Improved Neural Distinguishers with (Related-key) Differentials: Applications in SIMON and SIMECK

Jinyu Lu[1,2,3], Guoqiang Liu[1,2,4⋆], Yunwen Liu[1,2,3], Bing Sun[1,2,3], Chao Li[1,2,3], and Li Liu[4,5]

[1] College of Liberal Arts and Sciences, National University of Defense Technology, Hunan, Changsha 410073, China
[2] Hunan Engineering Research Center of Commercial Cryptography Theory and Technology Innovation, Hunan, Changsha 410073, China
[3] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[4] State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)
[5] College of Systems Engineering, National University of Defense Technology, Hunan, Changsha 410073, China
[6] Center for Machine Vision and Signal Analysis, University of Oulu, Oulu 90570, Finland
liuguoqiang87@hotmail.com

Abstract. In CRYPTO 2019, Gohr made a pioneering attempt, and successfully applied deep learning to the differential cryptanalysis against NSA block cipher Speck32/64, achieving higher accuracy than the pure differential distinguishers. By its very nature, mining effective features in data plays a crucial role in data-driven deep learning. In this paper, in addition to considering the integrity of the information from the training data of the ciphertext pair, domain knowledge about the structure of differential cryptanalysis is also considered into the training process of deep learning to improve the performance. Besides, based on the SAT/SMT solvers, we find other high probability compatible differential characteristics which effectively improve the performance compared with previous work. We build neural distinguishers (NDs) and related-key neural distinguishers (RKNDs) against Simon and Simeck. The ND and RKND for Simon32/64 reach 11-, 11-round with an accuracy of 59.55% and 97.90%, respectively. For Simon64/128, the ND achieve an accuracy of 60.32% in 13-round, while it is 95.49% for the RKND. For Simeck32/64, ND and RKND of 11-, 14-round are obtained, reaching an accuracy of 63.32% and 87.06%, respectively. And we build 17-round ND and 21-round RKND for Simeck64/128 with an accuracy of 64.24% and 62.96%, respectively. Currently, these are the longest (related-key) neural distinguishers with higher accuracy for Simon32/64, Simon64/128, Simeck32/64 and Simeck64/128.

⋆ Corresponding author

Keywords: Deep Learning, (Related-key) Differential Distinguisher, SIMON, SIMECK, Domain Knowledge

1 Introduction

The security analysis of many cryptographic primitives (such as pseudo-random number generators, hash functions, etc.) is usually attributed to attacks on the underlying block ciphers. Various cryptanalytic methods have been proposed over the past few decades, including differential cryptanalysis [6], linear cryptanalysis [24], integral cryptanalysis [18], zero-correlation linear cryptanalysis [8], etc. A block cipher must be able to resist all known cryptanalysis to obtain a strong security statement. In recent years, solver-based automatic tools and dedicated heuristic search algorithms have been extensively adopted to improve the accuracy and efficiency in cryptanalysis of block ciphers, where the cryptanalytic models are often transformed into Mixed Integer Linear Programming (MILP) problems [27, 29], SAT/SMT problems [20, 26] or CP problems [11, 25]. Automatic search technology has improved the analysis ability of block ciphers. The improvement and development of these automatic search technologies provide an inexhaustible source of thought for the design and analysis of block ciphers. However, these search technologies do not extract any new features that are not available manually. Therefore, once optimal distinguishers are obtained, these automatic tools would exert less influence in improving attacks.

Recently, under the joint driven form of big data and the availability of computing hardware, deep learning [4, 21] has made remarkable progress and spread over almost every field of science and technology. Some researchers have started to explore the feasibility of applying machine learning to the field of cryptography. In ASIACRYPT 1991, Rivest [28] made preliminary explorations of the possible connection between cryptography and machine learning, and some researchers applied machine learning in side channel analysis successfully, such as [14, 23]. However, few researchers focused on the application of machine learning to black box cryptanalysis, until Gohr published an article on CRYPTO 2019 [12] that accelerated the process of applying deep learning to black box cryptanalysis.

Deep learning algorithms can analyze data and learn effective patterns for predicting new samples. Based on this, Gohr trained a deep neural network using the labeled (labels 0 and 1) ciphertext pairs as training data, where the data with label 1 comes from the encrypted plaintext pair with fixed input difference, and the data with label 0 is a random number. The trained neural network then was used to distinguish between the real ciphertext pairs and random pairs. When his network is applied to Speck32/64, higher accuracy than the pure differential distinguisher is achieved. Although the number of rounds using his network has not yet surpassed the number of rounds achieved by the most advanced technology, the neural distinguisher under the same number of rounds uses some information that the pure differential distinguisher has not tapped, that is, the neural distinguisher contains more information compared

with the pure differential cryptanalysis. This also raises a key question: If there is better cryptanalysis to make the neural distinguisher outstrip pure differential distinguisher in terms of the number of rounds and accuracy?

Contributions. In this paper, our contributions are as follows:

- In this paper, we analyze the information contained in different data formats to explore the useful and limited factors. The exploration results indicate that the integrity of the ciphertext pair information and the domain knowledge about the structure of the differential cryptanalysis have an essential impact on the accuracy of the network. Therefore, we propose the data format $(\Delta L, \Delta V, C_l, C_r)$ combining the information integrity with the domain knowledge as the input to the network. Experiments (See Section 4) demonstrate that this data format can greatly improve the accuracy of the neural distinguishers.
- In addition, we find that plaintext differences have a significant impact on the performance of the (related-key) neural distinguisher (See Section 5). Therefore, to effectively improve the accuracy of the neural distinguishers, we use SAT/SMT automatic tools to search for high probability compatible differential characteristics.
- We have successfully constructed the related-key neural distinguishers against Simon32/64, Simon64/128, Simeck32/64 and Simeck64/128 for the first time. Meanwhile, we build neural distinguishers for Simon32/64, Simon64/128, Simeck32/64 and Simeck64/128. The results are shown in Table 1, which shows that we have significantly improved the number of rounds and accuracy of the distinguishers.

Organization. We recall Simon-like ciphers, (related-key) differential cryptanalysis and CNN network in Section 2. In Section 3, we introduce our improved neural distinguishers, including the motivation, the new data pre-processing approach, and the network architecture. The improved neural distinguishers are applied in Simon and Simeck in Section 4. Section 5 compares the performance with different input difference. Section 6 concludes this paper.

