Improve Neural Distinguisher for Cryptanalysis*

ZeZhou Hou, JiongJiong Ren and ShaoZhen Chen

Information Engineering University, ZhengZhou, P.R.China, jiongjiong_fun@163.com

Abstract. At CRYPTO'19, Gohr built a bridge between deep learning and cryptanalysis. Based on deep neural networks, he trained neural distinguishers of Speck32/64 using a plaintext difference and single ciphertext pair. Compared with purely differential distinguishers, neural distinguishers successfully use features of the ciphertext pairs. Besides, with the help of neural distinguishers, he attacked 11-round Speck32/64 using Bayesian optimization. At EUROCRYPTO'21, Benamira *et al.* proposed a detailed analysis about the inherent workings of Gohr's distinguishers. Although their work opened a new direction of machine learning aided cryptanalysis, there are still two research gaps that researchers are eager to fill in. (1) How to further improve neural distinguishers? (2) Can we conduct effective key recovery on large-size block ciphers adopting neural distinguishers?

In this paper, we propose a new algorithm and model to improve neural distinguishers in terms of accuracy and the number of rounds and present effective neural aided attack on large-size block ciphers. First, we design an algorithm based on SAT to improve neural distinguishers. With the help of SAT/SMT solver, we obtain new effective neural distinguishers of SIMON using the input differences of high-probability differential characteristics. Second, we propose a new neural distinguisher model using multiple output differences. Inspired by Benamira's work and data augmentation in deep learning, we use the output differences to exploit more derived features and train neural distinguishers, by splicing output differences into a matrix as a sample. Based on the new model, we construct neural distinguishers of SIMON and Speck with round and accuracy promotion. Utilizing our neural distinguishers, we can distinguish reduced-round NSA block ciphers from pseudo-random permutation better.

Moreover, we perform practical key recovery attacks on different versions of SIMON. For SIMON32/64 and SIMON48/96, we append additional 2-round optimal characteristics searched by SAT/SMT solver to the beginning of our neural distinguishers and attack 13-round SIMON32/64, 14-round SIMON48/96 using Gohr's key recovery frame. For SIMON64/128, it costs too much time in precomputation, especially in wrong key response profile, which is unbearable for most of researchers. However, we show with experiments that the distribution of the wrong key profile is pseudo-periodic. Based on this, we make use of partial wrong key profile to describe the whole wrong key response profile, and then propose a generic key recovery attack scheme which can attack large-size block ciphers. As an application, we perform a key recovery attack on 13-round SIMON64/128 using a 11-round neural distinguisher. All our results are confirmed with experiments (source code available online).

Keywords: Deep Learning · Block Cipher · Neural distinguisher · Key Recovery · SIMON · Speck

1 Introduction

As a chosen plaintext attack, differential cryptanalysis is one of the most powerful analysis techniques used in modern block ciphers. It can achieve key recovery attacks utilizing

*All the code and data used in this paper will be released on Github

plain-cipher difference pair, which is expressed in input difference and output difference. Differential cryptanalysis has been introduced in 1990 by Biham and Shamir in [BS91] to break the DES block cipher. This statistical cryptanalysis exploits how a specific input difference propagates through the cipher into the output difference. The most important step of differential cryptanalysis is to find differential trails with high probabilities. This technique has been widely applied to block ciphers and hash functions, and many new constructions of these primitives are specifically designed to withstand this attack. In traditional differential cryptanalysis, it is key to construct high-probability differential characteristics. Recently some tools for automatically searching for differential distinguishers were presented [SHY16, KLT15]. In traditional cryptanalysis, we normally use the optimal differential characteristics or the optimal differentials, which causes that only an output difference is considered given a fixed input difference. But the differential distribution also contains some other information about block cipher, which has not been fully utilized in traditional cryptanalysis. Therefore, these still cannot fully make use of the differential characteristics and differential distribution.

Deep learning (DL) has played an important role in many fields, but its development is bumpy. In 1943, McCulloch and Pitts [MP43] 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. In 1958, on the basis of MP neural, Rosenblatt [Ros58] proposed the single-layer feedforward neural network named single-layer perceptron, which can distinguish between triangle, square and other basic shapes. In 1986, the second generation of neural networks was put forward by Rumelhart [RHW86]. 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, which effectively solved the problem that the first generation only can be used in linear classification. However, shallower neural network cannot complete the corresponding task with the amount of data increasing. In 2006, Hinton *et al.* [HOT06] put forward the concept of deep learning for the first time, which played a huge role. With the development of deep learning, there are diverse neural networks used in diverse fields [LGTB97, WZ89, GPM+14].

Related work: Our work is most closely related to combine deep learning and differential cryptanalytic techniques. At Crypto2019, Gohr [Goh19] showed 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 used an input difference to train neural distinguishers of Speck32 [BSS+13] based on the deep residual neural networks (ResNet) [HZRS16]. If the accuracy of a neural distinguisher exceeds 0.5, the neural distinguisher can distinguish target cipher E from pseudo-random permutation. 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. But there are some questions worthy of exploring:

Why are neural distinguishers effective? How to improve neural distinguishers in terms of accuracy and the number of rounds? Can we use neural distinguishers to attack large-size block ciphers?

In [CY21c], Chen *et al.* designed a new neural distinguisher model using multiple ciphertext pairs instead of single ciphertext pair. The new neural distinguisher can be used to improve the key recovery attack on 11-round Speck32/64. Ulteriorly, it makes sense to explore theoretical attacks using neural distinguishers. Chen *et al.* proposed neural aided statistical attack (NASA) for cryptanalysis in [CY20, CY21b], which makes the use of the neural distinguishers not limited to practical attacks. By a theoretical attack based

on neural distinguishers, Chen *et al.* attacked reduced-round Speck32/64 and Speck48/x, where the computation complexity of their attack is similar to the previous traditional attacks. Chen's work extends the scope of application of the neural distinguishers from practical attack to theoretical attack, and the new model is effective to improve accuracy of neural distinguishers. In [CY21a], Chen *et al.* explained phenomena related to neural distinguishers via EDLCT (Extended Differential-Linear Connectivity Table). Chen *et al.* partially answered the second question from the perspective of data format. But they don't explore to improve the accuracy from the perspective of input difference or output difference, which is not conducive to finding a longer-round neural distinguisher. Meanwhile, a study into the inherent workings of neural distinguisher is of both theoretical and applied interest. In [BGPT21], Benamira *et al.* proposed a detailed analysis and thorough explanations of the inherent workings of Gohr's distinguishers. They showed that Gohr's neural distinguisher was in fact inherently building a very good approximation of the Differential Distribution Table (DDT). Based on this, Benamira *et al.* also constructed a 8-round distinguisher of SIMON32/64. In [BGPT21], Benamira *et al.* answered the first question and partially answered the second question. Whereas now, there are no related effective key recovery attacks on large-size block ciphers using neural distinguishers. And further improvement of the neural distinguishers is still worth studying. Because if the distinguishing accuracy is promoted, the complexity of key search can also be reduced. Inspired by their interesting work, our core target is to further improve neural distinguishers and propose effective key recovery policy to attack large-size block ciphers using neural distinguishers.

Our contribution: In this paper, our contributions are as follows:

- **Design an algorithm based on SAT to improve neural distinguishers.** In [Goh19], Gohr chose $(0x40, 0x0)$ as the input difference to train his distinguisher because it transitioned deterministically to the low-weight output difference. But such input differences are hard to find, which makes it difficult to find effective distinguishers. To solve this problem, we propose an algorithm based on SAT to improve neural distinguishers. With the help of this automatic search tool, we search for the exact nr-round differential characteristics with probability $\left[2^{-\frac{n}{4}} \times P_{\max}, P_{\max}\right]$ and choose their input differences to train nr-round neural distinguishers, where $P_{\max}$ is the optimal probability, n is the block size. Utilizing the algorithm, we obtain some neural distinguishers of 9-round SIMON32/64, 10-round SIMON48/96 and 11-round SIMON64/128 with the accuracy exceeding 57% for the first time. Compared with the choice of input difference presented in [BGPT21], our algorithm obtains higher-accuracy neural distinguishers.

- **Propose a new neural distinguisher model using multiple output differences and construct neural distinguishers for NSA block ciphers.** In image recognition based on deep learning, a deep learning researcher will enhance some objective features of pictures so that the neural network can learn more effective features, which will improve the accuracy of the network. In [BGPT21], Benamira *et al.* explored the connection between Gohr's distinguisher and DDT, which enlightens us the output difference is helpful to improve neural distinguishers. This also implies that we can selectively enhance certain features from output difference to improve neural distinguishers. Unlike [Goh19, CY21c] using ciphertext pairs as training data, we use the output differences to train neural distinguishers, by splicing output differences into a matrix as a sample. For a matrix, we treat it as an image and each output difference of the matrix is treated as an objective feature. Our goal is not only to learn each objective features, but to learn the connections between output differences. If all output differences of the matrix are from the same input difference, the matrix will be labeled 1, otherwise it will be labeled 0. Thanks to the new model learning more features than using ciphertext pairs, we improve neural

distinguishers of SIMON32/64, SIMON48/96 and SIMON64/128. Besides, we obtain new neural distinguishers of 8-round Speck32/64, 7-round Speck48/96 and 8-round Speck64/128, which is better than the existing neural distinguishers. Using our improved neural distinguishers, we can distinguish reduced NSA block ciphers from pseudo-random permutation better. As a footnote, we show with experiments that the improvement in the accuracy of distinguishers is not due to the increase in the number of plaintexts, but learning more features from the relationship between the output differences. The summary of our neural distinguishers together with other neural distinguishers is shown in Table 1.

