Cryptanalysis of Round-Reduced SIMON32 Based on Deep Learning

Zezhou Hou, Jiongjiong Ren(✉) and Shaozhen Chen

State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou
Email: jiongjiong_fun@163.com

Deep learning has played an important role in many fields. It shows significant potential to cryptanalysis. Differential cryptanalysis is an important method in the field of block cipher cryptanalysis. The key point of differential cryptanalysis is to find a differential distinguisher with longer rounds or higher probability. Firstly, we describe how to construct the ciphertext pairs required for differential cryptanalysis based on deep learning. Based on this, we train 9-round and 8-round differential distinguisher of SIMON32 based on deep residual neural networks. Secondly, we explore the impact of the input difference patterns on the accuracy of the distinguisher based on deep learning. For the input difference with Hamming weight of 1, the accuracy of 9-round distinguisher is different between the first 16 bits and the last 16 bits for non-zero bit positions. This is mainly caused by that its nonlinear operation is mainly concentrated in the first 16 bits. We also find that the accuracy of the distinguisher is different even if the input differences come from the differential characteristics with the same probability. Finally, we construct a last subkey recovery attack on 11-Round SIMON32 with practical data and time complexities. Our attack only uses about 2^9 chosen plaintexts and only needs about 45s for an attack with a success rate of over 90% using our workstation, which does not exceed $2^{18.5}$ 11-round encryption. At the same time, we extend the neural 9-round distinguisher to a 11-round distinguisher based on SAT, and propose a last subkey recovery attack on 13-Round SIMON32 using $2^{12.5}$ chosen plaintexts with a success rate of over 90%. Compared with traditional approach, the complexity of the method based on deep learning is lower, both in time complexity and data complexity.

Keywords: Deep Learning; SIMON32; Differential Distinguisher; Input Difference Pattern; Key Recovery Attack

1. INTRODUCTION

With long-term development, deep learning (DL) has been applied to various fields such as autonomous driving, machine translation and so on. In 1943, McCulloch and Pitts [1] proposed the MP neuron model, which was an abstract and simplified model constructed according to the structure and working principle of biological neurons. It opened the simulation of the neural network, but adjusting the weights relied heavily on manual work, very bad for study. In 1958, on the basis of MP neural, Rosenblatt [2] proposed the first-generation neural network named single-layer perceptron, which can distinguish between triangle, square and other basic shape. In 1986, the second generation of neural network was put forward by Rumelhart [3]. It changed the single fixed feature layer in the first-generation neural network to multiple hidden layers, using Sigmoid as the activation function. At the same time, it used the idea of Back Propagation (BP), which effectively solved the problem that the first generation only can be used in linear classification. Convolutional Neural Networks (CNN) [4], Recurrent Neural Networks (RNN) [5] and other networks have also developed to a certain extent. In 2006, Hinton [6] et al. put forward the concept of deep learning for the first time, and pointed out that deep learning can inhibit the gradient disappearance by using the method of layer by layer initialization. In 2011, Glorot [7] et al. proposed Rectified Linear Unit (ReLU), which can effectively inhibit the gradient disappearance. With these theories and networks, deep learning has played an important role in signal processing, image processing and other fields.

As an optimized version of the CNN, the deep residual neural networks (ResNet) [8] was applied to the ImageNet competition as soon as it was proposed in 2015, and achieved the first place. Due to the back propagation during the training process, it is easy to have the problem of gradients disappearance. This makes the accuracy of the model continue to improve as the number of layer continues to increase, but when the number increases to a certain number, the accuracy of validation data will decline rapidly. This is the reason why deep networks will become more difficult to train when the network becomes very deep. ResNet introduces a residual tower, which allows the network to increase without degradation as the depth increases.

Deep learning has brought significant improvement in many fields, and it enlightened cryptanalysis. As early as 1991, Ronald Rivest [9] discussed the similarities and differences between machine learning and cryptography, and analyzed the application of machine learning in the field of cryptography. In recent years, deep learning has also been applied to side channel analysis [10, 11, 12], and pointed out that the sensitive information on embedded devices can be effectively extracted by training neural networks.

In 2019, there was a giant leap in differential cryptanalysis based on deep learning because of Gohr's work [13]. At the Crypto2019, Gohr show that deep learning can produce very powerful cryptographic distinguishers and indicated that the neural distinguisher was better than the distinguisher obtained by traditional approach. He trained a neural distinguisher of SPECK32 [14] based on ResNet, which can distinguish the ciphertext pairs from random data roughly five times lower than a distinguisher using the full difference distribution table. At the same time, he developed a highly selective key search policy based on a variant of Bayesian optimization by using neural distinguishers. With this policy, Gohr described a practical key recovery attack on 11-round SPECK, and explained that the complexity of the attack based on deep learning was much lower than the traditional attack.

Enlightened by Gohr's work, Anubhab Baksi [15] et al. proposed to use multi-layer perceptron (MLP) to construct a distinguisher for longer block size and non-Markov ciphers. Compared with the traditional differential characteristics, the number of plaintexts selected is less. Similarly, Aayush Jain [16] et al. constructed a distinguisher for PRESENT [17]based on the MLP model, and explained that the MLP model with a lesser number of hidden layers can still be used in the construction of a block cipher distinguisher.

Although there are multiple researchs about the differential distinguishers based on deep learning, the influencing factors of the accuracy of neural distinguishers are unexplored, as far as we know. At the same time, it is meaningful to explore how to construct the neural distinguisher of the SIMON. What's more, it is significative to research about a practical key recovery attack of SIMON32 based on deep learning.

Our contribution: Inspired by Gohr's work, this paper tries to teach neural networks to exploit differential properties of round-reduced SIMON32. In order to achieve this goal, we train neural networks to distinguish the output of SIMON32 with a given input difference from random data. More importantly, we explore the influence of input difference pattern on the accuracy of the distinguisher and explain why the input differences affects the differential distinguisher. To show that our distinguisher is more effective than the traditional one, we construct a practical last subkey recovery attack with lower complexity in time and data. Our contributions can be summarised as follows:

