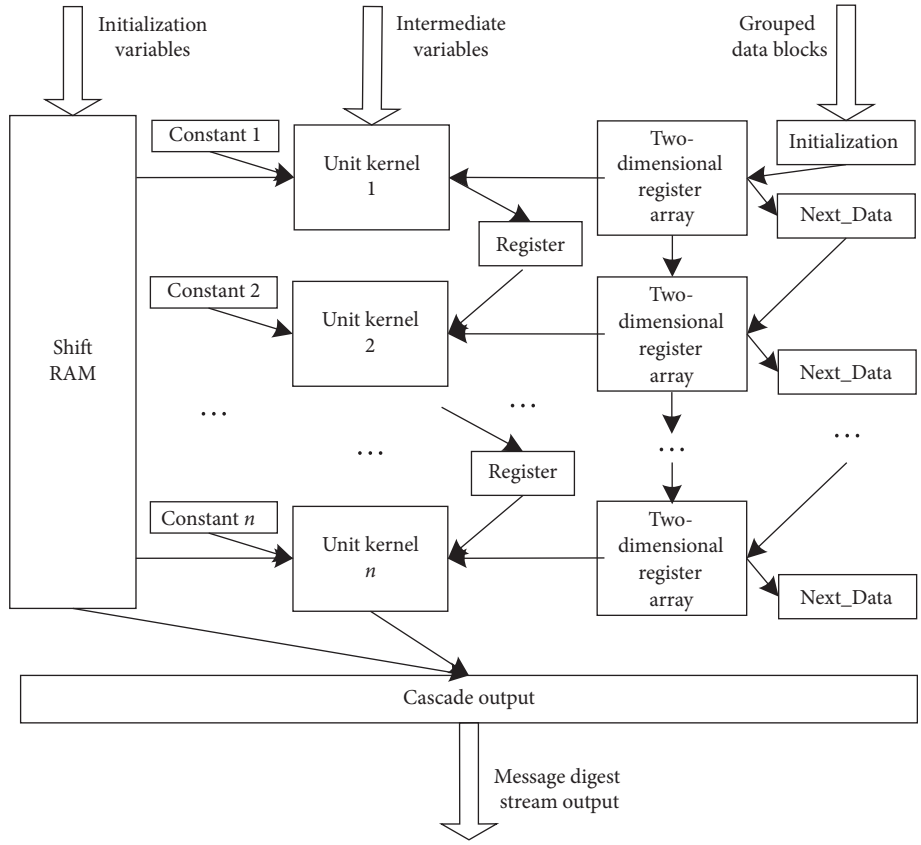


Figure 5: Full-pipeline architecture of the hash algorithm.

of the Sbox is better, as this method can reduce the occupation of resources and increase the routing frequency.

Finally, symmetric encryption algorithms have four commonly used encryption modes: ECB, CBC, CFB, and OFB. Among them, in ECB and CBC, the data are divided into blocks and padding operations are performed, while in CFB and OFB, change block ciphers can be changed into self-synchronized stream ciphers. Therefore, to pad the last data block, the CFB and OFB encryption methods are used. In addition, the previous block of encrypted data are regarded as the IV (initial vector), and the current data are XORed with IV after calculation and output.

3.3.3. Elliptic Curve Algorithm. In the prime domain, the description of the elliptic curve $E(F_p)$ is as follows: $y^2 = x^3 + ax + b \pmod{p}$ [41]. The security of the elliptic curve cryptosystem is mainly based on the difficulty of point multiplication inversion. Point multiplication, also called a multiple point operation, refers to the multiplication

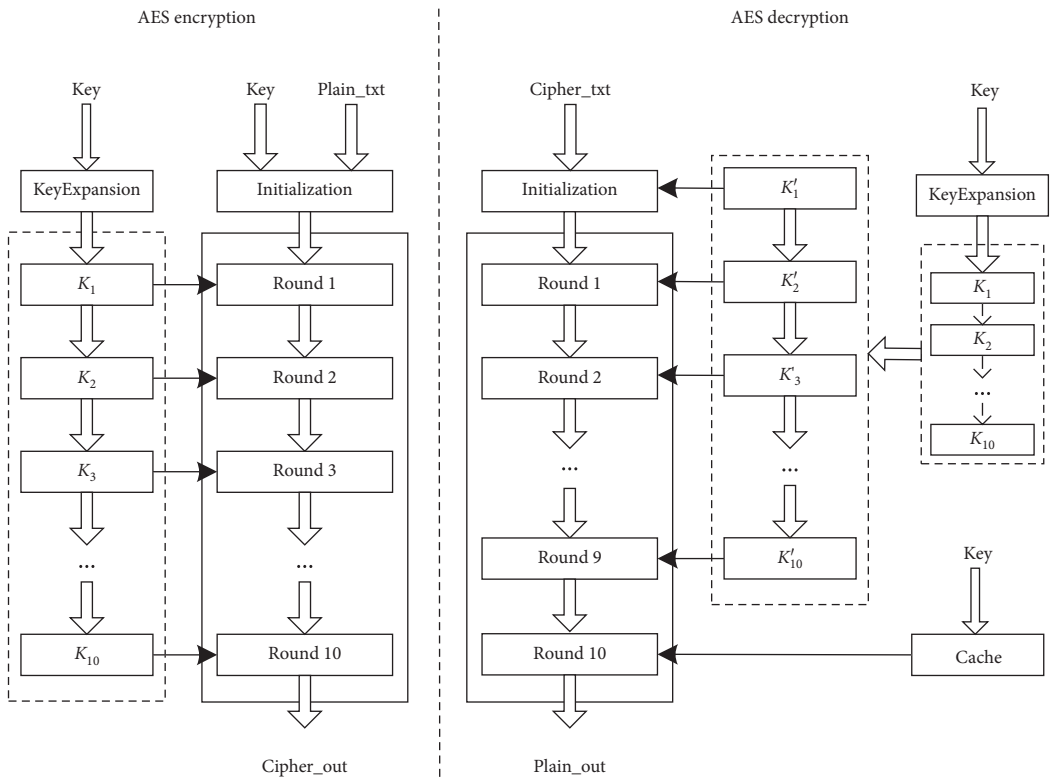


Figure 6: AES encryption and decryption structure.

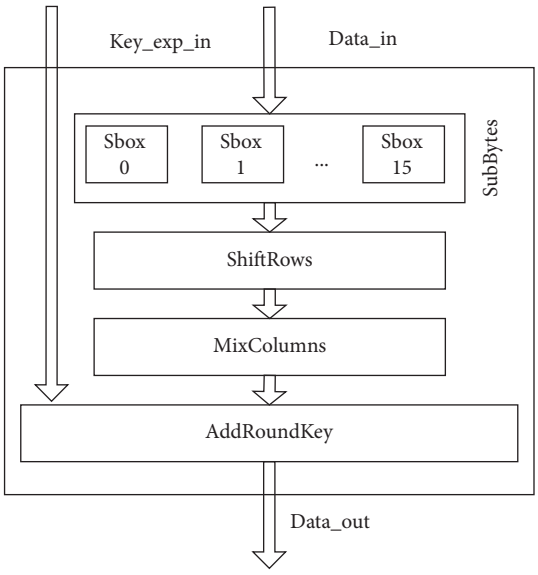


Figure 7: AES single-round encryption structure.

operation of a base point P on the curve with an integer k, that is, the addition of P for k times:

$$kP = \sum_{1}^{k} P = P + P+, \ldots, +P. \quad (4)$$

The point multiplication process includes double point and point addition operations; therefore, the optimization of double point and point addition is an important way to improve the efficiency of elliptic curve calculation.

Here, the Karatsuba–Ofman algorithm (KOA), fast modular reduction, radix-4 modular inversion, Montgomery point multiplication, and other optimization methods are combined to achieve an energy-efficient and antiattack ECC algorithm. Through the bottom-up design method, the most basic operations are realized with modular addition and subtraction, modular reduction, modular inversion, and modular multiplication, and then, the point multiplication operation is optimized by point addition and double point. Finally, the functions of elliptic curve signature, verification, encryption and decryption, and key agreement are realized. The overall structure is shown in Figure 8.

In the point multiplication calculation, the point addition and double point operations will be called many times, while the coordinate transformation is only calculated once, i.e., at the last time. Therefore, the point addition and double point operations are optimized by performance optimization, while the coordinate transformation operation is optimized by resource optimization. Second, the master state machine is used to schedule and manage the point multiplication module in order to meet the calculation requirements of different functions. Finally, the reuse of resources is realized through the coordinate transformation operation's shared modular addition and subtraction modules, and to reduce the consumption of FPGA resources, the data transmission is completed through an asynchronous FIFO method.

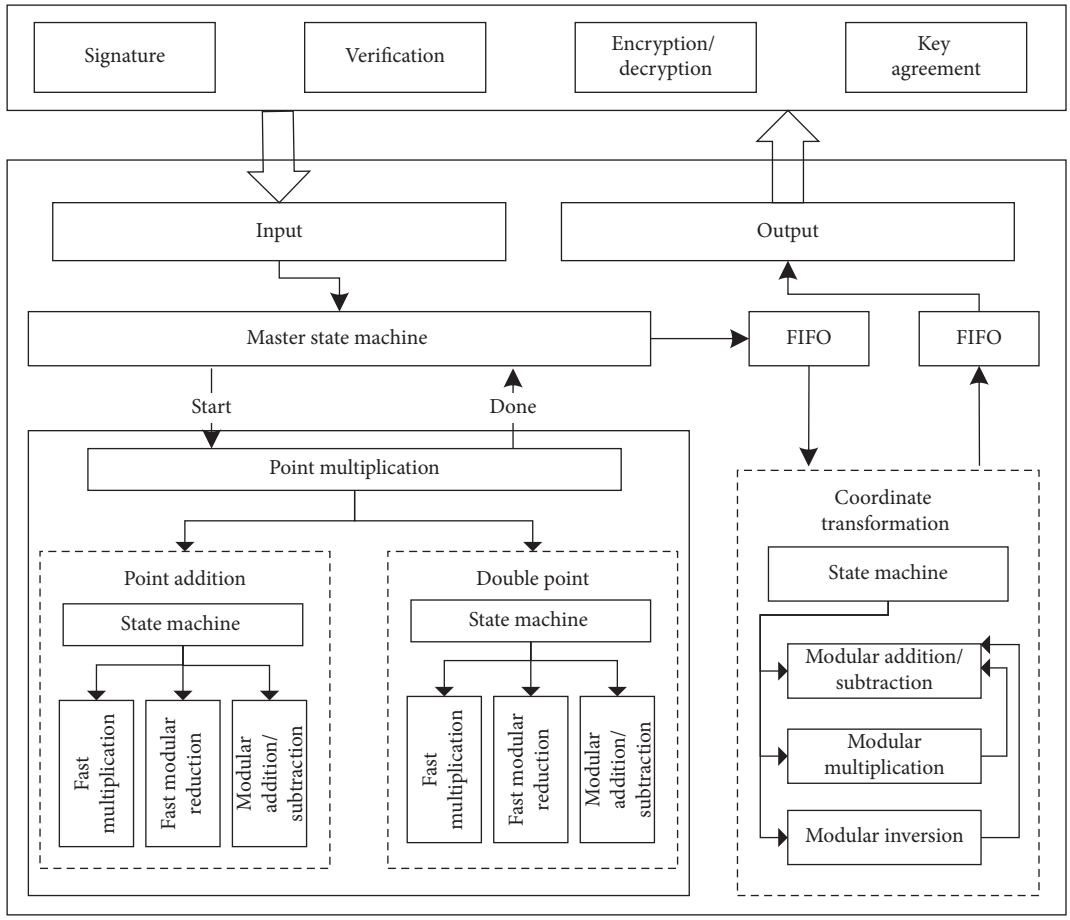


Figure 8: ECC algorithm overall architecture.

(1) KOA Fast Multiplication. The core idea of KOA [42] is "divide and conquer," i.e., a calculation approach in which a complex multiplication operation is decomposed into multiple simple multiplication operations by recursion. It is faster and more efficient than traditional calculation. For two n-bit numbers, if they are directly multiplied, the complexity is $O(n^2)$, and the complexity can be reduced to $O(n^{\log_2 3})$ by using KOA.

For the n-bit A, this can be expressed as $A = (\underbrace{\alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_{n/2}}_{A^H}, \underbrace{\alpha_{n/2-1}, \ldots, \alpha_0}_{A^L})$, where $\alpha_i \in \{0, 1\}$, $0 \le i < n$.

Then, A and B can be expressed equivalently as follows:

$$\begin{aligned} A &= A^H \times 2^{n/2} + A^L, \\ B &= B^H \times 2^{n/2} + B^L. \end{aligned} \tag{5}$$

Therefore, $C = A \times B$ can be calculated by the following formula:

$$\begin{aligned} C &= \left(A^H \times 2^{n/2} + A^L\right) \times \left(B^H \times 2^{n/2} + B^L\right) \\ &= z_2 \times 2^n + z_1 \times 2^{n/2} + z_0, \end{aligned} \tag{6}$$

where $z_2 = A^H \times B^H$, $z_1 = A^H \times B^L + A^L \times B^H$, and $z_0 = A^L \times B^L$.

The ECC parameter is 256 bits, while FPGA DSP supports multiplication operations with a maximum bitwidth of 64 bits. Then, 256 bits can be divided into 128 bits, and then, 128 bits can be divided into 64 bits. After two recursive operations, the result is obtained, as shown in Algorithm 1.

(2) Fast Modular Reduction. For fast modular reduction, the ECC special parameters can be optimized. Because of the particularity of prime number p, several addition and subtraction operations can be used to obtain the result of modular reduction. Compared with the current universal Montgomery algorithm, the fast modular reduction algorithm can save several 256-bit multiplication operations and significantly improve the performance.

When $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, for a large number A: $A = A_{15} \times 2^{480} + A_{14} \times 2^{448} + \ldots + A_1 \times 2^{32} + A_0$, each A_i is a 32 bit integer; then, A can be expressed as follows: $A = (A_{15} \| A_{14} \| \ldots \| A_1 \| A_0)$.

Then, $B = A \bmod p = (T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4) \bmod p$, where each 256 bit operand is represented as shown in Table 1.

(3) Extended Euclidean Modular Inversion. The extended Euclidean algorithm uses the toss and turns division method to obtain the modular inverse, and according to the nature of

Input: A, B, $n//n$ is the bit width
Output: C
(1) if $(n == 64)$ return $C = A \times B$;
(2) $A = A^H \times 2^{n/2} + A^L$;
(3) $B = B^H \times 2^{n/2} + B^L$;
(4) $C_1 = \text{KOA}\ (A^H, B^H, n/2)$;
(5) $C_2 = \text{KOA}\ (A^L, B^L, n/2)$;
(6) $C_3 = \text{KOA}\ (A^H + A^L, B^H + B^L, n/2)$;
(7) $C = C_1 << n + (C_3 - C_2 - C_1) << (n/2) + C_2$;

Algorithm 1: KOA multiplication.

Input: a, b, p
Output: $c = b/a \bmod p$
(1) $u = a$; $v = p$; $x_1 = b$; $x_2 = 0$;
(2) while $(v > 0)$
(3) if $(u[1:0] == 2'\text{b00})$
(4) $u = u \gg 2$; $x_1 = x_1/4 \bmod p$;
(5) else if $(v[1:0] == 2'\text{b00})$
(6) $v = v \gg 2$; $x_2 = x_2/4 \bmod p$;
(7) else if $(u[1:0] == v[1:0])$
(8) if $(u > v)$ $u = (u - v) \gg 2$;
(9) $x_1 = (x_1 - x_2)/4 \bmod p$;
(10) else $v = (v - u) \gg 2$;
(11) $x_2 = (x_2 - x_1)/4 \bmod p$;
(12) else if $(u[1:0] == 2'\text{b10})$
(13) if $((u >> 1) > v)$ $u = ((u \gg 1) - v) \gg 1$;
(14) $x_1 = (x_1/2 - x_2)/2 \bmod p$;
(15) else $u = u \gg 1$; $x_1 = x_1/2 \bmod p$;
(16) $v = (v - (u \gg 1)) \gg 1$;
(17) $x_2 = (x_2 - x_1/2)/2 \bmod p$;
(18) else if $(v[1:0] == 2'\text{b10})$
(19) if $(u > (v >> 1))$ $u = (u - (v \gg 1)) \gg 1$;
(20) $x_1 = (x_1 - x_2/2)/2 \bmod p$;
(21) $v = v \gg 1$; $x_2 = x_2/2 \bmod p$;
(22) else $v = ((v \gg 1) - u) \gg 1$;
(23) $x_2 = (x_2/2 - x_1)/2 \bmod p$;
(24) else if $(u \geq v)$
(25) $u = (u - v) \gg 1$; $x_1 = (x_1 - x_2)/2 \bmod p$;
(26) else
(27) $v = (v - u) \gg 1$; $x_2 = (x_2 - x_1)/2 \bmod p$;
(28) endwhile
(29) return $c = x_1$;

Algorithm 2: Extended Euclidean modular inversion.

Input: $k = (k_{l-1}, \ldots, k_0)$, point G
Output: $Q = kG$
(1) Initialize $R_0 = G$; $R_1 = 2G$; $i = l - 2$;
(2) while $(i \geq 0)$
(3) if $(k_i == 0)$ $R_1 = R_0 + R_1$; $R_0 = 2R_0$;
(4) else if $(k_i == 1)$ $R_0 = R_0 + R_1$; $R_1 = 2R_1$;
(5) $i = i - 1$;
(6) endwhile
(7) $Q = R_0$;

Algorithm 3: Montgomery point multiplication.

Table 1: Fast modulus reduction parameter representation.

	255–224	223–192	191–160	159–128	127–96	95–64	63–32	31–0
T	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
S_1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	0	0	0
S_2	0	A_{15}	A_{14}	A_{13}	A_{12}	0	0	0
S_3	A_{15}	A_{14}	0	0	0	A_{10}	A_9	A_8
S_4	A_8	A_{13}	A_{15}	A_{14}	A_{13}	A_{11}	A_{10}	A_9
D_1	A_{10}	A_8	0	0	0	A_{13}	A_{12}	A_{11}
D_2	A_{11}	A_9	0	0	A_{15}	A_{14}	A_{13}	A_{12}
D_3	A_{12}	0	A_{10}	A_9	A_8	A_{15}	A_{14}	A_{13}
D_4	A_{13}	0	A_{11}	A_{10}	A_9	0	A_{15}	A_{14}

the common divisor, all divisions can be changed to addition and subtraction operations, and the division by 2 operation is completed by binary shift, which is beneficial to hardware implementation. Here, it is expressed in a quaternary system; state machine cycle control is used to optimize the extended Euclidean algorithm, and the value of $b/a \bmod p$ can be directly obtained. The specific process is shown in Algorithm 2.

In Algorithm 2, the first line is the initialization state; the second line is the loop judgment state; the lines 3–27 are various judgments and calculations; the last line is the output. Second, the calculation of u and v always ensures that the result value is less than p, and no additional processing is required. However, the addition and subtraction of $x1$, $x2$ may be greater than p or overflow, and division by 2 and division by 4 require additional judgment. The specific calculation formula is as follows:

$$\frac{x}{2} \bmod p = \begin{cases} x \gg 1, & \text{if } x[0] == 1'b0, \\ x \gg 1 + p \gg 1 + 1, & \text{if } x[0] == 1'b1, \end{cases}$$

$$\frac{x}{4} \bmod p = \begin{cases} x \gg 2, & \text{if } x[1:0] == 2'b00, \\ (x \gg 1 + p \gg 1 + 1) \gg 1, & \text{if } x[1:0] == 2'b01, \\ x \gg 2 + p \gg 1 + 1, & \text{if } x[1:0] == 2'b10, \\ (x \gg 1 + p \gg 1 + 1) \gg 1 + p \gg 1 + 1, & \text{if } x[1:0] == 2'b11. \end{cases} \tag{7}$$

(4) Point Multiplication Optimization. At present, the Montgomery point multiplication algorithm is the most efficient and widely used algorithm [43, 44]. The specific operation process is shown in Algorithm 3.

