Improved (Related-key) Differential-based Neural Distinguishers for SIMON and SIMECK Block Ciphers

Jinyu Lu[1,2,3], Guoqiang Liu[1,2,4*], Bing Sun[1,2,3], Chao Li[1,2,3] and Li Liu[5,6]

[1] *College of 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*
Email: liuguoqiang87@hotmail.com

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. Meanwhile, taking the performance of the differential-neural distinguisher of Simon32/64 as an entry point, we investigate the impact of input difference on the performance of the hybrid distinguishers to choose the proper input difference. Eventually, we improve the accuracy of the neural distinguishers of Simon32/64, Simon64/128, Simeck32/64, and Simeck64/128. We also obtain related-key differential-based neural distinguishers on round-reduced versions of Simon32/64, Simon64/128, Simeck32/64, and Simeck64/128 for the first time.

Keywords: Deep Learning; (Related-key) Differential Distinguisher; Simon*;* Simeck*; Input Difference*

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 [1], linear cryptanalysis [2], integral cryptanalysis [3], zero-correlation linear cryptanalysis [4], 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 MILP problems [5,6], SAT/SMT problems [7,8] or CP problems [9,10]. 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 [11, 12] has made remarkable progress and spread over almost every field of science and technology. Some researchers explored the feasibility of applying machine learning to the field of cryptography. In ASIACRYPT 1991, Rivest [13] 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, 15]. However, few researchers focused on the application of machine learning to black box cryptanalysis, until the process of applying deep learning to black box cryptanalysis was accelerated by the remarkable work of Gohr [16].

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 is used to distinguish between the real ciphertext pairs and random pairs. When his network is applied to SPECK32/64, higher accuracy than the classical differential (CD) 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 (ND) under the same number of rounds uses some information that the CD has not tapped.

More importantly, a potent key recovery attack is created by combining NDs with CDs and highly selective key search strategies. In essence, the NDs are too short to be used in key recovery and must be prepended with CDs to get the hybrid distinguishers (HDs). Making the resulting HDs usable in a key recovery attack requires better NDs or prepended CDs. Researchers have provided solutions from various angles. Benamira *et al.* [17] analyzed and explained the inner workings of Gohr's neural network and enhanced the accuracy of the NDs by creating batches of ciphertext inputs instead of pairs. Bao *et al.* [18] enhanced the CD's neutral bits and trained better NDs by investigating different neural networks, enabling key recovery attacks for the 13-round SPECK32/64 and 16-round SIMON32/64.

Our contribution:

- In this paper, we present (related-key) differential-based neural distinguishers on SIMON and SIMECK block ciphers. To better match our neural network and increase the accuracy of the neural distinguisher, we adopt the multiple ciphertext pairs (8 ciphertext pairs) to train the neural network fed with the data of form $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$. Fig. 1 shows a schematic representation of these notations. Also, we employ the SE-ResNet network (Fig. 2) due to the success of ResNet on SPECK [16] and SENet on SIMON [18], as well as their superior performance on classification tasks.
- We notice that the choice of the ND or connecting difference is critical to obtain the best hybrid distinguishers. Therefore, taking the performance of the differential-neural distinguisher of SIMON32/64 as an entry point, we investigate the impact of input difference of the ND on the performance of the hybrid distinguishers to choose the proper input difference. As a result, the input difference $(0, e_i)$ is a good choice to obtain hybrid distinguishers for SIMON-like ciphers.
- Eventually, 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 improve the accuracy of the distinguishers. Meanwhile, we successfully construct the related-key neural distinguishers against SIMON32/64, SIMON64/128, SIMECK32/64 and SIMECK64/128 for the first time.

In this paper, the experiment is conducted by Python 3.6.10 in Ubuntu 18.04. The models are implemented by Tensorflow 2.5.0. The experiment uses a server with Intel(R) Xeon(R) Gold 6248 CPU *4 with 2.50GHz, 512GB RAM, and NVIDIA Tesla T4 16GB. The source code is available on Github[1].

Organization. Section 2 recalls SIMON-like ciphers, (related-key) differential cryptanalysis and CNN network. Section 3 introduces improved (related-key) differential-based neural distinguishers, including the batches of ciphertext pairs with new data format, and the network architecture. Section 4 compares the performance of the hybrid distinguisher with different input difference. Section 5 gives the (related-key) differential-neural distinguishers for round-reduced SIMON32/64 and SIMON64/128. Section 6 provides the (related-key) differential-neural distinguishers for round-reduced SIMECK32/64 and SIMECK64/128. Section 7 concludes this paper.