2 Related works

2.1 Notations

Table 2 presents the notations we use throughout the paper.

2.2 A Brief Description of Simon and Simeck Ciphers

Simon. The lightweight family of AND-RX block ciphers Simon was proposed by the National Security Agency (NSA) in 2013. It adopts the Feistel structure and the round function consists of bitwise AND ($\odot$), bitwise XOR ($\oplus$) and cyclic left shift γ bit (S^{γ}) operation composition. The designer provides 10 versions, all marked as Simon$2n/mn$, where $2n$ represents the block size, mn represents the

Table 1: The comparison of (related-key) neural distinguishers attacks on Simon32/64, Simon64/128, Simeck32/64 and Simeck64/128.

Ciphers	Attack Model	Round	Input difference	Accuracy	Source
Simon 32/64	DD	9[1]	(0x0,0x80)	82.27%	[15]
		9	(0x0,0x80)	99.34%	4.1
		10[1]	(0x0,0x80)	61.09%	[15]
		10	(0x0,0x80)	86.08%	4.1
		11[1]	(0x0,0x40)	51.73%	[1]
		11	(0x20,0x88)	**59.55%**	4.1
	R-k DD	10	(0x0,0x4),(0x0,0x0,0x0,0x4)	99.89%	4.1
		11	(0x0,0x4),(0x0,0x0,0x0,0x4)	**97.90%**	
Simeck 32/64	PDD	9[1]	(0x0,0x1),(0x0,0x4)	75.84%	[10]
	DD	9	(0x0,0x40)	100%	4.2
	DD	10	(0x0,0x40)	89.70%	4.2
		11	(0x0,0x40)	**63.32%**	4.2
	R-k DD	13	(0x0,0x4),(0x0,0x0,0x0,0x4)	100%	4.2
		14	(0x0,0x4),(0x0,0x0,0x0,0x4)	**87.06%**	
Simon 64/128	DD	11[1]	(0x0,0x10)	73.79%	[15]
		11	(0x0,0x4)	99.28%	4.1
		12[1]	(0x0,0x10)	69.57%	[15]
		12	(0x0,0x4)	83.78%	4.1
		13	(0x0,0x40)	**60.32%**	4.1
	R-k DD	12	(0x0,0x4),(0x0,0x0,0x0,0x4)	99.98%	4.1
		13	(0x0,0x4),(0x0,0x0,0x0,0x4)	**95.49%**	
Simeck 64/128	DD	14	(0x0,0x1)	99.85%	4.2
		15	(0x0,0x1)	96.45%	
		16	(0x0,0x1)	82.28%	
		17	(0x0,0x1)	**64.24%**	
	R-k DD	18	(0x0,0x1),(0x0,0x0,0x0,0x1)	99.80%	4.2
		19	(0x0,0x1),(0x0,0x0,0x0,0x1)	95.80%	
		20	(0x0,0x1),(0x0,0x0,0x0,0x1)	80.71%	
		21	(0x0,0x1),(0x0,0x0,0x0,0x1)	**62.96%**	

[1] We choose the highest accuracy neural distinguishers in these paper.
[2] DD: Differential Distinguisher, R-k DD: Related-key Differential Distinguisher, PDD: Polytopic Differential Distinguisher.

Table 2: The notations used throughout the paper

Notation	Description
$x = (x_{n-1}, \ldots, x_0)$	Binary vector of n bits; x_i is the bit in position i with x_0 the least significant one.
$x \odot y$	Bitwise AND between x and y.
$x \oplus y$	Bitwise XOR between x and y.
$x \parallel y$	Concatenation of x and y.
$x \lll \gamma, S^{\gamma}(x)$	Circular left shift of x by γ bits.
$x \ggg \gamma, S^{-\gamma}(x)$	Circular right shift of x by γ bits.
(P_l, P_r, P'_l, P'_r)	A set of plaintext pairs with left and right branches where $P = P_l \parallel P_r$ and $P' = P'_l \parallel P'_r$.
(C_l, C_r, C'_l, C'_r)	A set of ciphertext pairs with left and right branches where $C = C_l \parallel C_r$ and $C' = C'_l \parallel C'_r$.
ΔL	$C_l \oplus C'_l$.
ΔV	$C_r \oplus C'_r$.
V_0	$C_l \oplus C_r$.
V_1	$C'_l \oplus C'_r$.

key length, $n \in \{16, 24, 32, 48, 64\}$, $m \in \{2, 3, 4\}$. The round function of SIMON algorithm is defined as:

$$f_{8,1,2}(x) = \left(S^8(x) \odot S^1(x)\right) \oplus S^2(x).$$

The round keys are generated using a linear key schedule through the $K = (k_{m-1}, k_{m-2}, \ldots, k_0)$. A more complete description can refer to paper [2].

Simeck. The SIMECK family of lightweight block ciphers was designed by Yang *et al.* [30], aiming at improving the hardware implementation cost of SIMON. SIMECK$2n/4n$ denotes an instance with a $2n$-bit block and a $4n$-bit key for $n \in \{16, 24, 32\}$. The round function of SIMECK algorithm is defined as:

$$f_{5,0,1}(x) = \left(S^5(x) \odot S^0(x)\right) \oplus S^1(x).$$

Conversely, SIMECK uses the non-linear key schedule which reuses the cipher's round function to generate the round keys. A more complete description can be found in [30].

2.3 (Related-key) Differential Cryptanalysis

Differential cryptanalysis is a chosen-plaintext attack introduced by Biham and Shamir in [6]. It analyzes the effect of the difference of a plaintext pair on the difference of succeeding round outputs in an iterated cipher. Because of its generality, differential cryptanalysis is an extensively exploited tool for cryptanalysis of encryption algorithms and for defining new attacks. Resistance to differential cryptanalysis became one of the basic criteria in the evaluation of the security of block ciphers.

Definition 1 (Difference). [6] Let X and X' be two bit strings of length n, then the difference between X and X' is defined as:

$$\Delta X = X \oplus X'.$$

Definition 2 (Differential Pair). [6] Let α, β be n-bit vectors, the difference value of the input pair (X, X') of the block cipher is $X \oplus X' = \alpha$, after r-round of encryption, the difference value of the output pair (Y, Y') is $Y \oplus Y' = \beta$, and let a round function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, then (α, β) is called an r-round differential pair of block cipher, where α is the input difference of round function f, β is the output difference of f. In particular, when $r = 1$, (α, β) characterizes the differential propagation characteristics of the round function f.