It can be seen from Algorithm 3 that regardless of the value of k_i, the point addition and double point will be calculated during each cycle; in addition, the two are independent of each other and can be executed in parallel. At the same time, due to the simultaneous calculation of point addition and double point, the power consumption information leaked during the point multiplication operation is unruly, which can effectively resist the simple power consumption attack (SPA).

In the standard projective coordinate, given point $P(X_1, Y_1, Z_1)$ and point $Q(X_2, Y_2, Z_2)$, the calculation formula of point addition and double point is as follows.

The point addition formula is as follows:

$$\begin{cases} X(P+Q) = (X_1X_2 - aZ_1Z_2)^2 - 4bZ_1Z_2(X_1Z_2 + X_2Z_1), \\ Z(P+Q) = x_G(X_1Z_2 - X_2Z_1)^2. \end{cases} \tag{8}$$

The double point formula is as follows:

$$\begin{cases} X(2P) = \left(X_1^2 - aZ_1^2\right)^2 - 8bX_1Z_1^3, \\ Z(2P) = 4Z_1\left(X_1^3 + aX_1Z_1^2 + bZ_1^3\right). \end{cases} \tag{9}$$

Finally, the result is converted to an affine coordinate as follows:

$$x_1 = \frac{X_1}{Z_1}, \tag{10}$$

$$x_2 = \frac{X_2}{Z_2}, \tag{11}$$

$$y_1 = \frac{2b + (a + x_Gx_1)(x_G + x_1) - x_2(x_G - x_1)^2}{2y_G}. \tag{12}$$

Then, (x_1, y_1) is the result. Among them, (x_G, y_G) is the coordinate of the base point G.

It can be seen that, under standard projection coordinates, the projection point and the affine point will be mapped one by one. The affine coordinates will be

transformed into the projection coordinates at the beginning and will be mapped back to the affine coordinates when the operation is finished. Therefore, in the entire calculation process, only one modular inversion operation is used at the last time, and there is no modular inversion participation in the intermediate iteration process.

Finally, in order to further optimize the calculation efficiency of point addition and double point, the data stream is deeply optimized to complete the calculation in the shortest time. Fast modular multiplication consists of two modules, KOA multiplication and fast modular reduction, and the results can be calculated within one clock. Therefore, the part of the calculation process of point addition and double point is adjusted, and the KOA multiplication and fast modular reduction modules are alternately called to give full play to the calculation efficiency. After optimization, the point addition and double point calculation can finally be completed within 12 clocks, which have a very high efficiency.

3.4. Key Generation and Management. After the initial working key is obtained through the key management center, a pseudorandom number generator is used to generate the first group of keys to encrypt the first batch of packets. Then, the first batch of packets is used to generate the second group of keys, the second batch of packets is used to generate the third group of keys, and so on in order to generate all encryption keys, as shown in Figure 9.

3.4.1. Random Generation of Keys. A pseudorandom number generator [45] is widely used in information encryption. The selection of pseudorandom numbers starts from random seeds. Therefore, in order to ensure that the pseudorandom numbers obtained each time are sufficiently "random," the selection of random seeds is very important. If the random seeds are the same, the random numbers generated by the same random number generator will also be the same.

The most common method of generating pseudorandom numbers is to utilize a feedback shift register, which consists of two parts: the shift register and the feedback function. When the feedback function is linear, the feedback shift register is a linear feedback shift register (LFSR), as shown in Figure 10.

In the LFSR structure, f_n is the feedback coefficient, 1 means connection, and 0 means no connection.

Obviously, the output sequence of the LFSR is periodic, and an n-level LFSR provides up to $2n-1$ states (excluding all 0 states). According to the different feedback modes, the characteristic polynomial of the LFSR can be defined as follows:

$$p(x) = \sum_{i=0}^{n} f_i x^i = f_n x^n + f_{n-1} x^{n-1} + \ldots + f_1 x + 1. \quad (13)$$

In this paper, a 128 bit random number *randum* is defined, and the initial working key is used as a random seed to randomly generate the first group of keys. The polynomial used is $p(2) = 2^{128} + 2^{126} + 2^{101} + 2^{99} + 1$.

Similarly, the polynomial $p(2) = 2^{32} + 2^{26} + 2^{23} + 2^{22} + 2^{16} + 2^{12} + 2^{11} + 2^{10} + 2^8 + 2^7 + 2^5 + 2^4 + 2^2 + 2 + 1$ is used to complete the calculation of CRC32. Its initialization parameter is 0xFFFFFFFF, which uses big-endian alignment, and the end is not enough to be filled with 0x00. To meet the calculation requirements, the parallel method of the use of a lookup table is employed to realize the CRC32 calculation, which supports the direct calculation of 64-bit and 128-bit data and meets the calculation requirements of a 1G/10G network.

3.4.2. Key Storage and Update. During encryption and decryption, the key involved in the operation comes from the hash value of the last packet. To ensure the key synchronization of encryption and decryption, it is necessary to maintain a key storage (KS) module on the encryption and decryption channel. The depth of KS must be greater than the maximum number of hash iterations to ensure that the key is generated in advance before the next round of use. Here, dual-port RAM is used to store keys with a depth of 256 bits and a width of 128 bits, as shown in Figure 11.

In Figure 11, the initial key of KS is generated by the random number generator and, then, is continuously updated by the hash value of the packet. In the encryption direction, every time a packet that needs to be encrypted is received, the HKey of the corresponding entry is first taken from KS by using the PID as the key. Then, the hash value of the first 64 bytes of the encrypted packet updates KS. For the KS in the decryption direction, first, the HKey of the corresponding PID is taken from KS, and the encrypted packet is decrypted; at the same time, the hash value of the first 64 bytes of the current packet is fed back to the KS.

For the working key, it is not only used to initialize the random number generator but also the key for signature and verification. Moreover, the working key needs to be kept during the working period of the mimic encryption box, and there should be a backup and recovery mechanism. Therefore, the working key requires a higher security storage and update strategy. Here, flash memory is used to store the working key, and only the FPGA has read and write permissions. In addition, the key management center establishes a secure tunnel with the mimic encryption box and regularly updates the working key online.

3.5. Data Packet Processing

3.5.1. Rule Filtering. To monitor network data and prevent illegal access, for rule filtering, five tuples are used to form an access control list (ACL), including the protocol type, source and destination IP address, and source and destination port. At the same time, FPGA reconfigurability is used to form a pipeline for processing each step in order to meet the demand of high-speed data sending and receiving. The specific structure is shown in Figure 12.

Figure 12 shows that rule filtering is mainly composed of 4 functional modules: Parse, Key, Match, and Action.

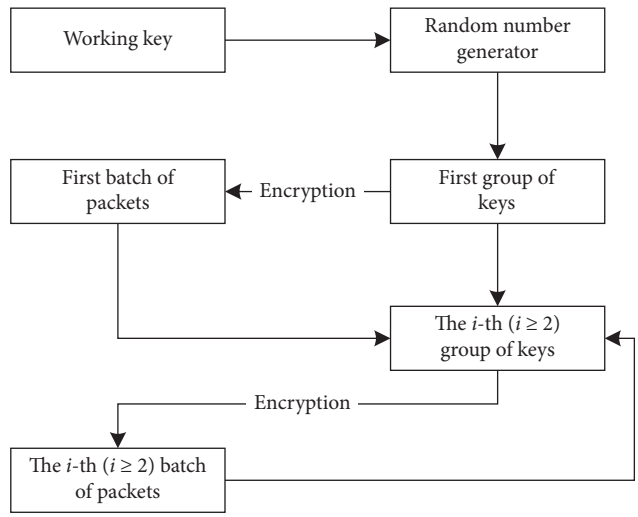


Figure 9: Key generation process.

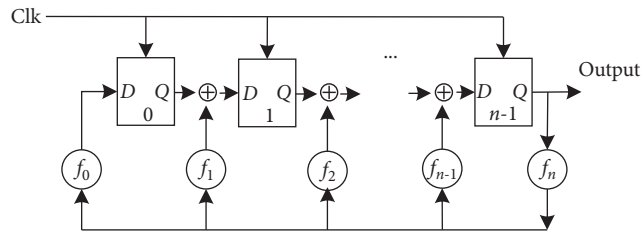


Figure 10: LFSR structure.

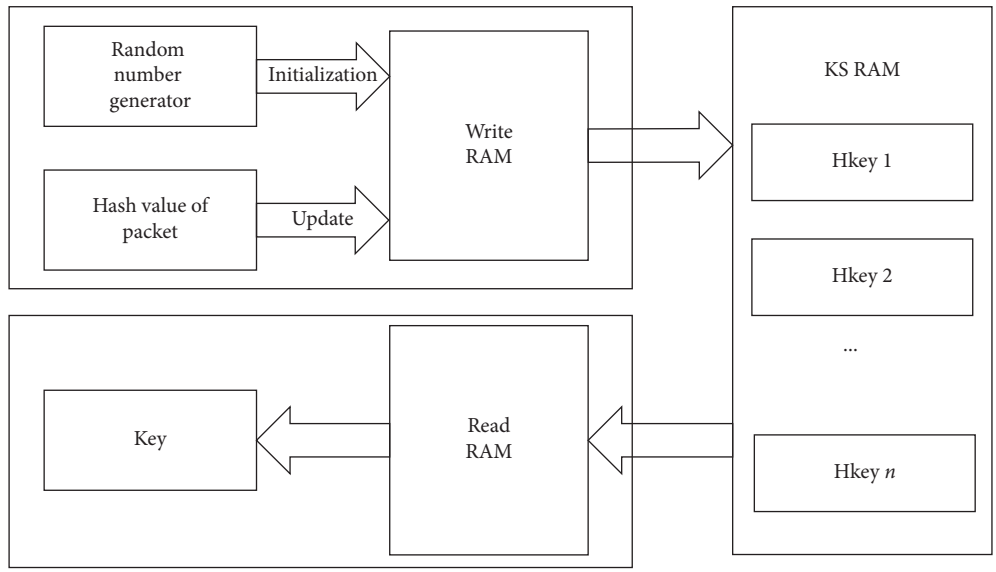


Figure 11: Storage and update for the key.

Among them, Parse represents the analysis of packet content; Key represents the extraction of keywords; Match represents matching, in which the matching is conducted by the rules of the ACL; Action represents action execution, indicating the processing of packets; and ACL represents the access control list and consists of five tuples. While the current packet is being parsed, FIFO is used for buffering and output according to the matching result.

Here, mainly IPv4 packets are analyzed. Multiple values for the Key module, ACL, and Action module are defined simultaneously, i.e., Key = $\{key_1, key_2, ..., key_{n1}\}$, ACL = $\{acl_1, acl_2, ..., acl_{n2}\}$, and Action = $\{action_1, action_2, ..., action_{n3}\}$. In

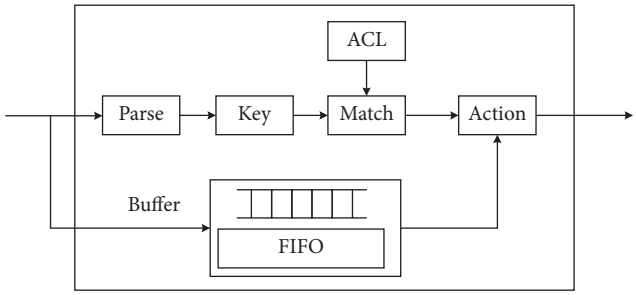


Figure 12: Rule filtering of packets.

this way, the packets are stripped layer by layer and extracted to a number of keywords, and then, according to the rules in a lookup table, the plaintext path, ciphertext path, discarding, encapsulation, decapsulation, and other data packets' processing is achieved.

3.5.2. DDR Cache Storage. To effectively deal with the impact of network blocking, prevent the mass dropping of packets, and ensure the smooth transmission of packets in each link, the packets are cached in memory. The FPGA provides the memory operation interface, but for each read and write, multiple signals need to be judged, which is quite tedious. To simplify the memory operation and improve the read and write efficiency, the FPGA memory interface was further encapsulated and optimized, as shown in Figure 13.

In the user FPGA logic module, four sets of FIFO are defined and are used to store the data and address for reading and writing memory. Then, the memory read-write control signals, such as app_addr, app_cmd, and app_en, are associated with FIFO read-write signals. In this way, as long as there are data and addresses in the four FIFOs, the state machine control will automatically read, arbitrate, and complete the memory read and write operations. Second, the memory read and write logic is independent, and the two do not affect each other, thereby improving the efficiency of the memory operations.

In addition, because the memory interface width is 256 bits and the MAC layer transmits 8/64 bits, bitwidth conversion is required for data written to memory. Similarly, data read from memory is converted from 256 bits to 8/64 bits. Finally, due to the need to complete reading and writing memory data according to the address and form a one-to-one mapping between data and address, the size of the address list has a direct impact on the space for reading and writing memory data. To store more data, the depth of the FIFO read and write memory address is set to 8192. In this way, multiple data packets can be cached, and the impact of network blocking can be effectively alleviated.

3.6. Security Mechanism

3.6.1. Autonegotiation Network Transformation. IP address hopping, port number hopping, protocol camouflage, channel transformation, and other technologies can hide the service mark and confuse the attacker, achieving covert communication. To further increase the attack difficulty and cost of the attacker, an autonegotiation transformation network is established. Through automatic negotiation, the communication parties form a security policy combination mechanism by frequently changing their IP addresses, ports, protocols, and channels in order to improve their defense capability. The specific process is shown in Figure 14.

As seen from Figure 14, the transformation process of the self-transforming network is as follows:

(1) The initiator will combine the transformed IP address, port, protocol, and channel to generate information *A*. The public key of the responder is used to encrypt and sign the data, generate the encrypted data *EncA* and the signed data *SigA*, and send it to the responder.

(2) After receiving the *EncA* and *SigA* from the initiator, the responder uses the private key to decrypt the information *A*, signs to generate the signature data *SigATmp*, and verifies the signature data *SigA* and *SigATmp*. If the verification is correct, the process continues to the next step.

(3) Similarly, the responder will transform the IP address, port, protocol, and channel combination to generate information *B*, encrypt *A* and *B* with the public key of initiator to generate encrypted data *EncAB*, and generate signature data *SigB* for *B*. Then, the responder will send *EncAB* and *SigB* to the initiator.

(4) The initiator receives *EncAB* and *SigB* from the responder and decrypts *A* and *B* of *EncAB* with the private key. The initiator, then, signs *B* to generate signature data *SigBTmp* and verifies the signature data *SigB* and *SigBTmp*. If the verification is correct, the process continues to the next step.

(5) The communication parties will update the data packet filtering rules by information *A* and *B* and use the new IP address, port, protocol, and channel for data encryption transmission.

The communication parties themselves can negotiate the time interval, which can be configured by ARM, or the initiator can define a timer to automatically negotiate according to certain time rules. Alternatively, the key management center may notify both parties to complete the network transformation.

3.6.2. Feedback Reconstruction Mechanism. The dynamic configuration of system parameters is mainly realized by the ARM, and the data are written into the inRAM of FPGA through memory address mapping. Moreover, the FPGA writes its state information, including controlling responses, number of incoming and outgoing packages, packet loss rate, decryption failure, and checksum error, into outRAM and sends it to the ARM. This structure is shown in Figure 15.

In this way, the collected information is analyzed by the ARM, and if an abnormality is found, the feedback

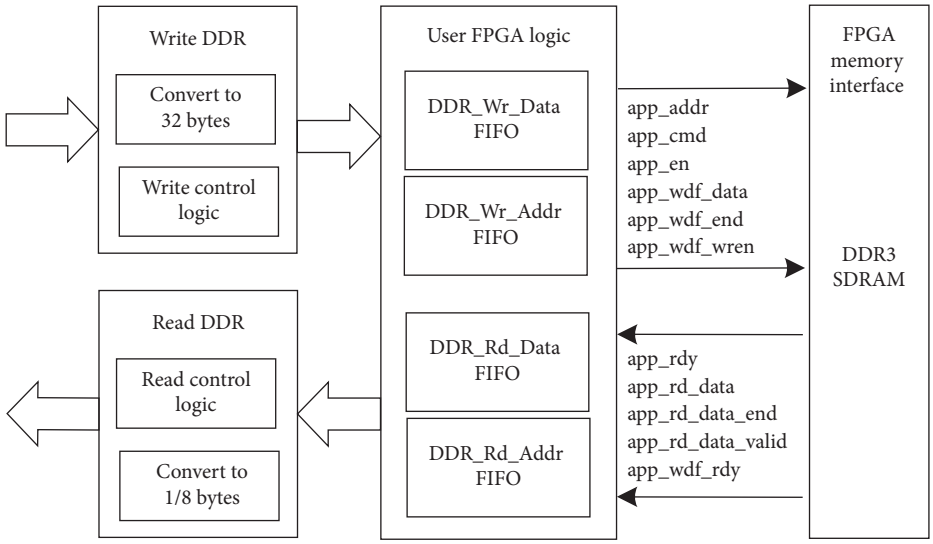


Figure 13: Memory read and write control optimization.

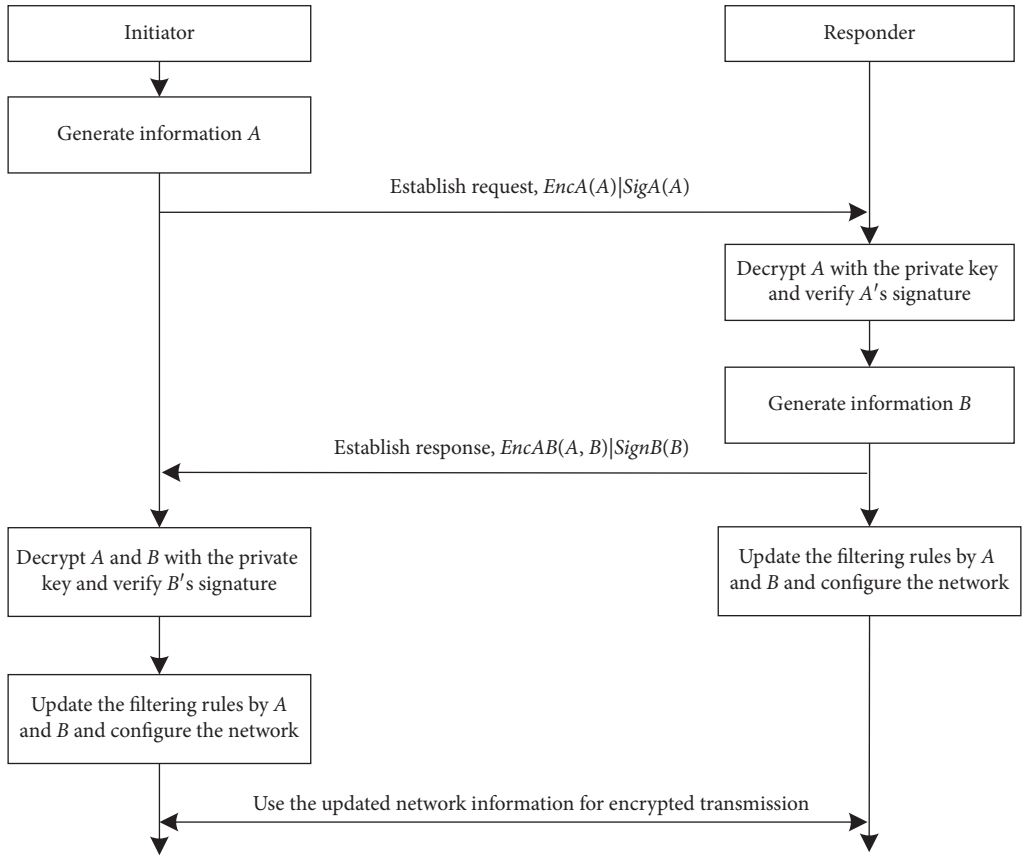


Figure 14: Self-transforming network negotiation process.

reconstruction mechanism is activated. First, a decision tree is built on the ARM side. In this decision tree, each branch node represents the choice between multiple alternatives, and each leaf node represents a decision. Then, whenever an abnormality is detected, the depth-first algorithm is used to traverse down from the root node, and if the judgment conditions of the current node are met, the execution is performed sequentially. Third, a heartbeat mechanism is added to the judgment condition, and the ARM sends a heartbeat packet to confirm whether the current network is under attack. If no response is received for a long time, the key management center will be notified and try to select another network for communication. Finally, according to the results of the decision tree judgment, the FPGA is

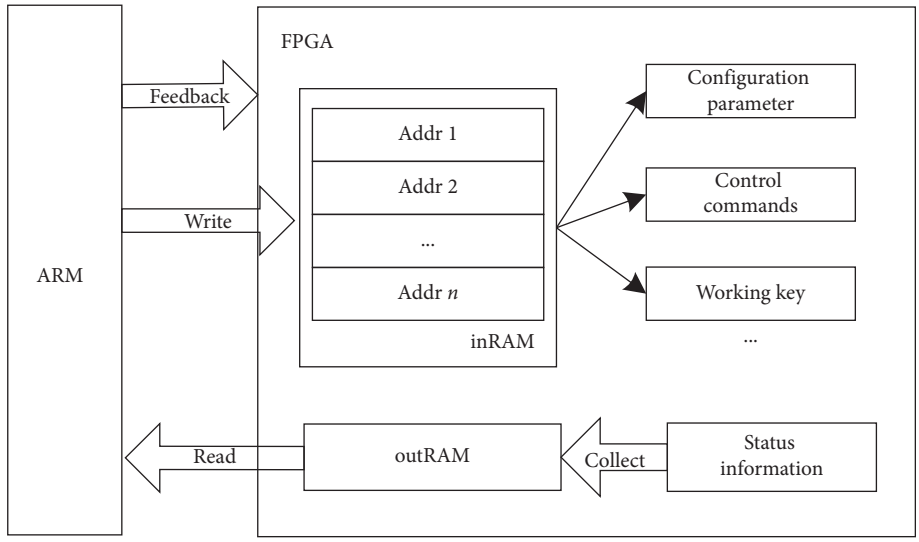


Figure 15: FPGA management structure.