Table 1: Comparison of neural differential distinguishers

Block Cipher	Source of Neural Distinguisher/Input difference	Round	Accuracy
SIMON32/64	[BGPT21]	8	82.2%
	[ALLW14][1]	9	59.07%
	[SZM20][2]	9	63.73%
	Section 3	9	59.77%
	Section 4	10	61.09%
SIMON48/96	[ALLW14][1]	9	50.22%
	[BGPT21][3]	10	53.49%
	Section 3	10	57.89%
	Section 4	11	81.40%
SIMON64/128	[ALLW14][1]	10	58.61%
	Section 3	11	59.72%
	Section 4	12	69.57%
Speck32/64	[Goh19]	7	61.6%
	[CY21c][2]	7	70.74%
	Section 4	7	88.19%
	[Goh19]	8	51.40%
	Section 4	8	56.49%
Speck48/96	[CY21b]	5	- [4]
	Section 4	7	63.43%
Speck64/128	Section 4	8	63.20%

[1] In [ALLW14], Abed *et al.* constructed differential characteristics of SIMON. We train neural distinguishers by choosing the input differences in [ALLW14].
[2] We choose the highest-accuracy neural distinguisher in [SZM20, CY21c].
[3] We train neural distinguishers using Benamira's method presented in [BGPT21].
[4] Chen *et al.* used 5-round neural distinguisher to attack Speck48/x, but the accuracy were not presented in [CY21b].

- **Perform key recovery attacks on reduced-round SIMON32/64 and SIMON48/96.** As applications of our neural distinguishers, we perform key recovery attacks on 13-round SIMON32/64 and 14-round SIMON48/96 with the help of Gohr's scheme. Similar to Gohr's work, we first obtain the wrong key response profile. Then we search for the optimal 2-round characteristics by SAT so that we can append two additional rounds to the beginning of neural distinguishers in Section 3. Finally, we attack 13-round SIMON32/64 and 14-round SIMON48/96 based on extended neural distinguishers. All of our attacks are performed on a workstation configured with *Intel i9-10900K* and *Nvidia TITAN RTX*. Our attacks cost about 23s and 1550s and each time to recover the final subkey for 13-round SIMON32/64 and 14-round SIMON48/96. The average time complexity is no more than $2^{16.4}$ 13-round encryption of SIMON32/64 and $2^{22.21}$ 14-round encryption of SIMON48/96. The two

attacks need about $2^{12.5}$ and $2^{12.8}$ ciphertext pairs respectively, which is much lower than the complexity of traditional differential cryptanalysis. It is worth mentioning that our attacks are all practical and the success rate of attacking SIMON32 is over 90%.

- **Perform practical key recovery attack on reduced-round SIMON64/128.** Gohr's framework is effective in key recovery of short-size block ciphers. Unfortunately, it costs a lot of time in precomputation for large-size block ciphers, especially in wrong key response profile. For SIMON32/64, it only spends about 1500s to obtain the complete profile, while it spends more than 1300 days[1] for SIMON64/128 on a fast graphics card of our system, which is unbearable for most researchers. By experimentally observing the wrong key response profile of SIMON32/64 and SIMON48/96, we find an interesting property that the distribution in the mean and standard deviation of the wrong key is pseudo-periodic. In other words, we don't need to obtain the full wrong key response for large-size block ciphers. Based on this, we design a generic algorithm to attack large-size block ciphers. Firstly, we make use of partial wrong key profile to describe the whole wrong subkey response profile. Once we get the complete profile portrayed partial profile, we can recover partial subkey bits with Gohr's key recovery policy. Then we guess the complete subkey based on known subkey bits. As an application, we perform a key recovery attack on 13-round SIMON64/128 using a 11-round neural distinguisher. The total complexity is $2^{25.700}$ and the data complexity is $2^{13.24}$. It shows that our attack scheme is effective in large-size block ciphers. This is the first attack based on neural distinguisher for large-size block ciphers. Our results are shown in Table 2.

Table 2: Summary of attacks on SIMON32/64, SIMON48/96 and SIMON64/128.

Cipher	Round	Time complexity	Data(CP)	Source
SIMON32/64	11	$2^{30.9}$	2^{25}	[SZM20]
SIMON32/64	13	$2^{16.4}$	$2^{12.5}$	Subsection 5.2
SIMON48/96	14	$2^{22.21}$	$2^{12.8}$	Subsection 5.1
SIMON64/128	13	$2^{25.700}$	$2^{13.24}$	Subsection 6.2

Organisation of the paper: The remaining of this paper is organised as follows. Section 2 reviews Gohr's work. In Section 3, we design an algorithm based on SAT to help us finding high-accuracy neural distinguishers. In Section 4, we propose a new neural distinguisher model to ulteriorly improve neural distinguishers. Combined with the content of Section 3, we perform key recovery attacks on 14-round SIMON48/96 and 13-round SIMON32/64 in Section 5. In Section 6, we design a generic algorithm to attack large-size block ciphers and attack 13-round SIMON64/128. Conclusions are drawn in Section 7 where we also suggest further work.

2 Preliminaries

To make it easier to read this paper, we first list the major notations. Then an overview of Gohr's work is given.

[1] We calculate 2^{16} wrong key response and estimate the time in 2^{32} wrong key response.

2.1 Notations

SIMON2n/mn : SIMON acting on $2n$-bit plaintext blocks and using a mn-bit key
Speck2n/mn : Speck acting on $2n$-bit plaintext blocks and using a mn-bit key
$\oplus$: bitwise XOR
$\odot$: bitwise AND
$\vee$: bitwise OR
$+$: addition modulo 2^n
S^j : left circular shift by j bits
K : Master key
k_i : i-round subkey $k_i = k_i^{n-1}||...||k_i^0$

2.2 Overview of Gohr's Work

2.2.1 Gohr's Distinguisher Model

Given a fixed input difference $\Delta = (0x40, 0x0)$ and a plaintext pair $\left(P^0, P^1\right)$, the resulting ciphertext pair $\left(C^0, C^1\right)$ is regarded as s sample. Each sample will be attached a label Y:

$$Y = \begin{cases} 1, if\ P^0 \oplus P^1 = \Delta \\ 0, else \end{cases} . \tag{1}$$

A neural network is trained over enough samples labeled 1 and 0. In addition, half the training data comes from ciphertext pairs labeled 1, half from ciphertext pairs labeled 0. For these samples with label 1, their ciphertext pairs are from a specific distribution related to the fixed input difference. For these samples with label 0, their ciphertext pairs are from a uniform distribution due to their random input difference. If a neural network can obtain a stable distinguishing accuracy higher than 50% on a testing set, we call the trained neural network a neural distinguisher. In [Goh19], Gohr chose the deep residual neural networks to train neural distinguisher and obtained effective neural distinguishers of 5-round, 6-round, and 7-round Speck32/64.

In traditional differential attack, it is pivotal to distinguish encryption function from a pseudo-random permutation, which is done with the help of the differential characteristic. For a nr-round optimal characteristic $\Delta\alpha \xrightarrow{2^{-t}} \Delta\beta$ of a block cipher with block size n bits, we calculate the output difference given the fixed input difference $\Delta\alpha$. If the ratio of the output difference to $\Delta\beta$ is about 2^{-t}, then we can distinguish the block cipher from a pseudo-random permutation. This is called distinguishing attack for block ciphers.

For Gohr's neural distinguisher, we can obtain N ciphertext pairs encrypted by the input difference $(0x40, 0x0)$. We input the N ciphertext pairs, and the neural distinguisher will predict their labels. If the ratio of samples labeled 1 exceeds 0.5, we can distinguish the block cipher and pseudo-random permutation and the neural distinguisher is effective. This is called distinguishing attack based on neural distinguisher. In addition, it is obvious that the higher the accuracy of the neural distinguisher, the better the effect of the distinguishing attack. And the complexity of key search can also be reduced if the distinguishing accuracy is greatly promoted. So it is necessary to improve neural distinguisher.

In [Goh19], Gohr explained the reason for choosing $(0x40, 0x0)$ as the input difference that it transitioned deterministically to the low-weight difference $(0x8000, 0x8000)$. But it is pretty hard to find such input differences unless the full differential distribution table is used. However it is a time consuming task to calculate the full DDT, especially for large-size block ciphers.

2.2.2 Gohr's Key Recovery Policy

The R-round neural distinguisher is used for a key recovery attack on $(R+1)$-round Speck32/64. In the attack, using the R-round neural distinguisher ND_R, the key candidate can be scored. A key guess is returned if its score exceeds a threshold λ. The key rank score is formulated as

$$v_k = \sum_{i=1}^{n} \log_2 \left(\frac{Z_i^k}{1 - Z_i^k} \right), \tag{2}$$

where k is the key candidate. Use k to decrypt all ciphertext pairs and Z_i^k is the i_{th} R-round ciphertext pair's output signal under the ND_R.

Since the time of scoring all k $(0 \leqslant k < 2^{16})$ is huge, Gohr searched for high-mark candidate keys by iterative method based on Bayesian Optimization [ARG04]. In order to generate new candidate keys in iterations, Gohr exploited the uneven distribution of the output signal corresponding to the wrong key. For $(R+1)$-round Speck32/64, let C_0 and $C_0^{'}$ be a pair of $(R+1)$-round ciphertexts whose input difference is $\Delta\alpha$, and k_R is the real subkey. For $\delta \in GF(2)^{16}$, there is $k^{'} = k_R \oplus \delta$. Use $k^{'}$ as a subkey to decrypt the ciphertext pair, and get the output signal of R-round neural distinguisher ND_R as

$$R_\delta \left(C_0, C_0^{'} \right) = ND_R \left(E_{k^{'}}^{-1} \left(C_0, C_0^{'} \right) \right). \tag{3}$$

Then $R_\delta \left(C_0, C_0^{'} \right)$ can be regarded as a random variable related to δ, which follows a normal distribution with mean μ_δ and standard deviation σ_δ. Using the difference in the distribution of different wrong key, the guessed key can be generated. The iteration ends, if the score of a candidate key exceeds threshold α.

In the attack scheme above, Gohr's attack is shown as Algorithm 1.