For a specific cipher, the differential must be carefully selected to make the differential attack successful. This makes researchers need to study the internal process of the algorithm. The basic method is to track a path passed by a high probability differential at different stages of encryption. This is called differential characteristics in cryptography and is defined as follows.

Definition 3 (Differential Characteristics). [6] Let X, X' be n-bit vectors and β_i be an n-bit constant. When the difference value of the input pair (X, X') satisfies $X \oplus X' = \beta_0$, the difference value of the intermediate state (Y_i, Y_i') satisfies $Y_i \oplus Y_i' = \beta_i$ during the r-th round of encryption, where, $1 \leq i \leq r$. Then, $\Omega = (\beta_1, \beta_2, \ldots, \beta_r)$ can be named an r-round differential characteristic of an iterative block cipher.

For given differential characteristics, use the following definition to calculate its probability.

Definition 4. [6] The probability $DP(\Omega)$ corresponding to an r-round differential characteristic $\Omega = (\beta_1, \beta_2, \ldots, \beta_r)$ of the iterative block cipher refers to the case where the input X and the round keys are independent and random distributed, when the differential value of the input pair (X, X') is $X \oplus X' = \beta_1$, in the i-round encryption process, the difference value of the intermediate state (Y_i, Y_i') satisfies the probability of $Y_i \oplus Y_i' = \beta_i$, where $1 \leq i \leq r$. Under the above assumption, the probability of the differential characteristic is equal to the product of the differential propagation probabilities of each round, i.e.:

$$DP(\Omega) = \prod_{i=1}^{r} Pr(\beta_{i-1} \to \beta_i) = \prod_{i=1}^{r} \frac{\{Y_{i-1} | f(Y_{i-1}) \oplus f(Y_{i-1} \oplus \beta_{i-1}) = \beta_i\}}{2^n}. \quad (1)$$

When the input difference undergoes a linear operation, it will be propagated through the operation with probability 1, and the output difference is deterministic, such as XOR ($\oplus$) and cyclic shift ($\lll, \ggg$) in the ARX operation. When the input difference passes through a non-linear operation, the difference propagation is often probabilistic.

Related-key differential cryptanalysis was introduced by Biham in [5]. Unlike the single-key differentials that have differences only in the plaintexts, related-key differential distinguishers have differences in the master keys as well. It exploits the output differences given a pair of plaintexts P and P' encrypted by a pair of related keys K and K', respectively. Related-keys differential cryptanalysis is also one of the basic criteria in the evaluation of the security of block ciphers, which successfully attacked many block ciphers, such as [7, 16, 19].

2.4 Convolutional Neural Network

Convolutional neural network (CNN) is an important paradigm in deep learning. CNN is usually composed of the convolutional layer, non-linear layer, pooling layer and fully connected layer.

Convolution Layer. Convolution is the basic operation of CNN, and its main purpose is to extract features. The core task of CNN is to learn parameters to extract effective patterns. In the forward propagation, the training data will go through the convolution kernel with initial parameters to obtain the initial output. In the back propagation, a loss function will be applied to adjust the parameters to minimize the gap between the initial output and the target label. After several iterations, when the loss stabilizes, the training process will be finished.

Non-linear layer. The main purpose of the non-linear layer is to introduce non-linear characteristics into the system. The most common non-linear layer in a CNN network is the ReLU function, defined as $f(x) = max(0, x)$, which not only accelerates the training speed, but also solves the gradient dispersion problem when the network is deep.

Fully connected layer. The fully connected layer is generally located in the back layers of the network for performing the classification task. Usually, the input of fully connected layer is the flatten feature map generated by convolution layer.

In addition, some functional layers may be used in CNN. For example, **Batch Normalization (BN)** can be applied after convolution layer to reduce the internal covariate shift, which can effectively prevent the gradient disappearance problem and speed up network training.

Residual Network (ResNet) is one of the most representative CNNs, which was proposed by He *et al.* [13] in 2015. ResNet can train a deeper CNN model to achieve higher accuracy. The core idea is to establish "shortcuts (skip) connections" between the front layer and the back layer. It is composed of a series of residual blocks. A residual block can be expressed as:

$$x_{l+1} = x_l + \mathcal{F}(x_l).$$

It is divided into two parts: the direct mapping part and the residual part. $\mathcal{F}(x_l)$ is the residual part, which is generally composed of two or three convolution operations. Many residual block variants can be designed by rearranging the activation functions of ReLU and BN. Figure 1 shows the residual block used in Gohr's work.

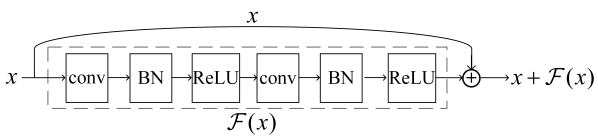


Fig. 1: The residual block used in Gohr's work.

3 Improved (Related-key) Neural Distinguishers

3.1 Revisiting Previous Work and Our Motivation

In Gohr's work, the (C_l, C_r, C'_l, C'_r) data format is used as the input of the network, which can be expressed in matrix form as:

$$\begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} C_l \\ C_r \\ C'_l \\ C'_r \end{bmatrix}, \text{ let } A_1 \triangleq \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}. \tag{2}$$

Then, $\text{rank}(A_1) = 4$.

Subsequently, Benamira *et al.* [3] conject the first convolution layer of Gohr's neural network transforms the input (C_l, C_r, C'_l, C'_r) into $(\Delta L, \Delta V, V_0, V_1)$ and a linear combination of those terms. Then, they verify the correctness of their conjecture through experiments, and keep $(\Delta L, \Delta V, V_0, V_1)$ as inputs for their pipeline to achieve better performance. It can be written as:

$$\begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,1\,0\,0 \\ 0\,0\,1\,1 \end{bmatrix} \cdot \begin{bmatrix} C_l \\ C_r \\ C'_l \\ C'_r \end{bmatrix}, \text{ let } A_2 \triangleq \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,1\,0\,0 \\ 0\,0\,1\,1 \end{bmatrix}. \tag{3}$$

Then, $\text{rank}(A_2) = 3$.

