Hindawi
Security and Communication Networks
Volume 2022, Article ID 7564678, 12 pages
https://doi.org/10.1155/2022/7564678

WILEY | Hindawi

*Research Article*

# Multiple Differential Distinguisher of SIMECK32/64 Based on Deep Learning

**Huijiao Wang, Jiapeng Tian, Xin Zhang, Yongzhuang Wei, and Hua Jiang**

*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*

Correspondence should be addressed to Yongzhuang Wei; walker_wyz@guet.edu.cn

Received 19 May 2022; Accepted 20 August 2022; Published 14 September 2022

Academic Editor: Zhen Wang



Currently, deep learning has provided an important means to solve problems in various fields. Intelligent computing will bring a new solution to the security analysis of lightweight block cipher as its analysis becomes more and more intelligent and automatic. In this study, the novel multiple differential distinguishers of round-reduced SIMECK32/64 based on deep learning are proposed. Two kinds of SIMECK32/64's 6–11 rounds deep learning distinguishers are designed by using the neural network to simulate the case of the multiple input differences and multiple output differences in multiple differential cryptanalysis. The general models of the two distinguishers and the neural network structures are presented. The random multiple ciphertext pairs and the associated multiple ciphertext pairs are exploited as the input of the model. The generation method of the data set is given. The performance of the two proposed distinguishers is compared. The experimental results confirm that the proposed distinguishers have higher accuracy and rounds than the distinguisher with a single difference. The relationship between the quantity of multiple differences and the performance of the distinguishers is also verified. The differential distinguisher based on deep learning needs less time complexity and data complexity than the traditional distinguisher. The accuracy of filtering error ciphertext of our 8-round neural distinguisher is up to 96.10%.

## 1. Introduction

The size of computing device has been decreasing, and some Internet of things applications such as smart homes and wearable systems have widely emerged in recent years [1, 2]. In these resource-constrained applications, lightweight block cipher usually plays a key role in ensuring data security [3]. This has also led to design new lightweight block cipher [4]. The balance between security and low consumption of lightweight block ciphers should be fully considered in design, which may be accompanied by the reduction of security. Therefore, the security analysis of lightweight block cipher is the first thing in its wide application.

The cipher distinguisher distinguishes the random permutation from the cipher according to the structure of the cipher algorithm or the characteristics of its components. The common construction methods of distinguisher include differential distinguisher [5], linear distinguisher [6], and integral distinguisher [7]. Among them, the differential distinguisher is the primary tool for differential attack, which was first proposed by Biham et al. in 1990 [8]. Differential attack takes advantage of the unbalanced distribution of difference statistics in the iterative process of block cipher algorithm. It has become the most basic and effective means for cryptanalysis of block cipher. The multiple differential cryptanalysis was introduced in reference [9]. It usually has the general form of multiple inputs and multiple outputs.

The deep learning technology is developing rapidly in the area of the artificial intelligence, such as computer vision [10], biological information [11], and natural language processing [12]. Deep learning uses the rule that is discovered from data to predict and judge the future moment and unknown situation. The cipher distinction task is coincided with the classification in form. A neural network model for classification can be established by abstracting plaintext pairs, ciphertext pairs, and the round function. The author in reference [13] first proposed a lightweight block cipher cryptanalysis based on deep learning. It successfully

provides a new cryptanalysis approach using deep learning. Since then, the cryptanalysis of lightweight block cipher based on deep learning has become more and more active.

Lightweight block cipher will be fully considered to resist differential cryptanalysis at the beginning of design. Nevertheless, it will be of great significance to design a new distinguisher and explore the unknown defects of block cipher. Based on the attack idea of multiple differential cryptanalysis in this study, the deep learning is applied to construct the distinguisher in multiple differential cryptanalysis for round-reduced SIMECK32/64. The structure of multiple ciphertext pairs with multiple input differences is exploited to train the distinguisher. Our contributions are as follows:

(1) The deep learning distinguisher of SIMECK32/64's 6–11 rounds for multiple differential cryptanalysis is presented. Our neural network structure adopts different multiple input differences, which is different from others.

(2) The general models of two novel kinds of distinguishers are given. The inputs of the two models adopt random ciphertext pairs and associated ciphertext pairs respectively, both of them are based on the multiple input differences. The method of data set generation is put forward.

(3) The multiple differential distinguishers based on deep learning have low complexity and high accuracy of filtering error ciphertext.

The rest of the study is structured as follows: in Section 2, the security analysis of relevant lightweight block cipher and the study of the cryptography combined with deep learning are presented. In Section 3, the preliminaries are provided, which include the overview of SIMECK, the differential distinguisher, and the deep learning model. In Section 4, two kinds of neural distinguishers are proposed and discussed in detail. In Section 5, the experiment of our distinguishers and the comparisons are carried out, and the results are given. Section 6 summarizes the work of the full study. Table 1 lists the main symbols and their meanings used in this study.

Table 1: Symbols and meanings.

| Symbol | Description of the symbol meaning |
|---|---|
| $t$ | Quantity of the input difference |
| $n$ | Block size |
| $r$ | Round |
| $\Delta_i$ | Difference |
| $(P_0, P_1, ..., P_i)$ | Plaintext pairs |
| $(C_0, C_1, ..., C_i)$ | Ciphertext pairs |
| $N$ | Training sample size |
| $ND_{rm}$ | Distinguisher trained with random ciphertext pairs |
| $ND_{am}$ | Distinguisher trained with associated ciphertext pairs |

## 2. Related Work

The family of SIMECK cipher similar to SIMON's structure was designed in reference [14] on CHES' 2015. The ciphers of SPECK and SIMON were released by the National Security Agency (NSA) in 2013 [15]. SIMECK continued the good design component of SPECK and SIMON and has an excellent performance in both hardware and software implementation. The designer has initially done the security analysis of SIMECK. The differential cryptanalysis, impossible differential cryptanalysis, and linear cryptanalysis were given.

The ability of SIMECK to resist linear attack was evaluated in reference [16]. The better results of differential cryptanalysis for SIMECK were presented in reference [17]. The authors in reference [18] proposed the novel algorithm for finding the more perfect differential trails and gave the differential trails of rounds 14, 21, and 27 of SIMECK32/48/64, respectively. The authors in reference [19] studied different versions of related-key impossible differential distinguisher of SIMECK. They proposed distinguishers of SIMECK32 for round 15 using the middle encounter method. With the help of MILP, when the difference between input and output was limited to one active key bit, the optimal related-key impossible differential of SIMECK was proposed. The authors in reference [20] introduced a cube attack based on SMT by using the additional information provided by the intermediate state cube feature for the SMT solver in attacking the round-reduced SIMECK32/64. A series of new distinguishers for statistical fault analysis of SIMECK based on the ciphertext-only attack was presented in reference [21]. The results showed that the distinguishers can restore the keys on the basis of reducing errors and improve the reliability and accuracy.

Deep learning technology is very useful for big data analysis, which can help to discover the small links among data. In cryptography, identifying the subtle relationship among data plays a very important role because the subtle relationship usually defines the security strength. The authors in reference [22] applied deep learning to side channel analysis and discussed the applicability of deep learning technology in classical cryptanalysis. The authors in reference [23] evaluated the various relations between deep learning and cryptography and proposed some possible research directions of using deep learning in cryptanalysis. A feedforward neural network (FNN) was developed in reference [24], which can find plaintext from the ciphertext of AES without using key information.