"cleaned" through various operations, such as resetting all data paths and memory, closing the current network interface, and changing the network configuration.

For example, when the FPGA receives a large number of packets that do not comply with the rules in a short timeframe, it can be judged that the current network has been illegally attacked. Then, the current network interface can be closed, and the key management center can be notified to reselect a new network with which to connect. For another example, when no response from the other party is received for a long time, another network port is selected and a heartbeat packet is sent to verify the working status of the other party. In addition, when an attack causes the FPGA to work abnormally, the ARM can first perform a reset operation. If it still does not work normally, it can directly load other bitstreams to reconstruct the FPGA.

4. Experimental Results and Analysis

4.1. System Implementation. The server, FPGA, and switch used in this paper are shown in Table 2. The development environment is Vivado v2019.2 (64 bit).

The FPGA is composed of dual-core ARM processors and a Kintex-7 programmable chip, which can communicate by memory mapping. Its structure is shown in Figure 16. Among the FPGA's components, the ARM side has a 1G Ethernet port; the FPGA side has two 10G Ethernet ports and two 1G Ethernet ports; the FPGA is connected with DDR3 memory and flash. By writing bitstream into flash, the automatic loading and reconstruction of FPGA can be completed.

4.1.1. Implementation of Each Module. The following are implemented on the FPGA: 10G Ethernet interfaces; 1G Ethernet interfaces; packet parsing and encapsulation modules; and key management and update modules. The realization of each functional module is shown in Table 3.

To prevent packet overflow, each module is interconnected through an asynchronous FIFO and a set *prog_full* flag. If the FIFO is about to be full, the MAC layer frame flow control function will be triggered immediately and the data transmission will be suspended. In addition, 1G networks and 10G networks have the same processing flow, except for the fact that they work at different frequencies and the data width is 1 byte and 8 bytes, respectively. Therefore, bitwidth conversion is needed before packet encryption and decryption can be performed.

Second, the mimic encryption box in this paper encapsulates the AES128 and the 3DES algorithms to realize encryption and decryption and encapsulates the SHA1 and the SHA256 pipeline algorithms to form the hash algorithm. The implementation of each algorithm is shown in Table 4.

Finally, for the ECC algorithm, the specific situation of clock frequency, resource consumption, and calculation cycle of each module is shown in Table 5.

Table 5 shows that when the point multiplication frequency is 25 MHz, the calculation can be completed after 3064 clocks, which is a very high computational speed.

When the FPGA is configured as a dual 1G network, AES128 and SHA1 are used to complete the encryption. The occupancy ratio of LUTs is 60.17%, REGs is 32.15%, and DSP is 48%. When configured as a dual 10G network, 3DES and SHA256 are used to complete the encryption. The occupancy ratio of LUTs is 73.88%, REGs is 37.63%, and DSP is 48%. Therefore, the design requirements of the mimicry encryption box are fully met.

4.2. Performance Analysis

4.2.1. Encryption and Decryption Throughput. The throughput calculation formula is as follows:

$$T = \frac{B \times f_{\max} \times N}{d}, \tag{14}$$

Table 2: Configuration information of each component.

Component	Name	Configuration information
Server	IBM X3650 M3	CPU type: X5650 2.66 GHz; memory: 24 GB
FPGA	Xilinx XC7Z035	ARM: 800 MHz; memory: 1 GB; flash: 256 Mb; 2 SFP+ 10G Ethernet ports; 3 1G Ethernet ports; logic cells: 270K; number of DSP: 900
Switch	H3C 24-port 10G switch	24 1/10G SFP + Ethernet ports; 4 10/100/1000 M electrical ports

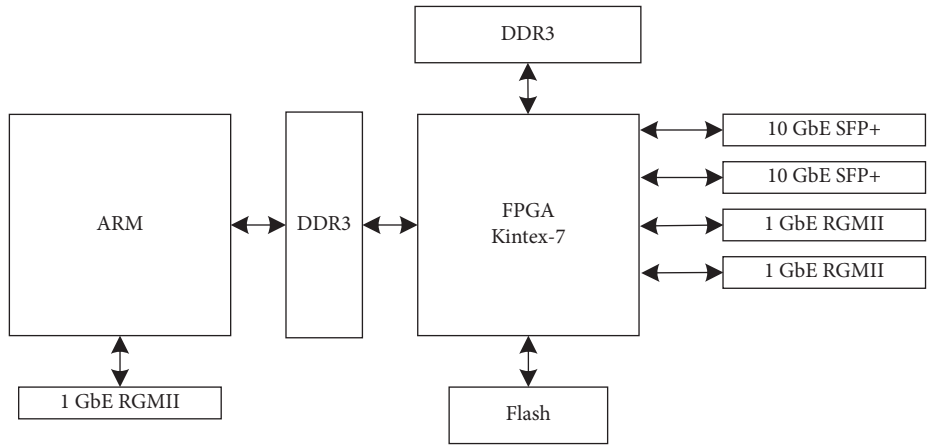


Figure 16: Hardware structure of the FPGA.

Table 3: Implementation of FPGA each functional module.

Functional module	Description	Frequency (MHz)	LUTs	REGs
ARM	ARM communication module	100	6487	8296
SFP_10GE_MAC	10G Ethernet port	156.25	4816	5457
Tri_Mode_Ethernet_MAC	1G Ethernet port	125	637	879
Memory_WR	Memory and read-write interfaces	666.67	9321	7549
Packet_Parse	Packet parsing	156.25	900	2336
Packet_Filter	Packet filtering	156.25	1026	2673
Packet_Package	Packet encapsulation	156.25	274	476
Parm_Mng	Parameter and key management	100	979	1832
Key_Update	Key update	156.25	517	1016

Table 4: Implementation of each encryption algorithm and hash algorithm.

Main algorithm	Algorithm structure	Highest frequency (MHz)	LUTs	REGs
AES128	Serial, 23 clocks	350	1060	402
3DES	48-stage pipeline	410	4436	5661
SHA1	80-stage pipeline	400	12336	23860
SHA256	64-stage pipeline	280	21472	24374
CRC32_64	64-bit lookup table	450	266	100
CRC32_128	128-bit lookup table	320	445	164
LFSR	Lookup tables, state machine	310	124	186

where T is the throughput, B is the data block size, $f_{\max}$ is the maximum clock frequency of each scheme, N is the pipeline stages, and d is the calculation delay.

To ensure the performance of encryption and decryption, AES128, 3DES, and LFSR all work at 200 MHz, while due to the use of pipeline technology, SHA1 and SHA256 work at 125 MHz or 156.25 MHz. In addition, for a 10G network, multiple encryption algorithm modules are designed to work in parallel in order to further improve the efficiency of execution. The throughput of each algorithm is calculated by formula (14), and the result is shown in Figure 17.

In Figure 17, in order to meet the demand of 10G encryption, 10 AES modules are designed to execute in parallel

Table 5: Implementation of each ECC module.

Module	Frequency (MHz)	LUTs	REGs	DSP	Calculation period
Point addition	25	8625	4380	144	12
Double point	25	9685	3620	144	12
Point multiplication control	25	1489	7217	0	3064
Coordinate transformation	25	17820	4401	144	225
Master state machine	50	252	1294	0	—

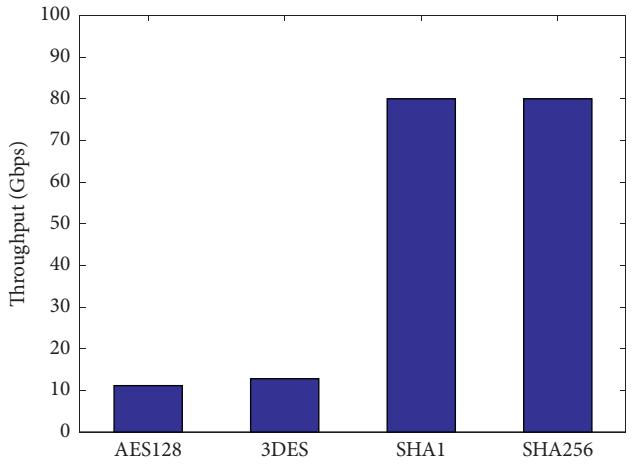


Figure 17: Throughput of each encryption algorithm.

and can work in four modes: ECB, CBC, CFB, and OFB. However, 3DES only works in the ECB mode. As seen from Figure 17, the algorithm throughputs achieved in this paper are all above 10 Gbps and can, thereby, fully meet the computing requirements of encryption and decryption of a 1G/10G network.

In the process of ECC signature/verification, encryption, and decryption of the key exchange, point multiplication will be called many times, and hash operation and coordinate transformation are also needed. Based on a frequency of 25 MHz, the corresponding speed of the ECC applications is shown in Table 6.

Table 6 reveals that the ECC can complete signature/verification, key encryption, and decryption, at least, 3000 times per second, which is a considerable execution speed.

4.2.2. Network Performance. Under 1G/10G networks, the maximum transmission unit is configured as 1495 bytes, and files of different sizes are encrypted and transmitted to the opposite end. After the peer receives the file, it decrypts the file. The CPU and mimic encryption box were used for testing. As the file size changes, the communication time between the two is shown in Figure 18.

Figure 18 shows that the processing time of the mimic encryption box encryption and decryption is significantly lower than that of the CPU. This is mainly because the mimic encryption box completes encryption and decryption while transmitting data. The CPU needs to encrypt data before transmitting and to decrypt the data after receiving it, which is time consuming. Especially under the 10G network, the encryption time occupies more than 88% of the total time;

Table 6: Calculation speed of the ECC applications.

Application	Clock number	Speed (p/s)
Signature	4068	6145
Verification	7629	3276
Encryption	7439	3360
Decryption	3908	6397

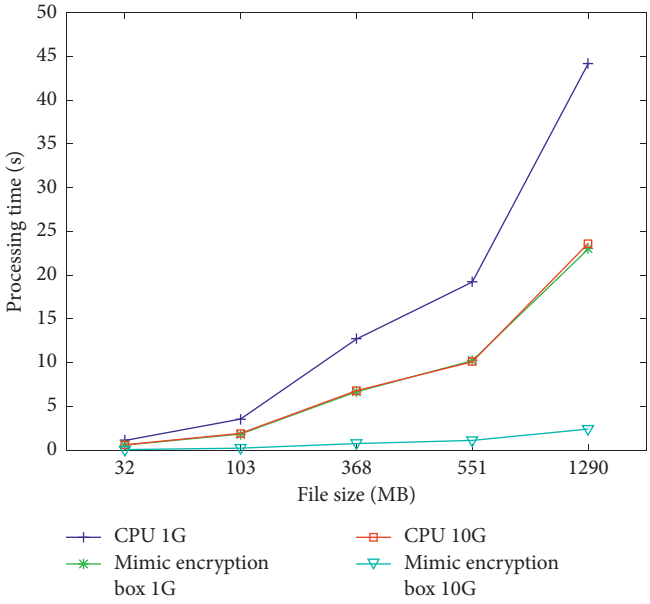


Figure 18: Comparison of the file encryption and decryption transmission performance between the CPU and the mimic encryption box.

therefore, the encryption speed of the CPU becomes the bottleneck.

In the 1G/10G networks, 100,000 packets are sent each time. As the packet length increases, the processing time of the CPU and the mimic encryption box changes, as shown in Figure 19.

Figure 19 shows that the mimic encryption box takes significantly less time to process encrypted packets than the CPU does. This is mainly because the FPGA omits system scheduling and speeds up network transmission and encrypted data processing.

In the 1G network, one byte is transmitted per clock at 125 MHz. However, AES128 needs to input 16 bytes each time to participate in the operation, and generates the result after 23 clocks. For continuous data streams, AES working at 200 MHz can meet the computing needs of the 1G network. At this time, the DDR cache is mainly used to avoid data loss

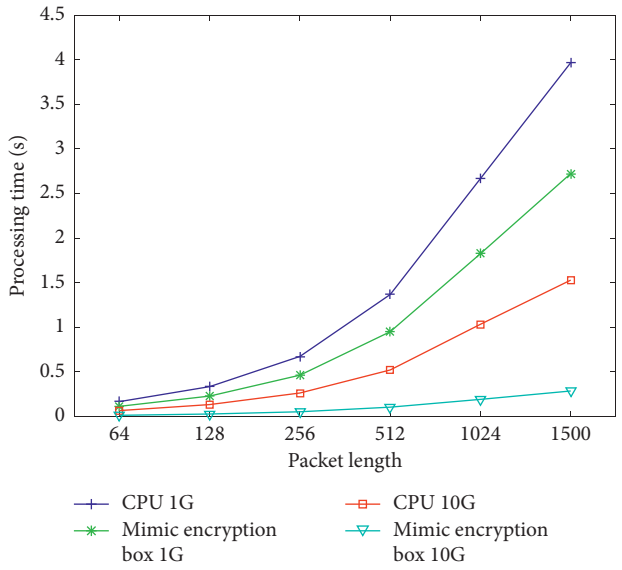


FIGURE 19: Comparison of the processing time of a large number of packets of different lengths between the CPU and the mimic encryption box.

when switching network interfaces. In the 10G network, 8 bytes are transmitted per clock at 156.25 MHz, and 10 AESs are required to work at 200 MHz under the continuous data stream. To ensure that the packets are first in, first out, the packets need to be distribution and collected. Also, it is necessary to wait for 10 groups of packets to be encrypted before sending out. As a result, the data throughput of the network side is greater than that of the encryption side. In addition, once the FIFO of the cached packet is about to be full, the flow control mechanism will be triggered to inform the other party to stop sending data. Obviously, if the DDR cache is not used, flow control frames will be sent frequently, resulting in a decrease in data throughput. After testing, the maximum throughput of the mimic encryption box under continuous data flow is as shown in Table 7.

It can be seen from Table 7 that using DDR cache improves data throughput to a certain extent. Moreover, the pause time of the flow control can be increased by the DDR cache, which can reduce the frequent transmission of flow control frames.

4.3. Security Analysis

4.3.1. Mimic Security Analysis. If the mimic encryption box is represented by the symbol Ω, it can be described by a 6-tuple as follows: $\Omega = \{\mathrm{Ec}, \mathrm{Key}, \mathrm{IP}, \mathrm{Port}, \mathrm{NPT}, \mathrm{NI}\}$, where Ec represents the encryption algorithm, Key represents the key, IP represents the IP address, Port represents the port number, NPT represents the network protocol type, and NI represents the network interface. The multiple stages of the system have multiple different encryption combination schemes. If a state vector $\Omega(t) = \{\mathrm{Ec}(t), \mathrm{Key}(t), \mathrm{IP}(t), \mathrm{Port}(t), \mathrm{NPT}(t), \mathrm{NI}(t)\}$ is used to represent a state at a certain moment, a set of reachable finite states can be used to represent all the different states of the system; that is,

$$\Omega = \begin{Bmatrix} \Omega(t_1) \\ \Omega(t_2) \\ \cdots \\ \Omega(t_l) \end{Bmatrix} = \begin{Bmatrix} \mathrm{Ec}(t_1), \mathrm{Key}(t_1), \mathrm{IP}(t_1), \mathrm{Port}(t_1), \mathrm{NPT}(t_1), \mathrm{NI}(t_1) \\ \mathrm{Ec}(t_2), \mathrm{Key}(t_2), \mathrm{IP}(t_2), \mathrm{Port}(t_2), \mathrm{NPT}(t_2), \mathrm{NI}(t_2) \\ \cdots \\ \mathrm{Ec}(t_l), \mathrm{Key}(t_l), \mathrm{IP}(t_l), \mathrm{Port}(t_l), \mathrm{NPT}(t_l), \mathrm{NI}(t_l) \end{Bmatrix}. \quad (15)$$

Among them, the components of the vector represent the changes of the system encryption algorithm, key, IP address, port number, protocol type, and network interface channel.

The traditional encryption system's component vectors do not change during operation; therefore, $\Omega(t_1) = \Omega(t_2) = \ldots = \Omega(t_l)$; that is, the traditional encryption system is static and deterministic. The characteristics of the mimic encryption box are its dynamics, diversity, and randomness; that is, at the time ti, the state of the system changes, so $\Omega(t_1) \neq \Omega(t_2) \neq \ldots \neq \Omega(t_l)$.

Ω can be described by information entropy: $H(x) = -\sum_{j=1}^{l} p_j \log(p_j)$, where p_j represents the probability of occurrence of each component vector $\Omega(t_j)$. Therefore, the external uncertainty of Ω can be transformed into the size of the information entropy; that is, the maximum information entropy can be determined as follows: $H_{\max}(X) = \max\left\{-\sum_{j=1}^{l} p_j \log(p_j)\right\}$.

Obviously, when the probability of occurrence of $l\Omega(t_j)$ is the same and equal to $1/l$, $H(X)$ reaches the maximum value, namely, $H_{\max}(X) = \log l$. Therefore, the greater the change state of Ω is, the greater the information entropy and the greater the external uncertainty are. As a result, this paper uses multiple encryption algorithms and multiple groups of keys, combined with IP address hopping, port number hopping, protocol camouflage, and network interface selection, to jointly realize a number of different element changes and combinations, which have a high degree of uncertainty.

4.3.2. Encryption Security Analysis. The mimic encryption box encrypts the plaintext load of the TCP layer and changes the IP address, port, and protocol so that the attacker cannot obtain user-related information, increasing the difficulty of the attack. The selected AES and 3DES encryption

Table 7: Maximum throughput of the mimic encryption box.