2. RELATED WORKS

2.1. Notations

Table 2 presents the notations used in this 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

[1] https://github.com/JIN-smile/Improved-Related-key-Differential-based-Neural-Distinguishers

TABLE 1: The comparison of (related-key) neural distinguishers attacks on SIMON32/64, SIMON64/128, SIMECK32/64, and SIMECK64/128 with 8 ciphertext pairs as a sample. ND: neural distinguisher, RKND: related-key neural distinguisher. TPR: True Positive Rate, TNR: True Negative Rate. †: For NDs fed with single ciphertext pairs, the combine-response distinguisher (CRD) obtained for the case of 8 ciphertext pairs. *: This neural distinguisher is trained using the staged training method.

Ciphers	Attack Model	Round	Input difference	Accuracy	TPR	TNR	Source
SIMON 32/64	ND	9†	(0x0,0x40)	0.8940	0.8728	0.9152	[18]
		9	(0x0,0x40)	0.9176	0.9052	0.9299	Sect. 5
		10*†	(0x0,0x40)	0.6865	0.6817	0.6912	[18]
		10	(0x0,0x40)	0.6975	0.6662	0.7287	Sect. 5
		11*†	(0x0,0x40)	0.5568	0.5419	0.5717	[18]
		11	(0x0,0x40)	0.5609	0.5366	0.5852	Sect. 5
		12	(0x1,0x4)	0.5152	0.4799	0.5505	Sect. 5
		12*	(0x0,0x40)	0.5142	0.5029	0.5254	
	RKND	10	(0x0,0x40),(0x0,0x0,0x0,0x40)	1	1	1	Sect. 5
		11	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.9604	0.9639	0.9569	
		12	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.6477	0.6518	0.6435	
		13	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5262	0.5437	0.5081	
SIMECK 32/64	ND	9	(0x0,0x40)	0.9952	0.9989	0.9914	Sect. 6
		10	(0x0,0x40)	0.7354	0.7207	0.7501	
		11	(0x0,0x40)	0.5646	0.5356	0.5936	
		12*	(0x0,0x40)	0.5146	0.4770	0.5522	
	RKND	13	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.9950	0.9990	0.9910	Sect. 6
		14	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.6679	0.6425	0.6933	
		15	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5467	0.5173	0.5762	
SIMON 64/128	ND	11	(0x0,0x40)	0.9181	0.9045	0.9318	Sect. 5
		12	(0x0,0x40)	0.7117	0.6705	0.7530	
		13	(0x0,0x40)	0.5722	0.5230	0.6215	
		14	(0x0,0x40)	0.5148	0.4697	0.5600	
		14*	(0x0,0x40)	0.5185	0.4663	0.5707	
	RKND	12	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.9880	0.9894	0.9865	Sect. 5
		13	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.8398	0.8389	0.8408	
		14	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5788	0.5894	0.5682	
SIMECK 64/128	ND	14	(0x0,0x40)	0.9142	0.8914	0.9371	Sect. 6
		15	(0x0,0x40)	0.7663	0.6981	0.8345	
		16	(0x0,0x40)	0.6356	0.5245	0.7467	
		17	(0x0,0x40)	0.5577	0.4301	0.6853	
		18	(0x0,0x40)	0.5202	0.3917	0.6486	
		18*	(0x0,0x40)	0.5218	0.3927	0.6510	
	RKND	18	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.9066	0.8837	0.9295	Sect. 6
		19	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.7558	0.6845	0.8270	
		20	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.6229	0.5104	0.7354	
		21	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5519	0.4248	0.6790	
		22	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5180	0.3906	0.6455	

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 $C' = C'_l \parallel C'_r$.

($\odot$), bitwise XOR ($\oplus$) and cyclic left shift γ bit (S^{γ}) operation composition. The designer provides ten versions, all marked as SIMON$2n/mn$, where $2n$ represents the block size, mn represents the 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 [19].

Simeck. The SIMECK family of lightweight block ciphers was designed by Yang *et al.* [20], 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 [20].