- We describe a universal method of collecting data sets used for deep learning training, which can be applied to block ciphers with the block size of 32bits or other block size. At the same time, we adopt the ResNet to train neural distinguishers. Based on this, we train the 8-round and 9-round neural distinguisher of SIMON32 based on ResNet, whose accuracy are more than 70% and 60%, respectively.
- We investigate the influence of input difference pattern on the accuracy of neural distinguisher. Firstly, we study the influence of input difference with Hamming weight of 1 on accuracy of 9-round distinguisher. The accuracy is about 50% if the non-zero bit is in the first 16 bits of SIMON32, while the accuracy is around 60% if the non-zero bit is in the last 16 bits. This is mainly caused by that its nonlinear operation is mainly concentrated in the first 16 bits. In addition, we investigate the influence of input difference of differential characteristics with same probability on the accuracy. We find that the difference between the highest accuracy and the lowest accuracy exceeds 40%. This is due to that the neural distinguisher considers more the output differences under the same input difference.
- Thanks to the research about the influence of input difference pattern on the accuracy of neural distinguisher, we choose $(0x0, 0x200)$ as the input difference and train 8-round and 9-round neural distinguisher of SIMON32. Inspired by Gohr's key search policy, we complete a practical 11-round key recovery attack based on the 8-round and 9-round distinguisher on a workstation configured with *Intel i9-10900K* and *Nvidia TITAN RTX*. We use preconstructing-plaintext technique to help us obtain appropriate plaintext pairs. At the same time, the neutral bits [18] are used to enhance the distinguishing ability. In order to filtrate the better neutral bits for key recovery, we randomly generate 10000 sets of neutral bits and save the neutral bits which makes the success rate more than 90%. Our attack takes about 45s to recover the final subkey, with a success rate of more than 90%. Its time complexity is no more than $2^{18.5}$ 11-round encryption of SIMON32, and the data complexity is about 2^9, which is much lower than the complexity of traditional differential cryptanalysis.
- With the automatic searches based on SAT, we extend our neural 9-round distinguisher to a 11-

round distinguisher by prepending the 2-round differential characteristic $(0x800, 0x2200) \xrightarrow{2^{-4}} (0x0, 0x200)$. Similar to 11-round attack, we construct a last subkey recovery attack on 13-round SIMON32 with practical data and time complexities. Our attack only uses about $2^{12.5}$ chosen plaintexts and only needs about 45s for an attack, which does not exceed $2^{18.5}$ 13-round encryption.

Outline: The remaining of this paper is organised as follows. Section 2 gives a brief description of SIMON32 and illustrates the model of differential distinguisher based on deep learning. In Section 3, we construct a 8-round and 9-round distinguisher based on deep learning, and the relationship between the input difference pattern and the accuracy of distinguisher is discussed. Combining with the content of Section 3, we perform a key recovery attack on 11-round SIMON32 and 13-round SIMON32 in Section 4. Conclusions are drawn in Section 5 where we also suggest further work.

2. PRELIMINARIES

2.1. Brief Description of SIMON32

SIMON [14] is a lightweight block cipher proposed by the NSA. The aim of SIMON is to fill the need for secure, flexible, and analyzable lightweight block ciphers. It is a family of lightweight block ciphers with block sizes of 32, 48, 64, 96, and 128 bits. The constructions are Feistel ciphers using a word size n of 16, 24, 32, 48 or 64 bits, respectively. Since this paper is studying the version with the block size of 32bits and the key size of 64bits, we only introduce SIMON32 here.

The round function of SIMON is composed of AND, rotation, and XOR operations on the n-bits word as follows:

$$\begin{aligned} \oplus &: \text{bitwise XOR} , \\ \& &: \text{bitwise AND} , \\ S^j &: \text{left circular shift by j bits.} \end{aligned}$$

For $k_i \in GF(2)^{16}$, the key-dependent SIMON32 round function is the map $R_{k_i}:GF(2)^{16} \times GF(2)^{16} \rightarrow GF(2)^{16} \times GF(2)^{16}$ defined by

$$R_{k_i}(x_{i+1}, x_i) = (x_i \oplus f(x_{i+1}) \oplus k_i, x_{i+1})$$

where $f(x_{i+1}) = \left(S^1 x_{i+1} \& S^8 x_{i+1}\right) \oplus S^2 x_{i+1}$, k_i is the round key. This round function is pictured in Figure 1.

As it is out of scope for our purpose, we refer to [14] for the description of the key-scheduling.

Since the SIMON algorithm was proposed, cryptographers have analyzed its various versions. The earliest differential attack on SIMON32 was presented by Abed et al. in [19], which attacked 18-round SIMON32. With Abed's work, Wang Q et al. [20] studied the security of SIMON32 by using integral, zero-correlation linear

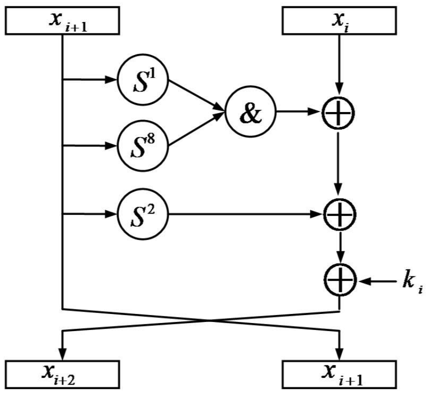

FIGURE 1. Round function of SIMON32

and impossible differential cryptanalysis. In addition, Sun S et al. [21] got a better differential distinguisher by mixed integer linear programming (MILP). What's more, Reihaneh Rabbaninejad et al. [22] presented cube and dynamic cube attacks on reduced-round SIMON32.

2.2. Model for differential distinguisher based on Deep Learning

As an effective method, differential cryptanalysis [23] is widely used in block cipher, where the differential distinguisher is indispensable, and often used to distinguish the ciphertext from random data. In traditional differential cryptanalysis, the first thing is to find a high-probability differential characteristic, and use the characteristic to construct a differential distinguisher. A distinguisher is considered to be useful if it can distinguish longer rounds or have higher probability. In traditional differential cryptanalysis, the construction of a differential distinguisher depends more on the possible defects of the algorithm itself. In recent years, automatic search [21, 24, 25] has gradually become the mainstream method to find differential distinguishers, which greatly accelerate the process of cryptanalysis.

The traditional differential distinguishers are used to distinguish the output of the block ciphers with a given input difference from random data. Deep learning is widely used in classification in the fields of image and signal processing, which is the similar to the traditional distinguishers. Therefore, deep learning can also be used to assist the construction of differential distinguishers.

In this section, we focus on how to obtain the ciphertext pair sets used to train neural distinguishers. In addition, we describe the construction of the neural distinguisher.

2.2.1. Collection of data set

In deep learning, the selection of data set affects the accuracy of the neural network model for distinguishing unknown data. Therefore, when we construct the differential distinguisher based on deep learning, the selected data set should be as random as possible and cover all possible situations. When we use the traditional differential cryptanalysis to construct a key recovery attack, it is the first step to find a differential distinguisher with the longer rounds and the higher probability, which is inseparable from finding a good input difference. Same as the traditional cryptanalysis, the neural distinguisher also needs a given input difference. We choose a fixed difference as the input difference of the neural differential distinguisher. The method of collecting data set is shown in Figure 2.