In paper [15], Hou *et al.* use the $(\Delta L, \Delta V)$ data format as the network input but the reason is not clarified. It can be written as:

$$\begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} C_l \\ C_r \\ C'_l \\ C'_r \end{bmatrix}, \text{ let } A_3 \triangleq \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix}. \tag{4}$$

Then, $\text{rank}(A_3) = 2$.

Considering key-recovery attacks, Bao *et al.* [1] use partial values combined with partial differences between ciphertext pairs, which are $(C_l, C'_l, \Delta V)$. It can be written as:

$$\begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix} \cdot \begin{bmatrix} C_l \\ C_r \\ C'_l \\ C'_r \end{bmatrix}, \text{ let } A_4 \triangleq \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix}. \tag{5}$$

Then, $\text{rank}(A_4) = 3$.

By observing the ranks of these matrices above, we find that the current approaches are suffering from the following drawbacks: (i) the existing works have not paid much attention to the importance of data integrity, which has a very large impact on the performance; and (ii) the data format does not well amplify the essential features of differential cryptanalysis.

To overcome these problems, we propose new data format I to train the neural distinguishers, considering both the information integrity and the domain knowledge about the structure of differential cryptanalysis. On the one hand, to make the data format I include as much as possible information, we design $I = A \cdot ((C_l, C_r, C_l{}', C_r{}'))^{\mathrm{T}}$, where A is a non-singular matrix. On the other hand, the data format I contain the ciphertext pair and their difference as the domain knowledge. Therefore, we consider the following data format: $(\Delta L, \Delta V, C_l, C_r)$, $(\Delta L, \Delta V, V_0, V_1, C_l, C_r)$, $(\Delta L, \Delta V, C_l, C_r, C_l', C_r')$. The respective matrix expression forms are:

$$A_5 = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \end{bmatrix}, A_6 = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,1\,0\,0 \\ 0\,0\,1\,1 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \end{bmatrix}, A_7 = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}. \tag{6}$$

Then, $\mathrm{rank}(A_5) = \mathrm{rank}(A_6) = \mathrm{rank}(A_7) = 4$.

Under the same conditions, the experiment (see Section 4.3) demonstrates the accuracy values of $(C_l, C_r, \Delta L, \Delta V)$, $(C_l, C_r, \Delta L, \Delta V, V_0, V_1)$ and $(C_l, C_r, C_l', C_r', \Delta L, \Delta V)$ input format are close, but higher than (C_l, C_r, C_l', C_r'), $(\Delta L, \Delta V, V_0, V_1)$, $(C_l, C_l', \Delta V)$ and $(\Delta L, \Delta V)$. Meanwhile, the longer the data format is, the more complex the training time and memory requirement are. Thus, the input data format of $(\Delta L, \Delta V, C_l, C_r)$ is more suitable. At the same time, notice that even if the internal sequence of these data formats is changed, there is almost no effect on the accuracy.

3.2 A New Data Pre-processing Approach

Chen *et al.* [9] propose multiple groups of ciphertext pairs that can achieve higher distinguishing accuracy. Thus, as shown in Figure 2, for differentials distinguishers, we first use a random key to encrypt the s-groups of plaintext pairs:

$$\left((P_l^1, P_r^1, P_l^{1\prime}, P_r^{1\prime}), (P_l^2, P_r^2, P_l^{2\prime}, P_r^{2\prime}), \ldots, (P_l^s, P_r^s, P_l^{s\prime}, P_r^{s\prime})\right). \tag{7}$$

to get the s-groups of ciphertext pairs:

$$\left((C_l^1, C_r^1, C_l^{1\prime}, C_r^{1\prime}), (C_l^2, C_r^2, C_l^{2\prime}, C_r^{2\prime}), \ldots, (C_l^s, C_r^s, C_l^{s\prime}, C_r^{s\prime})\right). \tag{8}$$

Then, use the s-groups of ciphertext pairs to get the data:

$$\begin{array}{c} (\Delta L^1, \Delta V^1, C_l^1, C_r^1), \\ (\Delta L^2, \Delta V^2, C_l^2, C_r^2), \\ \vdots \\ (\Delta L^s, \Delta V^s, C_l^s, C_r^s). \end{array} \tag{9}$$

where $\Delta L^i = C_l^i \oplus C_l^{i\prime}$, $\Delta V^i = C_r^i \oplus C_r^{i\prime}$, and the set $\left(\Delta L^i, \Delta V^i, C_l^i, C_r^i\right)$ of row i is denoted by Ω^i.

Finally, we splice Ω^i and convert it into a string of binary as a sample, and each sample will be attached a label Y:

$$Y\left(\Omega^1||\Omega^2\cdots||\Omega^s\right) = \begin{cases} 1, & \text{if } (P_l^i, P_r^i) \oplus (P_l^{i\prime}, P_r^{i\prime}) = \Delta_p, 1 \le i \le s, \\ 0, & \text{else.} \end{cases} \tag{10}$$

where Δ_p is a fixed plaintext difference.

Unlike differential distinguisher uses a random key K to encrypt the s-groups of plaintext pairs, related-key differential distinguisher uses a pair of keys (K, K') with a difference of Δ_k to encrypt the s-groups of plaintext pairs.