Algorithm 1 [Goh19] Gohr's Key Research for $(R+1)$-round Speck32/64

Input: Ciphertext pairs set $C = \{C_0, C_1, ..., C_{n-1}\}$, R-round neural distinguisher ND_R, number of candidates to be generated t, number of iterations l.

Output: Key set L.

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

In Algorithm 1, the $L||[(k, \sum_{i=0}^{n-1} w_{i,k})$ for $k \in S]$ means adding $[(k, \sum_{i=0}^{n-1} w_{i,k})$ for $k \in S]$ to L. And the $argsort_k(\lambda)$ means sorting k by the value of λ in descending order. In addition, Gohr found that the output signal from R-round neural distinguishers will be rather weak. Therefore, he used k neutral bits [BC04] to create from each plaintext pair a plaintext structure consisting of 2^k plaintext pairs that were expected to pass R-round neural distinguisher. Given N ciphertext structures, his algorithm is firstly tried on each

structure, and then the best ciphertext structure is selected, which can enhance output signal from R-round neural distinguisher to perform key recovery.

3 An Approach based on SAT to Improve Neural Distinguisher

In traditional differential cryptanalysis, it is a primary task to find a high-probability differential characteristic, which takes advantage of the unevenness of the differential distribution. The distribution of output differences is different for different input differences. For a neural distinguisher, it actually learns the distribution of output difference given a fixed input differences. Therefore, the input difference directly affects the accuracy of the neural distinguisher.

In [Goh19], Gohr chose $(0x40, 0x0)$ as the input difference to train the distinguisher because it transitioned deterministically to a low-weight output difference. But such input differences are hard to find, which makes it difficult to find effective distinguishers. In [BGPT21], Benamira *et al.* chose the input difference from $nr-1$-round or $nr-2$-round optimal differential characteristics for nr-round neural distinguishers.

In this section, we will introduce our algorithm for improving nr-round neural distinguishers by searching for the nr-round differential characteristics. With the help of SAT/SMT solver, we search for high-probability differential characteristics with probability in $\left[2^{-\frac{n}{4}} \times P_{\max}, P_{\max}\right]$, where $P_{\max}$ is the optimal probability and n is the block size. Using our algorithm, we can obtain high-accuracy neural distinguishers for 9-round SIMON32/64, 10-round SIMON48/96 and 11-round SIMON64/128.

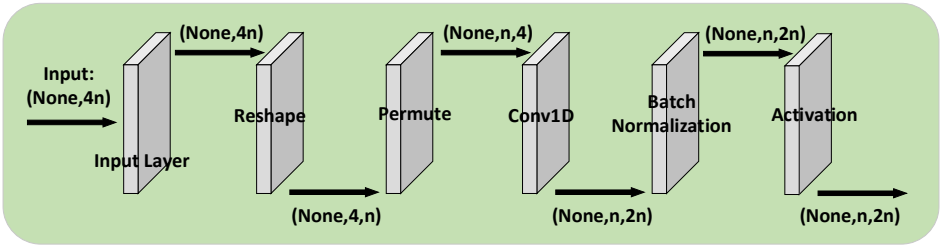
3.1 Generic Network Architecture

Gohr converted the distinguisher of ciphertext pairs into a binary classification problem. His method is not only applicable to Speck, but also applicable to SIMON. With his method, we can construct a generic network architecture for other ciphers. We refer to [Goh19] for the description of the method of constructing the network architecture.

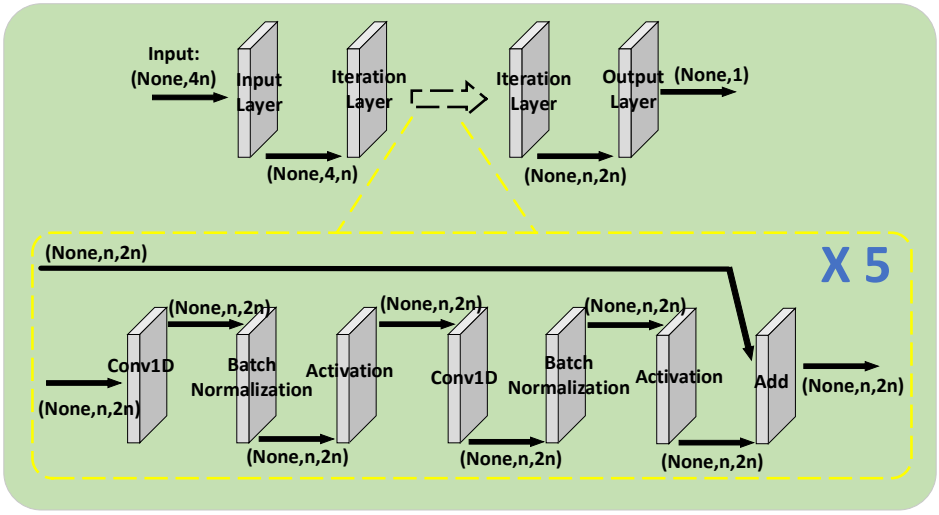
There are multiple neural networks available to train neural distinguishers, such as MIP, ResNet and so on. We choose the ResNet to train a neural distinguisher.

Our networks comprise three main components: input layer, iteration layer and predict layer, which is shown in Figure 1. The n in Figure 1 refers to the word size of SIMON$2n/mn$. The input layer receives training data with fixed length. In the iteration layer, we use 5 residual blocks. In each residual block, we use two Conv1D layers, and each Conv1D layers is followed by a batch normalization layer and a activation layer. After flattening data from iteration layer, data will be sent into a fully connected layer. The fully connected layer consists of a hidden layer and a output unit.

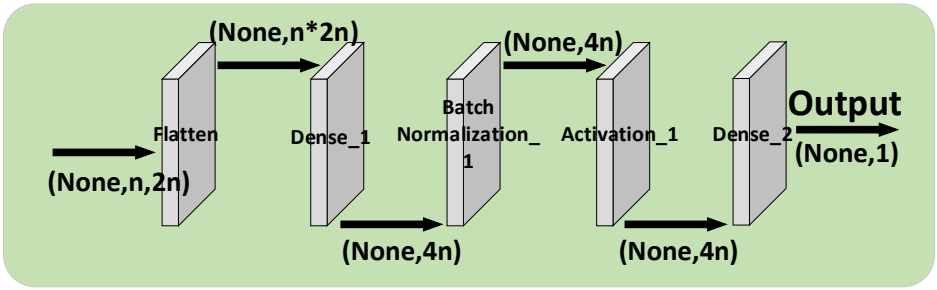
In our network, we choose that the kernel size of the first Conv1D layer is 1 and the kernel size of other Conv1D layer is 3. In addition, the number of filters in each convolutional layer is $2n$ and the padding method is SAME. At last, we train our network based on L2 weights regularization to avoid overfitting. The other details of the hyper-parameters used are given in Table 3.



(a) Input Layer



(b) Iteration Layer



(c) Output Layer

Figure 1: Network Architecture

Table 3: List of Hyper-parameters

Hyper-Parameters	Value
Batch Size	10000
Epochs	100
Train size	10^7
Validation size	10^5
Regularization parameter	10^{-4}
Optimizer	Adam
Loss function	MSE(mean-squared-error)

In Table 3, we choose the similar hyper-parameters as in Gohr's choice, so we can ignore the influence of the neural network and its parameters. After the neural distinguisher is trained, we can use it to distinguish the output of target cipher with a given input difference from random data. The higher its accuracy on the test set, the better it distinguishes ciphertext data.

3.2 An Algorithm based on SAT to Improve Neural Distinguisher

SAT is the Boolean Satisfiability Problem. It is an NP-complete problem and considers whether there is a valid assignment to Boolean variables satisfying a given set of Boolean conditions. As the key issue of computer science and artificial intelligence, SAT solvers have gained a lot of attention since it was proposed. It has great advantages with the open source, good interface, high efficiency and perfect compatibility. There are many cryptanalysis results based on SAT [MP13, KLT15, LWR16].

At present, there are two main ways to select the input differences of neural distinguishers. One way is to directly choose an existing optimal differential characteristic [CY21c], the other is to choose $nr-1$-round or $nr-2$-round optimal differential characteristics for nr-round neural distinguishers [BGPT21]. But these methods cannot effectively promote the distinguishing accuracy.

Taking into account the unevenness of the distribution of output differences for different input differences, we decide to choose the input differences of high-probability differential characteristics as the candidate differences. We search for high-probability differential characteristic by a SAT-based automatic search tool, and train neural distinguishers with these input differences of differential characteristics. Based on this, we design an algorithm to help us search for neural distinguishers with higher accuracy, which is shown in Algorithm 2. In Algorithm 2, we expand the search space of input difference by expanding the range of the probability. We choose $2^{-\frac{n}{4}} \times P_{\max}$ as the lower bound of the probability, where the P_{max} is the probability of the optimal differential characteristics. By experimental experience, we find that if the differential probability is lower than $2^{-\frac{n}{4}} \times P_{\max}$, there are almost no high-accuracy neural distinguishers. So there is nearly no need to spend time on the differential characteristics with the probability lower than $2^{-\frac{n}{4}} \times P_{\max}$.

Using the Theorem 3 in [KLT15] and open source SAT/SMT solver Z3 [dMB08], we search for high-probability differential characteristics of SIMON. Then, with Algorithm 2, we get the 9, 10 and 11 rounds neural distinguishers of SIMON32/64, SIMON48/96 and SIMON64/128, respectively. This is the first time that there is a neural distinguisher of 11-round SIMON64/128. The results of the neural distinguishers are shown in Table 4.

Algorithm 2 Search for neural distinguisher based on SAT

Input: Network Architecture Net, Cipher C with block size n bits, Round R.
Output: Neural distinguisher ND, Input difference of distinguisher I_d.