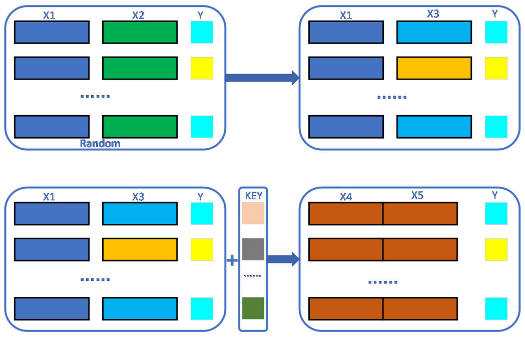

FIGURE 2. Collection of data set

Algorithm 1 gives a description about Figure 2. We choose a fixed difference defined by Δ with size of $n$bits. The method of collecting the data set is explained in Algorithm 1, and it can be applied to block ciphers with the block size of n bits.

Algorithm 1 Process for collecting data set

```
Input: Input difference $\Delta$ with size of $n$bits, Size of data set $N$, Rounds $R$
Output: Data set that meets the conditions
1: Randomly generate 4 sets containing $N$ elements, namely X1, X2, Y and KEY. The elements in X1 and X2 are integers with size of $n$bits, the elements in Y are 0 or 1. The elements in 4 sets are correspond to each other in order;
2: If an element in Y is 1, replace the value of the element at the corresponding position in X2 with the value by XOR the corresponding element in X1 and $\Delta$, and record the transformed X2 as X3;
3: The elements in X1 and X3 are regarded as plaintext, and the elements in the KEY at the corresponding position are used as the master key for $R$ rounds of encryption for the plaintexts. The ciphertext sets are denoted as X4 and X5;
4: The element at the corresponding position in X4 and X5 is spliced into a new element with size of $2n$bits, and obtain a new set namely X6. The element at the corresponding position in Y is used as the label of the data;
5: return X6 and Y;
```

If $n = 32$, the algorithm can be applied to SIMON32 and other block ciphers with block size of 32bits. And it can also be applied to block ciphers with block size of 64bits such as PRESENT and SPECK64, if $n = 64$. Supervised learning is currently a common type of machine learning. Given a set of labeled samples, it can learn how to map input datas to known labels. As shown in Figure 2, we take the $2n$-bit elements spliced by X4 and X5 as a new data set, which are used as a set of labeled samples for training. With deep learning, we can extract features from the known ciphertext pairs, and the unknown ciphertext pairs can be calculated by using these extracted features to obtain the classification result.

2.2.2. Construction of the differential distinguisher model

There are multiple neural networks available to train neural distinguishers, such as MIP, ResNet and so on. We choose the ResNet to help train a neural distinguisher, since ResNet introduces a residual tower which allows the network to increase without degradation as the depth increases.

In order to facilitate understanding of the network we constructed, the residual tower of ResNet is shown in Figure 3. In Figure 3, we choose "ReLU" as the activation and "Conv1D" as the basic convolution layer. What's more, The hidden layer of the ResNet we

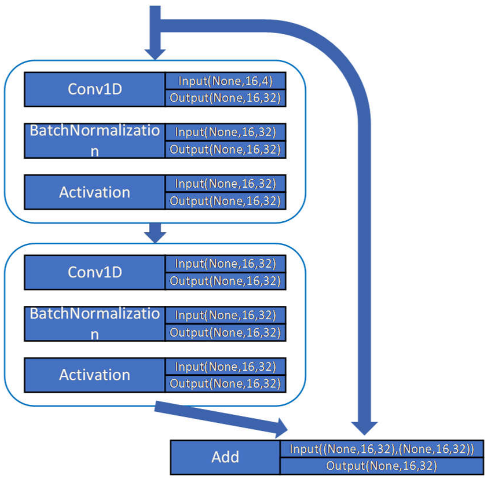

FIGURE 3. Residual tower

constructed contains a total of 5 residual towers as shown in Figure 3. Firstly the formatted original data is calculated by the "Conv1D" layer, and then transferred to the residual tower. Finally the final result is output by the output layer. When we finish the model training, we have a distinguisher model.

Using the model, we can distinguish whether the ciphertext pair is derived from the plaintext pair encrypted R rounds with a given input difference Δ.

3. NEURAL DISTINGUISHERS FOR REDUCED SIMON32

The differential distinguisher based on deep learning is used for key recovery attacks of SPECK32 in [13]. The SIMON family and SPECK family are proposed together, and the distinguisher of SIMON32 can also be constructed by emulating the method in [13]. At the same time, the influence of input difference on the accuracy of distinguisher is not mentioned in [13]. In this section, we construct an 8-round and 9-round differential distinguisher of SIMON32, and discuss the influence of input difference pattern on the accuracy of the distinguisher.

3.1. Training of differential distinguisher of SIMON32

We choose the input difference $(0x0, 0x200)$, from the differential characteristics shown in the appendix of [19], as the input difference of distinguisher based on deep learning. The size of training sets in the training process is 10^7, and the size of the validation set is 10^5. When training 8-round distinguisher, the data set is the ciphertext pairs set encrypted 8 rounds, and for 9-round distinguisher, it is ciphertext pairs encrypted 9 rounds. At the same time, we set the network to carry out 100 epochs. We use the Keras [26] for deep learning, and the training process of the model is mainly completed on the workstation configured with *Intel i9-10900K* and *Nvidia TITAN RTX*. In order to investigate the changes of the model in the training process, the change of accuracy and loss of the validation set during the training is shown in Figure 4.

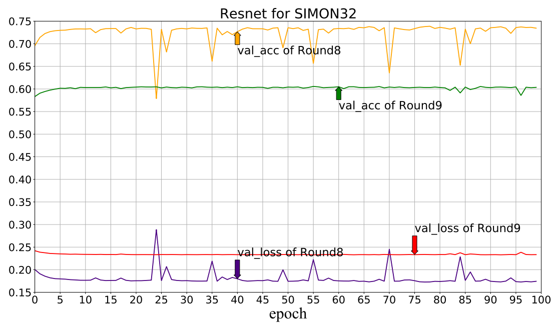

FIGURE 4. ResNet for SIMON32

In Figure 4, the x-coordinate represents the epochs in the training process, and the y-coordinate represents the accuracy and loss of validation set. The 4 broken lines in Figure 4 represent the changes in the accuracy and loss of the verification set when training the 8-round distinguisher and the 9-round distinguisher, respectively. From Figure 4, we can see that the accuracy of 8-round distinguisher on verification set is about 73%, and the accuracy of the 9-round distinguisher is about 60%.

For the model trained with 8-round encrypted data, we call this network model ResNet-based 8-round differential distinguisher. Using this distinguisher, we can distinguish the ciphertext pairs encrypted 8 rounds with a given input difference from random data. If the unknown data is the ciphertext pair encrypted 8 rounds with a given input difference, the distinguisher will return 1; and the distinguisher will return 0, if not. Similarly, for the model obtained from 9-round encrypted data, we can also use it to distinguish the ciphertext pair after 9-round encryption. In order to evaluate the accuracy of the model, we use the equation

$$acc = \frac{N_1}{N} \tag{1}$$

to evaluate the model, where N is the size of data judged by the model, N_1 is the size of data whose judgment result is the same as the real result, and acc is the accuracy of the model.