On CRYPTO'19, the author in reference [13] proposed the improved differential attack of round-reduced SPECK32/64 based on deep learning. The distinguisher was constructed by training the ResNet to distinguish the ciphertext pairs with the encryption using fixed plaintext difference and random data. It confirmed the effectiveness of deep learning on security analysis of symmetric ciphers and put forward a new research direction. The authors in reference [25] proposed a framework of using machine learning to extend the classical distinguisher. The distinguisher used a single differential trail and was implemented on SPECK, SIMON, and GIFT64. This method can

reduce the data complexity, but the accuracy was not as high as that of the distinguisher trained by random difference. In reference [26], convolutional neural network (CNN) and multilayer perceptron (MLP) were applied to construct the neural network distinguisher of round-reduced TEA and RAIDEN cipher. The distinguishing task was proposed where the traditional distinguisher could not be applied. The 3–6 rounds of neural network distinguisher for PRESENT cipher were constructed [27]. The distinguisher can distinguish ciphertext data from random data with high probability, which further expanded the application of deep learning in block cipher. In reference [28], deep learning was applied to perform the cryptanalysis on the simplified DES, SPECK, and SIMON under the limited key space, and the key bits were recovered successfully. However, this method was not applicable when the key space was not limited. The authors in reference [29] used neural network to simulate the "single input-multiple output" difference of non-Markov cipher and simplified the distinguish task into classification task. Several distinguishers of four ciphers Gimli, ASCON, Knot, and Chaskey were shown. It was proved that the complexity of each distinguisher was very low. The authors in reference [30] proposed the neural network distinguisher for Chaskey, PRESENT, and DES. The multiple ciphertext pairs use one difference as the input. The module was added to extract the derived features, so as to improve the accuracy of the distinguisher.

## 3. Preliminaries

*3.1. Description of SIMECK.* The family of SIMECK cipher is denoted as SIMECK 2*n*/*mn*, where 2*n* ($n$ =16,24,32) is the block size, and *mn* is the master key size [14]. For example, SIMECK 32/64 refers to the encryption of 32-bit block, and the length of master key is 64 bit. SIMECK is designed to follow Feistel structure. The plaintext is firstly divided into $L_0$ and $R_0$, then these two parts are encrypted by round function for $r$ rounds, and the last two outputs $L_r$ and $R_r$ consist of a complete ciphertext. Figure 1 shows the single round encryption process of SIMECK. The round function $F_{k_i}$ (the round $i$) is defined as follows:

$$F_{k_i}(L_{i+1}, R_{i+1}) = (R_i \oplus f(L_i) \oplus k_i, L_i), \tag{1}$$

where $L_i$ and $R_i$ are the intermediate state of SIMECK, and $k_i$ is the round key. The function $f$ is defined as follows:

$$f(x) = (x \odot (x \ll 5)) \oplus (x \ll 1), \tag{2}$$

where $\odot$ is bitwise AND, $\oplus$ is exclusive-or (XOR), and $x \ll i$ represents that $x$ is cyclically shifted left by $i$ bits. The encryption process of SIMECK is given in Algorithm 1.

Algorithm 2 gives the process of generating round key $k_i$ from the master key $K$. In order from high to low, the 64-bit master key $K$'s initial condition $(t_2, t_1, t_0, k_0)$ is 4 words. $Z_0$ and $Z_1$ are described in reference [14]. $C$ is a constant. The function $f$ is the reusing of SIMECK's round function. The update operation of intermediate state can be defined as follows:

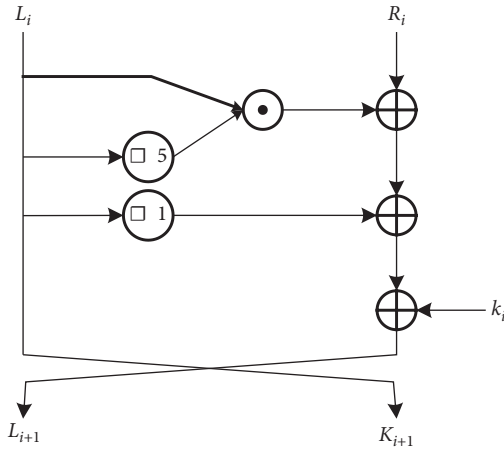


Figure 1: Single round encryption of SIMECK.

**Input:**
$P = (L_0, R_0) \in \{0,1\}^{2n}$
**Output:**
$C = (L_r, R_r) \in \{0,1\}^{2n}$
(1) **for** $i = 1$ to $r$ **do**
(2) $L_i = R_{i-1}, \oplus ((L_{i-1} \ll 5) \wedge L_{i-1}) \setminus$
(3) $L_i = L_i \oplus k_{i-1} \oplus (L_{i-1} \ll 1)$
(4) $R_i = L_{i-1}$
(5) **end for**
(6) **return** $(L_r, R_r)$

Algorithm 1: Encryption of SIMECK.

**Input:**
$K \in \{0,1\}^{mn}$
**Output:** $k_i$, $0 \le i \le r-1$
(1) **if** $n = 16$ **or** $n = 24$ **then**
(2) $j = 0$
(3) **else** $j = 1$
(5) $i = 0$
(4) **end if**
(6) **for** $i = 1$ to $r-1$ **do**
(7) $k_{i+3} = t_i$
(8) $t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_j)_i$
(9) **end for**
(10) **return** $k_i$

Algorithm 2: Encryption of SIMECK.

$$\begin{cases} k_{i+3} = t_i \\ t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_j)_i \end{cases}, \tag{3}$$

where $0 \le i \le r-1$, and $k_i$ is key of round $i$.

*3.2. Differential and Distinguisher.* Differential cryptanalysis exploits the differential $(\alpha, \beta)$ with high probability to

distinguish random data from ciphertext, and the key recovery attack is carried out on this basis. Therefore, the first step of differential cryptoanalysis is to look for high probability differential. The differential trail of a block cipher with $j$ rounds is defined as $(\alpha, \beta)$:

$$\alpha = \Delta_i \longrightarrow \Delta_{i+1} \longrightarrow \cdots \longrightarrow \Delta_{i+j} = \beta, \tag{4}$$

where $\Delta_i$ and $\Delta_{i+1}$ are the input difference and the output difference of round $i$ respectively. The difference operation generally adopts XOR and modular subtraction. In this study, the difference operation uses XOR; that is, $\Delta = P_0^i \oplus P_1^i$, $P_0^i$ and $P_1^i$ are the input pairs.

Let $\alpha$ and $\beta$ be two correlation differences with $n$ bits, $x$ is a $n$-bit input. The differential probability of the block cipher is denoted as $\mathrm{DP}(\alpha \longrightarrow \beta)$, which refers to the probability that $\alpha$ propagates to $\beta$ under round function $F$. It can be calculated as follows:

$$DP(\alpha \longrightarrow \beta) = 2^{-n} \cdot |\{x\colon F(x \oplus \alpha) \oplus F(x) = \beta\}|. \tag{5}$$

The differential trail generally is composed with a sequence of the triplet $(\Delta_i, \Delta_{i+1}, p_i)$, respectively representing the input differences, output differences, and corresponding probability of the round $i$. The differential gives all states in the whole difference chain.