Search for the optimal probability as P_{max}
Search for the differential characteristics with probability in $\left[2^{-\frac{n}{4}} \times P_{\max}, P_{\max}\right]$, and save their input differences as $DIFF$
$ND = [\,]$
$I_d = [\,]$
for d in $DIFF$ **do**
    S = C(d,R) *#Generate train set using d as the input difference*
    V = C(d,R) *#Generate test set using d as the input difference*
    $D_d = Net$(S)*#Training using train data*
    acc_d = Evaluate(D_d,V)*#Get the accuracy of the model* D_d
    if $acc_d > 0.51$ **then**
        $ND = ND \,||\, D_d$
        $I_d = I_d \,||\, d$
    end if
end for
return (ND,I_d)

In order to show that Algorithm 2 is effective, we use the other two methods [CY21c, BGPT21] of selecting the input difference to train the neural distinguishers with the same rounds. For the method presented in [CY21c], we choose $(0x10100, 0x44040)$ in [ALLW14] as input difference to train 9-round neural distinguisher. Besides, for the other method presented in [BGPT21], we train 10-round neural distinguishers of SIMON48/96 using 9-round and 8-round optimal differential characteristics. And the specific results are shown in Appendix B.

Table 4: Summary of neural distinguishers

Block Cipher	Source of neural distinguisher (Input difference)	Round	Input difference	Accuracy
SIMON32/64	[ALLW14]	9	(0x0,0x40)	59.07%
	Algorithm 2	9	(0x0,0x80)	59.77%
SIMON48/96	[ALLW14]	9	(0x10100,0x44040)	50.22%
	[BGPT21]	10	(0x80000,0x222000)	53.49%
	Algorithm 2	10	(0x0,0x100000)	57.89%
SIMON64/128	[ALLW14]	10	(0x100,0x440)	58.61%
	Algorithm 2	11	(0x0,0x10)	59.72%

In Table 4, we show the comparison of the accuracy from three methods of selecting the input difference. As we can see, compared with selecting the input difference in [ALLW14] and [BGPT21], the accuracy of neural distinguishers obtained by Algorithm 2 has been significantly promoted, which can be used to reduce the complexity of key recovery attack. Although both methods select the input difference from differential characteristics, Algorithm 2 selects the exact rounds of the differential characteristics according to the rounds of neural distinguisher.

We also try to search for neural distinguishers for more rounds. Unfortunately, as the number of rounds increases, the non-random features of the ciphertext pairs become weaker and weaker. So it is difficult for us to find a neural distinguisher with longer round, even

if using Algorithm 2. In addition, the higher the Hamming weight of the input difference, the weaker the non-random feature of the ciphertext pair. So we should firstly search for input differences with lower Hamming weight adopting Algorithm 2 if time limit.

4 A New Neural Distinguisher Model Using Multiple Output Differences

In [BGPT21], Benamira *et al.* show that the neural distinguisher generally not only relies on the differential distribution of ciphertext pairs, but also on the differential distribution in penultimate and antepenultimate rounds. This enlightens us whether we can directly use the output differences to train neural distinguishers. Unlike [Goh19, CY21c] using ciphertext pairs as samples, we design a new neural distinguisher model with multiple output differences as a sample. Using the new model, we obtain the high-accuracy neural distinguishers for 10-round SIMON32/64, 11-round SIMON48/96, 12-round SIMON64/128, 8-round Speck32/64, 7-round Speck48/96 and 8-round Speck64/128. Additionally, we show with experiments that the promotion in the accuracy of distinguishers is not due to the increase of the number of plaintexts, but learning more features from the relationship between the output differences.

4.1 New Neural Distinguisher Model

Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. As we know, the deep learning is data-driven, and the quality of the data determines the quality of the model to some extent. For neural distinguishers, the choice of ciphertext pairs directly affects the accuracy of the neural distinguishers, which has been solved in Section 3. In deep learning field, the format of train data also affects the quality of the trained model to some extent. This enlightens us that we can improve neural distinguishers from the perspective of data format. In image recognition, the deep learning researchers currently rotate the image or crop it to enhance some objective features, which has been experimentally proven to be effective. Inspired by Benamira *et al.*'s work and data augmentation in deep learning, we use the output differences to train neural distinguishers, by splicing output differences into a matrix as a sample. For a matrix, we treat it as an image and each output difference of the matrix is treated as a objective feature. Our goal is not only to learn each objective feature, but to learn the connections between output differences.

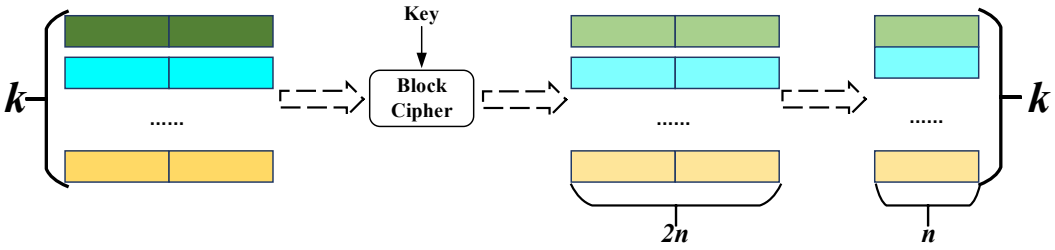


Figure 2: A New Data Format

As shown in Figure 2, the k plaintext pairs $\left(\left(P_1^1, P_2^1\right), \left(P_1^2, P_2^2\right), ..., \left(P_1^k, P_2^k\right)\right)$ are encrypted by a random master key. The k ciphertext pairs $\left(\left(C_1^1, C_2^1\right), \left(C_1^2, C_2^2\right), ..., \left(C_1^k, C_2^k\right)\right)$ are converted to output differences, where n is the block size of ciphers. We splice multiple output differences into a matrix as a sample, which is described as $O_1||O_2||...||O_k$. Similar

to Gohr's method, given an input difference I_d, each sample will be attached a label Y according to Equation 4.

$$Y(O_1||O_2||...||O_k) = \begin{cases} 1, if\ P_1^i \oplus P_2^i = I_d, i \in [1,k] \\ 0, else \end{cases} . \quad (4)$$

If the label is 1, the matrix is denoted as a positive sample. Otherwise it is denoted as a negative sample. By randomly generating plaintext and key, we make our distinguishers learn the features of target block cipher instead of the features of the plaintext or key. In the experiment, we make the neural network learn more features by using more output differences in a matrix. As we can see, the new data format needs more ciphertext pairs. For the same number of training sets, the new model requires k times more data than Gohr's model.

Because only the channel dimension is changed, we refer to Figure 1 for the description of network architecture.

4.2 Applications to NSA Block Ciphers

Application to SIMON:

We choose the input difference in Table 4 to train new neural distinguishers. Other hyper-parameters are posted in Table 3. The accuracy comparison is presented on Table 5. For 11-round SIMON48/96, we don't obtain an effective neural distinguisher using the input difference in Table 4. So we re-search other high-probability differential transmissions.

As shown in Table 5, compared with using ciphertext pairs, the round and accuracy of new neural distinguisher are greatly promoted. In addition, the new distinguishers can be further promoted by increasing k, which shows that the superposition of output difference can help the neural network to learn more unknown features.

Table 5: Comparison of SIMON using different data format. SCP: Single Ciphertext Pair. MOD: Multiple Output Differences.

Ciphers	Data Format	k	Round	Input difference	Accuracy
SIMON32/64	SCP	-	9	(0x0,0x80)	59.07%
	MOD	2	9	(0x0,0x80)	58.58%
	MOD	4	9	(0x0,0x80)	62.27%
	MOD	32	9	(0x0,0x80)	82.27%
	MOD	32	10	(0x0,0x80)	61.09%
SIMON48/96	SCP	-	10	(0x0,0x100000)	57.89%
	MOD	2	10	(0x0,0x100000)	57.31%
	MOD	4	10	(0x0,0x100000)	61.15%
	MOD	48	10	(0x0,0x100000)	81.40%
	MOD	48	11	(0x1000,0x4400)	61.43%
SIMON64/128	SCP	-	11	(0x0,0x10)	59.72%
	MOD	2	11	(0x0,0x10)	59.17%
	MOD	4	11	(0x0,0x10)	63.53%
	MOD	64	11	(0x0,0x10)	73.79%
	MOD	64	12	(0x0,0x10)	69.57%

Application to Speck:

The new format is not limited to the neural distinguisher of SIMON, but can also be found to be effective in Speck. In [Goh19, CY21c], the $(0x40, 0x0)$ is used to train

neural distinguisher of 7-round Speck32/64. Using the difference, we obtain a new higher-accuracy neural distinguisher of 7-round Speck32/64. Not only that, with the help of [MP13, LWR16], we obtain a good input difference $(0x2800, 0x10)$ and an effective 8-round neural distinguisher. As far as we know, this is the first effective 8-round neural distinguisher of Speck32/64 with accuracy more than 55%. Besides, we also obtain neural distinguishers of 7-round Speck48/96 and 8-round Speck64/128. Summary of the existing results is shown in Table 6.

Table 6: Comparison of Speck using different data format. SCP: Single Ciphertext Pair. MCP: Multiple Ciphertext Pairs. MOD: Multiple Output Differences.

Ciphers	Data Format	Round	Input difference	Accuracy	Source
Speck32/64	SCP	7	$(0x40, 0x0)$	61.6%	[Goh19]
	MCP	7	$(0x40, 0x0)$	70.74%[1]	[CY21c]
	MOD[2]	7	$(0x40, 0x0)$	88.87%	Section 4
	SCP	8	$(0x40, 0x0)$	51.4%	[Goh19]
	MOD[2]	8	$(0x2800, 0x10)$	56.49%	Section 4
Speck48/96	MCP	5	-[3]	-	[CY21b]
	MOD[4]	7	(0x20082,0x120200)	63.43%	Section 4
Speck64/128	MOD[5]	8	(0x1202,0x2000002)	63.20%	Section 4

[1] The highest accuracy of 7-round Speck32/64 in [CY21c].
[2] $k = 32$.
[3] Chen *et al.* used 5-round neural distinguisher to attack Speck48/x, but the accuracy were not presented in [CY21b].
[4] $k = 48$.
[5] $k = 64$.