We generate a new data set containing 10^5 ciphertext pairs and evaluated the model using Equation (1). We find that the accuracy of the model is comparable to that of the validation set. This satisfies the basic requirement for distinguisher in cryptanalysis, which is accuracy over 50%. At the same time, by observing the changes of accuracy and loss, we find that the broken line is fluctuating during the training process, which is caused by the setting of the optimizer, but this does not affect our use of the model. Finally, from the change in the accuracy, it can be seen that 100 epochs of training may not be required, and a distinguisher can be obtained in the first 20 epochs of training, which helps reduce the running time.

3.2. The effect of input difference pattern on accuracy of distinguisher

In traditional differential cryptanalysis, the most important thing is to find a high-probability differential characteristic. The input difference of the differential characteristic will directly affect the probability of the characteristic. So it is meaningful to investigate into the influence of the input difference pattern on the accuracy of the neural distinguisher. In this section, we explore the effect of two kinds of input difference pattern on accuracy of neural distinguishers.

3.2.1. The effect of input difference with hamming weight of 1

We use all differences with a Hamming weight of 1 as the input difference of the 9-round differential distinguisher for training. Similarly, the size of training sets is 10^7, and the size of the validation set is 10^5. At the same time, we set the network to carry out 100 epochs. We choose $(0x0, 0x1)$, $(0x0, 0x2)$, $(0x1, 0x0)$ and $(0x2, 0x0)$ as representatives, and draw the change of accuracy and loss.

As shown in Figure 5, if the input difference is $(0x0, 0x1)$ or $(0x0, 0x2)$, the accuracy of the

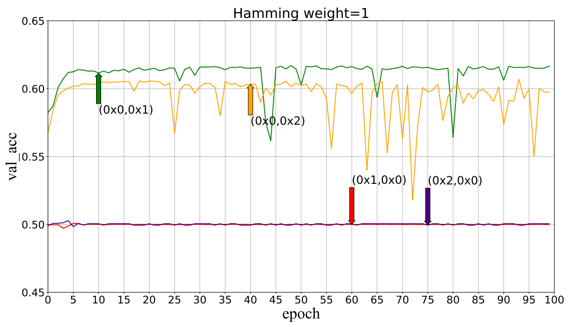

FIGURE 5. Training process with four input differences

TABLE 1. Accuracy with different input difference

Accuracy	Input difference
60%−62%	$(0x0, 0x1), (0x0, 0x10), (0x0, 0x100), (0x0, 0x1000),$ $(0x0, 0x2), (0x0, 0x20), (0x0, 0x200), (0x0, 0x2000),$ $(0x0, 0x4), (0x0, 0x40), (0x0, 0x400), (0x0, 0x4000),$ $(0x0, 0x8), (0x0, 0x80), (0x0, 0x800), (0x0, 0x8000)$
50%−51%	$(0x1, 0x0), (0x10, 0x0), (0x100, 0x0), (0x1000, 0x0),$ $(0x2, 0x0), (0x20, 0x0), (0x200, 0x0), (0x2000, 0x0),$ $(0x4, 0x0), (0x40, 0x0), (0x400, 0x0), (0x4000, 0x0),$ $(0x8, 0x0), (0x80, 0x0), (0x800, 0x0), (0x8000, 0x0)$

distinguisher exceeds 60%, and if the input difference is $(0x1, 0x0)$ or $(0x2, 0x0)$, the accuracy is within about 50%. It can be seen that the difference in the accuracy of the distinguisher with different input differences is huge, with a difference of more than 10%. We count the accuracy of the distinguisher with different input difference as shown in Table 1.

As shown in Table 1, we can see that the difference in accuracy from different input difference is huge. Our distinguisher is used to distinguish the ciphertext pair with a given input difference from random data, which is the binary classification in deep learning. Therefore, for the distinguisher model, if its accuracy is about 50%, we can think that the distinguishing effect of the distinguisher is not obvious.

The accuracy of the distinguisher is basically divided into the first 16 bits and the last 16 bits in Table 1. In the input difference with Hamming weight of 1, the accuracy is about 50% if the non-zero bit is in the first 16 bits. If the non-zero bit is in the last 16 bits, the accuracy is around 60%. This situation is mainly related to the round function of SIMON32. In the round function of SIMON32, its nonlinear operation is mainly concentrated in the first 16 bits. Assuming that the input plaintext pair is (L_0, R_0) and $\left(L_0^{'}, R_0^{'}\right)$ and the subkey is k, with $L_0 \oplus L_0^{'} = \Delta_L$ and $R_0 \oplus R_0^{'} = \Delta_R$, the plaintext pair can be obtained after 1-round encryption as follows:

$$(L_1, R_1) = \left(R_0 \oplus \left(S^1 L_0 \& S^8 L_0\right) \oplus S^2 L_0 \oplus k, L_0\right);$$

$$\left(L_1^{'}, R_1^{'}\right) = \left(R_0^{'} \oplus \left(S^1 L_0^{'} \& S^8 L_0^{'}\right) \oplus S^2 L_0^{'} \oplus k, L_0^{'}\right).$$

TABLE 2. Differential characteristics of SIMON32

Round	ΔL	ΔR	$\log_2(p)$
0	$0x0$	$0x200$	0
1	$0x200$	$0x0$	0
2	$0x80$	$0x200$	-2
3	$0x220$	$0x80$	-2
4	$0x8$	$0x220$	-4
5	$0x222$	$0x8$	-2
6	$0x8080$	$0x222$	-6
7	$0x2202$	$0x8080$	-4
8	$0x800$	$0x2202$	-6
9	$0x2002$	$0x800$	-2
10	$0x8000$	$0x2002$	-4
11	$0x2$	$0x8000$	-2
12	$0x0$	$0x2$	-2
13	$0x2$	$0x0$	0

If there are $\Delta_L = 0$ and $\Delta_R \neq 0$, there are $L_1 \oplus L_1^{'} = \Delta_R$ and $R_1 \oplus R_1^{'} = 0$. And if there are $\Delta_L \neq 0$ and $\Delta_R = 0$, there are $L_1 \oplus L_1^{'} = \left(S^1 L_0 \& S^8 L_0\right) \oplus \left(S^1 L_0^{'} \& S^8 L_0^{'}\right) \oplus S^2 \Delta_L$ and $R_1 \oplus R_1^{'} = \Delta_L \neq 0$.

That is, if the first 16 bits of the input difference are 0, the non-zero bits will spread less in the next round; and if there are non-zero bits in the first 16 bits of the input difference, due to the non-linear operations, the non-zero bits will diffuse into the next round of input. This is consistent with the analysis of the differential characteristic in the traditional differential cryptanalysis. This shows that the traditional differential cryptanalysis is helpful for the neural distinguisher.