Multiple differential cryptanalysis is a case of multiple linear cryptanalysis [9]. A group of input differences has no structure, and the corresponding output differences may be different due to the input differences. In multiple differential cryptanalysis, an attacker will adopt a collection $\Delta$ of differentials. The input differences set is denoted as $\Delta_\alpha$, $\Delta_\alpha = \{\Delta_\alpha^0, \Delta_\alpha^1, \ldots, \Delta_\alpha^t\}$. For a given input difference $\Delta_\alpha^i \in \Delta_\alpha$, the set of the output differences $\Delta_\beta$ can be obtained, and $\Delta_\beta$ is expressed as $\Delta_\beta = \left\{\Delta_\beta^{i,j} | (\Delta_\alpha^i, \Delta_\beta^{i,j}) \in \Delta\right\}$. Therefore, the collection $\Delta$ of differentials is described as $\Delta = \{(\Delta_\alpha^i, \Delta_\beta^{i,j}) | i = 0, 1, \ldots, t \text{ and } j = 0, 1, \ldots, |\Delta_\beta^{i,j}|\}$.

The cryptography distinguisher $\mathscr{D}$, or simply called distinguisher, is a probability algorithm. A random permutation $O$ or cipher encryption $\mathscr{C}$ as its input. If the distinguisher infers that the input comes from $\mathscr{C}$, then the output will be 1; otherwise, the output will be 0. When determining the success rate of a distinguisher, there are usually two situations. One is to identify the positive sample as the correct output, the other is to identify the negative sample as the correct output. A useful distinguisher often requires that its success rate be greater than 0.5. In deep learning, we use the accuracy to represent the success rate of the distinguisher. The accuracy of the distinguisher can be improved by learning more about the hidden statistical rules and structural features in ciphertexts.

*3.3. Neural Network.* Feedforward neural network (FNN), or called multilayer perceptron (MLP), is one of the deep learning models. FNN approximates a function $f^*$. In classification, $y = f(\mathbf{x}; \theta)$ means that the input $\mathbf{x}$ is mapped to a category $y$, and the parameter $\theta$ is learned, so that the optimal approximation of the function is obtained. The framework of FNN is generally composed of the input layer, some hidden layers, and the output layer. Finally, the learning model is a chain structure formed by many different functions. The depth of the network usually refers to the length of the chain. Assumed that the layer $l$ has $M$ units, the layer $l+1$ has $N$ units, $w_{ij}^l$ is the weight of the unit $i$ in the layer $l$ to the unit $j$ of the layer $l+1$, and $b_j^{l+1}$ and $f_j^{l+1}$ are the bias and activation functions of the unit $j$ in the layer $l+1$, respectively. The output of the unit $j$ in the layer $l$ of the FNN model is formally defined as follows:

$$x_j^{l+1} = f_j^{l+1}\left(\sum_{i=1}^{M} w_{ij}^l x_i^j + b_j^{l+1}\right). \tag{6}$$

Convolutional neural network (CNN) uses the convolution operations. CNN is a special structure of FNN. CNN can accept matrices as input and has repetitive neuron blocks (convolution kernels) that can across space (images) or time (audio signals). This special design structure makes the convolution network has partial translation invariance. Convolution operations are generally expressed as follows:

$$s(\mathbf{t}) = (\mathbf{x} * \mathbf{w})(\mathbf{t}), \tag{7}$$

where $\mathbf{x}$ represents the input and $\mathbf{w}$ represents the kernel function. $s(\mathbf{t})$ represents the output, which is called feature mapping, and $\mathbf{t}$ represents the current coordinate. CNN learning framework generally consists of the input layer, several convolution layers, the pooling layer, the full connection layer, and the output layer. The convolution layer and pooling layer are responsible for data processing, followed by the full connection layer. Therefore, CNNs are also known as FNN with data preprocessing.

## 4. Multiple Differential Distinguisher of SIMECK 32/64

In this section, the multiple differential distinguishers of SIMECK32/64 based on deep learning are given. The input structure of the model adopts the composite multiple ciphertexts with multiple differences. Ciphertexts use two composite methods: random ciphertext pairs and associated ciphertext pairs.

*4.1. Distinguisher Model.* The plaintexts are generated through multiple differences $\Delta_i$, and the corresponding ciphertexts are obtained by encrypting these plaintexts. The distinguisher will classify the random data and the ciphertext. The label vector $\mathbf{Y}$ is used to represent the classification results. If $Y_i$ is 0, it represents the random data. If $Y_i$ is 1, it represents the encrypted ciphertext. The definition of $Y$ is given as follows:

$$Y = \begin{cases} 1, P_i \oplus P_{i+1} = \Delta \dfrac{i}{2} \text{ and } \Delta_a \neq \Delta_b, (a,b) \in \{0,1,\ldots,t-1\}, i \in \{0,2,4,\ldots,2t\} \\ 0, Pi \in \text{Random}, i \in \{0,1,2,\ldots,2t+1\} \end{cases}. \tag{8}$$

Then, the general model of distinguisher $\mathcal{D}$ is defined as follows:

$$\Pr(Y = 1 | X_1, X_2, \ldots, X_k) = F(f_2(f_1(X_1), \ldots, f_1(X_k))), k \in \{0,1,\ldots,N-1\}, \tag{9}$$

where $\mathbf{X}_k$ is the ciphertext sample. $f_1(\mathbf{X}_k)$ represents the feature extracted from the set of ciphertext pairs $\mathbf{X}_k$, and $f_2(y_1, y_2, \ldots, y_m)$ represents the composite feature re-extraction among multiple ciphertext information. The function $F$ is the posteriori probability estimation function after integrating global features. In this study, two composite forms are designed. Formula (8) gives the structure of plaintext pair using random multiple differences. Another structure of plaintext pairs using multiple differences is proposed that link the plaintext pairs in series, named associated multidifferences. Multiple ciphertext pairs are combined to investigate the extraction of composite features by the distinguisher. The associated composite form is defined as follows:

$$Y = \begin{cases} 1, Pi - 1 \oplus Pi = \Delta i - 1 \quad an\ d\ \Delta a \neq \Delta b, (a,b,i) \in \{1,2,\ldots,t-1\} \\ 0, Pi \in Ran\ do\ m, i \in \{0,1,2,\ldots,t\} \end{cases}. \tag{10}$$

*4.2. Design of Neural Network Distinguisher*

(1) *Input Data Format*. The input layer adopts the form of ciphertext pairs set. CNN accepts the matrix as input. Two input data formats of SIMECK32/64 are constructed: $(2t+2) \times 16$ and $4t \times 16$. The input data formats represent the byte-oriented structure of ciphertext pairs based on multiple differences.

(2) *Network Structure*. The network structure consists of four modules, as shown in Figure 2. Module 1 uses 32 parallel convolution kernels with size 1 for bit slicing operation. The matrices of $(2t+2) \times 16$ and $4t \times 16$ are mapped to $16 \times 32$ matrix. The size of the matrix remains unchanged during the mapping process. Then, several repeated modules 2 are connected to adjust the depth of the network. Each module 2 contains two convolution layers. Each layer uses 32 convolution kernels whose size is $k_s$, which is able to study the features from the input ciphertext pairs. The result of the addition of the output of module 2 and the output of module 1 is a residual connection. Finally, module 3 and module 4 are fully connected layers. Each layer has $d_1$ and $d_2$ neurons respectively, which are used to synthesize global features. Each layer of the neural network applies L2-based kernel regularizer, batch normalization module, and a rectifier nonlinearly to ensure the universality of the network.