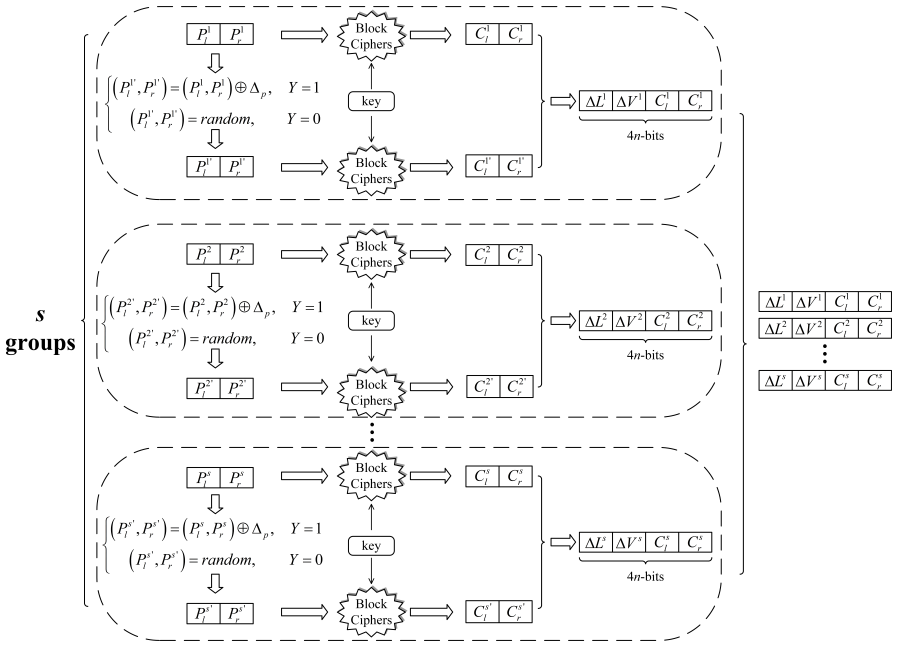


Fig. 2: A new data format using the multiple s-groups ciphertext pairs and difference

In our training process, the size of the training dataset is 10^7, and the validation dataset is 10^6. The validation dataset is only used to observe the model performance trained by training data and it does not participate in the parameter learning and selection process, so we directly use the same data from the validation dataset as the test dataset in our test process. Note that there is an

independent key used for each sample. Therefore, our training dataset has 10^7 corresponding random keys, and our test dataset has 10^6 corresponding random keys.

3.3 Network Architecture

A deep learning architecture is a multilayer stack of simple modules, most of which are subject to learning, and many of which compute non-linear input–output mappings. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details.

To illustrate the effect of data format we proposed on the accuracy of the neural distinguishers, we use a network similar to Gohr's [12] work. The network consists of three main components: input layer, iteration layer and predict layer, which is shown in Figure 3. The input layer receives training data with fixed length and uses one Conv1D layer. In the iteration layer, we use 5 residual blocks where each residual block contains two Conv1D layers. To make the network learning more stable and alleviate the problem of gradient disappearance, a BN layer is applied after each Conv1D layer, and then followed by an activation layer with ReLU function. Finally, in predict layer, to make the data smoothly transform from the convolutional layer to the fully connected layer, we need to introduce a flatten layer to perform one-dimensional flattening of the data output from the convolutional layer. The fully connected layer consists of two hidden layers where each has 64 neurons and an output unit with only one neuron.

We use Adam optimization algorithm [17], MSE loss function and L2 regularization parameterized by $c = 0.0001$. At the same time, we use a natural exponential decay learning rate.

4 Applications

In this paper, the experiment is conducted by Python 3.6.10 in Ubuntu 18.04 operating system. The models we use are implemented by Tensorflow 2.3.0. Our experiment uses a server with Intel(R) Xeon(R) Gold 6248 CPU *4 with 2.50GHz, 512GB RAM, and NVIDIA Tesla T4 16GB. We fix 64-groups to generate the dataset (See Section 4.4 for the reason). Each distinguisher is trained for 100 or 150 epochs.

4.1 (Related-key) Neural Distinguishers for Reduced Simon32/64 and Simon64/128

Hou *et al.* [15] train effective 9-, 10-round NDs against SIMON32/64 with the input difference `(0x0000,0x0080)`, but they fail in 11-round. Bao *et al.* use the input difference `(0x0000,0x0040)` to build 7-, 8-, 9-, 10-, 11-round NDs against SIMON32/64 in paper [1], but the 11-round ND they build achieves an accuracy

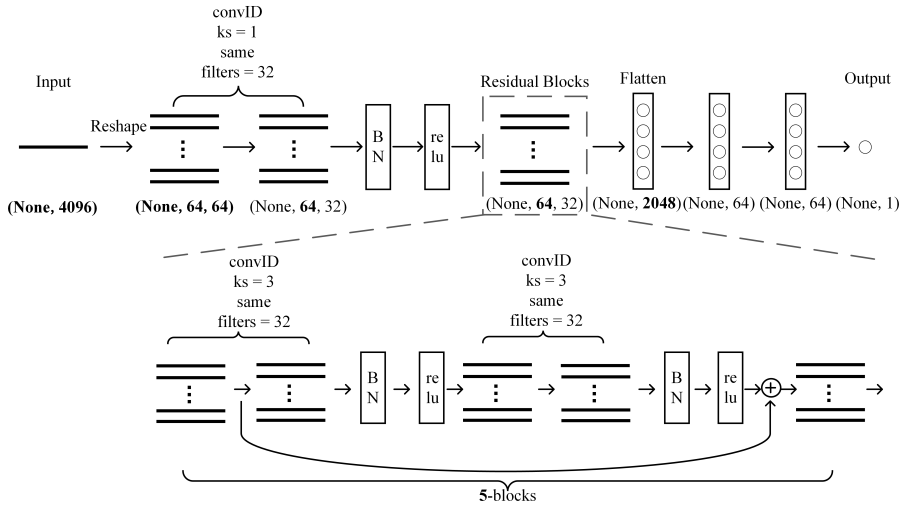


Fig. 3: Network Architecture proposed in this paper.

of 51.73%. We guess that the main reason for the invalidation of 11-round distinguisher using the difference `(0x0000,0x0080)` and `(0x0000,0x0040)` is that the optimal probability of these input differences propagating to 11-round is 2^{-34}. Thus, the ciphertext pairs generated by these input difference are almost irregular, which leads to the failure of the classification task. Therefore, in our work, we use the SAT/SMT solvers to search for the compatible differential distinguishers with a probability of no more than 2^{-2n} ($2n$ is the block size).

Based on the plaintext difference $\Delta_p =$ `(0x0000,0x0080)`, we build NDs to against Simon32/64 cover to 9-, 10-round with 99.34% and 86.08% accuracy, respectively. We obtain NDs against Simon32/64 cover to 11-round achieve an accuracy 59.55% with the input difference $\Delta_p =$ `(0x0020,0x0088)`. For Simon64/128, the NDs we train with `(0x0,0x4)` reach 99.28% and 83.78% accuracy for 11-, 12-round, respectively. We enjoy a 60.32% accuracy for the 13-round ND with the input difference `(0x0,0x40)`. The results are presented in Table 3. Compared with previous NDs, the number of rounds and the accuracy of our NDs are greatly improved.