Utilizing the new model, we improve neural distinguishers in terms of length and accuracy. We can achieve better results in distinguishing attack utilizing the new neural distinguishers. Moreover, we give a further illustration of our model. Since we use more data in the new model than using ciphertext pairs, this makes our improved results seem to be related to increase of data. We perform supplementary experiments to show that the improvement of the accuracy of distinguishers is not due to the increase of the number of plaintexts, but because of learning more features from the relationship between the output differences.

4.3 A Supplementary Explanation to Our New Model

Although the accuracy is higher using the new data format, the performance may be likely improved by training on more train samples. So we use same number of ciphertext pairs to train neural distinguishers shown in Table 7.

As shown in Table 7, the accuracy of new distinguisher is higher, even if they use the same amount of data. In addition, it takes up less memory using output differences, which can reduce training time in the training process.

Table 7: Comparison with using ciphertext pairs and output differences

Ciphers	Use the new data format?	k	Round	Size of train data	Accuracy
SIMON32/64	No	-	10	32×10^7	50.28%
	Yes	32	10	10^7	61.09%
SIMON48/96	No	-	11	48×10^7	50.43%
	Yes	48	11	10^7	61.43%
SIMON64/128	No	-	12	64×10^7	50.37%
	Yes	64	12	10^7	69.57%

To further illustrate the effectiveness of new distinguishers, we conduct additional experiments. As shown in Figure 3, we use k same output differences as a sample. Based on the new neural distinguishers, 10^6 positive and negative ciphertext pairs are randomly generated. And each output difference is reused k times and filled in a matrix as a sample. Then new neural distinguishers are performed on 10^6 samples. We calculate the accuracy of the new neural distinguishers for these special data. Table 8 shows the corresponding test results.

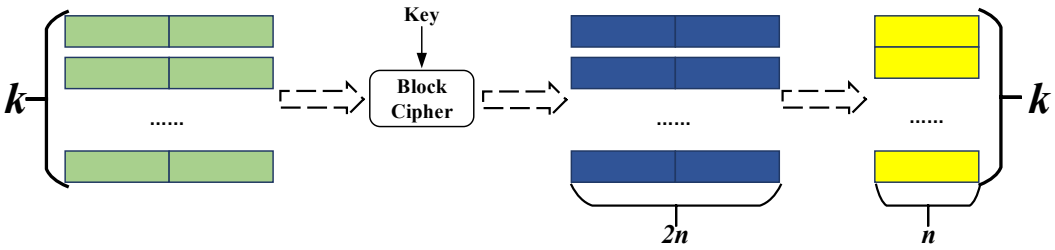


Figure 3: A new data format using the same output difference

Table 8: Accuracy of the new neural distinguishers for special data.

Ciphers	k	Round	Accuracy	Accuracy of Neural Distinguisher
SIMON32/64	32	10	51.69%	61.09%
SIMON48/96	48	11	54.81%	61.43%
SIMON64/128	64	12	49.78%	69.57%
Speck32/64	32	8	51.18%	56.49%
Speck48/96	48	7	50.06%	63.43%
Speck64/128	64	8	50.84%	63.20%

In Table 8, the 'Accuracy' refers to the accuracy of neural distinguishers for special samples like Figure 3. The 'Accuracy of Neural Distinguisher' refers to the accuracy of neural distinguishers for normal samples like Figure 2. As shown in Table 8, the accuracy for special samples is lower than for normal samples. This illustrates that the new distinguishers learn more unknown features especially in the connection of different output differences.

5 Key Recovery Attack on Round-reduced SIMON32/64 and SIMON48/96

Using Algorithm 2, we choose $(0x0, 0x100000)$ as the input difference to train 9-round and 10-round neural distinguisher of SIMON48/96. With the help of automatic search tool, we extend our 9-round and 10-round distinguisher to a 11-round and 12-round distinguisher by prepending the 2-round differential characteristic $(0x400000, 0x100001) \xrightarrow{2^{-4}} (0x0, 0x100000)$. Using 11-round and 12-round distinguisher, we complete practical 14-round key recovery attack of SIMON48/96 on a workstation configured with *Intel i9-10900K* and *Nvidia TITAN RTX*. It takes about 1550s to recover the final subkey each time. Our attack only needs no more than $2^{22.21}$ 14-round encryption and the data complexity does not exceed $2^{12.8}$. With the traditional differential attack in [ALLW14], it requires 2^{35} plaintext pairs at least with the optimal 12-round optimal probability 2^{35} [SWW21]. In addition, we perform key recovery attack of 13-round SIMON32/64 with a success rate more than 93%. Therefore the data complexity and time complexity based on deep learning is far lower than that of the traditional differential cryptanalysis.

5.1 Practical Key Recovery Attack on 14-round SIMON48/96

5.1.1 Overview

We choose $(0x0, 0x100000)$ as the input difference and train 9-round and 10-round neural distinguisher of SIMON48/96 with an accuracy over 57.5%. We extend our 9-round neural distinguisher to 11-round and 12-round distinguishers by prepending the 2-round differential characteristic $(0x400000, 0x100001) \xrightarrow{2^{-4}} (0x0, 0x100000)$. Combine 11-round and 12-round distinguishers together, we can construct 14-round key-recovery attack.

Attack Pattern 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 passes through the 11-round distinguisher is the highest. That is to say, the response of the 11-round distinguisher is the highest if 2-round subkeys 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 return a value below 0.5 if the guessed key is wrong. This can help us develop the key recovery attack of 14-round SIMON48/96. We guess possible keys in the last round, then we use the guessed subkey to perform 1-round decryption, and use 12-round distinguisher to score and sort the guessed subkeys. If the score of a key exceeds the threshold λ_1, 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 λ_2, the last guessed subkey will be returned as the result.

We use Gohr's attack scheme to perform a practical key recovery attack on 14-round SIMON48/96. Our attack parameters are shown in Table 9.

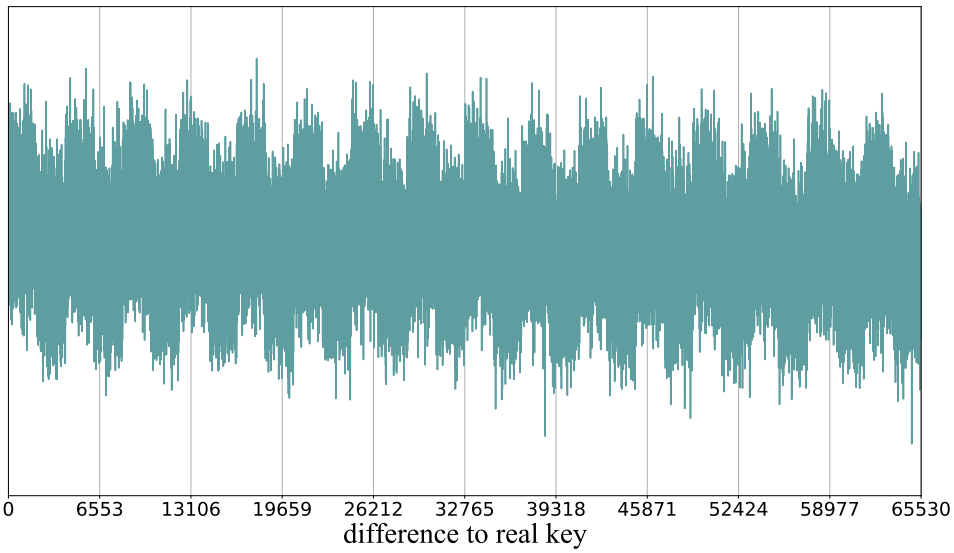
Table 9: Attack parameters for 14-round SIMON48/96

Parameter	Value
Initial difference	(0x400000,0x110001)
λ_1	10
λ_2	50
Neutral bit [BC04]	[44,47,21,39,3,28]
Number of iterations	100
Experimental configuration	*Intel i9-10900K*, *Nvidia TITAN RTX*

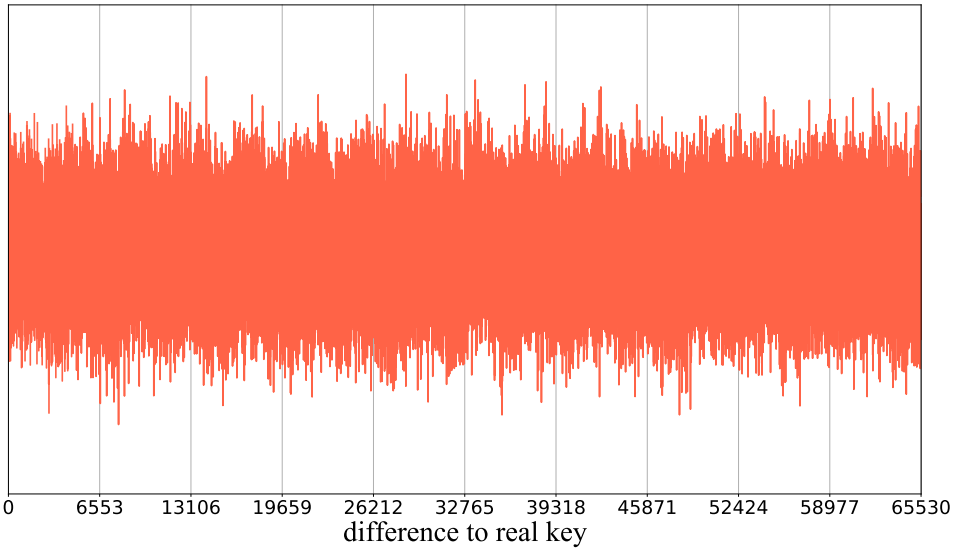
5.1.2 Wrong Key Randomization

Let C_0 and $C_0^{'}$ be a pair of ciphertexts whose input difference is $(0x0, 0x100000)$, and k is the real subkey of the last round. For $\delta \in GF(2)^{24}$, there is $k^{'} = k \oplus \delta$. Use $k^{'}$ as a subkey to decrypt the ciphertext pair, and get the response as $R_\delta\left(C_0, C_0^{'}\right) = N_{10}\left(E_{k^{'}}^{-1}\left(C_0, C_0^{'}\right)\right)$, where N_{10} is the 10-round neural distinguisher. 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 calculate the wrong key response profile for our 10-round distinguishers of SIMON48/96. For $\delta \in GF(2)^{24}$ and a random master key K, we generate 1 ciphertext pair $\left(C_0, C_0^{'}\right)$ whose input difference is $(0x0, 0x100000)$ and save its last real subkey 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 10-round distinguisher. We repeat the above steps 2^{24} times and calculate the mean value and standard deviation of the response values. The part of wrong key response profile is shown in Figure 4.