3.2.2. The effect of input differences with the same probability

In order to explore the impact of the input difference of the differential characteristics with the same probability on the accuracy, we need to find some differential characteristics that meet the requirements, and use its input difference to compare the impact of the input difference. Abed F et al. searched for 13-round differential characteristics of SIMON32 in [19], and gave the probability between each round, as shown in Table 2. We use the differential characteristics shown in Table 2 to find the input difference that meets the requirements.

In the difference characteristics shown in Table 2, we find two groups of appropriate input differences, and the corresponding rounds is 6 and 7 respectively. Their probability are 2^{-24} and 2^{-26} respectively, as shown in Table 3.

As shown in Table 3, we select the input difference $(0x80, 0x200)$, $(0x220, 0x80)$, $(0x8, 0x220)$ and $(0x222, 0x8)$ as the input difference to train the 6-round distinguisher based on deep learning. Similarly, we select the difference $(0x200, 0x0)$, $(0x80, 0x200)$, $(0x8, 0x220)$, and $(0x222, 0x8)$ as input difference

TABLE 3. Differential characteristics with same probability

Round	$\Delta input$	$\Delta output$	$\log_2(p)$
6	$(0x80, 0x200)$	$(0x800, 0x2202)$	-24
	$(0x220, 0x80)$	$(0x2002, 0x800)$	-24
	$(0x8, 0x220)$	$(0x8000, 0x2002)$	-24
	$(0x222, 0x8)$	$(0x2, 0x8000)$	-24
7	$(0x200, 0x0)$	$(0x800, 0x2202)$	-26
	$(0x80, 0x200)$	$(0x2002, 0x800)$	-26
	$(0x8, 0x220)$	$(0x2, 0x8000)$	-26
	$(0x222, 0x8)$	$(0x0, 0x2)$	-26

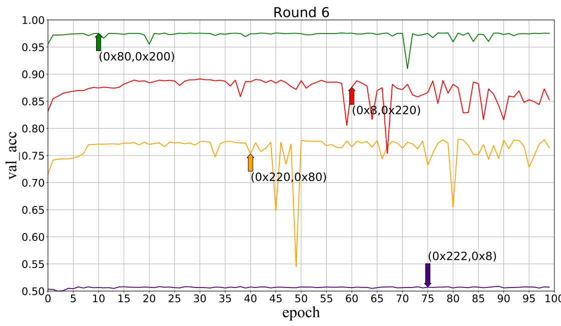

FIGURE 6. 6-round distinguisher with different input difference

to construct the 7-round distinguisher.We plot the accuracy changes of the distinguisher with different input difference on the verification set during the training process as Figure 6 and Figure 7. As shown in Figure 6 and Figure 7, it can be seen that although the probability of the differential characteristics is same, the accuracy of the distinguisher obtained is different given different input differences. The difference between the highest accuracy and the lowest accuracy of the distinguisher exceeds 40%.

The traditional distinguisher uses only one or some differential characteristics. For the neural distinguisher, since it only needs a fixed input difference, the differential distinguisher can learn more features from a given input difference. This is why although the probability of the differential characteristics is the same, the accuracy is obviously different. Due to that the neural distinguisher is essentially learning multiple characteristics at the same time, so the distinguisher is better at distinguishing than traditional one. This also inspires us to use the differential distinguisher based on deep learning to perform differential attacks, and its effect may exceed the traditional differential attack.

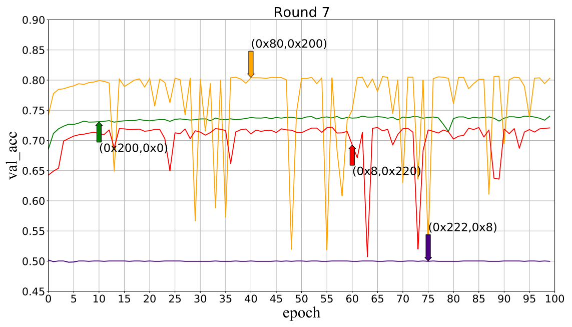

FIGURE 7. 7-round distinguisher with different input difference

4. KEY RECOVERY ATTACK OF ROUND-REDUCED SIMON32

In order to show that the neural distinguisher is effective for key recovery, we complete practical 11-round key recovery attack on a workstation configured with *Intel i9-10900K* and *Nvidia TITAN RTX*. It takes about 45s to recover the final subkey, with a success rate of more than 90%. Our attack only needs no more than $2^{18.5}$ 11-round encryption and the data complexity does not exceed 2^9. With the traditional differential attack in [19], it requires 2^{28} plaintext pairs and more than 2^{29} 11-round encryption. Therefore the data complexity and time complexity based on deep learning is far lower than that of the traditional differential cryptanalysis. In addition, we improved the rounds of key recovery based on SAT, we extend our neural 9-round distinguisher to a 11-round distinguisher by prepending the 2-round differential characteristic $(0x800, 0x2200) \xrightarrow{2^{-4}} (0x0, 0x200)$. Then, we propose a attack on 13-round SIMON32/64. Our attack uses about $2^{12.5}$ chosen plaintexts and only needs about 45s for an attack, which does not exceed $2^{18.5}$ 13-round encryption.

4.1. Basic Attack Idea

4.1.1. Overview

Thanks to the research about the influence of input difference pattern on the accuracy of neural distinguisher, we choose $(0x0, 0x200)$ as the input difference and train 8-round and 9-round neural distinguisher of SIMON32. Due to the round function of the SIMON32, the key-addition occurs after the non-linear operation, we can construct 11-round key-recovery attack by adding 1 round before and after 9-round distinguisher. At the same time, we choose these plaintext pairs whose difference is $(0x0, 0x200)$ after 1-round encryption. These plaintext pairs which meet the requirements are encrypted for 11 rounds, and the ciphertext pairs is obtained.

It is based on a hypothesis: if the final subkey is correctly guessed, the probability that the intermediate state obtained by the correct last subkey and the correct second-to-last subkey passs through the 8-round distinguisher is the highest. That is to say, the response of the 8-round distinguisher is the highest if 2-round subkey are guessed correctly. This is due to the fact that the intermediate state decrypted by the error key is a random sequence for distinguisher given by the fixed difference, that is, the distinguisher will returns a value of approximately 0.5 if the guessed key is wrong. Based on the above assumptions, the approach which recover the last subkey is explained. We guess possible keys in the last round, we use the guessed subkey to perform 1-round decryption, and use 9-round distinguisher to score and sort the guessed subkeys. If the score of a key exceeds the threshold $t1$, we use the key to decrypt one round. At the same time, guess the second-to-last subkey, and similarly, score and sort them. If the score exceeds the threshold $t2$, the last guessed subkey will be returned as the result.