Similarly, based on the plaintext difference $\Delta_p =$ `(0x0000,0x0004)` and the key difference $\Delta_k =$ `(0x0000,0x0000,0x0000,0x0004)`, we enjoy 99.89% and 97.90% accuracy for 10-, 11-round RKNDs against Simon32/64, respectively. Based on the plaintext difference $\Delta_p =$ `(0x0,0x4)` and the key difference $\Delta_k =$ `(0x0,0x0,0x0,0x4)`, we build RKNDs cover to 12-, 13-round with 99.98% and 95.49% accuracy for Simon64/128, respectively. To the best of our knowledge, this is the first successful application of the RKNDs against the block ciphers.

Table 3: Comparison of neural distinguishers for SIMON32/64 and SIMON64/128.

Ciphers	Attack	Data	Round	Input difference	Acc	Source
SIMON 32/64	DD	SCP	9	(0x0,0x40)	65.32%	[1]
		MOD	9	(0x0,0x80)	82.27%	[15]
		MOD-CP	9	(0x0,0x80)	**99.34%**	4.1
		SCP	10	(0x0,0x40)	56.08%	[1]
		MOD	10	(0x0,0x80)	61.09%	[15]
		MOD-CP	10	(0x0,0x80)	**86.08%**	4.1
		SCP	11	(0x0,0x40)	51.73%	[1]
		MOD-CP	**11**	(0x20,0x88)	**59.55%**	4.1
	R-k DD	MOD-CP	10	(0x0,0x4),(0x0,0x0,0x0,0x0,0x4)	99.89%	4.1
		MOD-CP	**11**	(0x0,0x4),(0x0,0x0,0x0,0x0,0x4)	**97.90%**	4.1
SIMON 64/128	DD	MOD	11	(0x0,0x10)	73.79%	[15]
		MOD-CP	11	(0x0,0x4)	**99.28%**	4.1
		MOD	12	(0x0,0x10)	69.57%	[15]
		MOD-CP	12	(0x0,0x4)	83.78%	4.1
		MOD-CP	**13**	(0x0,0x40)	**60.32%**	4.1
	R-k DD	MOD-CP	12	(0x0,0x4),(0x0,0x0,0x0,0x0,0x4)	99.98%	4.1
		MOD-CP	**13**	(0x0,0x4),(0x0,0x0,0x0,0x0,0x4)	**95.49%**	4.1

[1] We choose the highest accuracy NDs in these papers.

[2] SCP: Single Ciphertext Pair. MOD: Multiple Output Differences. MCP: Multiple Ciphertext Pairs. MOD-CP: Multiple Output Differences and Ciphertext Pairs.

4.2 (Related-key) Neural Distinguishers for Reduced Simeck32/64 and Simeck64/128

For SIMECK32/64, based on the plaintext difference Δ_p = (0x0000,0x0040), we get 9-, 10-, 11-round NDs against SIMECK32/64 with 100%, 89.70% and 63.32% accuracy, respectively. For SIMECK64/128, based on the plaintext difference Δ_p = (0x0,0x1), we obtain NDs against SIMECK64/128 cover to 14-, 15-, 16- and 17-round with 99.85%, 96.45%, 82.28% and 64.24% accuracy, respectively. The results are presented in Table 4. Compared with previous work, we have not only improved the number of rounds, but also the accuracy rate has been greatly improved. Meanwhile, it is interesting to see that although the parameters of the round functions of SIMON and SIMECK are different, the number of rounds and accuracy of the NDs are close.

Analogously, based on the plaintext difference Δ_p = (0x0000,0x0004) and the key difference Δ_k = (0x0000,0x0000,0x0000,0x0004), we build RKNDs cover to 13-, 14-round with 100% and 87.06% accuracy for SIMECK32/64, respectively. For SIMECK64/128, based on the plaintext difference Δ_p = (0x0,0x1) and the key difference Δ_k = (0x0,0x0,0x0,0x1), we build RKNDs cover to 18-, 19-, 20- and 21-round with 99.80%, 95.80%, 80.71% and 62.96% accuracy for SIMECK64/128, respectively. This seems interesting. Although SIMON and SIMECK have similar NDs performance, the gap of RKNDs is obvious. And SIMON's key-expansion algorithm offers the better resistance. This is consistent with the conclusion Lu *et al.* get using rotational-XOR cryptanalysis in [22].

4.3 Experiment with Different Data Format

In order to show that our new data format $(\Delta L, \Delta V, C_l, C_r)$ is indeed better than others, based on the experiment of 10-round ND against SIMON32/64, we only change the data format, and other parameters are fixed for comparison experiments.

The results are presented in Table 5, the accuracy of the NDs with the data format $(C_l, C_r, C_l{}', C_r{}')$ is no more than 50% for 10-round SIMON32/64. The accuracy of neural distinguishers using data formats of $(\Delta L, \Delta V, V_0, V_1)$, $(\Delta L, \Delta V)$, $(C_l, C_l', \Delta V)$, $(\Delta L, \Delta V, C_l, C_r)$, $(\Delta L, \Delta V, V_0, V_1, C_l, C_r)$ and $(\Delta L, \Delta V, C_l, C_r, C_l', C_r')$ are all over 50%, but the accuracy rate improved by using the latter three is greater than that of the first three. In addition, the use of $(\Delta L, \Delta V, V_0, V_1, C_l, C_r)$ and $(\Delta L, \Delta V, C_l, C_r, C_l', C_r')$ has higher memory requirements. Therefore, $(\Delta L, \Delta V, C_l, C_r)$ is a more appropriate choice considering both the accuracy need and memory and time cost.

4.4 How to Choose Parameter S

Chen *et al.* [9] explain that why it is better to consider multiple groups ciphertext pairs: if ciphertext pairs corresponding to plaintext pairs with a specific plaintext difference obey a non-uniform distribution, there are some derived features from multiple ciphertext pairs. Once the network captures these features,

Table 4: Comparison of neural distinguishers for SIMECK32/64 and SIMECK64/128.