(a) Mean response for 10-round distinguisher



(b) Standard deviation for 10-round distinguisher

Figure 4: Wrong key response profile for 10-round distinguisher

As described in Section 2, the output signal of the $R-$round neural distinguisher ND_R follows a normal distribution with mean μ_δ and standard deviation σ_δ, where δ is the difference between the correct key and the wrong key. In Figure 4, the x-axis refers to the difference δ. In Figure 4(a), the y-axis refers to the value of μ_δ. And the y-axis refers to the value of σ_δ in Figure 4(b). It can be seen from Figure 4 that the mean and standard deviation are larger when there are fewer error bits. This will help recover the last subkey. Similar to 10-round neural distinguisher, we can get wrong key response profile for 9-round neural distinguisher.

5.1.3 Result and Complexity Analysis

For calculating the time complexity, we calculate the time for 2^{10} 14-round encryptions, which costs about 0.32s for our hardware. And our attack only needs no more than 1550s each time. Therefore, the time complexity can be calculated by $1550 \div \frac{0.32}{2^{10}} \approx 2^{22.21}$. At the same time, the data complexity does not exceed $2^{12.8}$. With complexity analysis, we can see that this is a practical attack. Our result in 100 trials is shown in Table 10. It is considered a success if the guess for the last subkey is only incorrect in 5 bits. And then we can eliminate error bits by exhaustive methods.

Table 10: Results of 14-round SIMON48/96

The number of error bits	$\leqslant 4$	$\leqslant 5$	$\leqslant 6$	$\leqslant 7$	$\leqslant 8$
Number of experiments	50	58	62	68	72

To show that the neural distinguisher has advantages in key recovery attacks, we use the traditional cryptanalysis to perform the recovery attacks for last key on 14-round SIMON48/96. Since key-addition occurs after non-linear operation in the round function of SIMON48/96, we can construct 14-round attack by adding one round before and after 12-round optimal differential characteristics with probability 2^{-35} [SWW21]. Utilizing the traditional method, the time complexity is more than 2^{35} 14-round encryption.

5.2 Practical Key Recovery Attack on 13-round SIMON32/64

Similar to the key recovery attack of 14-round SIMON48/96, we choose $(0x0, 0x80)$ as the input difference to train 8-round and 9-round neural distinguishers. With the automatic search based on SAT, we extend our 8-round and 9-round distinguishers to a 10-round and 11-round distinguishers by prepending the 2-round differential characteristic $(0x200, 0x880) \xrightarrow{2^{-4}} (0x0, 0x80)$. Using 10-round and 11-round distinguishers, we complete practical 13-round key recovery attack of SIMON32/64. Our attack parameters are shown in Table 11.

Table 11: Attack parameters for 13-round SIMON32/64

Parameter	Value
Initial difference	$(0x200, 0x880)$
λ_1	10
λ_2	10
Neutral bit	[21, 30, 26, 5, 14, 18]
Number of iterations	100
Experimental configuration	*Intel i9-10900K*, *Nvidia TITAN RTX*

The key recovery attack based on our neural distinguishers is performed 100 times. In one hundred trials, the last subkey is correctly guessed in 60 cases and there are 24 cases that the guess for the last subkey is incorrect in just one bit. There are only 9 cases that have 2 bits differences from the correct last subkey. It takes about 23s each time to recover the final subkey, with a success rate more than 90%. And we calculate the time for 2^{10} 13-round encryptions, which costs about 0.27s for our hardware. Our attack only needs no more than $2^{16.4}$ 13-round encryption and the data complexity does not exceed $2^{12.5}$.

6 Practical Key Recovery Attack on Reduced-round SIMON64/128

Gohr's framework is effective in key recovery of short-size block ciphers. Unfortunately, it costs a lot of time in precomputation for large-size block ciphers, especially in wrong key response profile. For example, for SIMON32/64, it only spends about 1500s to obtain complete profile on a fast graphics card, while it spends more than 1300 days for SIMON64/128, which is unbearable for most of researchers.

We find with further experiments of SIMON32/64 and SIMON48/96 that the distribution of the wrong key response is pseudo-periodic. In other words, we can re-use known partial wrong key profile to fill the remaining wrong key profile. Based on this, we design a generic scheme to attack large-size block ciphers. Firstly, we make use of partial wrong key profile to describe the whole wrong subkey response profile. Once we get the complete profile portrayed partial profile, we can recover partial subkey bits with Gohr's key recovery policy. Then we guess the complete subkey based on the known subkey bits.

6.1 Observations on Wrong Key Response

In [CY21c], Chen *et al.* proposed two properties for most symmetric ciphers and neural networks as follows.

Property 1. [CY21c] Let a ciphertext C be decrypted one round with two different subkey, $C_1 = Dec_{one}(C, sk_1)$, $C_2 = Dec_{one}(C, sk_2)$. If sk_1 and sk_2 are different at a few bits, the Hamming distance between C_1 and C_2 will be very small.

For SIMON, let $C = (lc, rc)$. The $C_i = (lc_i, rc_i), i = 1, 2$ are generated by

$$\begin{aligned} lc_i &= rc \\ rc_i &= \left(S^1 rc \& S^8 rc\right) \oplus S^2 rc \oplus lc \oplus sk_i. \end{aligned} \tag{5}$$

In other words, the different bits of C_1 and C_2 depend on the different bits of sk_1 and sk_2. So $Hw(C_1, C_2) = Hw(sk_1, sk_2)$, where Hw denotes the Hamming distance.

Property 2. [CY21c] Given a trained neural network $N(\cdot)$ for solving a binary classification problem, if two input samples X_1, X_2 are very close to each other in the input space, two outputs $N(X_1)$ and $N(X_2)$ obtained from the neural network may satisfy $N(X_1) \approx N(X_2)$ with a high probability.

According to Property 1 and Property 2, when two guessed subkeys are different in a few bits, the corresponding output signals should be similar.

Property 3. For a block cipher with subkey size $n$bits and $I = [0, 2^n)$, then I can be equally divided into 2^{n-k} parts, denoted by $I_i = \left[i \times 2^k, (i+1) \times 2^k\right) (0 \leqslant i < 2^{n-k})$. For $x_i \in I_i$, $x_{i+1} \in I_{i+1}$, two outputs $ND(Dec_{one}(C_p, x_i \oplus sk))$ and $ND(Dec_{one}(C_p, x_{i+1} \oplus sk))$ obtained from the neural distinguisher ND may satisfy $ND(Dec_{one}(C_p, x_i \oplus sk)) \approx ND(Dec_{one}(C_p, x_{i+1} \oplus sk))$, where $0 \leqslant i < 2^{n-k} - 1$, $x_{i+1} - x_i = 2^k$, sk is the correct subkey, C_p is a pair of ciphertexts given a fixed input difference. In addition, $|ND(Dec_{one}(C_p, x_i \oplus sk)) - ND(Dec_{one}(C_p, x_{i+1} \oplus sk))|$ gets closer to 0, as k increases.

Property 4. For a block cipher with subkey size $n$bits and $I = [0, 2^n)$, then I can be equally divided into 2^{n-k} parts, denoted by $I_i = \left[i \times 2^k, (i+1) \times 2^k\right) (0 \leqslant i < 2^{n-k})$. For $x \in I$, the output $ND(Dec_{one}(C_p, x \oplus sk))$ obtained from the neural distinguisher follows a normal distribution with mean μ_x and standard deviation σ_x, where sk is the correct subkey, C_p is a pair of ciphertexts given a fixed input difference. In addition, for $x_i \in I_i$, $x_{i+1} \in I_{i+1}$, there may satisfy $\mu_{x_i} \approx \mu_{x_{i+1}}$ and $\sigma_{x_i} \approx \sigma_{x_{i+1}}$, where $0 \leqslant i < 2^{n-k} - 1$, $x_{i+1} - x_i = 2^k$. In addition, $|\mu_{x_i} - \mu_{x_{i+1}}|$ and $|\sigma_{x_i} - \sigma_{x_{i+1}}|$ get closer to 0, as k increases.

Due to Property 4, we can find that the distribution of the wrong key response is pseudo-periodic. So we can use partial wrong key response instead of the complete wrong key response profile to recover subkey.

6.2 Key Recovery Attack on 13-round SIMON64/128

6.2.1 Key Recovery Policy

As described in Section 2, the output signal of R-round neural distinguisher ND_R follows a normal distribution with mean μ_δ and standard deviation σ_δ, where the δ is the difference between the correct key and the wrong key. Affected by Bayesian optimization, we need all μ_δ and σ_δ, where $\delta \in [0, 2^{32})$. But limited by our hardware, we only obtain partial wrong key response profile, where $\delta \in [0, 2^{24})$. In other word, we only have partial μ_δ and σ_δ ,where $\delta \in [0, 2^{24})$. Thanks to Property 4, the μ_i will be replaced by $\mu_{i \bmod 2^{24}}$, where $i \in [2^{24}, 2^{32})$. For δ_i, it will be replaced by $\delta_{i \bmod 2^{24}}$. By this way, we can obtain a pseudo wrong key response profile, which is used in the key recovery attack of SIMON64/128. This will reduce the time in precomputation from more than 1300 days to about 5.3 days. Based on this, we improve Gohr's key search policy to Algorithm 3.