Simon-like ciphers. Iterated ciphers that use SIMON's round function and generalize it to accept arbitrary rotational parameters are known as SIMON-like ciphers (a, b, c). The SIMON-like function is then $f_{a,b,c}(x) = \left(S^a(x) \odot S^b(x)\right) \oplus S^c(x)$, which the rotational parameters (a, b, c) are (8,1,2) and (5,0,1) for all SIMON and SIMECK versions, respectively.

2.3. (Related-key) Differential Cryptanalysis

Differential cryptanalysis is a chosen-plaintext attack introduced by Biham and Shamir in [1]. It analyzes the effect of the difference of a plaintext pair on the difference of succeeding round outputs in an iterated cipher. Differential cryptanalysis is a widely used tool for the cryptanalysis of encryption algorithms and the development of new attacks due to its generality. Resistance to differential cryptanalysis became one of the basic criteria in the evaluation of the security of block ciphers.

DEFINITION 2.1 (Difference). [1] 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 2.2 (Differential Pair). [1] 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 2.3 (Differential Characteristics). [1] 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 2.4. [1] 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.*,:

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

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 [21]. 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 has successfully attacked many block ciphers, such as [22–24].

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. According to the convolution dimension of the feature map, it can be divided into one-, two-, and three-dimensional convolutional neural network (*i.e.*, 1D-CNN, 2D-CNN and 3D-CNN), where the 1D-CNN applies a convolution over a fixed (multi-)temporal input signal.

Convolution Layer (CONV). 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. Note that in this paper we apply 1D-CNN, then the convolution layer can be denoted by Conv1D.

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 rectified linear unit (**ReLU**) function, defined as $f(x) = max(0, x)$. Effectively, it removes negative values from an activation map by setting them to zero. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase nonlinearity, such as the sigmoid function. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

Fully connected layer (FC). The fully connected layer is generally located in the back layers of the network for performing the classification task. Usually, the input of the 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 the 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.* [25] 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. The activation functions of ReLU and BN can be rearranged to create a variety of residual block variants.

Squeeze-and-Excitation Network (SENet) is a new network structure proposed by Hu *et al.* that won the first place in ILSVRC 2017 classification competition [26]. The "Squeeze-and-Excitation" (SE) block adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. It can be integrated into standard architectures by insertion after the non-linearity following each convolution. In this paper, SE block is used directly with the residual network, *i.e.*, the SE-ResNet network.

3. IMPROVED (RELATED-KEY) DIFFERENTIAL-BASED NEURAL DISTINGUISHERS

3.1. Dateset: Multiple Ciphertext Pairs with New Data Format

Data plays a very important role in deep learning, data preparation is a fundamental step for deep learning model development. Some researchers explored the use of multiple ciphertext pairs to improve the performance of differential-based neural distinguishers [17, 27, 28]. Some researchers also performed additional transformations on each pair of ciphertexts before feeding them into the network. Concretely, in Gohr's work, the n-round NDs fed with data of form (C_l, C_r, C'_l, C'_r). Subsequently, Benamira *et al.* [17] conjected the first convolution layer of Gohr's neural network transforms the input (C_l, C_r, C'_l, C'_r) into $(C_l \oplus C'_l, C_l \oplus C'_l \oplus C_r \oplus C'_r, C_l \oplus C_r, C'_l \oplus C'_r)$ and a linear combination of those terms. In [28], Hou *et al.* designed the NDs model with multiple output differences as a sample, *i.e.*, the n-round NDs fed multiple pairs with data of form $(C_l \oplus C'_l, C_r \oplus C'_r) \triangleq (\Delta^r_L, \Delta^r_R)$. In [18], Bao *et al.* accepted the r-round NDs fed with data of form $(C_r, C'_r, \Delta^{r-1}_R)$, where $\Delta^{r-1}_R = ((C_r \lll 8) \odot (C_r \lll 1) \oplus (C_r \lll 2) \oplus C_l) \oplus ((C'_r \lll 8) \odot (C'_r \lll 1) \oplus (C'_r \lll 2) \oplus C'_l)$ for SIMON ciphers.

In this paper, we employ multiple ciphertext pairs with new data of form

$(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$ to improve the performance of neural distinguishers (the reason for choosing this data format is given in Section 5.3). Then, the process of constructing a dataset can be described.

For the differential-neural distinguisher, first encrypt the s plaintext pairs $((P,P')^1, (P,P')^2, \ldots, (P,P')^s)$ with a random key to get the s ciphertext pairs. Then, use the s ciphertext pairs to get the data:

$$\begin{array}{c}(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})^1,\\ (\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})^2,\\ \vdots\\ (\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})^s.\end{array}$$

where the set $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})^i$ of row i is denoted by Ω^i.