(3) *Activation Function*. The gradient may disappear when the function *sigmoi d* is used as the activation function. The reason is that the gradient value of sigmoid function is too small in the interval of $|x| > 4$. In contrast, the output of the function ReLU is relatively stable, which is a linear function when $x > 0$. At the same time, the problem of sparsity is solved. Therefore, ReLU is used as activation function while training the distinguisher for lower number of rounds. The function of ReLU is defined as follows:

$$\text{Re}LU(x) = \max(0, x). \tag{11}$$

While training the distinguisher with a high number of rounds, the gradient explosion often occurs in the training process because there is no obvious difference between ciphertext pairs and random data in the data set. The training model may bias to two directions. Therefore, this study selects tanh function, whose function expression is as follows:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{12}$$

(4) *Super Parameter Setting*. The mean square error is added to *L*2 norm. The effect of parameter penalty and regularization are achieved. The loss function is set as follows:

$$L(y; f(x)) = \frac{1}{n} \sum_{i=1}^{n} (yi - f(xi))^2 + \lambda \|w\|^2, \tag{13}$$

where $f(x_i)$ represents the neural network's output, $w$ represents the neural network's parameters, $y_i$ represents the true label, and $\lambda$ represents the penalty factor, which is 0.0001. The optimizer uses the Adam algorithm, which corrects the learning rate of each parameter in real time by

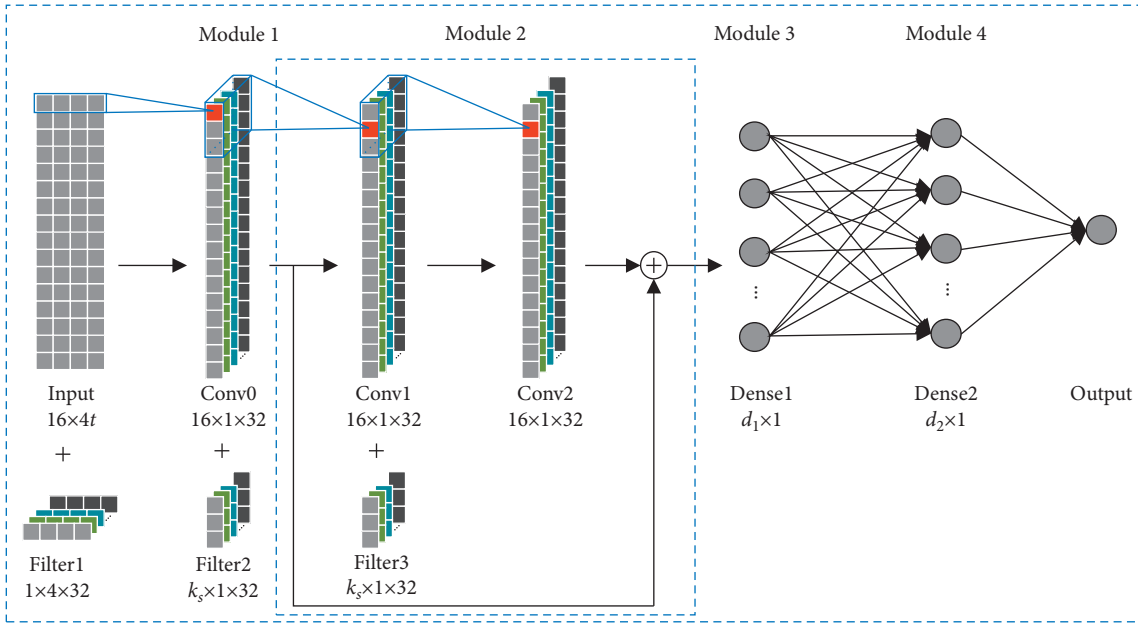


Figure 2: Neural network structure of CNN for SIMECK32/64 distinguisher.

Input:
multiple differences ($\Delta_0$, $\Delta_1$, ..., $\Delta_{t-1}$)
sample number $N$
**Output:** TD"
(1) TD" ← (·)/∗initial data set∗/
(2) $K$ ← Random()
(3) **for** $i = 0$ to $t - 1$ **do**
(4) $P_{2i}$ = Random()
(5) **end for**
(6) **for** $i = 0$ to $t - 1$ **do**
(7) $P_{2i+1} = P_{2i} \oplus \Delta_i$
(8) **end for**
(9) **for** $j = 0$ to $N - 1$ **do**
(10) $\mathbf{C}_j$ ← encrypt ($\mathbf{P}_j$, $K_j$)
(11) **end for**
(12) **for** $i = 0$ to $N - 1$ **do**/∗set label∗/
(13) **if** $i$&1 = 0 **then**
(14) $\mathbf{C}_i$ ← Random()
(15) $Y_i$ ← 0
(16) **else**
(17) $Y_i$ ← 1
(18) **end if**
(19) **end for**
(20) **return** TD" ← ($\mathbf{X}(\mathbf{C_0}...\mathbf{C_{N-1}})$, Y)

Algorithm 3: Data generation for $ND_{rm}$.

Input:
multiple differences ($\Delta 0$, $\Delta 1$, ..., $\Delta_{t-1}$)
sample number $N$
**Output:** model
(1) TD ← (·); /∗initial training set∗/
(2) $K$ ← Random()
(3) $P_0$ = Random()
(4) **for** $i = 1$ to $t$ **do**
(5) $P_i = P_{t-1} \oplus \Delta_{i-1}$
(6) **end for**
(7) **for** $j = 0$ to $N - 1$ **do**
(8) $\mathbf{C}_j$ ← encrypt ($\mathbf{P}_j$, $K_j$)
(9) **end for**
(10) **for** $i = 0$ to $N - 1$ **do**/∗set training label∗/
(11) **if** $i$&1 = 0 **then**
(12) $\mathbf{C}_i$ ← Random()
(13) $Y_i$ ← 0
(14) **else**
(15) $Y_i$ ← 1
(16) **end if**
(17) **end for**
(18) TD ← ($\mathbf{X}(\mathbf{C_0}...\mathbf{C_{N-1}})$, Y)
(19) model ← Training with TD
(20) **return** model

Algorithm 4: Training $ND_{am}$ model.

first-order and second-order matrix estimates of gradients. The learning rate decreases with the increment of training rounds and does not decrease after 40 epochs. Therefore, the training cycle in the algorithm is set to 40 epochs. Meanwhile, the ModelCheckPoint method is triggered by the callback function to save the best learning model.

### 4.3. Model Training and Testing

*4.3.1. Data Generation.* The data set generation of the distinguisher $ND_{rm}$ is shown in Algorithm 3. The initial plaintext of the distinguisher $ND_{rm}$ is obtained by the random generator. They are expressed as $(P_{i0}, P_{i2}, \ldots, P_{i(2k)})$, where $i \in \{0, 1, 2, \ldots, N-1\}$ and

**Input:** model $F$
  **sample** number $M$
**Output:** acc
(1) TD'← (**X** ($\mathbf{C}_0 \ldots \mathbf{C}_{m-1}$), Y)/*generate test data set*/
(2) sum = 0
(3) **for** $i = 0$ to $M - 1$ do
(4)  **if** $F(\mathbf{C}_i) = 1$
(5)  sum++
(6)  **end if**
(7) **end for**
(8) **return** acc = sum/m

Algorithm 5: Testing NDam model.

$k \in \{0, 1, 2, \ldots, t-1\}$. For the given $t$ different differences $\Delta_i$, $i \in \{0, 1, \ldots, t\}$, the bitwise XOR is performed between the initial plaintexts and the given differences. The corresponding other half of plaintexts are $(P_{i1}, P_{i3}, \ldots, P_{i(2k+1)})$, where $i \in \{0, 1, 2, \ldots, N-1\}$ and $k \in \{0, 1, 2, \ldots, t-1\}$ are obtained. Finally, the plaintexts are encrypted to generate a sample data $C_i = (C_{i0}, C_{i1}, \ldots, C_{i(2k+1)})$, $i \in \{0, 1, 2, \ldots, N-1\}$, and $k \in \{0, 1, 2, \ldots, t-1\}$. The bitwise $AND$ operation is performed between the sample data and 1 to ensure that the encrypted data and random data in the sample space account for half respectively.