Algorithm 3 A New Key Research for $(R+1)$-round SIMON64/128

Input: Ciphertext pairs set $C = \{C_0, C_1, ..., C_{n-1}\}$, R-round neural distinguisher ND_R, 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: $sk_g \leftarrow 0$
3: $Score \leftarrow 0$
4: **for** $j \in \{0, 1, ..., l-1\}$ **do**
5: **for** $i \in \{0, 1, ..., t-1\}$ **do**
6: Calculate the score of k_i using Equation 2, denoted by $score_{k_i}$
7: **if** $score_{k_i} > Score$ **then**
8: $Score \leftarrow score_{k_i}$
9: $sk_g \leftarrow k_i$
10: **end if**
11: **end for**
12: $\phi_k \leftarrow \sum_{i=0}^{t-1} \left(\frac{\frac{score_{k_i}}{n} - \mu_{(k_i \oplus k) \bmod 2^{24}}}{\sigma_{(k_i \oplus k) \bmod 2^{24}}} \right)^2$ for $k \in \{0, 1, ..., 2^{32}-1\}$
13: $S \leftarrow argsort_k(\phi)[0 : t-1]$ *#Get the new key candidates corresponding to the largest t elements in ϕ_k*
14: **end for**
15: return sk_g

In Algorithm 3, the key candidate will be returned if the iteration ends. We choose the key with the highest score as the guessed subkey, denoted by sk_g. Due to the difference between pseudo and real wrong key response profile, Algorithm 3 only recovers partial subkey of SIMON64/128.

As shown in Figure 5, the whole wrong key response profile is described by partial profile. The search for the correct subkey is the search for δ to be 0, where the δ is the difference between subkey candidate and real subkey. In the step 12,13 of Algorithm 3, we choose all possible keys and score them by partial μ_δ and σ_δ, where $\delta \in [0, 2^{24})$. And we use little ciphertext pairs to recover subkey. Therefore, in the experiment of Algorithm 3, the difference δ between subkey candidate and real subkey is easy to get the best value in I_i. In other word, Algorithm 3 will make $hw(\delta \odot 0x00ffffff)$ close to 0, where hw

denotes the Hamming weight. In fact, we just make $hw(\delta^{23\sim0})$ close to 0 by Algorithm 3 rather than that $hw(\delta)$ close to 0, where δ is the difference between sk_g and the real subkey. In other words, for the returned subkey sk_g, there will be a few error bits in $sk_g^{31\sim24}$. So we design Algorithm 4 to recover $sk_g^{31\sim24}$.

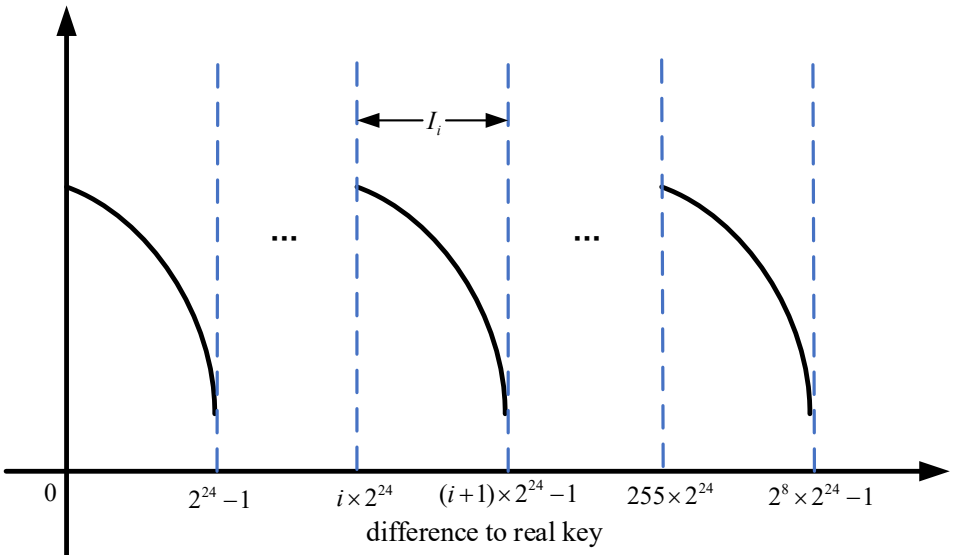


Figure 5: Schematic of mean response using pseudo-period

Algorithm 4 Recover complete subkey for SIMON64/128

Input: Ciphertext pairs $C = \{C_0, C_1, ..., C_{n-1}\}$, ND_R: R-round neural differential distinguisher, sk_g: Returned guess subkey with Algorithm 3.
Output: The complete subkey sk.

$score \leftarrow 0$
$guess_{sk} \leftarrow 0$
for $j \in \{0, 1, ..., 2^8 - 1\}$ **do**
  $guess_{sk} \leftarrow (j \ll 24) \oplus sk_g$
  $P_i \leftarrow Decrypt_{one}(C_i, guess_{sk})$ for all $i \in \{0, 1, ..., n-1\}$
  $v_i \leftarrow ND_R(P_i)$ for all $i \in \{0, 1, ..., n-1\}$
  $w_i \leftarrow \log_2\left(\frac{v_i}{1-v_i}\right)$ for all $i \in \{0, 1, ..., n-1\}$
  $m_j \leftarrow \sum_{i=0}^{n-1} w_i/n$
  if $m_j > score$ **then**
    $score \leftarrow m_j$
    $sk \leftarrow guess_{sk}$
  end if
end for
return sk

Although we choose SIMON64/128 as an example in Algorithm 3 and Algorithm 4, they can also be used in other block ciphers whose subkey size exceeds 64 bits. Besides, we can only obtain 2^{24} wrong key response profile limited by the hardware. The more wrong key response profile will be obtained utilizing better computing resources.

6.2.2 Application to Attack 13-round SIMON64/128

Different from the attacks of SIMON32/64 and SIMON48/96, we only use the 11-round neural distinguisher to perform 13-round attack without appending additional rounds. This attack consists of two stages:

Stage 1(Algorithm 3):

In this stage, we complete the screening of the all possible keys using Algorithm 3, 11-round and 10-round neural distinguishers. And the parameters used in **Stage 1** are shown in Table 12.

Table 12: Attack parameters for 13-round SIMON64/128

Parameter	Value
Input difference	$(0x0, 0x10)$
λ_1	10
λ_2	10
Neutral bit	[27,19,3,23,18,22]
Number of iterations	500
Experimental configuration	*Intel Xeon 6226R@2.90Ghz, Nvidia GeForce RTX3090*

After performing 100 times attack using Algorithm 3, 11-round and 10-round neural distinguishers, we find:

1. In one hundred trials, sk_g is output after processing on average $2^{12.703}$ ciphertext pairs.

2. Average runtime is about 21000s. This yields an estimated computational attack complexity of $2^{25.700}$ 13-round SIMON64/128 encryptions.

3. The specific results are shown in Table 13.

Table 13: Results of 13-round SIMON64/128 in low 24 bits

The number of error bits	$\leqslant 4$	$\leqslant 5$	$\leqslant 6$	$\leqslant 7$	$\leqslant 8$
Number of experiments	49	63	78	83	86

For low 24 bits, it is considered a successful attack if the guessed last subkey is incorrect in 5 bits, and the success rate exceeds 63%.

Stage 2(Algorithm 4):

In this stage, we obtain the best key with highest score from 2^8 key candidates using Algorithm 4 and 3000 ciphertext pairs. In this stage, the calculation method of success rate is designed as:

1. Randomly generate 3000 plaintext pairs with the difference $(0x0, 0x10)$ and encrypt them for 12 rounds with the right key;

2. Randomly generate a masking value $m(0 \leqslant m < 2^{24})$ and $hw(m) = t$, where hw denotes the Hamming weight;

3. Perform Algorithm 4 with 3000 ciphertext pairs and $sk_g = rk \oplus m$, where rk is the real last subkey;

4. Save $((sk \oplus rk) \gg 24)$, where the sk is the output of Algorithm 4;

5. Perform steps $1-4$ 300 times and calculate the ratio of $hw((sk \oplus rk) \gg 24) \leqslant 2$.

For $t\,(0 \leqslant t \leqslant 5)$, the ratio is shown in Table 14.

Table 14: Results in high 8 bits

t	0	1	2	3	4	5
Ratio	93.00%	93.33%	90.67%	92%	92.67%	91.33%

Computation Complexity: In Stage 1, about $2^{12.703}$ plaintext pairs are needed and an attack needs about $2^{25.700}$ 13-round SIMON64/128 encryption. In Stage 2, an attack needs 3000 plaintext pairs. At the same time, compared with Stage 1, the computation complexity is negligible in Stage 2. Thus the total complexity is $2^{25.700}$ and the data complexity is $2^{13.24}$. In addition, the success rate is $90.67\% \times 63\% \approx 57.12\%$, if it is considered a success in that the guess for the last subkey is only incorrect in 7 bits.

In this section, we design an algorithm to attack large-size block cipher based on neural distinguishers. Limited by the computing resources, we can only obtain 2^{24} wrong key response profile to describe a complete wrong key response profile. Compared to complete profile, the partial profile has some flaws in key recovery attack. Therefore the success rate in **Stage 1** will rise if we can obtain more wrong key response. At the same time, there will be less error bits in **Stage 1** if the success rate in **Stage 1** rises, which will increase the success rate of **Stage 2**. In short, the success rate will increase and the error bits will decrease, if we can obtain more wrong key responses.

Moreover, for large-size SIMON, the key recovery attacks are mainly based on Property 3 and Property 4. And Property 3 and Property 4 can be used in other block ciphers, not limited to SIMON. In other words, our techniques also work for other complex ciphers apart from SIMON.