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

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

where Δ_p is a constant input difference. It examines how to select the Δ_p in Section 4.

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

We construct the dataset based on the above steps and set $s = 8$. In the basic training process, the size of the training set is 2×10^7, and the test set is 2×10^6. Meanwhile, there is an independent key used for each sample. Therefore, the training set has 2×10^7 corresponding random keys, and the test set has 2×10^6 corresponding random keys.

3.2. 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.

Given the success of ResNet on SPECK [16] and SENet on SIMON [18], as well as their superior performance on classification tasks, we use the SE-ResNet network. As shown in Fig. 2, the network consists of three main components: input layer, iteration layer and predict layer. The input layer uses one Conv1D layer and two Dense layers to receive fixed length training data. In the iteration layer, use 5 SE-ResNet modules where each module contains two Conv1D layers and one SE block. 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 introduce a flatten layer to perform one-dimensional flattening of the data output from the convolutional layer. The fully connected layer consists of two Dense layers where each has 64 neurons and an output unit with only one neuron.

We set the batch size to 30000, cyclic learning rate $l_i = \alpha + \frac{(n-i) mod (n+1)}{n} \cdot (\beta - \alpha)$ with $\alpha = 0.0001$, $\beta = 0.003$, $n = 29$ for epoch i, which is denoted as cyclic_lr$(30, 0.003, 0.0001)$. Adam [29] is used as the optimizer with mean squared error (MSE) loss function and L2 regularization parameterized by $c = 0.00001$. Each dataset is trained with 120 epochs for the basic training method. The accuracy, TPR, and TNR of the ND are the average results after 5 repetitions.

4. COMPARING THE PERFORMANCE OF THE HYBRID DISTINGUISHER WITH DIFFERENT INPUT DIFFERENCE

In this section, we investigate the effect of input difference of the NDs on the performance of the hybrid distinguishers. Essentially, to be used in key-recovery, the NDs are too short such that they have to be prepended with classical differentials. Whether the resulting HDs can be used in a key-recovery attack depends on whether the input difference of NDs leads to better accuracy and, at the same time, leads to prepended CDs with high differential probability.

Therefore, taking the performance of the hybrid distinguisher of SIMON32/64 as an entry point, we investigate the issue in two phases. In the first stage, we study the performance of all input differences with Hamming weights of 1, 2, and 3 on the 11-round ND, and filter the input differences that can obtain a non-marginal advantage (accuracy above 0.50). Then study the performance of these filtered input differences on 12-round ND. In the second stage, we study the probability of the prepended CDs with these filtered input differences.

The First Stage

Let $\mathrm{HW}(\Delta_p)$ denote the Hamming weight of the input difference, then there are $32 + 496 + 4960 = 5488$ input difference with $\mathrm{HW}(\Delta_p) \le 3$. Based on Section 3, traversing these input difference Δ_p with the batch size 30000 and cyclic_lr$(30, 0.003, 0.0001)$, we construct 11-round ND of SIMON32/64, respectively. There are 128 input differences filtered, of which 48 have an accuracy between 0.51-0.52 and 80 have an accuracy between 0.54-0.56. Therefore, we mainly focus on the