The data set generation of the distinguisher $ND_{am}$ is shown in Algorithm 4 from line 1 to line 18. The initial plaintext $P_0$ of the distinguisher $ND_{am}$ is obtained by the random generator. Given a difference sequence $(\Delta_0, \Delta_1, \ldots, \Delta_{t-1})$, next plaintext $P_i$ is obtained by the bitwise XOR between the previous plaintext $P_{i-1}$ and the corresponding given difference $\Delta_{i-1}$, namely, $P_i = P_{i-1} \oplus \Delta_{i-1}$. $t$ plaintexts are generated. Finally, the plaintexts are encrypted to generate a sample data $C_i = (C_{i0}, C_{i1}, \ldots, C_{i(k+1)})$, where $i \in \{0, 1, 2, \ldots, N-1\}$ and $k \in \{0, 1, 2, \ldots, t-1\}$. The construction of the sample space adopts the method similar to that in $ND_{rm}$.

*4.3.2. Training.* Algorithm 4 is the training of $ND_{am}$ model. The set used for training is composed of ciphertexts and random data. First, $t$ associated plaintext pairs are generated by the given $P_0$ and the multiple input differences $(\Delta_0, \Delta_1, \ldots, \Delta_{t-1})$. Then, the ciphertext pairs are obtained. In order to make the encrypted data and random data account for half of the sample, half of the data is replaced with random data. Finally, the trained neural network model is saved and returned.

*4.3.3. Testing.* The data sets of testing and training are generated in the same way. The model can be regarded as a function $F$, which accepts inputs, judges, and outputs. If the label is 0 and it is judged that the data belong to random data, the output is 0. Similarly, if the label is 1 and the model judged that the data belong to cipher, the output is 1. Finally, the test accuracy of neural network model is returned. Algorithm 5 describes the process of testing neural network distinguisher model. The process of training and testing of the distinguisher $ND_{rm}$ is similar to the above $ND_{am}$ model, which will not be repeated here.

## 5. Experiment and Performance Evaluation

The performance of some distinguishers and the comparisons among these distinguishers are presented in this section. The proposed distinguishers in this study are $ND_{rm}$ and $ND_{am}$. The distinguisher $ND_S$ is realized by using the ideas of reference [13], and the distinguisher $ND_M$ is realized by using the ideas of multiple ciphertext and single difference [30]. All experiments were conducted on a computer with a GTX 1650 graphics card, 16 GB memory. Tensorflow is used at the back end and Keras is used at the front end.

*5.1. Results of the Proposed Distinguishers.* Figure 3 shows the accuracy of distinguisher $ND_{am}$ and $ND_{rm}$ of round-reduced SIMECK 32/64 (6–11 rounds). The network of the distinguisher shown in Figure 3 adopted CNN. In our experiments, the two distinguishers have the same parameters. The number of differences is set as $t = 3$, and the threshold is set as $\delta > 0.5$. The training sample size is $2^{24}$, and the testing sample size is $2^{18}$. The positive and negative samples in training set and testing set account for 1/2 of them, respectively. The input plaintext differences $\Delta_0 = 0x0/0x1, \Delta_1 = 0x0/0x2, \Delta_2 = 0x0/0x4$ were selected.

The experimental results show that the deep learning distinguisher can easily learn the distinguishing characteristics of encrypted data and random data for the low rounds, but as the number of iteration rounds increases, the accuracy will continue to decrease. According to the confusion and diffusion of the block cipher, the higher iteration rounds is, the weaker the statistical information between plaintexts and ciphertexts become. And the corresponding positive and negative samples have the high similarity. It becomes difficult for deep learning to select the feature effectively. In order to make the neural network distinguisher have a strong generalization ability, the neural network model can be improved to a certain extent by increasing the size of sample in training and testing or prolonging the training epoch.

It can also be seen from Figure 3 that distinguisher $ND_{rm}$ and $ND_{am}$ have similar trends, but the accuracy of $ND_{am}$ is slightly higher than that of $ND_{rm}$. The convergence speed of $ND_{am}$ is faster than that of $ND_{rm}$ when the number of rounds is high. It proves that $ND_{am}$ has learned more features from the associated ciphertext pairs. Compared with the ciphertext pairs generated by random multiple differences, there are hidden features in the associated ciphertext pairs. Therefore, the performance of the distinguisher is improved. However, $ND_{am}$ distinguisher also has restrictions on data requirements in training and testing. Data generation of the distinguisher $ND_{rm}$ is easier than that of distinguisher $ND_{am}$.

In training and testing of the distinguisher model, the sample data required for the distinguisher of lower rounds can be reduced. Taking $ND_{am}$ ($r = 6, t = 3$) as an example, the