7 Conclusion and Future Work

In this paper, we proposed a new algorithm and model to further improve neural distinguishers, and then performed practical key recovery attack on SIMON. On the one hand, by carefully selecting the input differences utilizing SAT/SMT algorithm, we managed to searched for exact nr-round differential characteristics with high probability and trained nr-round neural distinguishers. On the other hand, by adopting the new data format, we spliced multiple output differences into a matrix as a sample to capture more derived features, thus can improve the round and accuracy of neural distinguishers. Application to SIMON and Speck have proved the superiorities of our new models. Moreover, based on the improved distinguishers, we performed key recovery attack on SIMON32/64 and SIMON48/96 adopting Gohr's key recovery policy. We also proposed a new generic key recovery policy using partial wrong key profile to significantly reduce the precomputation time, that could complete practical attack on SIMON64/128.

With our results we obtain new effective neural distinguishers, which can be used to distinguish reduced-round NSA block ciphers from pseudo-random permutation better. Besides, we believe the new key recovery scheme of neural aided cryptanalysis has great potential for better assessing the security of large-size block ciphers. Since there are numerous network architectures now with the development of deep learning, it is meaningful to explore other appropriate network models to improve neural distinguishers and key recovery.

References

[ALLW14] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential cryptanalysis of round-reduced simon and speck. In Carlos Cid and Christian

Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 525–545. Springer, 2014. https://doi.org/10.1007/978-3-662-46706-0_27.

[ARG04] Chang Wook Ahn, Rudrapatna S. Ramakrishna, and David E. Goldberg. Real-coded bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell, editors, *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004, Proceedings, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 840–851. Springer, 2004. https://doi.org/10.1007/978-3-540-24854-5_86.

[BC04] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[BGPT21] Adrien Benamira, David Gérault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2021:287, 2021. https://eprint.iacr.org/2021/287.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991. https://doi.org/10.1007/BF00630563.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, 2013:404, 2013. http://eprint.iacr.org/2013/404.

[CY20] Yi Chen and Hongbo Yu. Neural aided statistical attack for cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2020:1620, 2020. https://eprint.iacr.org/2020/1620.

[CY21a] Yi Chen and Hongbo Yu. Bridging machine learning and cryptanalysis via EDLCT. *IACR Cryptol. ePrint Arch.*, 2021:705, 2021. https://eprint.iacr.org/2021/705.

[CY21b] Yi Chen and Hongbo Yu. Improved neural aided statistical attack for cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2021:311, 2021. https://eprint.iacr.org/2021/311.

[CY21c] Yi Chen and Hongbo Yu. A new neural distinguisher model considering derived features from multiple ciphertext pairs. *IACR Cryptol. ePrint Arch.*, 2021:310, 2021. https://eprint.iacr.org/2021/310.

[dMB08] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008.*

Proceedings, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008. https://doi.org/10.1007/978-3-540-78800-3_24.

[Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019. https://doi.org/10.1007/978-3-030-26951-7_6.

[GPM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. http://arxiv.org/abs/1406.2661.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006. https://doi.org/10.1162/neco.2006.18.7.1527.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. https://doi.org/10.1109/CVPR.2016.90.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015. https://doi.org/10.1007/978-3-662-47989-6_8.

[LGTB97] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997. 10.1109/72.554195.

[LWR16] Yunwen Liu, Qingju Wang, and Vincent Rijmen. Automatic search of linear trails in ARX with applications to SPECK and chaskey. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve A. Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, volume 9696 of *Lecture Notes in Computer Science*, pages 485–499. Springer, 2016.

[MP43] Warren S. McCulloch and Walter H. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics.*, 5:115–133, 1943. https://doi.org/10.1007/BF02478259.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. *IACR Cryptol. ePrint Arch.*, 2013:328, 2013. https://eprint.iacr.org/2013/328.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back propagating errors. *Nature*, 323:533–536, 1986. https://doi.org/10.1038/323533a0.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 10.1037/h0042519.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, volume 9723 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2016. https://doi.org/10.1007/978-3-319-40367-0_24.

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021. https://doi.org/10.46586/tosc.v2021.i1.269-315.

[SZM20] Heng-Chuan Su, Xuan-Yong Zhu, and Duan Ming. Polytopic attack on round-reduced simon32/64 using deep learning. In Yongdong Wu and Moti Yung, editors, *Information Security and Cryptology - 16th International Conference, Inscrypt 2020, Guangzhou, China, December 11-14, 2020, Revised Selected Papers*, volume 12612 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2020. https://doi.org/10.1007/978-3-030-71852-7_1.

[WZ89] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989. https://doi.org/10.1162/neco.1989.1.2.270.

Appendix

A Brief Description of SIMON and Speck

A.1 SIMON

SIMON [BSS+13] 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. Table 15 makes explicit all parameter choices for all versions of SIMON.

Table 15: SIMON parameters

block size $2n$	key size mn	word size n	rounds T
32	64	16	32
48	72	24	36
	96	24	36
64	96	32	42
	128	32	44
96	96	48	52
	144	48	54
128	128	64	68
	192	64	69
	256	64	72

For SIMON2n/mn, the key-dependent SIMON2n/mn round function is the map R_{k_i}:$GF\left(2\right)^n \times GF\left(2\right)^n \rightarrow GF\left(2\right)^n \times GF\left(2\right)^n$ defined by

$$R_{k_i}\left(x_i, y_i\right) = \left(y_i \oplus f\left(x_i\right) \oplus k_i, x_i\right), \tag{6}$$

where $f\left(x_i\right) = \left(S^1 x_i \odot S^8 x_i\right) \oplus S^2 x_i$, $k_i (k_i \in GF\left(2\right)^n)$ is the round subkey.

A.2 Speck

Similar to SIMON, there are some different variants of Speck. And these parameter about Speck are shown in Table 16.

Table 16: Speck parameters

block size $2n$	key size mn	word size n	rot α	rot β	rounds T
32	64	16	7	2	22
48	72	24	8	3	22
	96	24	8	3	23
64	96	32	8	3	26
	128	32	8	3	27
96	96	48	8	3	28
	144	48	8	3	29
128	128	64	8	3	32
	192	64	8	3	33
	256	64	8	3	34

For Speck2n/mn, the key-dependent Speck2n/mn round function is the map R_{k_i}:$GF\left(2\right)^n \times GF\left(2\right)^n \rightarrow GF\left(2\right)^n \times GF\left(2\right)^n$ defined by

$$R_{k_i}\left(x_i, y_i\right) = \left(\left(S^{-\alpha} x_i + y_i\right) \oplus k_i, \left(S^{-\alpha} x_i + y_i\right) \oplus k_i \oplus S^{\beta} y_i\right), \tag{7}$$

where $k_i (k_i \in GF\left(2\right)^n)$ is the round subkey.

As it is out of scope for our purpose, we refer to [BSS+13] for the description of the key-scheduling.

B Accuracy of neural distinguishers of 10-round SIMON48/96 using Benamira's method

Table 17: 10-round neural distinguishers using 9-round optimal characteristics

Input difference	Accuracy	Input difference	Accuracy	Input difference	Accuracy
(0x2,0x880008)	0.5258	(0x100,0x444)	0.5279	(0x800,0x2220)	0.5281
(0x8000,0x22200)	0.5286	(0x4000,0x11100)	0.5286	(0x80000,0x222000)	0.5288
(0x100000,0x444000)	0.5290	(0x20000,0x88800)	0.5293	(0x20000,0x88800)	0.5293
(0x20,0x800088)	0.5301	(0x200000,0x888000)	0.5301	(0x200,0x888)	0.5303
(0x2000,0x8880)	0.5303	(0x4,0x100011)	0.5308	(0x80,0x222)	0.5310
(0x400,0x1110)	0.5311	(0x40,0x111)	0.5312	(0x10,0x400044)	0.5312
(0x400000,0x110001)	0.5312	(0x8,0x200022)	0.5315	(0x1,0x440004)	0.5327
(0x800000,0x220002)	0.5327	(0x1000,0x4440)	0.5330	(0x40000,0x111000)	0.5334

Table 18: 10-round neural distinguishers using 8-round optimal characteristics

Input difference	Accuracy	Input difference	Accuracy	Input difference	Accuracy
(0x400004,0x10)	0.5264	(0x100,0x444)	0.5282	(0x8,0x200022)	0.5283
(0x88,0x200)	0.5285	(0x2000,0x8880)	0.5286	(0x400,0x1110)	0.5287
(0x40,0x111)	0.5289	(0x440,0x1000)	0.5290	(0x40000,0x111000)	0.5291
(0x100001,0x4)	0.5294	(0x400000,0x110001)	0.5296	(0x20000,0x88800)	0.5298
(0x800000,0x220002)	0.5298	(0x880,0x2000)	0.5298	(0x11000,0x40000)	0.5300
(0x88000,0x200000)	0.5301	(0x110000,0x400000)	0.5301	(0x800,0x2220)	0.5302
(0x800,0x2220)	0.5302	(0x220,0x800)	0.5304	(0x8800,0x20000)	0.5304
(0x200,0x888)	0.5305	(0x2200,0x8000)	0.5306	(0x2,0x880008)	0.5306
(0x100000,0x444000)	0.5307	(0x880000,0x2)	0.5307	(0x10000,0x44400)	0.5308
(0x1100,0x4000)	0.5309	(0x1,0x440004)	0.5311	(0x110,0x400)	0.5313
(0x200002,0x8)	0.5313	(0x44000,0x100000)	0.5314	(0x220000,0x800000)	0.5314
(0x22,0x80)	0.5318	(0x8000,0x22200)	0.5319	(0x200000,0x888000)	0.5320
(0x80,0x222)	0.5322	(0x1000,0x4440)	0.5323	(0x10,0x400044)	0.5325
(0x4400,0x10000)	0.5326	(0x4,0x100011)	0.5327	(0x44,0x100)	0.5327
(0x440000,0x1)	0.5329	(0x20,0x800088)	0.5339	(0x4000,0x11100)	0.5340
(0x22000,0x80000)	0.5341	(0x11,0x40)	0.5341	(0x80000,0x222000)	0.5349