	1G FIFO cache	1G DDR cache	10G FIFO cache	10G DDR cache
Continuous data stream	110 MB/s		812 MB/s	855 MB/s

algorithms have high security and can provide high-quality data protection. At the same time, the encryption algorithm provides four working modes: ECB, CBC, CFB, and OFB. Among them, under CBC, data blocks are interrelated during encryption and are not easy to be actively attacked.

Second, the 256-bit ECC algorithm is used to complete the negotiation between the communication parties. The ECC algorithm is based on the intractable problem of discrete logarithms, which has higher security and resistance to attacks. Although, according to the published $E(F_p(a, b))$, base point G and order n, 2G, 3G, ..., nG can be calculated, and $n\mathrm{G} = O$. In addition, when a large number k is given, $P = kG$ can be easily calculated. However, given P and G, it is very difficult to reversely infer k.

Finally, the hash operation is irreversible, and the hash value of the packet is used as the key of the next round in order to prevent the attacker from pushing back the key according to the content. If the output value of the hash is uniformly distributed and the bits of message digest are m bits, then there are $n = 2^m$ possible outputs. If k $(k \le n)$ random inputs are selected, the probability of, at least, one collision is as follows:

$$\begin{aligned} P(n,k) &= 1 - \frac{n!}{(n-k)! \times n^k} \\ &= 1 - \left[\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right), ..., \left(1 - \frac{k-1}{n}\right)\right] \\ &= 1 - \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \approx 1 - \prod_{i=1}^{k-1} e^{-i/n} = 1 - e^{-k(k-1)/2n}. \end{aligned} \tag{16}$$

To make $P(n,k) > 0.5$, that is, to achieve $1/2 = 1 - e^{-k(k-1)/2n}$, we have the following: $\ln 2 \approx k^2/2n$; then, $k \approx \sqrt{n}$.

According to the abovementioned calculation, if the hash function has m-bit output digests, then the probability of a collision occurring with only $k = 2^{m/2}$ attempts is, at least, 50%. In this way, any change in the input information will result in a significant change in the hash result, thereby ensuring that the key is greatly different. Table 8 shows the collision threshold of the hash function.

4.3.3. Antiattack Analysis. A test environment is set up through network cameras, mimic encryption boxes, and switches, and then, a simulation of attackers launching attacks on intermediate switches is conducted in order to verify the security of the mimic encryption boxes, as shown in Figure 20.

Table 8: Hash function collision threshold.

Hash function	Collision threshold
SHA1	$2\text{^}80 \approx 1.2 \times 10\text{^}24$
SHA256	$2\text{^}128 \approx 3.4 \times 10\text{^}38$

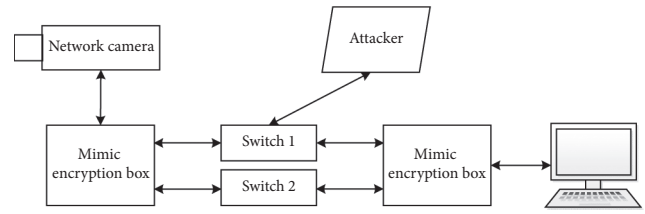


Figure 20: Security test environment of the mimic encryption box.

(1) Network Sniffing. The mimic encryption box uses dynamic network information hopping and pseudorandom encryption to make the system appear in a changing state. The IP address and port hopping mechanism essentially increase the difficulty of scanning by randomly and unpredictably migrating service entries. To obtain accurate target information, the attacker will increase the number and frequency of scanning and detection, which will significantly increase the cost of the attack. Assuming that the time of an attack is t, the number of IP addresses that can be hopped is n, the number of ports is m, and then, the time required for the attacker to successfully break a service is $T = t \times (1 + \sum_{k=1}^{n\times m-1} k \times C_{n\times m-1}^{k} / (C_{n\times m}^{k} \times C_{n\times m-k}^{1}))$, which is simplified as $T = t \times (1 + (n \times m - 1)/2)$. Then, taking 251 IP addresses 192.168.0.3–192.168.0.253 as an example, 64510 ports in 1025–65534 are detected. These IP addresses and ports are combined to form 251×64510 service entries. If the attacker scans every 5 milliseconds, it will take, at least, 40480 seconds. Therefore, the time of IP address and port number hopping can be set to 3600 seconds to reduce the scanning success rate. Second, if the scanning detection frequency is too high, the illegal data can be found quickly based on the statistical information of the filtering rules. In addition, even if the detection is successful, if the mimic encryption box subsequently switches the network, such as switching from switch 1 to switch 2, the attacker needs to sniff again.

(2) Tampering Attack. For tampered encrypted packets, if the attacker tampered with the PID, it will cause the decryption to fail. However, because it is decrypted first and, then, the CRC32 is checked, after decryption fails, the CRC32 cannot be passed, so the packet is discarded. If the first 64 bytes of the packet are tampered with, the key update and decryption will fail, and the packet will be discarded. If the data behind the 64 bytes of the packet are tampered with, the decryption will fail and the packet will be discarded. Obviously, as long as the attacker tampered with any byte of the encrypted packet, it will cause the decryption to fail and cause the packet to be discarded. At this point, the ARM can monitor the behavior and quantity of packet discards in real time. If a large number of packets are found to be discarded, it will

consider that the system has been attacked by an attacker, and then, communication is stopped or the network channel is changed.

(3) Replay Attack. When an attacker intercepts a packet and implements a replay attack, the system will follow the normal process. However, because PIDs increase in sequence, through the collection of statistics on the same PID packets, if it is found that the number of packets of a certain PID is significantly higher than other packets, it can be judged that it has received a replay attack. Second, because the TCP/IP protocol is used to transmit data, the repeated sending of the same packet will cause the system to fail to receive the packet with the corresponding sequence number. At this time, the system will not receive valid packets and will send new requests repeatedly. Finally, the sequence number field of the TCP header is encrypted; therefore, it is difficult for an attacker to forge the correct sequence number. In this way, through user monitoring, replay attacks in this situation can be found.

(4) Ciphertext-Only Attack. The mimic encryption box can effectively resist key exhaustive attacks, ciphertext-only attacks, and differential attacks. With the constantly updated key and the dynamically changing encryption algorithm, that is, the "one packet, one key," it effectively prevents attackers from using brute force attacks and ciphertext to reverse the key, ensuring that the attacker cannot decrypt all data normally. At the same time, the hash value of the first 64 bytes of the encrypted packet is used as the key for the next round. Since the hash value is basically irregular symbol data, it is difficult for an attacker to perform inference analysis. The mimic encryption box effectively resists the method of using differential attacks to decipher the ciphertext and prevents unauthorized leakage and undetected modification of data.

(5) DDos Attack. A DDos attack will cause the mimic encryption box to have a key update failure and a decryption failure; the attack will, then, affect normal system operation. If a large number of illegal interference packets are detected in a short period of time, the feedback reconstruction mechanism can be used to select other networks for communication. For example, when the attacker attacks the network where switch 1 is located, at this time, the FPGA can be reconstructed and the network where switch 2 is located can be selected. The transformation of the network mitigated the DDos attack to a certain extent.

(6) Vulnerability Attack. The mimic encryption box adopts the ARM + FPGA software and hardware cooperation method. All packets must be "reviewed" by the FPGA, enabling the prevention of unauthorized data access or network attacks, and the FPGA's security is much higher than that of software security products. Using the FPGA network interface for parameter configuration can realize one-way transmission control, protocol filtering, and the content filtering of packets. It can effectively prevent the use of the vulnerabilities or the weaknesses of multimedia equipment in order to carry out targeted intrusion and destruction. At the same time, by using FPGA hardware encryption, only the decrypted data can be transmitted to the upper-layer application; therefore, the packets containing malicious content are displayed as garbled after decryption, and normal attacks on the software system cannot be carried out.

4.3.4. Comparison with Other Schemes. Considering defense features such as dynamic, diversity, intrusion detection, encrypted transmission, virtualization, and hardware protection, a comparison of the defense mechanisms of different schemes is shown in Table 9.

Table 9 reveals that offering more advantages than other solutions, the mimic encryption box not only combines the randomness, dynamics, and diversity defense characteristics of MTD and CMD but also integrates hardware protection and reconfigurable encryption technology. In addition, most of the schemes in Table 9 adopt the software implementation and are built on the operating system. If the operating system itself has vulnerabilities, someone can bypass the protection mechanism and launch attacks. The mimic encryption box implements a security protection mechanism on the FPGA, does not rely on the operating system, and uses the FPGA's high anti-interference ability to filter some system attacks. Second, the mimic encryption box strictly controls the illegal traffic of the network to the front-end equipment with rule filtering. Third, the mimic encryption box uses FPGA encryption to increase the uncertainty of the system without reducing system performance, thereby increasing the difficulty of attack. However, some existing solutions use virtualization technology. Although layer-by-layer virtualization brings a certain degree of security, it loses performance. Fourth, the mimic encryption box uses a dynamically variable encryption algorithm and uses a different key to encrypt each packet. Even if an attacker intercepts a large number of packets, it is difficult to successfully implement the ciphertext-only attack. Finally, in view of the diversity and mobility of network multimedia data and equipment, the mimic encryption box can be easily connected to the original network. Compared with other solutions, it not only has higher security but also has portability and scalability for deployment.

5. Scope and Limitations

The mimic encryption box uses reconfigurable hardware to realize dynamic random encryption, network structure transformation, and data security filtering, which improves the security of sensitive data transmission and has good reliability. It can be deployed at the edge of the network and bound with terminal equipment to enhance the security of network data transmission, such as that associated with network cameras, video terminal equipment, data collection equipment, and self-service terminals. Second, the mimic encryption box can also be deployed in the Ethernet to complete end-to-end data encryption transmission. For the internal communications of the military and government

Table 9: Comparison between this scheme and other schemes.

Scheme	Dynamic	Diversity	Intrusion detection	Encrypted transmission	Virtualization	Hardware protection	Defensive effect
Aydeger et al. [9]	√	√	√				Defend against crossfire attacks
Aydeger et al. [10]	√	√	√		√		Resist crossfire attacks and provide network forensics
Wang et al. [12]	√	√					Increase the scanning space
Zhao et al. [15]	√	√					Defend against sniffer attack
Tang et al. [26]	√	√		√			Provide dynamic encryption; resist both exhaustive and analysis attacks
Lin et al. [34]			√			√	Detect the security of packets
Our system	√	√	√	√		√	Provide dynamic network and encryption and access control filtering; resist ciphertext-only and key exhaustion attacks

confidential departments, it can provide high-security services from the inside to outside. Finally, the mimic encryption box can also be applied to blockchain, cloud security, data security, and other fields. While using hardware to improve security, it can also be used to accelerate application calculations, such as signatures, verification, and data integrity verification.

The mimic encryption box provides the management of reconfigurable cryptographic algorithms and keys, enabling multimedia devices to conveniently call cryptographic services and to complete application layer encryption and authentication of video and voice data. It also provides fine-grained security and authentication services. Second, the mimic encryption box uses ACL to filter packets, strictly controls the access of illegal traffic, and has a certain anti-infiltration function. Finally, the mimic encryption box uses a "one packet, one key" encryption mechanism and has a certain self-cleaning function with a self-transforming network capability and a feedback mechanism. After being attacked, it can be "online" again in a reconstructed way.

At present, the mimic encryption box mainly completes the secure encryption of network multimedia data. In the future, we will consider using secure tunnel technology to provide better antireplay and antitampering functions with time factor and integrity verification. In addition, limited by FPGA resources, the current mimic encryption box has a mediocre performance in a 10G network. Therefore, how to use the idea of mimic defense to design a mimic encryption box or gateway with higher performance and higher security and apply it in 10 G/100G networks still needs further research.

6. Conclusions and Future Work

The mimic encryption box proposed in this paper optimizes the implementation of the reconfigurable hash algorithm, symmetric encryption algorithm, and elliptic curve algorithm, thereby improving the processing efficiency of data encryption and decryption. It uses a pseudorandom number generator to generate the initial key and updates the key with the hash value of the packet in order to realize the random encryption mode of "one packet, one key." Dynamic network information is realized by IP address hopping, port hopping, and protocol camouflage. The firewall function is realized by the access control list formed by five tuples, which limits the illegal access of the attacker. At the same time, the FPGA is managed through the ARM, and under abnormal conditions, the feedback information is used to realize the "cleaning" and reconstruction of the FPGA so that the mimic encryption box can work again. The experimental results and the analysis show that the mimic encryption box not only has higher encryption and decryption throughput but also has higher security. It can effectively prevent data leakage and tampering, disrupt attackers, and weaken network sniffing and vulnerability attacks. In addition, it can resist key exhaustive attacks and ciphertext-only attacks. It is suitable for applications with high security requirements.

Further study is suggested for the analysis of a mimic encryption gateway with higher performance, ways to improve its processing performance in 10G/100G networks, and the expansion of the use of the mimic encryption box across a range of system applications. At the same time, further study is suggested on ways to combine the mimic encryption box with SDN, as well as the use of the mimic encryption box to achieve a better mimic defense with a time-varying network, through changing the routing and composition of the network and forming a combination of multilayer changes, thereby effectively resisting network attacks.

Data Availability

All data generated or analyzed during this study are included in this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Multimedia Internet of Things: a comprehensive survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020.

[2] P. Chauhan, A. Choudhary, and A. K. Gupt, "Multimedia big data security,," in *Proceedings of the 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, pp. 112–117, Bhubaneswar, India, 2018.

[3] V. Nikhila and C. Rupa, "Intensifying multimedia Information security using comprehensive cipher," in *Proceedings of the 2019 Innovations in Power and Advanced Computing Technologies (I-PACT)*, pp. 1–4, Vellore, India, 2019.

[4] X. Li, J. Xu, H.-N. Dai, Q. Zhao, C. F. Cheang, and Q. Wang, "On modeling eavesdropping attacks in wireless networks," *Journal of Computational Science*, vol. 11, pp. 196–204, 2015.

[5] S. Ma, T. Zhang, A. Wu, and X. Zhao, "Lightweight and privacy-preserving data aggregation for mobile multimedia security," *IEEE Access*, vol. 7, pp. 114131–114140, 2019.

[6] H. Zhou, Y. Ma, L. Yan et al., "A Time-based self-cleaning control mechanism in moving target defense," in *Proceedings of the IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 983–986, Chongqing, China, 2018.

[7] X. Zhou, Y. Lu, Y. Wang et al., "Overview on moving target network defense," in *Proceedings of the IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 821–827, Chongqing, China, 2018.

[8] B. Zhang, X. Chang, and J. Li, "A generalized information security model SOCMD for CMD systems," *Chinese Journal of Electronics*, vol. 29, no. 3, pp. 417–426, 2020.

[9] A. Aydeger, N. Saputro, K. Akkaya et al., "Mitigating crossfire attacks using SDN-based moving target defense," in *Proceedings of the IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 627–630, Dubai, China, 2016.

[10] A. Aydeger, N. Saputro, K. Akkaya et al., "A moving target defense and network forensics framework for ISP networks using SDN and NFV," *Future Generation Computer Systems*, vol. 94, pp. 496–509, 2019.

[11] K. Zeitz, M. Cantrell, R. Marchany et al., "Changing the game: a micro moving target IPv6 defense for the Internet of Things," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 5778–6581, 2018.

[12] K. Wang, X. Chen, and Y. Zhu, "Random domain name and address mutation (RDAM) for thwarting reconnaissance attacks," *PLoS One*, vol. 12, no. 5, pp. 1–22, 2017.

[13] J. Zheng and A. S. Namin, "The Impact of address changes and host diversity on the effectiveness of moving target defense strategy," in *Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 553–558, Atlanta, GA, USA, 2016.

[14] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable & Secure Computing*, vol. 13, no. 2, pp. 163–177, 2015.

[15] Z. Zhao, D. Gong, B. Lu, F. Liu, and C. Zhang, "SDN-based double hopping communication against sniffer attack," *Mathematical Problems in Engineering*, vol. 2016, Article ID 8927169, 13 pages, 2016.

[16] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, 2015.

[17] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Adversary-aware IP address randomization for proactive agility against sophisticated attackers," in *Proceedings of the Computer Communications (INFOCOM)*, pp. 738–746, Kowloon, Hong Kong, 2015.

[18] H. Hu, Z. Wang, G. Cheng, and J. Wu, "MNOS: a mimic network operating system for software defined networks," *IET Information Security*, vol. 11, no. 6, pp. 345–355, 2017.

[19] H. Hu, J. Wu, Z. Wang, and G. Cheng, "Mimic defense: a designed-in cybersecurity defense framework," *IET Information Security*, vol. 12, no. 3, pp. 226–237, 2018.

[20] B. Ma and Z. Zhang, "Security research of redundancy in mimic defense system," in *Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2910–2914, Chengdu, China, 2017.

[21] C. Qi, J. Wu, G. Cheng, J. Ai, and S. Zhao, "An aware-scheduling security architecture with priority-equal multi-controller for SDN," *China Communications*, vol. 14, no. 9, pp. 144–154, 2017.

[22] W. Liu, F. Chen, H. Hu et al., "A novel framework for zero-day attacks detection and response with cyberspace mimic defense architecture," in *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 50–53, Nanjing, China, 2017.

[23] H. Li, J. Hu, H. Ma et al., "The architecture of distributed storage system under mimic defense theory," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, pp. 2658–2663, Boston, MA, USA, 2017.

[24] A. Abusukhon, Z. Mohammad, and M. Talib, "A novel network security algorithm based on encrypting text into a white-page image," in *Proceedings of the World Congress on Engineering and Computer Science*, pp. 1–5, San Francisco, CA, USA, 2016.

[25] A. Abusukhon, M. N. Anwar, Z. Mohammad, and B. Alghannam, "A hybrid network security algorithm based on Diffie Hellman and text-to-image encryption algorithm," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 1, pp. 65–81, 2019.

[26] H. Tang, Q. T. Sun, X. Yang, and K. Long, "A network coding and DES based dynamic encryption scheme for moving target defense," *IEEE Access*, vol. 6, pp. 26059–26068, 2018.

[27] A. Khan, Q. T. Sun, Z. Mahmood, and A. U. Ghafoor, "Energy efficient partial permutation encryption on network coded MANETs," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–10, 2017.

[28] W. Jiang, H. Xu, H. Dong, H. Jin, and X. Liao, "An improved security framework for web service-based resources," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 3, pp. 774–792, 2016.

[29] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.

[30] X. Jia, D. He, N. Kumar, and K.-K. R. Choo, "A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 560–571, 2020.

[31] N. Indira, S. R. Devi, and A. V. Kalpana, "R2R-CSES: proactive security data process using random round crypto security encryption standard in cloud environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2020.

[32] L. A. Maciel, M. A. Souza, H. C. Freitas et al., "Reconfigurable FPGA-based K-means/K-modes architecture for network

Intrusion detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 8, pp. 1459–1463, 2020.

[33] J. A. Joseph, R. Korah, and S. Salivahanan, "Efficient string matching FPGA for speed up network Intrusion detection," *Applied Mathematics & Information Sciences*, vol. 12, no. 2, pp. 397–404, 2018.

[34] S. Lin, D. Zhang, Y. Fu et al., "A design of the ethernet firewall Based on FPGA," in *Proceedings of the 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, Shanghai, China, 2017.

[35] S. M. Keni and S. Mande, "Packet filtering for IPV4 protocol using FPGA," in *Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 400–403, Madurai, India, 2018.

[36] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero et al., "Net FPGA-based firewall solution for 5G multi-tenant architectures," in *Proceedings of the IEEE International Conference on Edge Computing (EDGE)*, pp. 132–136, Milan, Italy, 2019.