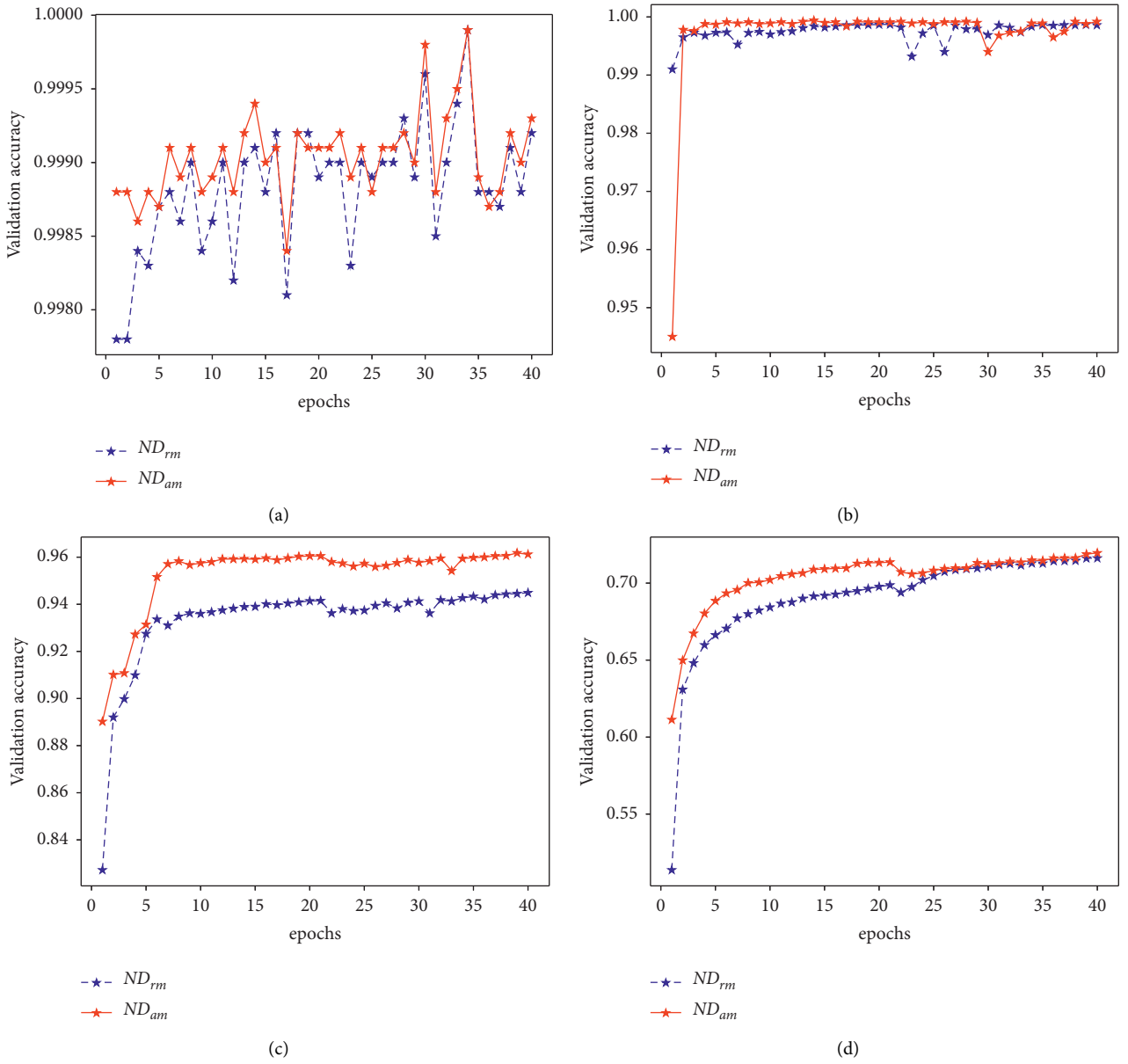


Figure 3: Continued.

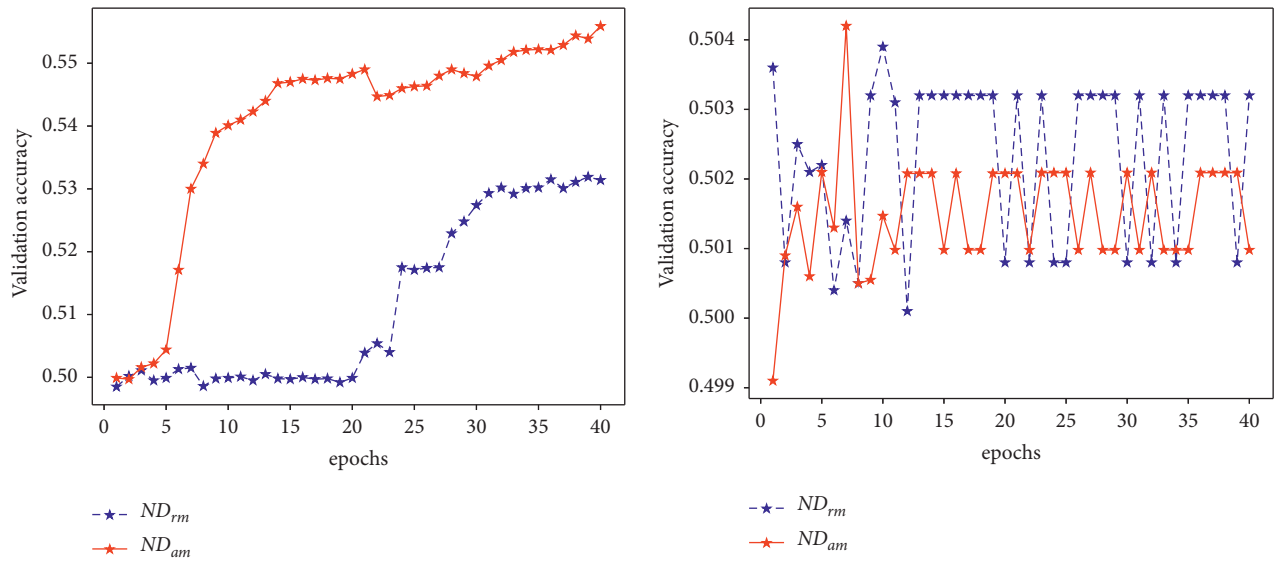


Figure 3: Accuracy of distinguishers for round-reduced SIMECK32/64 (6–11 rounds). (a) Round 6. (b) Round 7. (c) Round 8. (d) Round 9. (e) Round 10. (f) Round 11.

Table 2: Accuracy of SIMECK's distinguisher training by MLP.

| $r$ (round) | $ND_{am}$ | $ND_{rm}$ |
|---|---|---|
| 6 | 0.9999 | 0.9670 |
| 7 | 0.9441 | 0.9391 |
| 8 | 0.8785 | 0.7846 |
| 9 | 0.6124 | 0.5011 |
| 10 | 0.5017 | - |

Table 3: SIMECK32/64 distinguisher of 6–10 rounds MLP compared to CNN.

| $r$ (round) | Number of parameters | | Time of training | | Accuracy | |
|---|---|---|---|---|---|---|
| | MLP | CNN | MLP (s) | CNN (s) | MLP | CNN |
| 6 | 15585 | 44001 | 3362 | 5723 | 0.9999 | 0.9999 |
| 7 | 15585 | 44001 | 3481 | 5727 | 0.9432 | 0.9998 |
| 8 | 15585 | 44001 | 3401 | 5720 | 0.8801 | 0.9648 |
| 9 | 15585 | 44001 | 3408 | 5743 | 0.6128 | 0.7248 |
| 10 | 15585 | 44001 | 3361 | 5761 | 0.5016 | 0.5501 |

samples of the training set only need $2^8$. Using the settings in this study, the accuracy about 0.99 can be achieved in 25 s. For the $ND_{am}$ ($r = 7$, $t = 3$), the samples of the training set need $2^{16}$, which takes about 1 minute to achieve the accuracy about 0.98.

Using the same training parameters as CNN, this study also constructed the corresponding distinguisher using MLP. The MLP network consists four hidden layers, whose neurons for each hidden layer are set as $2n, 2n, n, n$. Each hidden layer abstracts the features of the input ciphertext pair to another dimension space to extract more abstract features. L2 kernel regularizer and nonlinear activation function ReLU are used in each layer. Since the number of layers of MLP network is not very deep, the batch normalization module is not used. The training of MLP adopts the circular learning rate scheme in reference [13], which is different from the decreasing learning rate scheme adopted in CNN. Table 2 lists the results of the distinguisher based on MLP. It can be seen from the results that the accuracy of MLP is equal to or less thanthat of CNN in each round. But it is up to only 10 rounds at most. In the case of rounds 8 and 9, the accuracy of $ND_{am}$ is significantly higher than that of $ND_{rm}$. This also shows that the associated difference has certain advantages.

Table 3 lists the number of learning parameters, training time, and accuracy of 6–10 rounds of distinguisher $ND_{am}$ ($t = 3$) when using MLP and CNN. Due to the simple structure of MLP, the training time and the number of parameters will be greatly reduced. It can be seen that CNN can better converge to a local minimum and is easier to be optimized, and the accuracy of CNN is slightly higher than that of MLP.

Figures 4 and 5 show the prediction distribution of distinguisher $ND_{am}$ and $ND_{rm}$ ($r = 8$, $t = 3$) based on CNN with 512 random positive samples, respectively. It can be seen that our 8-round CNN distinguishers have high reliability in the distribution of predicted values. It is basically consistent with the accuracy of training. Ciphertexts with high probability difference can be easily found.

*5.2. Comparisons of Distinguishers.* In our experiment, the distinguishers ($t = 2$, 3, and 4) were given. The sample sizes of training and testing are $2^{24}$ and $2^{18}$, respectively. The specific input differences used are listed in Table 4.