4.1.2. Preconstructing-plaintext technique

The early abort technique has been widely used in various key-recovery attacks [27], which aims to reduce the number of chosen plaintext pairs by using plaintext features and reduce the complexity of the attack. Inspired by the technique, we use preconstructing-plaintext technique to get appropriate plaintext pairs. Different from the traditional method of constructing and then screening plaintext pairs, we directly construct plaintext pairs that meet the conditions.

As shown in 4.1.1, we need these plaintext pairs whose difference is $(0x0, 0x200)$ after 1-round encryption. We choose a random pair, where the difference is $(0x0, 0x200)$. At the same time, we choose a random subkey to decrypt it and get a plaintext pair whose difference is $(0x0, 0x200)$ after 1-round encryption. This can been proven rigorously.

Assuming that the random pair are (L, R) and $\left(L^{'}, R^{'}\right)$ and the random subkey is rk, with $L \oplus L^{'} = 0x0$ and $R \oplus R^{'} = 0x200$, the plaintext pair can be obtained after 1-round decryption as follows:

$$\begin{aligned}(L_0, R_0) &= \left(R, \left(S^1R\&S^8R\right) \oplus S^2R \oplus rk \oplus L\right) \\ \left(L_0^{'}, R_0^{'}\right) &= \left(R^{'}, \left(S^1R^{'}\&S^8R^{'}\right) \oplus S^2R^{'} \oplus rk \oplus L^{'}\right)\end{aligned} \tag{2}$$

We use the subkey sk to encrypt it for 1 round and get:

$$\begin{aligned}(L_1, R_1) &= \left(R_0 \oplus \left(S^1L_0\&S^8L_0\right) \oplus S^2L_0 \oplus sk, L_0\right) \\ \left(L_1^{'}, R_1^{'}\right) &= \left(R_0^{'} \oplus \left(S^1L_0^{'}\&S^8L_0^{'}\right) \oplus S^2L_0^{'} \oplus sk, L_0^{'}\right)\end{aligned} \tag{3}$$

According to Equation 2 and Equation 3, we can get:

$$\begin{aligned}L_1 \oplus L_1^{'} = R_0 &\oplus \left(S^1L_0\&S^8L_0\right) \oplus S^2L_0 \oplus \\ R_0^{'} &\oplus \left(S^1L_0^{'}\&S^8L_0^{'}\right) \oplus S^2L_0^{'} \\ &= \left(S^1R\&S^8R\right) \oplus S^2R \oplus rk \oplus L \oplus \\ &\left(S^1L_0\&S^8L_0\right) \oplus S^2L_0 \oplus \\ &\left(S^1R^{'}\&S^8R^{'}\right) \oplus S^2R^{'} \oplus rk \oplus L^{'} \oplus \\ &\left(S^1L_0^{'}\&S^8L_0^{'}\right) \oplus S^2L_0^{'} \\ &= \left(S^1R\&S^8R\right) \oplus S^2R \oplus rk \oplus L \oplus \\ &\left(S^1R\&S^8R\right) \oplus S^2R \oplus \\ &\left(S^1R^{'}\&S^8R^{'}\right) \oplus S^2R^{'} \oplus rk \oplus L^{'} \oplus \\ &\left(S^1R^{'}\&S^8R^{'}\right) \oplus S^2R^{'} \\ &= 0x0 \\ R_1 \oplus R_1^{'} = L_0^{'} \oplus L_0 &= R \oplus R^{'} = 0x200.\end{aligned} \tag{4}$$

Therefore, our above method of constructing plaintext pair is effective. We give a detailed description of our method as shown in Algorithm 2.

Algorithm 2 Process for generating plaintext pair

```
Input: Input difference $\Delta = (0x0, 0x200)$
Output: Ciphertext pair
1: Randomly generate plaintext $P = (L, R)$ and key $rk$;
2: Calculate $L^{'} = L \oplus 0x0$ and $R^{'} = R \oplus 0x200$, and get $P^{'} = (L^{'}, R^{'})$;
3: Use $rk$ to decrypt $P$ and $P^{'}$ for 1 round and get $M$ and $M^{'}$;
4: Return $M$ and $M^{'}$;
```

4.2. Key recovery for SIMON32

4.2.1. Precomputation

Let C_0 and $C_0^{'}$ are a pair of ciphertext and k is the subkey of the last round. For $\delta \in GF(2)^{16}$, there is $k^{'} = k \oplus \delta$. Use $k^{'}$ as a subkey to decrypt the ciphertext pair, and then distinguish the intermediate state with the neural distinguisher, and get the response as $R_\delta\left(C_0, C_0^{'}\right)$. Then $R_\delta\left(C_0, C_0^{'}\right)$ can be regarded as a random variable related to δ. We can assume that $R_\delta\left(C_0, C_0^{'}\right)$ follows a normal distribution with mean μ_δ and standard deviation σ_δ.

In order to compare the influence of wrong keys on the response of the neural distinguisher, we conduct the following experiments. For $\delta \in GF(2)^{16}$, $R = 11$ and a random master key K, we use Algorithm 2 to generate 1 ciphertext pair $\left(C_0, C_0^{'}\right)$ and save its last key rk. Calculate $k^{'} = rk \oplus \delta$ and use $k^{'}$ to decrypt $\left(C_0, C_0^{'}\right)$ for one round, we get the response value of the 9-round distinguisher to the intermediate state that has been decrypted for 1 round. We repeat the above steps 2^{16} times and calculate the mean value and standard deviation of the response values. Taking the mean value as an example, we plot the changes of the mean value corresponding to different δ as shown in Figure 8.

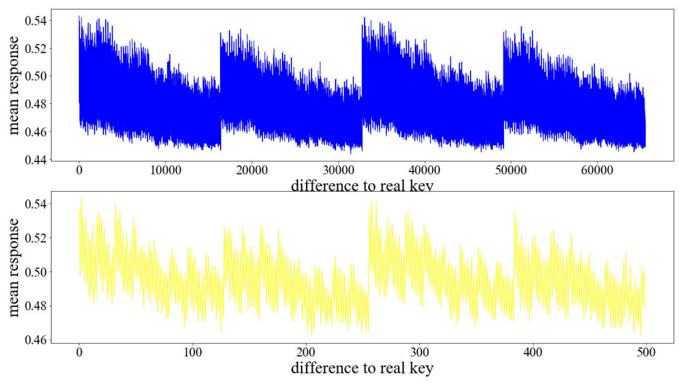

FIGURE 8. Mean response of difference to real key

As shown in Figure 8, we find that as δ changes, the distribution of the mean of the response value μ_δ is not uniform, but its change curve is regular. This is a similar distribution for standard deviation σ_δ. If $R = 10$, there are similar results. This regular change helps recover the last key.