[37] H. E. Michail, G. S. Athanasiou, V. I. Kelefouras et al., "Area-throughput trade-offs for SHA-1 and SHA-256 hash functions' pipelined designs," *Journal of Circuits, Systems and Computers*, vol. 25, no. 4, pp. 1–27, 2016.

[38] S. Suhaili and T. Watanabe, "High throughput evaluation of SHA-1 implementation using unfolding transformation," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 5, pp. 3350–3355, 2016.

[39] M. M. Wong, M. L. D. Wong, C. Zhang et al., "Circuit and system design for optimal lightweight AES encryption on FPGA," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 52–62, 2018.

[40] A. Hafsa, A. Sghaier, M. Machhout et al., "A new security approach to support the operations of ECC and AES algorithms on FPGA," in *Proceedings of the 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 95–100, Sousse, Tunisia, 2019.

[41] M. S. Hossain, Y. Kong, E. Saeedi et al., "High-performance elliptic curve cryptography processor over NIST prime fields," *IET Computers & Digital Techniques*, vol. 11, no. 1, pp. 33–42, 2016.

[42] S. Khan, K. Javeed, and Y. A. Shah, "High-speed FPGA implementation of full-word Montgomery multiplier for ECC applications," *Microprocessors and Microsystems*, vol. 62, pp. 91–101, 2018.

[43] W. Yu, K. Wang, B. Li et al., "Montgomery algorithm over a prime field," *Chinese Journal of Electronics*, vol. 28, no. 1, pp. 43–48, 2019.

[44] K. Javeed and X. Wang, "FPGA based high speed SPA resistant elliptic curve scalar multiplier architecture," *International Journal of Reconfigurable Computing*, vol. 2016, pp. 1–10, 2016.

[45] K. N. Devika and R. Bhakthavatchalu, "Design of reconfigurable LFSR for VLSI IC testing in ASIC and FPGA," in *Proceedings of the International Conference on Communication and Signal Processing (ICCSP)*, pp. 928–932, Chennai, China, 2017.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8857815, 7 pages
https://doi.org/10.1155/2020/8857815

Research Article

Lattice-Based Linearly Homomorphic Signature Scheme over $\mathbb{F}_2$

Jie Cai,[1] **Han Jiang**,[2] **Hao Wang**,[3] **and Qiuliang Xu**[2]

[1]*School of Science, Shandong Jianzhu University, Ji'nan, Shandong, China*
[2]*School of Software, Shandong University, Ji'nan, Shandong, China*
[3]*School of Information Science and Engineering, Shandong Normal University, Ji'nan, Shandong, China*

Correspondence should be addressed to Han Jiang; jianghan@sdu.edu.cn

Received 7 August 2020; Revised 2 September 2020; Accepted 22 September 2020; Published 29 October 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Jie Cai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, we design a new lattice-based linearly homomorphic signature scheme over $\mathbb{F}_2$. The existing schemes are all constructed based on hash-and-sign lattice-based signature framework, where the implementation of preimage sampling function is Gaussian sampling, and the use of trapdoor basis needs a larger dimension $(m \geq 5n \log q)$. Hence, they cannot resist potential side-channel attacks and have larger sizes of public key and signature. Under Fiat–Shamir with aborting signature framework and general SIS problem restricted condition $(m \geq n \log q)$, we use uniform sampling of filtering technology to design the scheme, and then, our scheme has a smaller public key size and signature size than the existing schemes and it can resist side-channel attacks.

1. Introduction

The idea of the linear homomorphic signature scheme comes from network coding routing mechanism. Specifically, after a signer sends a number of signatures for messages to router (verifier) in a computer network using network coding, the router can generate a random linear combination $\boldsymbol{\mu}$ of the received messages. Using the homomorphic property, the router computes a signature (σ) of $(\boldsymbol{\mu})$ and transmits $(\sigma, \boldsymbol{\mu})$ to the next router, and the process will be continued for different linear combined messages. The final router accepts properly signed signature and recovers the original message by solving a full-rank linear system over $\mathbb{F}_p$.

Then, one can easily abstract definition from applications. Informally, given n–dimensional message vectors $(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{F}_p^n)$ and signatures $\sigma_1, \ldots, \sigma_k$, anyone can create a signature for any vector $\boldsymbol{\mu} \in \text{span}\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$. At the same time, if any adversary cannot produce a valid signature for $\boldsymbol{\mu}' \notin \text{span}\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, we say the linear homomorphic signature scheme is secure. There exit many classical linearly homomorphic signatures [1–4] based on the difficulty in solving discrete logarithm or the difficulty in integer factoring. However, they have two obvious disadvantages.

First, the parameter p must be large enough to guarantee the difficulties in classical problems, but implementations are generally given over $\mathbb{F}_2$ in network coding. Second, these schemes cannot resist quantum computing attack as we all know. Hence, more and more people focus on designing postquantum linearly homomorphic signature scheme over $\mathbb{F}_2$, where lattice-based schemes are significant.

2. Related Work

2.1. Lattice-Based Signature Schemes. The existing lattice-based signature schemes are mostly based on short integer solution (SIS) problem first provided in [5] $(\mathbf{x}: \mathbf{As} = 0 \bmod q, \|\mathbf{s}\| \leq \beta, \mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{s} \in \mathbb{Z}^m)$. There are two frameworks to construct lattice-based signature schemes: hash-and-sign type [6, 7] and Fiat–Shamir type [8–11]. In schemes of hash-and-sign type, the signer uses trapdoor basis to compute preimage sampling function to create signature (σ) satisfying $\mathbf{A}\sigma = H(\boldsymbol{\mu}) \bmod q$, where H is a hash function. Unlike the hash-and-sign lattice-based signature framework, aborting technology is used in schemes of Fiat–Shamir type without trapdoor. This is the output of the signature $(\mathbf{z} = \mathbf{sc} + \mathbf{y})$ according to some probabilities (rejection sampling) or the norm of signature must be in a security range (filtering outputting), where

($\mathbf{c} = H(\mathbf{A}\mathbf{y} \bmod q, \boldsymbol{\mu})$) and ($\mathbf{y} \longleftarrow \mathcal{D}_{\mathbf{y}}$) ($\mathcal{D}$ is a Gaussian distribution or uniform distribution).

Furthermore, Gaussian distribution is utilized in preimage sampling and rejection sampling, which cannot resist partial side-channel attacks (see [12, 13]). Although, the author in [14] showed that an almost perfect implementation could resist these attacks, and the designed programme errors might occur.

2.2. Lattice-Based Linearly Homomorphic Signature. The first lattice-based linearly homomorphic signature scheme over $\mathbb{F}_2$ was proposed by Boneh and Freeman [15] in 2011, which was based on k–SIS problem, that is, finding a solution $\mathbf{s}$ satisfying $\mathbf{s} \notin \text{span}\{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ under giving ($A\mathbf{s}_i = 0 \bmod q$). In addition, the specific sign process is ($\mathbf{A}\sigma = H(\boldsymbol{\mu}) \bmod 2q$), where ($\mathbf{A} \in \mathbb{Z}_{2q}^{n\times m}$). Soon after, Wang et al. proposed an improved scheme [16] based on the general SIS problem, and the size of public key and signature is smaller than [15] by changing $2q$ into q. Compared to signature size, $(2m + 2m\log q + n)$ in scheme [15, 16] has the smaller size $(m\log q + n)$.

In fact, both of them are designed in terms of the hash-and-sign lattice-based signature framework [6], where Gaussian sampling is used inevitably. Meanwhile, the generation of trapdoor basis needs that lattice dimension is ($m \geq 5n\log q$) (see [17, 18]), which is larger than ($m \geq n\log q$) for SIS problem itself.

2.3. Our Contributions. In this paper, our scheme overcomes the drawbacks of existing schemes. Specifically, based on the SIS problem, we use filtering technology of Fiat–Shamir with aborting signature framework to design a new linearly homomorphic signature scheme over $\mathbb{F}_2$, and the advantages can be seen as follows:

(1) The signature size is smaller than existing lattice-based schemes. The signature of our scheme is $\sigma = (\mathbf{z}, \mathbf{h}, \tau)$, and the size is $(2m\log q + n)$, where $(m \geq n\log q)$. Since our design does not utilize preimage trapdoor sampling, the signature size is smaller than $(m'\log q + n)$ in [16] with the same n and q, where $(m' \geq 5n\log q)$. Here, we use a different lattice dimension m' to distinguish the difference in signature size.

(2) Our scheme can resist side-channel attacks. Using filtering technology, the masked element $\mathbf{y}$ is chosen uniformly at random under restriction ($\|\mathbf{y}\|_\infty \leq \gamma$), and $\mathbf{z}$ must satisfy the condition $\|\mathbf{z}\|_\infty \leq \gamma - \beta$; otherwise, $\mathbf{z} \longleftarrow \perp$ (aborting). Hence, the signature output can protect secret key, and the scheme can resist side-channel attacks without Gaussian sampling.

2.4. Organization of the Paper. We will provide two main technical descriptions to show how our scheme can have the above advantages in Section 3. Then, we propose the basic notations and definitions of linearly homomorphic signature in Section 4. We show the detailed design and security proof of our lattice-based linearly homomorphic signature in Section 5 and Section 6, respectively. In Section 7, we present efficiency comparisons. Finally, we give a conclusion and further work in Section 8. Data availability, conflicts of interest, and funding statement can be seen in the last three sections, respectively.

3. Technical Notes

In this part, we give detailed descriptions to show how we get a smaller signature size and the scheme can resist side-channel attacks.

3.1. Different Lattice Dimension Assumptions. As we know, given security parameter n, m influences the signature size directly. Thus, we want to reduce it. Fortunately, compared to hash-and-sign signature framework, the Fiat–Shamir signature framework has advantage in this aspect. We show the main reason as follows.

Definition 1. (the short integer solution problem SIS). Given m uniformly random elements ($\mathbf{a}_i \in \mathbb{Z}_q^n$), find a nonzero ($\mathbf{z} \in \mathbb{Z}^m$) of norm ($\|\mathbf{z}\| \leq \beta$) satisfying

$$\sum_{i=1}^{m} \mathbf{a}_i \mathbf{z}_i = 0 \in \mathbb{Z}_q^n. \tag{1}$$

Usually, it is denoted $\mathbf{A}\mathbf{z} = 0 \in \mathbb{Z}_q^n$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ forms the columns of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. To guarantee the hardness (existence of solution) of this problem, the parameters satisfy conditions $\sqrt{m} \leq \beta < q, m \geq n\log q, q \in \mathbb{Z}^+, n \geq 100$. Normally, people consider the inhomogeneous version of the SIS problem, which is to find a small solution of equation $\mathbf{A}\mathbf{z} = \mathbf{b}$.

To design Fiat–Shamir signature schemes, the lattice dimension m satisfies $m \geq n\log q$ enough. However, hash-and-sign type needs $m \geq 5n\log q$ to get trapdoor basis; thus, we provide the existing conclusion below.

Proposition 1 (see [6, 17, 18]). *Given any prime q and ($m \geq 5n\log q$), then there exists a PPT algorithm which outputs ($\mathbf{A} \in \mathbb{Z}_q^{n\times m}$) statistically close to uniform over ($\mathbb{Z}_q^{n\times m}$) and a full-rank set ($\mathbf{S} \subset \Lambda^{\perp}(\mathbf{A}), \|\mathbf{S}\| \leq m^{2.5}$) by input 1^n. Then, it further gets a good basis ($\mathbf{T} \subset \Lambda^{\perp}(\mathbf{A})$) satisfying ($\|\mathbf{T}\| \leq \|\mathbf{S}\|$).*

From what has been discussed above, we get the smaller m the better for the size of signature and secret key under the same n. Hence, we design a new scheme using Fiat–Shamir signature framework without trapdoor (basis).

3.2. Filtering Technology. Since the existing lattice-based linearly homomorphic signature schemes are based on the hash-and-sign signature framework in which they use preimage sampling function implemented by Gaussian sampling, the schemes cannot resist side-channel attacks. Hence, we unitize uniform sampling of Fiat–Shamir framework to generate signature.

The idea of filtering can be traced back to [10, 19], and it is formally provided in [20] to design a blind signature scheme. Here, we rewrite this lemma according to our construction.

Lemma 1. *For arbitrary* $\mathbf{a} \in \{\mathbf{x} \in \mathbb{Z}^k: \|\mathbf{x}\|_\infty \le \beta\}$ *and random* $(\mathbf{b} \longleftarrow \{\mathbf{x} \in \mathbb{Z}^k: \|\mathbf{x}\|_\infty \le \gamma\}, k = \Omega(n))$, *if* $(\gamma \ge \phi k\beta)$ *with* $(\phi \in \mathbb{N}^+)$, *then we have* $(Pr[\|\mathbf{a} - \mathbf{b}\|_\infty \le \gamma - \beta] > (1/((e^{(1/\phi)}) - o(1))))$.

Then, the repeat time of our scheme can be computed by $e^{(1/\phi)}$. Intuitively, bigger ϕ size is better. However, this value has influence on the size of $(\|\mathbf{a} - \mathbf{b}\|_\infty)$ directly; thus, it leads to increased communication costs. Hence, we can assign this value according to different efficiency requirements, which is a nice advantage in practice. According to [20], the authors provide the condition $(\phi = 4)$ is the best; then, the repeat time is no more than 2, which is also hold for our scheme.

In our scheme, the parameters (γ, β) satisfy $(\gamma, \beta < q)$. In addition, to facilitate comparisons with schemes [15, 16], we use $(\|\mathbf{a} - \mathbf{b}\|_\infty \le q)$ to compute signature size instead of $(\gamma - \beta)$ (see Table 1).

4. Preliminaries

4.1. Notions. We denote $(\mathscr{R} = \mathbb{Z})$. The elements in $\mathscr{R}$ (vector or matrix) are marked in bold, $\|\mathbf{y}\|$ is l_2 norm, and $\|\mathbf{y}\|_\infty$ is l_∞ norm. $(\mathbf{y} \longleftarrow \mathscr{D}_\mathbf{y})$ means that $\mathbf{y}$ is chosen according to some distribution $\mathscr{D}_\mathbf{y}$ (uniform or Gaussian) at random. If $\mathbf{y} \longleftarrow \mathscr{D}_\gamma$, it means that $\|\mathbf{y}\|_\infty \le \gamma$ using uniform sampling. Using Gaussian sampling, we denote $\mathbf{y} \longleftarrow \mathscr{D}_\sigma$, where σ is the standard deviation.

4.2. Definitions of Linearly Homomorphic Signature

Definition 2. Given a fixed ring $\mathscr{R}$, a linearly homomorphic signature over it contains a tuple of probabilistic polynomial-time algorithms (Setup, Sign, Verify, Combine) and the detailed descriptions can be seen as follows:

(1) Setup (n, params). It is a probabilistic algorithm that outputs (pk, sk) by inputting a security parameter n and other public parameters (params).

(2) Sign $(\boldsymbol{\mu}, sk, \tau)$. It is a probabilistic algorithm that outputs a valid signature σ by inputting secret key (sk), a basis vector $\boldsymbol{\mu}$ of message set $(\mathscr{M}\ (\boldsymbol{\mu} \in \mathscr{M} \subset \mathbb{F}_2^n))$, and a tag (or an identifier id) $(\tau \in \{0,1\}^*)$ of message $\mathscr{M}$.

(3) Verify $(\boldsymbol{\mu}, \text{pk}, \sigma, \tau)$. It is a deterministic algorithm that outputs a bit b by inputting the tuple $(\boldsymbol{\mu}, \text{pk}, \sigma, \tau)$. If σ is a valid signature of $\boldsymbol{\mu}$, the algorithm outputs $b = 1$ (accept); otherwise, $b = 0$ (reject).

(4) Combine $(\text{pk}, \tau, \{a_i, \boldsymbol{\mu}_i, \sigma_i\}_{i=1}^l, L)$. This algorithm outputs a valid combined signature (σ') of $(\boldsymbol{\mu}' = \sum_{i=1}^l a_i \boldsymbol{\mu}_i \bmod 2)$, where $(a_i \in \{0,1\}, l \le L)$. The parameter L is the maximum circuit depth.

In general, the security properties of a linearly homomorphic signature scheme contain correctness, unforgeability, and privacy. We will give the specific contents for them as follows:

(1) **Correctness**: the outputs from above algorithms Sign and Combine can be accepted by the Verify algorithm.

(2) **Unforgeability**: we will show a game between challenger $\mathscr{C}$ and a polynomial-time adversary $\mathscr{A}$.

(1) **Setup**: the challenger $\mathscr{C}$ runs algorithm $(\text{Setup}(n, \text{params}))$ to get (pk, sk) and gives (pk) to the adversary $\mathscr{A}$.

(2) **Sign queries**: the adversary $\mathscr{A}$ makes adaptive signature queries on k-dimensional subspaces $\mathscr{U}_i$ of message space $\mathscr{M}$, and he chooses a basis vectors $(\boldsymbol{\mu}_{i1}, \ldots, \boldsymbol{\mu}_{ik})$ for $\mathscr{U}_i$. For each subspaces $\mathscr{U}_i$, the challenger $\mathscr{C}$ chooses τ_i from $\{0,1\}^n$ at random and gives τ_i and j signatures $(\sigma_{ij} \longleftarrow \text{Sign}(\text{sk}, \tau_i, \boldsymbol{\mu}_{ij}))$ to the adversary $\mathscr{A}$, where $j = 1, 2, \ldots, k$.

(3) **Output**: the adversary $\mathscr{A}$ outputs a tag τ^*, a nonzero message $\mathscr{U}^*$, and a signature σ^*.

The adversary wins the game when his outputs satisfy the algorithm

$$\text{Verify}(\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1, \tag{2}$$

and this algorithm satisfies the following one of two conditions:

(1) Type 1. $(\tau^* \ne \tau_i)$ for all i.

(2) Type 2. $(\tau^* = \tau_i)$ but $(\mathscr{U}^* \notin \mathscr{U}_i)$.

Definition 3. A linearly homomorphic signature scheme (Setup, Sign, Verify, Combine) is unforgeability if the probability advantage of adversary winning above game is negligible with security parameter n. That is,

$$\left|\Pr\left[\text{Verify}(\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1 \mid \tau^* \ne \tau_i\right]\right| \le \varepsilon(n),$$
$$\text{or } \left|\Pr\left[\text{Verify}(\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1 \mid \tau^* = \tau_i, \mathscr{U}^* \notin \mathscr{U}_i\right]\right| \le \varepsilon(n). \tag{3}$$

(3) **Privacy**: a game between challenger $\mathscr{C}$ and a polynomial-time adversary $\mathscr{A}$ is shown as follows:

(1) **Setup**: the challenger $\mathscr{C}$ runs algorithm Setup $(n, (\text{params}))$ to get (pk, sk) and gives (pk) to the adversary $\mathscr{A}$.

(2) **Challenge**: the adversary $\mathscr{A}$ chooses two linear message subspaces $\mathscr{U}_0$ and $\mathscr{U}_1$ represented as vectors $(\boldsymbol{\mu}_1^{(b)}, \ldots, \boldsymbol{\mu}_k^{(b)})$ for $(b = 0, 1)$. In addition, he selects functions $(f_1, \ldots, f_s)$, which satisfy $(f_i(\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_k^{(0)}) = f_i(\boldsymbol{\mu}_1^{(1)}, \ldots, \boldsymbol{\mu}_k^{(1)}))$, where $(i = 1, 2, \ldots, s)$.

(3) **Response**: the challenger $\mathscr{C}$ chooses a random bit $(b \in \{0,1\})$, a tag $(\tau \in \{0,1\}^n)$, and signs the vector space $\mathscr{U}_b$. Then, the challenger uses the combine algorithm to generate signatures σ_i for $(f_i(\boldsymbol{\mu}_1^{(b)}, \ldots, \boldsymbol{\mu}_k^{(b)}))$ and sends σ_i to $\mathscr{A}$.

Table 1: Comparison.