Ciphers	Attack	Data Format	Input difference	Round	Acc	Source
SIMECK 32/64	PDD	SCP	(0x0,0x1)	9 [1]	75.84%	[10]
	DD	MOD-CP	(0x0,0x40)	9	100%	4.2
				10	89.70%	
				11	**63.32%**	
	R-k DD	MOD-CP	(0x0,0x4),(0x0,0x0,0x0,0x4)	13	100%	4.2
				14	**87.06%**	
SIMECK 64/128	DD	MOD-CP	(0x0,0x1)	14	99.84%	4.2
				15	96.45%	
				16	82.28%	
				17	**64.24%**	
	R-k DD	MOD-CP	(0x0,0x1), (0x0,0x0,0x0,0x1)	18	99.80%	4.2
				19	95.80%	
				20	80.71%	
				21	**62.96%**	

[1] We choose the highest accuracy neural distinguisher in paper [10].
[2] DD: Differential Distinguisher, R-k DD: Related-key Differential Distinguisher, PDD: Polytopic Differential Distinguisher. SCP: Single Ciphertext Pair. MOD-CP: Multiple Output Differences and Ciphertext Pairs.

Table 5: Experiment with Different Data Format[1].

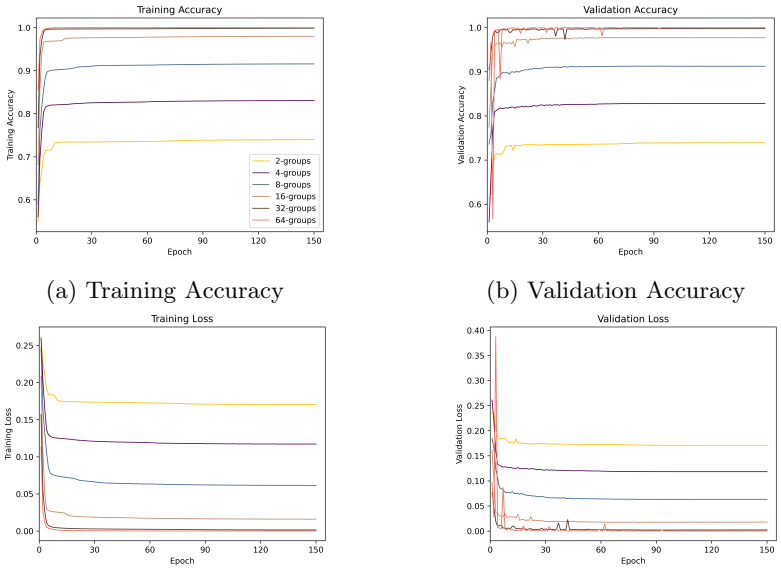
Ciphers	Attack	Round	Data Format	Accuracy	Source
SIMON32/64	DD	10	(C_l, C_r, C'_l, C'_r)	50.01%	[12]
			$(\Delta L, \Delta V, V_0, V_1)$	61.48%	[3]
			$(\Delta L, \Delta V)$	61.09%	[15]
			$(C_l, C'_l, \Delta V)$	61.20%	[1]
			$(\Delta L, \Delta V, C_l, C_r)$	77.96%	This Paper.
			$(\Delta L, \Delta V, V_0, V_1, C_l, C_r)$	76.12%	This Paper.
			$(\Delta L, \Delta V, C_l, C_r, C'_l, C'_r)$	77.13%	This Paper.

[1] Keeping other experimental conditions the same, these experiments are carried out with different data formats. We use 32-groups multiple ciphertext pairs and the input difference used is `(0x0000,0x0080)`.

the neural distinguishers would obtain some performance promotions. But they don't specify how many groups are appropriate.

Therefore, in order to explore how to select the parameter s, we design a comparative experiment that fixed all other parameters but only the parameter s is variable, where $s \in \{2, 4, 6, 8, 16, 32, 64\}$.

It can be found that the accuracy is growing when the parameter s (the number of groups) increases in Figure 4, and the accuracy does not increase after reaching 99.99%. But it should be noted that with the increase of s, the training time is also increasing. To make a trade-off between training time and accuracy, we select 64 as the value of s to build the (related-key) neural distinguishers in this paper.



(a) Training Accuracy

(b) Validation Accuracy

(c) Training Loss

(d) Validation Loss

Fig. 4: Training a neural network to distinguish 13-round SIMECK32/64 output for the input difference Δ_p = `(0x0000,0x0004)` and Δ_k = `(0x0000,0x0000,0x0000,0x0004)` from random data. Only the parameter s is different in these experiments while the other experimental conditions are the same.

5 Comparing the Performance with Different Input Difference

In Section 4, we use the SAT/SMT solvers to search for the compatible differential distinguishers with a probability of no more than 2^{-2n} ($2n$ is the block

size). We find a 19-round related-key differential distinguisher with probability 2^{-32} for SIMON32/64, and the characteristic is shown in Table 7. Then, based on the plaintext difference Δ_p = (0x0000,0x5555) and the key difference Δ_k = (0xaaaa,0xaaaa,0x0000,0x5555), we obtain an 8-round RKND with an accuracy of 91.22%. But for 9-round, the training scheme fails, i.e. the model does not learn a useful distinguishing function by mining the information contained in the training data. This is interesting, although the pure distinguisher has reached 19-round, it cannot reach 9-round ND when the input difference of the 19-round distinguisher is used in the network. Hou *et al.* [15] think the higher the hamming weight of the input difference, the weaker the non-random feature of the ciphertext pair. We agree with this view to a certain extent, and as described in Section 4.1, based on the plaintext difference Δ_p = (0x0000,0x0004) and the key difference Δ_k = (0x0000,0x0000,0x0000,0x0004) with low hamming weight, we enjoy 99.89% and 97.90% accuracy for 10-, 11-round RKNDs against SIMON32/64, respectively. The pure characteristic is also shown in Table 7.

But as shown in Table 6, it can be seen that the input difference (0x0400,0x1000), (0x0010,0x0040), (0x0000,0x0028) and (0x0000,0x0110) with hamming weight 2 fail for 11-round ND against SIMON32/64, but the input difference (0x0020,0x0088), (0x0200,0x0880), (0x2000,0x8800) and (0x0002,0x8008) with hamming weight 3 obtain 11-round ND with an accuracy of 58%-60% against SIMON32/64. Therefore, it needs to be further explored in the future that how the input differences influence the performance of the network.