4.2.2. Specific attack process

According to 4.2.1 and regular curve, we use μ_δ and σ_δ to help recover the last key. Different from the traditional key recovery attack, what we study is the distance between the guess key and the real key. If the distance between the real key and the guess key is 0, we get the correct subkey. In 4.2.1, we assume that δ is the distance between the guess key k_i and the real key k. So we can use $\sum_{i=0}^{n-1}\frac{\left(m_{k_i}-\mu_{k\oplus k_i}\right)^2}{\sigma_{k\oplus k_i}}$ to measure k_i and k, where m_{k_i} is the response and n is the number of ciphertext pair. This is the same as [13].

At the same time, to filter out the best guess subkey, we need to rate all guesses as shown in Algorithm 3.

Algorithm 3 KeyRate

Input: Ciphertext pair set: $C = \{C_0, C_1, ..., C_{n-1}\}$, Distinguisher based on deep learning: D, Guessed key: k
Output: Score of k: $Score_k$

```
1: $P_i \leftarrow Decrypt_1(C_i, k)$, for $i \in 0, 1, ..., n-1$;
2: $v_i \leftarrow D(P_i)$, for $i \in 0, 1, ..., n-1$;
3: $w_i \leftarrow \log_2\left(\frac{v_i}{1-v_i}\right)$, for $i \in 0, 1, ..., n-1$;
4: $Score_k \leftarrow \sum_{i=0}^{n-1} w_i$;
5: return $Score_k$;
```

Bayesian search algorithm is proposed to search the key in [13], which is effective to recover subkey. So we use the same method to search for keys, and the method is shown in Algorithm 4.

Algorithm 4 Gohr's KeySearch [13]

Input: Ciphertext pair set $C = \{C_0, C_1, ..., C_{n-1}\}$, Distinguisher based on deep learning D, number of candidates to be generated t, number of iterations l
Output: Key set L

```
1: $S \leftarrow \{k_0, k_1, ..., k_{t-1}\}$, $k_i \neq k_j$ if $i \neq j$;
2: $L \leftarrow \{\}$;
3: for $j \in \{0, 1, ..., l-1\}$ do
4:     $P_{i,k} \leftarrow Decrypt_1(C_i, k)$ for all $i \in \{0, 1, ..., n-1\}$ and $k \in S$
5:     $v_{i,k} \leftarrow D(P_{i,k})$ for all $i \in \{0, 1, ..., n-1\}$ and $k \in S$
6:     $w_{i,k} \leftarrow \log_2\left(\frac{v_{i,k}}{1-v_{i,k}}\right)$ for all $i \in \{0, 1, ..., n-1\}$ and $k \in S$
7:     $L \leftarrow L||[(k, \sum_{i=0}^{n-1} w_{i,k})$ for $k \in S]$
8:     $m_k \leftarrow \sum_{i=0}^{n-1} w_{i,k}/n$ for $k \in S$
9:     $\lambda_k \leftarrow \sum_{i=0}^{t-1}\left(\frac{m_{k_i}-\mu_{k_i\oplus k}}{\sigma_{k_i\oplus k}}\right)^2$ for $k \in \{0, 1, ..., 2^{16}-1\}$
10:    $S \leftarrow argsort_k(\lambda)[0 : t-1]$
11: end for;
12: return $L$;
```

In [13], Gohr proposed to use neutral bits [18] to enhance the ability of the distinguisher. We learn from Gohr's idea of using neutral bits to improve our key recovery accuracy. We choose a certain plaintext pair and construct a plaintext structure to enhance the distinguishing ability by choosing a suitable plaintext structure. In this way, we can get better plaintext pair set by using neutral bits which can enhance the distinguishing ability. The method of constructing plaintext pair set with neutral bits is shown in Algorithm 5.

Algorithm 5 Constructing plaintext pair set with neutral bits

1 **Input:** Input difference $\Delta = (0x0, 0x200)$, Number of ciphertext pair n , Neutral bits $[q_0, q_1 ..., q_{m-1}]$
Output: Ciphertext pair set

```
1: Randomly generate $n$ plaintexts;
2: Extended plaintext set using neutral bits and get $S = \{s_{0,0}, s_{0,1}, ..., s_{0,2^m-1}, ..., s_{n-1,2^m-1}\}$;
3: $S' = \{s_{0,0} \oplus \delta, s_{0,1} \oplus \delta, ..., s_{0,2^m-1} \oplus \delta, ..., s_{n-1,2^m-1} \oplus \delta\}$;
4: $M \leftarrow Decrypt_1(s, 0)$ for $s \in S$; $M' \leftarrow Decrypt_1(s', 0)$ for $s' \in S'$;
5: return $(M, M')$;
```

Using Algorithm 5, we obtain 2^m plaintext structures based on neutral bits. When performing key recovery, we don't select all ciphertext structures for key recovery attacks, but choose the one that can make the guess key score higher. In order to filtrate the better neutral bits for key recovery, we randomly generate 10000 sets of neutral bits and calculate its accuracy in 25 key recovery attacks. If the success rate of 25 key recovery attacks exceeds 93%, we use the neutral bits to perform 100 key recovery attacks, and save the neutral bits with a success rate of more than 90%.

We have obtained some neutral bit, and their success rate in 100 experiments exceeded 90%, of which the highest success rate is 92%. These neutral bits are as follows: [27,19,3,23,18,22], [27,28,7,3,20,2], [25,27,23,11,6,3], [3,18,4,19,21, 13], [7,19,2,8, 3,4] and [27,25,15,8,0,10].

4.2.3. Improved 13-round attack based on SAT

In [28], Stefan et al. proposed an algorithm to search for the differential characteristics of SIMON based on SAT. We use the algorithm to increase the round of attacks. First, we automatically search for for 2-round differential characteristic $(0x800, 0x2200) \xrightarrow{2^{-4}} (0x0, 0x200)$ based on SAT. Then we extend our neural 9-round distinguisher to a 11-round distinguisher by prepending the 2-round differential. Utilizing the differential characteristic, we can attack 13-round SIMON32/64 and recovery the last subkey similar to 11-round attack. We have conducted 100 experiments with the accuracy exceeded 90%. Our attack only uses about $2^{12.5}$ chosen plaintexts and only needs about 45s for an attack, which does not exceed $2^{18.5}$ 13-round encryption.

4.3. Complexity analysis and comparison

In the experiment, the complexity of our attack is calculated based on the average value of each experiment. For the 11-round key recovery attack, our attack only needs no more than 45s each time, which does not exceed $2^{18.5}$ 11-round encryption, and the data complexity does not exceed 2^9.