	Public key size	Signature size	Dimension	Resist side-channel attacks
[15]	$(m'n + m\prime n \log q)$	$(2m' + 2m\prime \log q + n)$	$(m' \geq 5n \log q)$	×
[16]	$(m'n \log q)$	$(m' \log q + n)$	$(m' \geq 5n \log q)$	×
Ours	$(mn \log q)$	$(2m \log q + n)$	$(m \geq n \log q)$	√

(4) **Outputs**: $\mathcal{A}$ outputs a guess bit b'. If $b' = b$ holds, the adversary $\mathcal{A}$ succeeds in the game.

Definition 4. A linearly homomorphic signature scheme (Setup, Sign, Verify, Combine) is privacy if the probability advantage of adversary winning above game is negligible with security parameter n. That is, $|\Pr[b' = b] - (1/2)| \leq \varepsilon(n)$.

5. Our Lattice-Based Linearly Homomorphic Signature

Setup. $(\mathbf{A} \longleftarrow \mathbb{Z}_q^{n \times m})$ and $(\mathbf{T} = \mathbf{AS} \bmod q)$, where the public key is $(\mathbf{T}, \mathbf{A})$ and secret key is $\mathbf{S}$. We denote the hash function as $(H: \{0,1\}^* \longrightarrow \mathbb{Z}_q^m)$ and another hash function $(h_\alpha(\boldsymbol{\mu}) = \langle \alpha, \boldsymbol{\mu} \rangle \bmod q)$. Obviously, this function satisfies a property $(\sum_{i=1}^m a_i h_\alpha(\boldsymbol{\mu}_i) = h_\alpha(\sum_{i=1}^m a_i \boldsymbol{\mu}_i), a_i \in \mathbb{Z})$. Especially, we assume $(a_i \in \{0,1\})$ in our scheme as below.

Sign: we suppose the massage satisfies $(\boldsymbol{\mu} \in \mathbb{Z}_2^m)$ and choose a basis of it, that is, $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m\}$ (for the sake of design, we have assumed that it is a full-rank space). In addition, the used linear function is $(f(\boldsymbol{\mu}) = \sum_{j=1}^m a_j \boldsymbol{\mu}_j)$. Then, signer does the following steps:

(1) He chooses $(\mathbf{y}_i \longleftarrow_R D_\gamma^m)$ $(\gamma < q)$ and computes m vectors $(\alpha_i = H(\mathbf{A}\mathbf{y}_i \bmod q, \tau))$, where $(\tau \in \{0,1\}^m)$ is a tag of massage basis vector $(\boldsymbol{\mu}_j \ (1 \leq j \leq m))$.
(2) He computes $(h_{\alpha_i}(\boldsymbol{\mu}_j) = h_{ij} = \langle \alpha_i, \boldsymbol{\mu}_j \rangle \bmod q)$. Then, fixing the parameter j for any message $\boldsymbol{\mu}_j$, he denotes a vector $(\mathbf{h}_j = (h_{1j}, \ldots, h_{\mathrm{jj}}, \ldots, h_{\mathrm{mj}})^T)$.
(3) He computes $(\mathbf{z}_j = \mathbf{S}\mathbf{h}_j + \mathbf{y}_j)$. If $(\|\mathbf{z}_j\|_\infty \leq \gamma - \beta)$, output the signature $(\mathbf{z}_j, \mathbf{h}_j, \tau)$, or else, go to the first step. Here, $(\mathbf{y}_j = \mathbf{y}_i)$ holds.

Verify: the verifier verifies the conditions as follows:

(1) He computes $(\alpha_j = H(\mathbf{A}\mathbf{z}_j - \mathbf{T}\mathbf{h}_j \bmod q, \tau))$.
(2) He computes whether the equation $(h_{jj} = \langle \alpha_j, \boldsymbol{\mu}_j \rangle \bmod q)$ holds or not.
(3) $\|\mathbf{z}_j\|_\infty \leq \gamma - \beta$.

Combine: given public key $\mathbf{A}$ and an array $(a_j, \boldsymbol{\mu}_j, \mathbf{z}_j, \mathbf{h}_j)$ for $(j = 1, \ldots, m)$. This algorithm outputs signature $(\sum_{j=1}^m a_j \mathbf{z}_j, \sum_{j=1}^m a_j \mathbf{h}_j)$ of message $(\sum_{j=1}^m a_j \boldsymbol{\mu}_j)$.

6. Proof of Security

6.1. Correctness. Since correctness refers to two verifications from the outputs of **Sign** and **Combine** algorithms, we prove it one by one:

(1) The signature from **Sign** algorithm is valid. For each j, when the verifier receives the signature $(\mathbf{z}_j, \mathbf{h}_j, \tau)$, he computes

$$\begin{aligned} \boldsymbol{\alpha}_j &= H\left(\mathbf{A}\mathbf{y}_j \mod q, \tau\right) \\ &= H\left(\mathbf{A}\left(\mathbf{z}_j - \mathbf{S}\mathbf{h}_j\right) \mod q, \tau\right) \\ &= H\left(\mathbf{A}\mathbf{z}_j - \mathbf{A}\mathbf{S}\mathbf{h}_j \mod q, \tau\right) \\ &= H\left(\mathbf{A}\mathbf{z}_j - \mathbf{T}\mathbf{h}_j \mod q, \tau\right). \end{aligned} \tag{4}$$

Then, he computes whether the equation $(\langle \alpha_j, \boldsymbol{\mu}_j \rangle \bmod q = h_{\mathrm{jj}})$ holds or not.

(2) The signature from **Combine** algorithm is valid. We let matrix $\mathbf{H}$ be a composition of vectors $(\alpha_j (1 \leq j \leq m))$; that is, $(\mathbf{H} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T)$. Hence, we have $(\mathbf{h}_j = \mathbf{H}\boldsymbol{\mu}_j \bmod q)$. Since condition $(\alpha_j = H(\mathbf{A}\mathbf{y}_j, \tau))$ holds for each j, we only need to verify the linear property of $(\sum_{j'=1}^m a_{j'} h_{jj'})$ and linear bound of $(\sum_{j=1}^m a_j \mathbf{z}_j)$. At first, we consider the following equation:

$$\begin{aligned} \sum_{j'=1}^m a_{j'} \mathbf{h}_{j'} &= \sum_{j'=1}^m a_{j'} \mathbf{H}\boldsymbol{\mu}_{j'} \mod q \\ &= \mathbf{H}\left(\sum_{j'=1}^m a_{j'} \boldsymbol{\mu}_{j'}\right) \mod q. \end{aligned} \tag{5}$$

Thus, $(\sum_{j'=1}^m a_{j'} h_{jj'} = \langle \alpha_j, \sum_{j'=1}^m a_{j'} \boldsymbol{\mu}_{j'} \rangle \bmod q)$ holds because of $(h_{jj'} = \langle \alpha_j, \boldsymbol{\mu}_j' \rangle \bmod q)$ when $(\alpha_j = H(\mathbf{A}\mathbf{z}_j - \mathbf{T}\mathbf{h}_j \bmod q, \tau))$ holds. Next, we can see that the bound of our signature size is linear obviously. That is,

$$\left\| \sum_{j'=1}^m a_{j'} \mathbf{z}_{j'} \right\|_\infty \leq m \left\| \mathbf{z}_{j'} \right\|_\infty. \tag{6}$$

Hence, as long as this in equation holds, the signature is accepted.

6.2. Unforgeability

Theorem 1. *Our scheme is unforgeability if the lattice problem SIS is hard.*

Proof. We suppose that the defined unforgeability game is correctly performed between a challenger $\mathscr{C}$ and a polynomial-time adversary $\mathscr{A}$. In addition, $q_{H,h}$ and q_{sig} are the times of random oracle and signature oracle. Specifically, given the public key $(\mathbf{A}, \mathbf{T})$, $\mathscr{A}$ adaptively chooses some m–dimensional subspaces $\mathscr{U}_{i(n)}$ and chooses basis $(\boldsymbol{\mu}_{i1}, \ldots, \boldsymbol{\mu}_{\text{im}})$ for $\mathscr{U}_i$. To return m signatures to $\mathscr{A}$, the challenger makes query to above oracles and outputs $(\sigma_{i1}, \ldots, \sigma_{\text{im}})$ for the chosen basis.

When above game is finished, the adversary $\mathscr{A}$ outputs a tag τ^*, a nonzero message $\mathscr{U}^*$, and a signature σ^*. Next, we consider two types forgeries, respectively. □

Type 1. If $(\tau^* \neq \tau_i)$ for all i and $\text{Verify}((\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1)$ holds, the adversary has ability to solve the SIS problem. Suppose $\mathscr{A}$ has gotten $((\mathbf{z}_{\text{ij}}, \mathbf{h}_{\text{ij}}), j = 1, \ldots, m)$, then he chooses (α_{ij}^*) randomly and computes $(\mathbf{h}_{\text{ij}}^*)$. Hence, he can combine equations as follows:

$$\begin{aligned} \mathbf{z}_{\text{ij}} &= \mathbf{S}\mathbf{h}_{\text{ij}} + \mathbf{y}_{\text{ij}}, \\ \mathbf{z}_{\text{ij}}^* &= \mathbf{S}\mathbf{h}_{\text{ij}}^* + \mathbf{y}_{\text{ij}}. \end{aligned} \tag{7}$$

Thus, he gets $(\mathbf{z}_{\text{ij}} - \mathbf{z}_{\text{ij}}^* = \mathbf{S}(\mathbf{h}_{\text{ij}} - \mathbf{h}_{\text{ij}}^*))$ and computes

$$\begin{aligned} \mathbf{A}\left(\mathbf{z}_{\text{ij}} - \mathbf{z}_{\text{ij}}^*\right) &= \mathbf{A}\mathbf{S}\left(\mathbf{h}_{\text{ij}} - \mathbf{h}_{\text{ij}}^*\right) \\ \mathbf{A}\Delta\mathbf{z} &= \mathbf{T}\Delta\mathbf{h}. \end{aligned} \tag{8}$$

Notice that if $(\Delta\mathbf{h} \neq 0)$, the adversary forgeries a signature if and only is he can solve SIS instance. Furthermore, since $|\Pr[\text{Verify}(\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1 \mid \tau^* \neq \tau_i]| = (|\Pr[\Delta\mathbf{h} = 0 \mid \alpha_{ij}^* \longleftarrow_R \mathbb{Z}_q^m]| = (1/q_{H,h}) + (1/q_{\text{sig}}) \leq \varepsilon(n))$, we can see that the probability of success for this type of attack is negligible.

Type 2. If conditions $(\tau^* = \tau_i)$ and $(\mathscr{U}^* \notin \mathscr{U}_i)$ hold, the adversary also can solve the SIS problem. In this case, for the same hash value (α_{ij}), the adversary chooses massage space $\mathscr{U}^*$ and one of its basis $(\boldsymbol{\mu}_j^* \notin \mathscr{U}_i)$. Then, he computes $(\langle \alpha_{ij}, \boldsymbol{\mu}_j^* \rangle \bmod q)$ and $\mathbf{h}_j^*$. Since (α_{ij}) is a fixed value, the adversary can compute:

$$\begin{aligned} \boldsymbol{\alpha}_j &= H\left(\mathbf{A}\mathbf{y}_{\text{ij}}, \tau\right) \\ &= H\left(\mathbf{A}\mathbf{z}_{\text{ij}} - \mathbf{T}\mathbf{h}_{\text{ij}}, \tau\right) \\ &= H\left(\mathbf{A}\mathbf{z}_j^* - \mathbf{T}\mathbf{h}_j^*, \tau\right). \end{aligned} \tag{9}$$

Then, he obtains equation $(\mathbf{A}(\mathbf{z}_{\text{ij}} - \mathbf{z}_j^*) = \mathbf{T}(\mathbf{h}_{\text{ij}} - \mathbf{h}_j^*))$, which is marked as $(\mathbf{A}\Delta\mathbf{z} = \mathbf{T}\Delta\mathbf{h})$.

Since $|\Pr[\text{Verify}(\mathscr{U}^*, \text{pk}, \sigma^*, \tau^*) = 1 \mid \tau^* = \tau_i, \mathscr{U}^* \notin \mathscr{U}_i]| = |\Pr[\Delta z = 0 \mid z_j^* \longleftarrow_R \mathbb{Z}_q^m, \boldsymbol{\mu}_j^* \notin \mathscr{U}_i]| = (1/q_{\text{sig}}) \leq \varepsilon(n)$, the adversary cannot forgery a valid signature; otherwise, he is able to search a SIS problem solution. In addition, the reason why he does not use hash oracle is that the condition $(\tau^* = \tau_i)$ determines this oracle has the same input.

6.3. Privacy

Theorem 2. *Our scheme is privacy.*

Proof. According to the definition of privacy game, challenger $\mathscr{C}$ and adversary $\mathscr{A}$ firstly finish the setup step. Then, $\mathscr{A}$ chooses two basis vectors $(\boldsymbol{\mu}_1^{(b)}, \ldots, \boldsymbol{\mu}_m^{(b)})$ from message spaces $\mathscr{U}_b$, where $(b = 0, 1)$. At the same time, he selects linear functions $(f_i(i = 1, 2, \ldots, s(n)))$ satisfying $(f_i(\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_m^{(0)}) = f_i(\boldsymbol{\mu}_1^{(1)}, \ldots, \boldsymbol{\mu}_m^{(1)}))$. Specifically, for a fixed i, the condition $(\sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(0)} = \sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(1)})$ holds.

When challenger $\mathscr{C}$ obtains $(\boldsymbol{\mu}_1^{(b)}, \ldots, \boldsymbol{\mu}_m^{(b)})$, he chooses b and $\tau \in \{0, 1\}^m$ randomly. Then, he signs basis vectors under f_i and outputs the signature σ_i computed using the Combine algorithm. And he sends σ_i to $\mathscr{A}$.

Finally, the adversary outputs a guess bit b'. Next, we will show that it is negligible for him to succeed in this game. That is, $|\Pr[b' = b] - 1/2| \leq \epsilon(n)$.

In fact, we let two distinguished signatures are $(\sigma_i^{(0)} = (\sum_{j=1}^m a_j^i \mathbf{z}_j^{(0)}, \sum_{j=1}^m a_j^i \mathbf{h}_j^{(1)}))$ of message $(\sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(0)})$, $(\sigma_i^{(1)} = (\sum_{j=1}^m a_j^i \mathbf{z}_j^{(1)}, \sum_{j=1}^m a_j^i \mathbf{h}_j^{(1)}))$ of message $(\sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(1)})$, and condition is $(\text{Con} = \sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(0)} = \sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(1)})$. Then, we have

$$\begin{aligned} \mathbf{H}\left(\sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(0)}\right) = \mathbf{H}\left(\sum_{j=1}^m a_j^i \boldsymbol{\mu}_j^{(1)}\right) \mod q \left(\sum_{j=1}^m a_j^i \mathbf{h}_j^{(0)}\right) \\ = \left(\sum_{j=1}^m a_j^i \mathbf{h}_j^{(1)}\right) \mod q, \end{aligned} \tag{10}$$

where $\mathbf{H} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T$. Hence, we get

$$\left|\Pr\left[\sigma_i^{(0)} \,\middle|\, \text{Con}\right] - \Pr\left[\sigma_i^{(1)} \,\middle|\, \text{Con}\right]\right| \leq \varepsilon(n). \tag{11}$$

Because $(\sigma_i^{(0)})$ and $(\sigma_i^{(1)})$ are indistinguishable under (Con) for the adversary, we show $(|\Pr[b' = b] - (1/2)| \leq \varepsilon(n))$ holds. □

7. Efficiency Comparisons

In schemes of hash-and-sign type [15, 16], the signer uses trapdoor basis to compute preimage sampling function to create signature σ. However, the lattice dimension m must satisfy $(m \geq 5n \log q)$ to get a trapdoor basis (see [17, 18]) and a larger m will result in a larger size of public key and signature.

Our design using the Fiat–Shamir signature framework without trapdoor has smaller public key and signature sizes mainly because $(m \geq n \log q)$ is enough. Then, compared to public key size $(m' n \log q \approx 5n^2 (\log q)^2)$ in [16], our result $(mn \log q \approx n^2 (\log q)^2)$ is smaller than theirs. In addition, signature size of our scheme $(2m \log q + n \approx 2n(\log q)^2 + n)$ is also smaller than $(m' \log q + n \approx 5n(\log q)^2 + n)$ in [16].

Furthermore, preimage sampling function utilizes Gaussian distribution which cannot resist partial side-channel attacks, and we use uniform distribution of aborting technology to resist such attacks effectively. The detailed comparisons can be seen in Table 1.

8. Conclusion and Further Work

8.1. Conclusion. In this paper, we provide a new lattice-based linearly homomorphic signature scheme over $\mathbb{F}_2$ based on the SIS problem. Since we use Fiat–Shamir signature framework instead of hash-and-sign signature framework to design this signature scheme, we do not need to construct a trapdoor basis, and then the whole design is simpler than the existing schemes. At the same time, without the trapdoor basis, our scheme has the smallest public key size $(n^2(\log q)^2)$ and signature size $(2n(\log q)^2 + n)$ in the existing schemes because of parameter m satisfying $(m \approx n \log q)$ rather than $(m \approx 5n \log q)$. In addition, under the Fiat–Shamir framework, the use of filtering technology with uniform sampling can resist side-channel attacks.

8.2. Further Work. Decreasing interaction and storage costs is the main work of our future research. In fact, new compression skill and decreasing parameters m and n must be improved efficiency. Since our scheme can be designed on R-SIS directly, we no longer give a special scheme. That is, if each element chosen forms the ring $(\mathscr{R} = (\mathbb{Z}[x]/f(x)))$ or $(\mathscr{R}_q = (\mathbb{Z}_q[x]/f(x)))$, where $(f(x) = x^n + 1)$, n is power of 2, and q is prime, then the parameter m only needs to satisfy $m \approx \log q$, rather than $m \approx n \log q$ for SIS. Hence, it can improve the efficiency.

Specifically, we focus on [9], where it also uses filtering technology (uniform distribution), and special compression methods are used. Meanwhile, module lattice form brings an advantage to parameters m and n, which can be the 1/4 of existing set. In addition, this form can be transformed into lattice hard problem over ring (R-SIS) and general problem (SIS) by setting relative parameters.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (grant nos. 61602287, 11531008, 11771252, and 61632020), State Key Program of National Natural Science of China, Natural Science Foundation of Shandong Province (grant no. ZR2017MF021), Major Innovation Project of Science and Technology, Shandong (grant no. 2018CXGC0702), Primary Research & Development Plan of Shandong Province (grant no. 2018GGX101037), National Innovation Demonstration Zone Development and Construction Fund Project of Shandong Peninsula (grant no. S190101010001), Innovative Research Team in University by Ministry of Education (grant no. IRT16R43), and Taishan Scholars Project.

References

[1] D. Boneh, D. M. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: signature schemes for network coding," in *Proceedings of the International Conference on Practice and Theory in Public Key Cryptography PKC*, S. Jarecki and G. Tsudik, Eds., pp. 68–87, Irvine, CA, USA, March 2009.

[2] D. X. Charles, K. Jain, and K. E. Lauter, "Signatures for network coding," *IACR Cryptology*, vol. 25, 2006, http://eprint.iacr.org/2006/025.

[3] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in *Proceedings of the International Conference on Practice and Theory in Public Key Cryptography PKC*, P. Q. Nguyen and D. Pointcheval, Eds., Paris, France, May 2010.

[4] F. Zhao, T. Kalker, M. Médard, and K. J. Han, "Signatures for content distribution with network coding," in *Proceedings of the IEEE International Symposium on Information Theory ISIT*, pp. 556–560, IEEE, Cambridge, MA, USA, July 2007.

[5] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, G. L. Miller, Ed., pp. 99–108, ACM, Philadelphia, PA, USA, May 1996.