Table 6: Comparing the ND Performance with Different Input Difference for SIMON32/64.

Cipher	Round	Accuracy	Input difference	Hamming weight
SIMON32/64	11	50%-51%	(0400,1000) (0010,0040) (0000,0028) (0000,0110)	2
		58%-60%	(0020,0088) (0200,0880) (2000,8800) (0002,8008)	3
		50%-51%	(0004,0410) (0080,0022) (1000,0440) (0040,0011)	3

[1] The optimal probability of these input differences propagating to 11-round is 2^{-32}.

6 Conclusion

In this paper, we provide an in-depth analysis of the (related-key) neural differential distinguishers. We find that the difference between the ciphertext pair and the integrity of the ciphertext pair information have an essential impact on the performance to the network. Therefore, we use the data format $(\Delta L, \Delta V, C_l, C_r)$

Table 7: The 12-, 19-round related-key characteristics for SIMON32/64

Round	SIMON32/64 key difference	SIMON32/64 data difference	SIMON32/64 key difference	SIMON32/64 data difference
0	0004	(0000||0004)	5555	(0000||5555)
1	0000	(0000||0000)	0000	(0000||0000)
2	0000	(0000||0000)	aaaa	(0000||0000)
3	0000	(0000||0000)	aaaa	(aaaa||0000)
4	0004	(0000||0000)	aaaa	(0000||aaaa)
5	c000	(0004||0000)	0000	(0000||0000)
6	1400	(c010||0004)	5555	(0000||0000)
7	03c6	(0427||c010)	aaaa	(5555||0000)
8	0040	(fc04||0427)	5555	(0000||5555)
9	de0c	(4800||fc04)	0000	(0000||0000)
10	4004	(0201||4800)	aaaa	(0000||0000)
11	cfa6	(0000||0201)	aaaa	(aaaa||0000)
12		(cda7||0000)	aaaa	(0000||aaaa)
13			0000	(0000||0000)
14			5555	(0000||0000)
15			aaaa	(5555||0000)
16			5555	(0000||5555)
17			0000	(0000||0000)
18			aaaa	(0000||0000)
19				(aaaa||0000)
Prob.	1	2^{-32}	1	2^{-32}

as the input of the network. In addition, to improve the accuracy of the neural distinguishers, we also use the SAT/SMT automatic tools to find differential characteristics with high differential probabilities. For SIMON32/64, SIMON64/128, SIMECK32/64 and SIMECK64/128, we construct the longest neural network (related-key) differential distinguishers with higher accuracy.

It is undeniable that there are many factors that can affect the performance of neural distinguishers, such as the data format discussed in this paper, network structure, methods of model training, and so on. This paper explores its impact on the performance of neural distinguishers from the perspective of data format. In the future, we plan to further explore ways that can improve the distinguishing performance of neural networks from multiple dimensions, such as using some classic deep learning models, using methods of feature engineering to extract more essential characteristics of the training data, combining pre-training models with the domain knowledge and so on.

References

1. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Conditional differential-neural cryptanalysis. Cryptology ePrint Archive (2021)
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference. pp. 1–6 (2015)
3. Benamira, A., Gerault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 805–835. Springer (2021)
4. Bengio, Y., Lecun, Y., Hinton, G.: Deep learning for ai. Communications of the ACM 64(7), 58–65 (2021)
5. Biham, E.: New types of cryptanalytic attacks using related keys. Journal of Cryptology 7(4), 229–246 (1994)
6. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY 4(1), 3–72 (1991)
7. Biryukov, A., Nikolić, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 322–344. Springer (2010)
8. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Designs, codes and cryptography 70(3), 369–383 (2014)
9. Chen, Y., Shen, Y., Yu, H., Yuan, S.: A new neural distinguisher considering features derived from multiple ciphertext pairs. Cryptology ePrint Archive (2021)
10. Fu, C., Duan, M., Wei, Q., Wu, Q., Zhou, R., Su, H.: Polytopic differential attack based on deep learning and its application (chinese version). Journal of Cryptologic Research (2021)
11. Gerault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: International Conference on Principles and Practice of Constraint Programming. pp. 584–601. Springer (2016)
12. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Annual International Cryptology Conference. pp. 150–179. Springer (2019)

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. Journal of Cryptographic Engineering 1(4), 293 (2011)
15. Hou, Z., Ren, J., Chen, S.: Improve neural distinguisher for cryptanalysis. Cryptology ePrint Archive (2021)
16. Jakimoski, G., Desmedt, Y.: Related-key differential cryptanalysis of 192-bit key aes variants. In: International Workshop on Selected Areas in Cryptography. pp. 208–221. Springer (2003)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: International Workshop on Fast Software Encryption. pp. 112–127. Springer (2002)
19. Ko, Y., Hong, S., Lee, W., Lee, S., Kang, J.S.: Related key differential attacks on 27 rounds of xtea and full-round gost. In: International Workshop on Fast Software Encryption. pp. 299–316. Springer (2004)
20. Kölbl, S., Leander, G., Tiessen, T.: Observations on the simon block cipher family. In: Annual Cryptology Conference. pp. 161–185. Springer (2015)
21. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436–444 (2015)
22. Lu, J., Liu, Y., Ashur, T., Li, C.: On the effect of the key-expansion algorithm in simon-like ciphers. The Computer Journal (2021)
23. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
24. Matsui, M.: Linear cryptanalysis method for des cipher. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 386–397. Springer (1993)
25. Minier, M., Solnon, C., Reboul, J.: Solving a symmetric key cryptographic problem with constraint programming. In: ModRef 2014, Workshop of the CP 2014 Conference. p. 13 (2014)
26. Mouha, N., Preneel, B.: A proof that the arx cipher salsa20 is secure against differential cryptanalysis. IACR Cryptol. ePrint Arch. 2013, 328 (2013)
27. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: International Conference on Information Security and Cryptology. pp. 57–76. Springer (2011)
28. Rivest, R.L.: Cryptography and machine learning. In: International Conference on the Theory and Application of Cryptology. pp. 427–439. Springer (1991)
29. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des (l) and other bit-oriented block ciphers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 158–178. Springer (2014)
30. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 307–329. Springer (2015)