To show that the neural differential distinguisher has advantages in key recovery attacks, we use the traditional cryptanalysis to perform the recovery attacks for last key on 11-round SIMON32. Due to the round function of the SIMON32, the key-addition occurs after the non-linear operation, we can construct 11-round attack by adding one round before and after 9-round differential characteristics. We perform attacks with the 9-round characteristic $(0x0, 0x200) \rightarrow (0x2002, 0x800)$ and attack approach mentioned in [18]. With the traditional method, the time complexity is more than 2^{29} 11-round encryption, and the data complexity is 2^{28}. At the same time, by reason of the differential characteristic added before the neural differential discriminator, the key recovery attack on 13-round SIMON32 needs more plaintexts and time. For the 13-round key recovery attack, our attack only needs no more than 45s each time, which does not exceed $2^{18.5}$ 13-round encryption. At the same time, the data complexity does not exceed $2^{12.5}$.

By comparing traditional differential cryptanalysis with differential cryptanalysis based on deep learning, we find that differential cryptanalysis based on deep learning have advantages in key recovery attacks. In traditional differential cryptanalysis, the traditional distinguisher uses a differential characteristic or multiple characteristics with given input difference and output difference. In contrast, the neural distinguisher only uses given input difference, which considers more the output differences effect under the same input difference. This makes the neural distinguisher to be more powerful in differential cryptanalysis.

5. CONCLUSION

This paper investigates how to construct a differential distinguisher of SIMON32 based on deep learning and constructs key recovery attack on 11-round SIMON32. We find that the input difference pattern has an impact on the accuracy of the neural distinguisher when we construct distinguishers. It is helpful to find appropriate input differences which is used to train neural distinguisher with higher accuracy. In addition, we complete practical 11-round key recovery attack with lower complexity than traditional differential cryptanalysis. At the same time, we improved the round of attack based on SAT, and the highest attack reached 13 rounds. To our excitement, the success rate is more than 90%.

This paper investigates the impact of two kinds of input difference patterns on the accuracy of the neural distinguisher. It is significative to study other input difference patterns, which is a direction of follow-up research. At the same time, a new key search policy for searching key is also worth studying.

REFERENCES

[1] McCulloch, W.S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.

[2] Rosenblatt, F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, 65, 386-408.

[3] Rumelhart, D., Hinton, G. and Williams, R. (1986) Learning representations by back-propagating errors. Nature, 323, 533-536.

[4] Steve, Lawrence. Giles, C.L. Tsoi, A.C. and Andrew, D,B. (1997) Face recognition: a convolutional neural-network approach. IEEE Transactions on Neural Networks, 8, 98-113.

[5] Williams, R. and Zipser, D. (2014) A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Computation, 1, 270-280.

[6] Hinton, G.E., Osindero, Simon. and Yee-Whye Teh. (2006) A fast learning algorithm for deep belief nets. Neural Computation, 18, 1527-1554.

[7] Glorot, X., Bordes, A. and Bengio, Y. (2011) Deep Sparse Rectifier Neural Networks. Journal of Machine Learning Research, 15, 315-323.

[8] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, 27-30 June, pp. 770-778. IEEE, New Jersey.

[9] Rivest, R.L. (1991) Cryptography and machine learning. International Conference on the Theory and Application of Cryptology. Japan, November, pp. 427-439. Springer, Berlin.

[10] Masure, L., Dumas, C. and Prouff, E. (2019) Gradient Visualization for General Characterization in Profiling Attacks. Constructive Side-Channel Analysis and Secure Design. Darmstadt, 3C5 April, pp. 145-467. Springer, Cham.

[11] Liu, Z., Zhu, T. (2020) Research on Side-Channel Attack Based on the Synergy between SNR and Convolutional Neural Networks. Journal of Physics: Conference Series. 1575, 012026.

[12] Benadjila, R., Prouff, E., Rmi, S., et al. (2020) Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering, 10, 163-188.

[13] Gohr, A. (2019) Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. In Boldyreva, A. and Micciancio, D. (eds), Advances in Cryptology - CRYPTO 2019. Springer, Cham.

[14] Beaulieu, R., Shors, D., Smith, J. et al. (2013) The SIMON and SPECK Families of Lightweight Block Ciphers. 52nd ACM/EDAC/IEEE Design Automation Conference, San Francisco, 8-12 June, pp. 1-6. IEEE, New Jersey.

[15] Baksi, A., Breier, J., Yang, X. and Yi, C. (2020) Machine Learning Assisted Differential Distinguishers For Lightweight Ciphers. IACR Cryptology ePrint Archive, 2020, 571.

[16] Jain, A., Kohli, V. and Mishra, G. (2020) Deep Learning based Differential Distinguisher for Lightweight Cipher PRESENT. IACR Cryptology ePrint Archive, 2020, 846.

[17] Bogdanov, A., Knudsen, L.R., Leander, G., et al. (2007) PRESENT: An Ultra-Lightweight Block Cipher. Cryptographic Hardware and Embedded Systems, Vienna, 10-13 September, pp. 450-466. Springer, Berlin.

[18] Biham, E. and Chen, R. (2004) Near-Collisions of SHA-0. In Franklin, M. (eds), Advances in Cryptology-CRYPTO 2004. Springer, Berlin, Heidelberg.

[19] Abed, F., List, E., Wenzel, J., et al. (2014) Differential Cryptanalysis of Round-Reduced Simon and Speck. International Workshop on Fast Software Encryption, London, 3-5 March, pp. 525-545. Springer, Berlin.

[20] Wang, Q., Liu, Z., Kerem, V., et al. (2014) Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In Meier, W. and Mukhopadhyay, D. (eds), Progress in Cryptology - INDOCRYPT 2014. Springer, Cham.

[21] Sun, S., Hu, L., Wang, P., et al. (2014) Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In Sarkar, P. and Iwata, T. (eds), Advances in Cryptology - ASIACRYPT 2014. Springer, Berlin.

[22] Rabbaninejad, R., Ahmadian, Z., Salmasizadeh, M., et al. (2014) Cube and dynamic cube attacks on SIMON32/64. 11th International ISC Conference on Information Security and Cryptology, Tehran, Iran, 3-4 September, pp. 98-103. IEEE, New Jersey.

[23] Biham, E. and Shamir, A. (1993) Differential Cryptanalysis of the Data Encryption Standard. Springer Nature, Switzerland.

[24] Biryukov, A., Nikolic, I. (2010) Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In Gilbert, H. (eds), Advances in Cryptology-EUROCRYPT 2010. Springer, Berlin.

[25] Fu, K., Wang, M., Guo, Y., Sun, S. and Hu, L. (2016) MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In Peyrin, T. (eds), Fast Software Encryption 2016. Springer, Berlin.

[26] Ketkar, N. (2017) Deep Learning with Python. Apress, Berkeley, CA.

[27] Lu, J., Kim, J., Keller, N., et al. (2008) Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In Malkin, T. (eds), Topics in Cryptology - CT-RSA 2008. Springer, Berlin, Heidelberg.

[28] Stefan, Kolbl. Gregor, Leander. and Tyge, Tiessen. (2015) Observations on the SIMON Block Cipher Family, Proceedings of the CRYPTO2015, CA, USA, 16-20 August, pp. 161-185. Springer Nature, Switzerland.