[6] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, C. Dwork, Ed., pp. 197–206, ACM, Victoria, Canada, May 2008.

[7] P.-A. Fouque, J. Hoffstein, and P. Kirchner, "Falcon: fast-fourier lattice-based compact signatures over nuru," in *Post-quantum Cryptography, NIST, Round 2 Summissions*, Springer, Berlin, Germany, 2018.

[8] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal Gaussians," in *Proceedings of the 33rd Annual Cryptology Conference (CRYPTO 2013)*, R. Canetti and J. A. Garay, Eds., Santa Barbara, CA, USA, August 2013.

[9] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: digital signatures from module lattices," *IACR Cryptology*, vol. 633, 2017.

[10] V. Lyubashevsky, "Fiat-Shamir with aborts: applications to lattice and factoring-based signatures," in *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2009)*, M. Matsui, Ed., Tokyo, Japan, December 2009.

[11] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)*, D. Pointcheval and T. Johansson, Eds., Cambridge, UK, April 2012.

[12] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom, "Flush, gauss, and reload—a cache attack on the BLISS lattice-based signature scheme," in *Proceedings of the 18th International Conference Cryptographic Hardware and Embedded Systems (CHES 2016)*, B. Gierlichs and A. Y. Poschmann, Eds., Santa Barbara, CA, USA, August 2016.

[13] P. Pessl, "Analyzing the shuffling side-channel countermeasure for lattice-based signatures," in *Proceedings of the 17th International Conference on Cryptology in India Progress in Cryptology INDOCRYPT 2016*, O. Dunkelman and S. K. Sanadhya, Eds., Kolkata, India, December 2016.

[14] D. Micciancio and M. Walter, "Gaussian sampling over the integers: efficient, generic, constant-time," in *Proceedings of the 37th Annual International Cryptology Conference Advances in Cryptology (CRYPTO 2017)*, J. Katz and H. Shacham, Eds., , Santa Barbara, CA, USA, August 2017.

[15] D. Boneh and D. M. Freeman, "Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures," in *Proceedings of the International Conference on Practice and Theory in Public Key Cryptography PKC*, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., pp. 1–16, Taormina, Italy, March 2011.

[16] F. H. Wang, H. U. Yupu, and B. C. Wang, "Lattice-based linearly homomorphic signature scheme over binary field," *Science China (Information Sciences)*, vol. 11, pp. 238–246, 2013.

[17] M. Ajtai, "Generating hard instances of the short basis problem," in *Proceedings of the 26th International Colloquium Automata, Languages and Programming (ICALP'99)*, J. Wiedermann, P. van Emde Boas, and M. Nielsen, Eds., Prague, Czech Republic, July 1999.

[18] D. Micciancio and S. Goldwasser, "Complexity of lattice problems-a cryptograhic perspective," *The Kluwer International Series in Engineering and Computer Science*, Springer, Berlin, Germany, 2002.

[19] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," in *Proceedings of the 11th International Workshop on Practice and Theory in Public-Key Cryptography (PKC)*, pp. 162–179, Springer, Barcelona, Spain, March 2008.

[20] M. Rückert, "Lattice-based blind signatures," in *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pp. 413–430, Springer, Singapore, December 2010.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8849902, 8 pages
https://doi.org/10.1155/2020/8849902

Research Article

An Antiforensic Method against AMR Compression Detection

Diqun Yan,[1,2] **Xiaowen Li**,[1] **Li Dong**,[1] **and Rangding Wang**[1]

[1]*Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China*
[2]*Guangdong Key Laboratory of Intelligent Information Processing and Shenzhen Key Laboratory of Media Security, Shenzhen University, Shenzhen 518060, China*

Correspondence should be addressed to Diqun Yan; yandiqun@nbu.edu.cn

Received 24 May 2020; Revised 23 July 2020; Accepted 19 August 2020; Published 2 September 2020

Academic Editor: Zhihua Xia

Copyright © 2020 Diqun Yan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Adaptive multirate (AMR) compression audio has been exploited as an effective forensic evidence to justify audio authenticity. Little consideration has been given, however, to antiforensic techniques capable of fooling AMR compression forensic algorithms. In this paper, we present an antiforensic method based on generative adversarial network (GAN) to attack AMR compression detectors. The GAN framework is utilized to modify double AMR compressed audio to have the underlying statistics of single compressed one. Three state-of-the-art detectors of AMR compression are selected as the targets to be attacked. The experimental results demonstrate that the proposed method is capable of removing the forensically detectable artifacts of AMR compression under various ratios with an average successful attack rate about 94.75%, which means the modified audios generated by our well-trained generator can treat the forensic detector effectively. Moreover, we show that the perceptual quality of the generated AMR audio is well preserved.

1. Introduction

AMR audio codec [1] is one of the most popular audio codec standards, which is optimized for speech signals and encodes narrowband (200–3400 Hz) signals, with sampling frequency of 8000 Hz [2]. As more and more AMR audio appears as evidence in the forensics scene, it is of extreme importance to verify their integrity [3]. Generally, to manipulate an AMR audio, attacker should decompress it into raw waveform first and then do the forgery operations and decompress it into AMR format. The double compressed audio becomes questionable because the manipulated audio is always through the double compression. In the past decade, many forensic techniques have been proposed to detect compression history of AMR audios based on traditional methods [3–5] and deep learning methods [2, 6, 7]. To represent the difference of single compressed audios and double compressed audios, traditional AMR compression detection techniques rely on low-level acoustic features such as sub-band energy and linear prediction coefficients (LPCs), which acquire professional acoustic knowledge. Recently, deep learning methods are gaining popularity in forensic research studies, which can capture the highly complex feature from a raw sample by training large-scale sample data with a neural network.

However, as many forensic techniques are proposed to detect the integrity of digital file, some antiforensic methods have also been proposed to expose the shortcomings and weakness of existing forensic techniques and thus help investigators better address the weaknesses and improve their forensic techniques. For example, Fontani et al. [8] firstly presented an antiforensic method of median filtering (MF), which made MF images undetectable by the MF detectors [9–11] while keeping the image quality in a good PSNR. Luo et al. [12] applied a GAN framework to improve the quality of JPEG images and fool the JPEG compression detectors successfully. Chen et al. [13] used the legacy traces of a designated camera to generate a forged image that can deceive the existing camera identification techniques successfully. Kim et al. [14] adopted a deep convolutional neural network (DCNN) to remove the forensic traces from MF images and effectively recover the MF images visually similar to the original image. Li et al. [15] modified the forensic traces using a data-driven manner to mislead the results of

three advanced audio source identification techniques [16–18].

These antiforensic methods have a little consideration about exposing the weakness of the robustness of AMR compression detection. Generally speaking, as more and more AMR audio appears as evidence in forensics scene, it is important to help the investigators to address the weakness of AMR compression detectors. Therefore, in this paper, we propose an antiforensic method utilizing a GAN framework which comprised of two networks: a generator and a discriminator. The generated data can statistically model the distribution of real data [19]. To improve the perceptual quality of the double compressed audio and remove the artifacts introduced by AMR compression procedure, we adopt the GAN to modify the double compressed audios to avoid forensic detection. For building our antiforensic attack, we design the architecture of GAN and the loss functions. In particular, three state-of-the-art detectors of AMR compression have been selected as the attack target to evaluate the performance of our method.

The rest of this paper is organized as follows. In Section 2, we introduce the related work of forensic method of AMR compression and the GAN framework. The detail of our proposed GAN framework has been provided in Section 3. Section 4 presents the experimental settings and extensive experiments against three AMR compression detectors. Conclusions are given in Section 5.

2. Related Work

In this section, we briefly introduce three advanced detection methods, which are considered as attack targets. Additionally, the GAN framework is also briefly reviewed.

2.1. Detection of AMR Compression. In general, traditional detection of AMR compression consists of two primary steps: feature extraction and model classification.

As the first work of the detection of AMR compression, Shen et al. [3] used the traditional acoustic features including average sub-band frequency energy ratio, average low-frequency sub-band energy ratio, bispectrum features, and linear prediction spectrum to represent the difference caused by AMR compression. And a standard SVM modelling technique was employed for classification. They achieved an accuracy about 87% for detecting the single compressed audio from the double one.

In [2], Luo et al. adopted an autoencoder network for automatic feature extraction. They demonstrated that the deep features differ greatly between the single compressed audio and the double one which were extracted from a well-trained autoencoder. And they designed a majority voting strategy for classification.

In [6], the authors delved into the stack autoencoder (SAE) network for obtaining better deep features in the AMR compression forensic task. Then, they applied a universal background model-Gaussian mixture model (UBM-GMM) for the identification of compression history. They improved the classification accuracy to 98% on the TIMIT [20] database.

2.2. Generative Adversarial Network. The generative adversarial network (GAN) was firstly proposed by Goodfellow et al. [21] for generating realistic images. In GAN, two networks are training against each other in a min-max two-player game. In their iterative training, the purpose of generator G is to capture the distribution of real data and that of discriminator D is to classify a sample that came from the real database rather than generated by G. The generator G tries to maximize the probability of making the discriminator D mistakenly classify the generated data as real, while the discriminator guides the generator to produce a more realistic sample. Generally, the adversarial training process can be denoted as a min-max game and it will be optimized by the loss function L_{GAN} as follows:

$$L_{\mathrm{GAN}} = \sum_{x} [\log(D(x))] + \sum_{z} [\log(1 - D(G(z)))], \quad (1)$$

where x denotes the real data and z denotes the random noise similar to x after the adversarial training of the generator G and discriminator D. In the training process, the purpose of G is to minimize the loss value while that of D is to maximize it.

Recently, GAN has gained growing popularity in various fields because of its effective generative capability. In this work, the GAN framework is assumed as the reverse procedure of AMR compression to improve the perceptual quality of double compressed audio and remove the forensic artifacts. Specifically, the generator and the discriminator can be regarded as an antiforensic model and AMR compression detector, respectively. Hence, the adversarial concept is suitable for antiforensic task in the AMR compression detection.

3. Proposed Antiforensic Framework

In this section, we briefly introduce three advanced detection methods, which are considered as attack targets. Additionally, the GAN framework is also briefly reviewed.

x_{db} is firstly sent into the generator to get a falsified audio x'_{db}. x'_{db} and x_{org} selected from the uncompressed audio are further fed into the discriminator for classification. Then, by freezing the parameters of discriminator, the loss from D will be fed back to G, which is represented by the dotted lines.

3.1. Overall Architecture. The overall goal of our attack is to remove the artifacts left by the AMR compression so that the resultant audio can fool the detectors. To deploy a successful attack, the generated audio should be decompressed back to AMR format because many investigators only accept the AMR file before the detection. Thus, the generated audio x'_{db} must statistically model the distribution of original audio x_{org} so that the decompressed ones will be similar to the single compressed audio x_{sg}.

As shown in Figure 1, the proposed framework consists of a generator G and a discriminator D. To remove the artifacts left by the compression, G is used to generate the falsified audio x'_{db} by adding a generated perturbation into x_{db}. The discriminator D is designed to distinguish an original audio x_{org}, which is never through compression from a falsified audio x'_{db}. In the adversarial training of G and D, G is encouraged to learn how to minimize the difference between x'_{db} and x_{org} and optimize the parameters to achieve a better performance in generating good perceptual quality of x'_{db}.

3.2. Architecture of Proposed Framework

3.2.1. Generator. Generator is used to generate the anti-forensic audios. In this framework, we use the SEGAN [22] as a reference architecture to design our adversarial network, which has been effectively applied in speech enhancement. As shown in Figure 2, the generator gets x_{db} (size = 1×8000) as the input and consists of 7 convolutional groups and 7 corresponding deconvolutional groups.

Each convolutional group includes a convolutional layer with 64 filters with 1×30 kernels and stride = 2, whereafter a batch normalization (BN) layer which can stabilize the training process makes the generated audios more realistic. And the Leaky-ReLU is chosen as the activation function. The deconvolutional group is constituted of a deconvolutional layer which is set up as the convolutional group, followed by a BN layer and ReLU as the activation function. To reconstruct the details of audio and diminish the loss when information flows through convolutional and deconvolutional groups, we apply the skip connection in the generator, which can make the convolutional group's output flow to its corresponding deconvolutional group. The skip connection can make the generator have a better performance, as the gradients can flow deeper through the skip connection without suffering much vanishing [23]. And the sigmoid activation is added to restrict the output for classification.

3.2.2. Discriminator. Since the key advantage of GAN is iterative training to obtain a better performance in generating samples, it seems that the architecture of D is a very important constraint to our framework. The discriminator D is intended to classify x_{org} and x'_{db} and force the generated audios to deceive the detector. Hence, the discriminator must perform well in distinguishing x_{org} and x'_{db}. Therefore, we build a CNN architecture for D. As shown in Figure 3, the discriminator is designed as a compression detector based on CNN. It comprises 6 convolutional groups and is followed by a group consisting of a global average pool layer. At the end of the network, a dense layer coupled with a softmax activation function is placed to output the categorical probability.

Before the iterative training, we firstly test the capability of the designed discriminator to distinguish the original audio x_{org} from double compressed audio x_{db}. Then, we test the capability of the designed discriminator in a sub-dataset including 6000 original audios selected from TIMIT database and its double compressed audios with a compression bit rate randomly selected from {4.75 kbps, 5.15 kbps, 5.9 kbps, 6.7 kbps, 7.4 kbps, 7.95 kbps, 10.2 kbps, 12.2 kbps}. The sub-dataset was then divided into training (70%) and validation (30%). The accuracy of the discriminator model is shown in Figure 4. It is observed that our designed discriminator achieves a good performance.

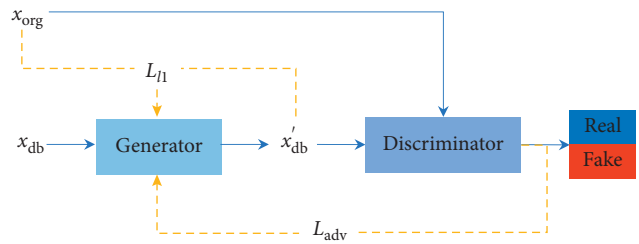


Figure 1: Overall structure of the proposed framework.

3.3. Loss Functions. In this section, we demonstrate the loss functions for the two networks. To achieve the goal of antiforensics, the generator G should be capable to learn how to minimize the difference of the modified double compressed audio x'_{db} and the original audio x_{org}, while maintaining an acceptable perceptual quality. In this work, we define the loss of generator L_G as

$$L_G = \alpha L_{l1} + \beta L_{adv}, \tag{2}$$

where L_{l1} represents the perceptual loss of x'_{db}, L_{adv} denotes the adversarial loss calculated from D, and α, β are the weights to balance the importance of L_{l1} and L_{adv}.

Considering that the attack needs to introduce lesser perceptual artifacts to improve the forensic undetectability, we employ the perceptual loss L_{l1} for improving the quality of x'_{db}. L_{l1} is defined as

$$L_{l1} = \sum_{i=1}^{N} \left(x_{org_i} - G\left(x_{db_i}\right)\right), \tag{3}$$

where $G(\cdot)$ presents the output of G, and N and i represent the batch size and the position of x in this batch, respectively.

Then, the adversarial loss L_{adv} is designed to force G to have a better performance in the iterative training. We define the L_{adv} as

$$L_{adv} = \sum_{i=1}^{N} \log\left(1 - D\left(G\left(x_{db_i}\right)\right)\right), \tag{4}$$

where $G(\cdot)$ denotes the class probabilities of the modified audio x'_{db} calculated by D.

In this adversarial task, for forcing G to modify x_{db} similar to x_{org}, D should have the ability to detect the original audio correctly from the decompressed x_{org} or the generated x'_{db}. Therefore, L_D is defined as follows:

$$L_D = \sum_{i=1}^{N} \left[\log\left(1 - D\left(G\left(x_{db_i}\right)\right)\right) - \log\left(D\left(x_{org_i}\right)\right)\right]. \tag{5}$$

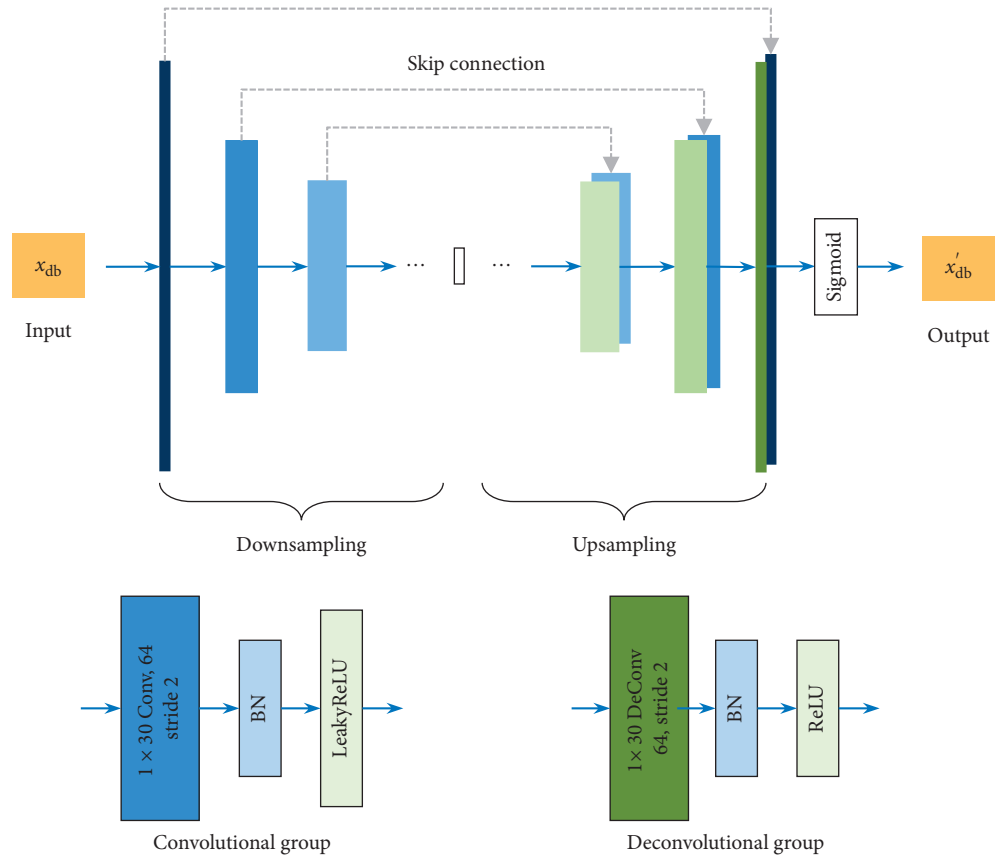


Figure 2: Architecture of the generator.

4. Experimental Results

In this section, we evaluate our antiforensic method against the three advanced forensic techniques [2, 3, 6]. First, we create an audio database designed especially for the experiment. Then, successful attack rate (SAR) is used to perform the forensic undetectability of our antiforensic audios and perceptual evaluation of speech quality (PESQ) [24] is adopted to present the quality of our antiforensic audios.

4.1. Database. TIMIT [20] is a typical speech database which consists of 630 speakers from different dialects of American English (192 females and 432 males) and each speaker reads ten sentences which are approximately three seconds. At first, to build the forensic database, we use the AMR codec to obtain the single compression audio from TIMIT database, with a random compression bit rate selected from {4.75 kbps, 5.15 kbps, 5.9 kbps, 6.7 kbps, 7.4 kbps, 7.95 kbps, 10.2 kbps, 12.2 kbps}. Then, we decode and recompress the AMR audios to get the double compressed AMR audio with random bit rates also selected from 4.75 to 12.2 kbps.

In the experiments, we first split those audios into 1 s clips and randomly divide those clips into the train set and test set. Therefore, we obtain 12000 1 s training audios and 6900 testing audios. Then, three detectors [2, 3, 6] are trained using the train set, and the average detection accuracies in test set are 87.52%, 92.60%, and 98.54%, respectively, which are essentially in agreement with the results reported in their works.