Table 5 lists the accuracy of $t$-differences neural distinguisher $D_{rm}$, $D_{am}$ of SIMECK's 6–11 rounds. In the case of the same round, the same input difference, and the same

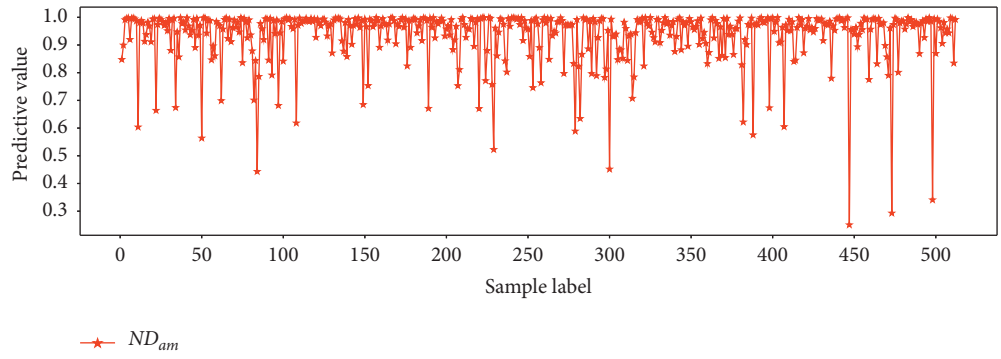


Figure 4: Prediction distribution of 8-round distinguisher $ND_{am}$.

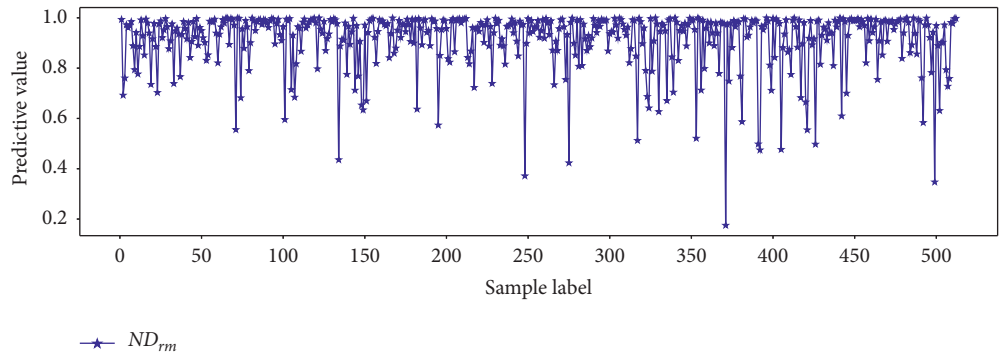


Figure 5: Prediction distribution of 8-round distinguisher $ND_{rm}$.

Table 4: Quantity of differences and selected differences.

| $t$ | Selected differences |
|---|---|
| 2 | $\Delta_0 = 0x0/0x1, \Delta_1 = 0x0/0x2$ |
| 3 | $\Delta_0 = 0x0/0x1, \Delta_1 = 0x0/0x2, \Delta_2 = 0x0/0x4$ |
| 4 | $\Delta_0 = 0x0/0x1, \Delta_1 = 0x0/0x2, \Delta_2 = 0x0/0x4, \Delta_3 = 0x0/0x8$ |

Table 5: Comparison of $t$-differences distinguishers' accuracy.

| $r$ (round) | $t=2$ | | $t=3$ | | $t=4$ | |
|---|---|---|---|---|---|---|
| | $ND_{rm}$ | $ND_{am}$ | $ND_{rm}$ | $ND_{am}$ | $ND_{rm}$ | $ND_{am}$ |
| 6 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 |
| 7 | 0.9980 | 0.9983 | 0.9980 | 0.9998 | 0.9990 | 0.9992 |
| 8 | 0.9245 | 0.9449 | 0.9382 | 0.9645 | 0.9336 | 0.9422 |
| 9 | 0.6875 | 0.6917 | 0.7060 | 0.7151 | 0.6849 | 0.7051 |
| 10 | 0.5205 | 0.5388 | 0.5319 | 0.5505 | 0.5325 | 0.5415 |
| 11 | 0.5024 | 0.5015 | 0.5039 | 0.5029 | 0.5019 | 0.5022 |

Table 6: Comparison of distinguishers' accuracy of SIMECK32/64.

| $r$ (round) | $D_{am}$ | $D_{rm}$ | $ND_S$ ([13]) | $ND_M$ ([30]) |
|---|---|---|---|---|
| 6 | 0.9999 | 0.9999 | 0.9999 | 0.9952 |
| 7 | 0.9990 | 0.9980 | 0.9846 | 0.9718 |
| 8 | 0.9645 | 0.9382 | 0.9389 | 0.8505 |
| 9 | 0.7151 | 0.7060 | 0.7023 | 0.5898 |
| 10 | 0.5505 | 0.5319 | 0.5018 | 0.5403 |
| 11 | 0.5042 | 0.5038 | — | 0.5014 |

Table 7: Complexity of SIMECK32/64.

| $r$ (round) | Data complexity | | Time complexity | | Memory complexity | |
|---|---|---|---|---|---|---|
| | $D_{am}$ (our) | [18] | $D_{am}$ (our) | [18] | $D_{am}$ (our) | [18] |
| 6 | $2^8$ | $2^{13}$ | $2^{5.42}$ | $2^{10.42}$ | $2^8$ | $2^{13}$ |
| 7 | $2^{11}$ | $2^{15}$ | $2^{8.19}$ | $2^{12.19}$ | $2^{11}$ | $2^{15}$ |
| 8 | $2^{13}$ | $2^{19}$ | $2^{10}$ | $2^{16}$ | $2^{13}$ | $2^{19}$ |
| 9 | $2^{14}$ | $2^{21}$ | $2^{10.83}$ | $2^{17.83}$ | $2^{14}$ | $2^{21}$ |
| 10 | $2^{16}$ | $2^{25}$ | $2^{12.68}$ | $2^{21.68}$ | $2^{16}$ | $2^{25}$ |
| 11 | $2^{24}$ | $2^{27}$ | $2^{20.54}$ | $2^{23.54}$ | $2^{24}$ | $2^{27}$ |

size of training sets and testing sets, the accuracy of $D_{am}$ is slightly higher than that of $D_{rm}$ regardless of the quantity of plaintext differences. It is indicated that $D_{am}$ has stronger feature selection ability and generalization ability. It also shows that when $t$ increases, the accuracies of $D_{rm}$ and $D_{am}$ increase. But it does not always increase with the increment of $t$. Table 5 also lists that the accuracies of $D_{rm}$ and $D_{am}$ are all the highest when $t=3$. It can be inferred from algorithm 4 and algorithm 5 that in the case of the same size of training sets and testing sets, the more the number of input differences is, the more complex the data become. The distinguishers are difficult to extract the features among ciphertexts, which will not increase the accuracy of the distinguishers.

The input difference selected by $D_{rm}$ and $D_{am}$ is the same as that used in the previous experiment when $t=3$. The input

Table 8: Accuracy statistical results of $ND_{am}$.

| $r$ (round) | TP (%) | FP (%) | TN (%) | FN (%) |
|---|---|---|---|---|
| 7 | 100.00 | 0.39 | 0.00 | 99.61 |
| 8 | 99.61 | 3.91 | 0.39 | 96.10 |
| 9 | 81.25 | 22.66 | 18.75 | 77.34 |

difference used in $ND_M$ and $ND_S$ is $\Delta_0 = 0x0/0x1$. The size of samples is the same as above. Table 6 lists the accuracy of distinguishers of SIMECK32/64's 6–11 rounds.