4.2. Experimental Setup and Evaluation Metrics

4.2.1. Experimental Setup. We train our network on patch sized audios with the pairs of sets: $\{x_{org}, x_{db}\}$. Considering the audio might be split into different sizes by the investigator before the detection, we stitch all 1 s audios x'_{db} to obtain more audios with difference sizes, including 13800 0.5 s clips, 6900 1 s clips, 3450 2 s clips, and 2300 3 s clips. Then, we compressed x'_{db} back to AMR format with random bit rates chosen from 4.75 to 12.2 kbps.

Adam [25] is adopted as the optimizer with a learning rate of 1×10^{-4} for G and 5×10^{-6} for D. Before the iterative training, we perform the generator training with batch size = 64 and weight terms of $\alpha = 1000$ and $\beta = 0$ for 5 epochs. Then, G and D are trained iteratively for 30 epochs with weight terms of $\alpha = 1000$ and $\beta = 1$, with an iteration ratio of 1 : 5, which gives the discriminator more iterations to get a better performance.

4.2.2. Evaluation Metrics. The successful attack rate (SAR) is used as the evaluation metric, which could well represent the forensic undetectability of our antiforensic audio. We define the SAR as

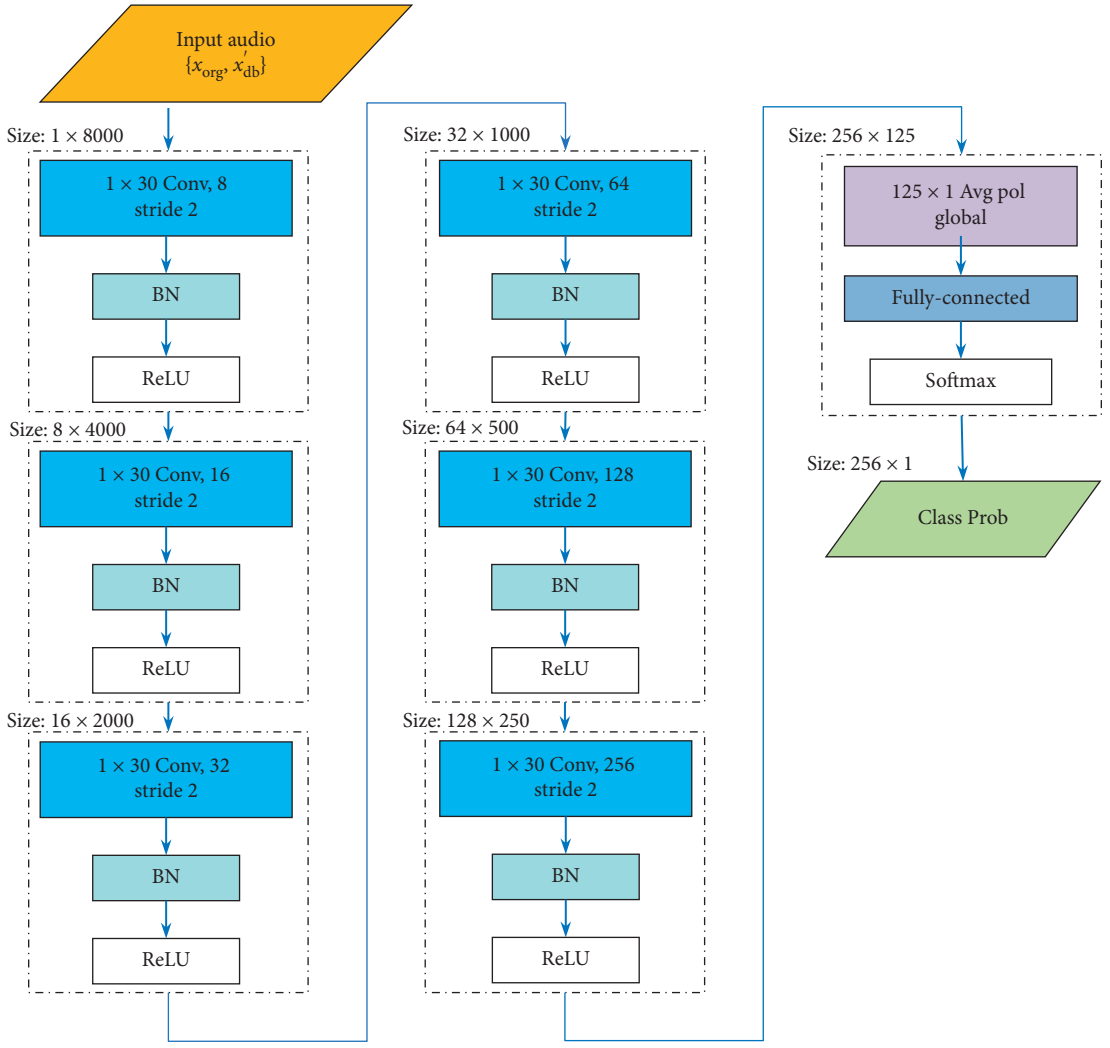


Figure 3: Architecture of the discriminator.

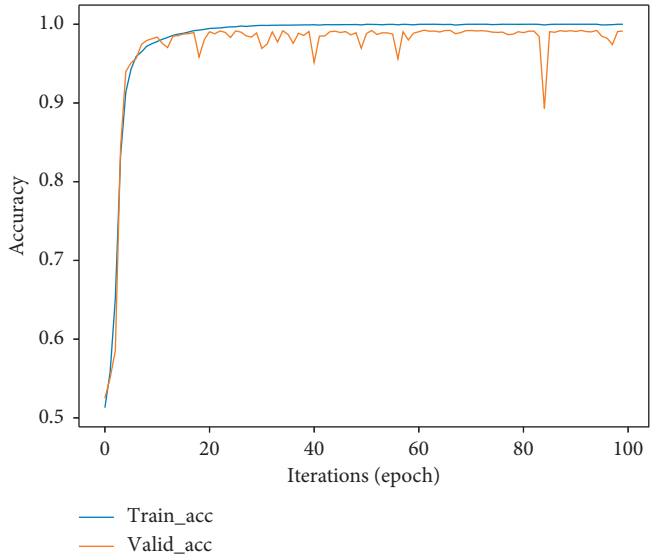


Figure 4: Accuracy in the training and validation performance of the designed discriminator.

Table 1: Successful attack rate (SAR) of antiforensic double AMR audio clips (%).

Size (s)	Method	Bit rates (kbps)								Average
		4.75	5.15	5.9	6.7	7.4	7.95	10.2	12.2	
0.5	[2]	97.07	93.96	98.41	95.28	96.52	97.40	96.88	97.59	96.63
	[3]	95.82	92.74	93.95	94.26	93.53	96.83	91.75	94.85	94.20
	[6]	94.95	96.49	92.60	97.25	96.55	91.64	98.28	93.75	95.69
1	[2]	97.66	98.05	98.15	91.72	95.78	98.42	92.17	94.82	95.82
	[3]	98.33	97.94	98.30	93.69	96.72	96.59	96.08	91.01	96.21
	[6]	94.05	96.88	92.20	90.63	86.08	91.26	91.73	90.23	91.63
2	[2]	96.12	90.95	97.17	91.56	95.86	98.49	98.12	97.32	95.70
	[3]	96.23	99.35	98.59	93.41	92.32	91.34	96.91	93.27	95.15
	[6]	94.77	92.88	95.26	89.14	89.00	95.25	90.19	96.85	92.91
3	[2]	98.86	96.06	93.32	98.05	95.35	93.36	85.95	96.74	94.71
	[3]	97.57	96.37	94.72	98.45	97.51	98.05	96.74	93.19	96.68
	[6]	93.36	96.61	94.72	92.20	93.86	91.56	94.14	92.53	93.62

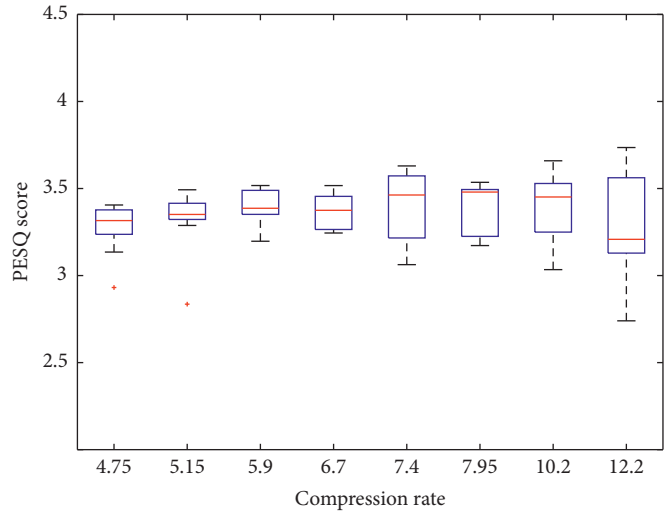


Figure 5: PESQ score of box plots calculated by antiforensic audios and single compressed audios via same compression bit rate.

$$L_D = \frac{1}{N}\sum_{i=1}^{N}\left(F\left(\mathrm{AF}_{x_{\mathrm{db}_i}}\right)\right), \tag{6}$$

where AF_x_{db} represents the audio decompressed with each bit rate selected from 4.75 to 12.2 kbps and $F(\cdot)$ is the classification result of forensic detector, that is, $F(\mathrm{AF}_{x_{\mathrm{db}}}) = 1$ while AF_x_{db} has been misclassified as x_{sg} and 0 otherwise.

Meanwhile, we apply the PESQ to test the perceptual quality of the antiforensic audio AF_x_{db}. PESQ is an industry-standard methodology for the assessment of speech quality. The range from −0.5 to 4.5 is the default PESQ score range, and higher score means better perceptual quality.

4.3. Experimental Performance and Analysis. We perform our attack on three advanced forensic methods [2, 3, 6]. Specifically, for each clip in the testing set, we generate a copy of it with the well-trained generator and then decompress the copy with eight different bit rates.AF_x_{db}. Finally, three trained detectors are used to classify our antiforensic audios.

As shown in Table 1, the experimental results are in line with expectations. The antiforensic audios AF_x_{db} can significantly deceive the three advanced AMR compression detectors, and the SAR of AF_x_{db} is significantly achieved with an average rate about 94.71% which means the forensic techniques cannot distinguish the antiforensic audios correctly. Obviously, our method can significantly make x_{db} undetectable by the forensic techniques.

To measure the quality of our antiforensic audios, we compute the PESQ score of AF_x_{db} comparing the original audios. As shown in Figure 5, it is obvious that our antiforensic audios can retain good perceptual quality and the PESQ values of most AF_x_{db} are over 3.3 compared with x_{sg}, which means that our method can improve the perceptual quality of x_{db} while achieving the antiforensic purpose. Figure 6 shows the spectrograms of an original audio x_{org} from the test set, its x_{sg} and x_{db}, and its antiforensic audio AF_x_{db} compressed with random bit rates. Compared with x_{sg}, x_{db} presents fewer losses of content in the high frequency than AF_x_{db}, and AF_x_{db} is similar to x_{sg}.

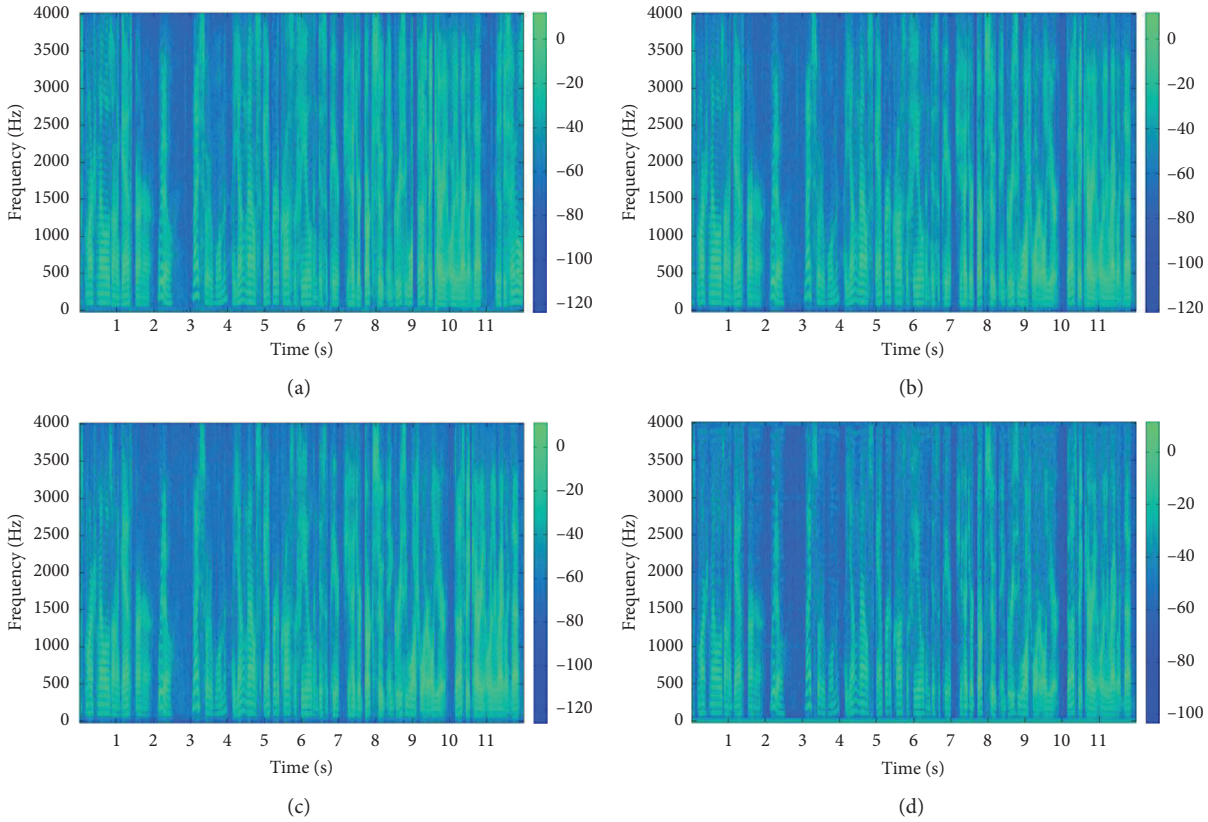


Figure 6: (a) Spectrogram of original audio. (b) Spectrogram of single compressed audio. (c) Spectrogram of double compressed audio. (d) Spectrogram of antiforensic audio generated by a well-trained generator.

5. Conclusion and Future Work

In this paper, we have proposed a new method to prove the weakness of the forensic detectors of AMR compression. To do this, we developed a GAN framework for the removal of AMR compression artifacts. Unlike the conventional antiforensic methods, our method can retain good perceptual quality with a better antiforensic capability in a data-driven manner. Through extensive experiments, the results demonstrate that the antiforensic double compressed audio can effectively avoid the detection of existing AMR compression methods with an average SAR about 94.75%, while retaining good perceptual quality.

However, there are still many remaining problems in the competition of forensics and antiforensics. In the future, we plan to consider the robustness of the forensic approach of AMR compression, i.e., whether adversarial framework could obtain a robust discriminator which can detect the antiforensic audios correctly by a well-trained generator or other attack strategy while distinguishing the double compressed audios from single compressed audios successfully.

Data Availability

The TIMIT dataset used to support the findings of the study is public and available at https://catalog.ldc.upenn.edu/LDC93S1.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China under grant nos. U1736215, 61672302, and 61901237, in part by the Zhejiang Natural Science Foundation under grant nos. LY20F020010 and LY17F020010, and in part by the K. C. Wong Magna Fund of Ningbo University.

References

[1] B. Bessette, R. Salami, R. Lefebvre et al., "The adaptive multirate wideband speech codec (AMR-WB)," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, pp. 620–636, 2002.

[2] D. Luo, R. Yang, and J. Huang, "Detecting double compressed AMR audio using deeplearning," in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2669–2673, Florence, Italy, May 2014.

[3] Y. Shen, J. Jia, and L. Cai, "Detecting double compressed AMR-format audio recordings," in *Proceedings of the 10th Phonetics Conference. China (PCC)*, pp. 1–5, Shanghai China, April 2012.

[4] J. Sampaio and F. Nascimento, "Double compressed AMR audio detection using linear prediction coefficients and support vector machine," in *Proceedings of the 22th Brazilian Conference on Automation*, João Pessoa, Brazil, September 2018.

[5] J. F. P. Sampaio and F. A. D. O. Nascimento, "Detection of AMR double compression using compressed-domain speech features," *Forensic Science International: Digital Investigation*, vol. 33, Article ID 200907, 2020.

[6] D. Luo, R. Yang, B. Li, and J. Huang, "Detection of double compressed AMR audio using stacked autoencoder," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 432–444, 2016.

[7] K. Valanchery, "Analysis of different classifier for the detection of double compressed AMR audio," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 4, pp. 98–107, 2018.

[8] M. Fontani and M. Barni, "Hiding traces of median filtering in digital images," in *Proceedings of the 2012 20th European Signal Processing Conference (EUSIPCO)*, IEEE, Bucharest, Romania, pp. 1239–1243, August 2012.

[9] M. Kirchner and J. Fridrich, "On detection of median filtering in digital images,"in Proceedings of the SPIE Electronic Image, Security, Steganography, vol. 7541, pp. 1–6, Watermarking on Multimedia Contents, San Jose, CA, USA, August 2010..

[10] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian, "Forensic detection of median filtering in digital images," in *Proceedings of the 2010 IEEE International Conference on Multimedia and Expo*, pp. 89–94, Singapore, July 2010.

[11] H.-D. Yuan, "Blind forensics of median filtering in digital imagesfiltering in digital images," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 4, pp. 1335–1345, 2011.

[12] Y. Luo, H. Zi, Q. Zhang, and X. Kang, "Anti-forensics of JPEG compression using generative adversarial networks," in *Proceedings of the 2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 952–956, IEEE, Rome, Italy, September 2018.

[13] C. Chen, X. Zhao, and M. C. Stamm, "Mislgan: an anti-forensic camera model falsification framework using a generative adversarial network," in *Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 535–539, Athens, Greece, October 2018.

[14] D. Kim, H.-U. Jang, S.-M. Mun, S. Choi, and H.-K. Lee, "Median filtered image restoration and anti-forensics using adversarial networks filtered image restoration and anti-forensics using adversarial networks," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 278–282, 2018.

[15] X. Li, D. Yan, L. Dong, and R. Wang, "Anti-forensics of audio source identification using generative adversarial networkfication using generative adversarial network," *IEEE Access*, vol. 7, pp. 184332–184339, 2019.

[16] C. Hanilci, F. Ertas, T. Ertas, and O. Eskidere, "Recognition of brand and models of cell-phones from recorded speech signals," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 625–634, 2012.

[17] C. Kotropoulos and S. Samaras, "Mobile phone identification using recorded speech signals," in *Proceedings of the 2014 19th International Conference on Digital Signal Processing*, pp. 586–591, IEEE, Hong Kong, China, August 2014.

[18] D. Luo, P. Korus, and J. Huang, "Band energy difference for source attribution in audio forensics," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2179–2189, 2018.

[19] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, How to train a GAN? tips and tricks to make GANs work (2017), 2016.

[20] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, *Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database*, Vol. 107, National Institute of Standards and Technology (NIST), Gaithersburgh, MD, USA, 1988.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, Université De Montréal, Montreal, Canada, 2014.

[22] S. Pascual, A. Bonafonte, and J. Serra, "SEGAN: speech enhancement generative adversarial network," 2017, https://arxiv.org/abs/1703.09452.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[24] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs,"in Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings, vol. 2, pp. 749–752, IEEE, Salt Lake City, UT, USA, February 2001..

[25] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.