$D_{am}$ and $D_{rm}$ can reach higher rounds than $ND_S$ obviously. They can be applied to the SIMECK 32/64 algorithm with higher rounds; that is, the number of the round is expanded to round 11. And the accuracy is equivalent to the accuracy of round 10 in $ND_S$. The $D_{am}$ and $D_{rm}$ have achieved higher accuracy than $ND_M$ under the same number of rounds and still maintain a high accuracy in round 9 without a precipitous decline. It fully proves that our $D_{am}$ and $D_{rm}$ have learned some additional features from multiple ciphertext pairs. At the same time, it also proves that when multiple differences are introduced, the accuracy of the distinguisher can be appropriately improved under the same round or a higher round of the distinguisher can be obtained. It also proves that it is feasible to train distinguishers by using multiple differences.

In reference [18], the authors presented the optimized differential trails. The deep learning multiple differential distinguishers proposed in this study have better performance compared with reference [18]. Table 7 lists the detailed comparison of the complexity.

To further verify our neural distinguisher as a cryptographic tool, 512 random samples are used to test the ability of filtering error ciphertext of the multiple differential distinguishers. Using 3 input differences and 7–9 rounds of encryption, $ND_{am}$ is tested. Among 512 chosen-plaintext sets, half are positive samples and half are negative samples. Table 8 lists the probabilities of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) of our $D_{am}$. It can be seen that the accuracy of filtering error ciphertext (i.e., FN) of our neural distinguisher is 99.61%, 96.10%, and 77.34%, respectively, and our distinguisher has high reliability.

## 6. Conclusions

In this study, we proposed a new deep learning distinguisher based on multiple differences for SIMECK32/64's 6–11 rounds. Two kinds of ciphertext pairs by using multiple differences are designed as the input of neural network. The distinguishers with good performance are verified. We also show some distinguishers and the accuracy of our distinguishers under different conditions. And the differential distinguishers based on deep learning consume less time and data than traditional distinguisher. The accuracy of filtering error ciphertext of our neural distinguisher is high. It is further proved that the deep learning method provides feasible means to simulate the case of the multiple input differences and multiple output differences. We are also trying to study deep learning distinguisher for other ciphers with different structure and block size. In the future, we will also adopt data preprocessing and other methods to study the deep learning distinguisher.

## Data Availability

The data used to support the findings of this study are available from the first author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

The work was supported by the Natural Science Foundation of Guangxi (2019GXNSFGA245004 and 2019GXNSFAA245053), Guangxi Science and Technology Major Project (guike AA22068072 and guike AA19254016), and the Innovation Project of GUET Graduate Education (2022YCXS088).

## References

[1] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 391–406, 2020.

[2] S. Seneviratne, Y. Hu, T. Nguyen et al., "A survey of wearable devices and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2573–2620, 2017.

[3] R. T. Tiburski, L. A. Amaral, E. de Matos, D. F. G. de Azevedo, and F. Hessel, "The role of lightweight approaches towards the standardization of a security architecture for IoT middleware systems," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 56–62, 2016.

[4] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. Elsayed, and M. M. Dessouky, "SLIM: a lightweight block cipher for internet of health things," *IEEE Access*, vol. 8, pp. 203747–203757, 2020.

[5] E. Bagherzadeh and Z. Ahmadian, "MILP-based automatic differential search for LEA and HIGHT block ciphers," *IET Information Security*, vol. 14, no. 5, pp. 595–603, 2020.

[6] M. Matsui, "The first experimental cryptanalysis of the data encryption standard," *Advances in Cryptology-CRYPTO*, vol. 94, pp. 1–11, Berlin, Heidelberg, 1994.

[7] L. Knudsen and D. Wagner, *Integral Cryptanalysis*, pp. 112–127, Fast Software Encryption, 2002.

[8] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.

[9] C. Blondeau and G. Benot, "Multiple differential cryptanalysis: theory and practice," in *Proceedings of the 18th international conference on Fast software encryption (FSE'11)*, pp. 35–54, Berlin, Heidelberg, February 2011.

[10] A. Esteva, A. Robicquet, B. Ramsundar et al., "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[11] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, "A method of information protection for collaborative deep learning under GAN model attack," *IEEE/ACM Transactions on*

*Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 871–881, 2021.
[12] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2021.
[13] A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," *Advances in Cryptology-CRYPTO 2019*, pp. 150–179, Springer, Berlin, Germany, 2019.
[14] G. Yang, B. Zhu, and V. Suder, "The SIMECK family of lightweight block ciphers," in *Proceedings of the 17th International Workshop on Cryptographic Hardware and Embedded Systems 2015*, vol. 9293, pp. 307–329, 2015.
[15] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, *The SIMON and SPECK Families of Lightweight Block Ciphers*, p. 404, Cryptology ePrint Archive, 2013.
[16] N. Baherl, "Linear cryptanalysis of reduced-round SIMECK variants," in *Proceedings of the 16th International Conference on Cryptology*, pp. 140–152, 2015.
[17] S. KÖlbl and A. Roy, "A brief comparison of SIMON and SIMECK," in *Proceedings of the 5th International Workshop on Lightweight Cryptography for Security and Privacy*, pp. 69–88, Aksaray, Turkey, 2016.
[18] Z. Liu, Y. Li, and M. Wang, "Optimal differential trails in SIMON-like ciphers," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 1, pp. 358–379, 2017.
[19] S. Sadeghi and N. Bagheri, "Security analysis of SIMECK block cipher against related-key impossible differential," *Information Processing Letters*, vol. 147, no. 12, pp. 14–21, 2019.
[20] M. Zaheri and B. Sadeghiyan, "SMT-based cube attack on round-reduced SIMECK32/64," *IET Information Security*, vol. 14, no. 5, pp. 604–611, 2020.
[21] W. Li, J. Li, D. Gu, C. Li, and T. Cai, "Statistical fault analysis of the SIMECK lightweight cipher in the ubiquitous sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4224–4233, 2021.
[22] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-DeepSCA: cross-device deep learning side channel attack," in *Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.
[23] R. L. Rivest, "Cryptography and machine learning," in *Proceedings of the International Conference on the Theory and Application of Cryptology*, pp. 427–439, 1991.
[24] X. Hu and Y. Zhao, "Research on plaintext restoration of AES based on neural network," *Security and Communication Networks*, vol. 2018, no. 15, pp. 1–9, 2018.
[25] T. Yadav and M. Kumar, "Differential-ML distinguisher: machine learning based generic extension or differential cryptanalysis," *IACR Cryptol. ePrint Arch.*, 2020.
[26] B. Emanuele and R. Matteo, "Performance comparison between deep learning -based and conventional cryptographic distinguishers," *ICAR Cryptol ePrint Archive*, 2020.
[27] A. Jain, V. Kohli, and G. Mishra, *Deep Learning Based Differential Distinguisher for Lightweight Cipher PRESENT*, Cryptology ePrint Archive, 2020.
[28] J. So, "Deep learning-based cryptanalysis of lightweight block ciphers," *Security and Communication Networks*, vol. 2020, no. 3, 11 pages, 2020.
[29] A. Baksi, J. Breier, Y. Chen, and X. Dong, "Machine learning assisted differential distinguishers for lightweight ciphers," in *Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 176–181, 2021.
[30] L. Zhang and Z. Wang, *Improving Differential-Neural Distinguisher Model for DES, CHASKEY, and PRESENT*, arXiv e-prints, 2022.