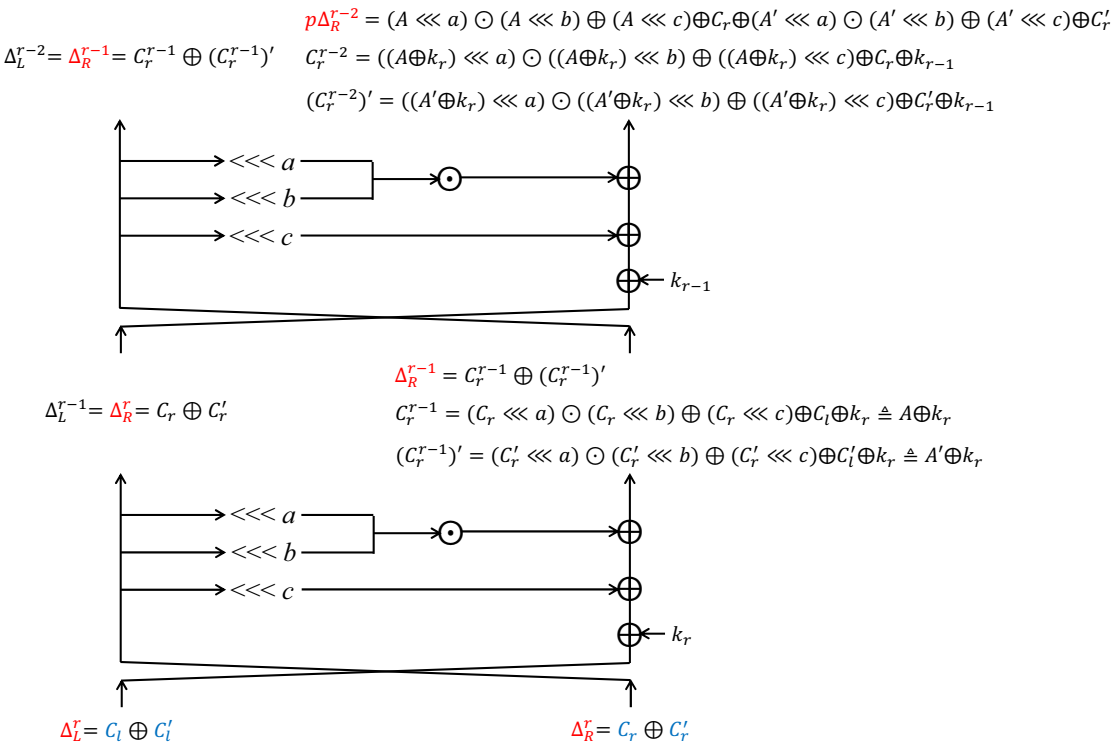

FIGURE 1: Notation of the data format.

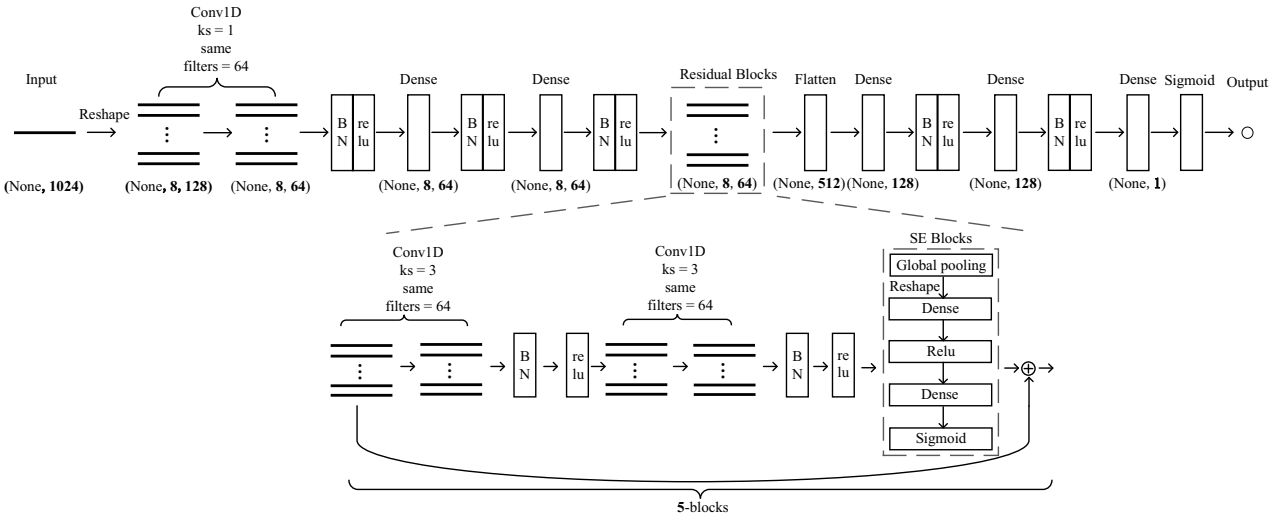

FIGURE 2: Network architecture proposed in this paper.

performance of these 80 input differences. The results with these 80 input differences are shown in Fig. 3.

It is discovered that 11-round ND with input difference $\Delta_p = (a, b)$ and input difference $\Delta_p' = (a \lll i, b \lll i)$ have similar accuracy, for $0 \leq i < 16$. Thus, we only list one of these 16 input differences in Table 4. Specifically, for $\mathrm{HW}(\Delta_p) = 1$, using the input difference (omit the $0x$ symbol):

(0000,0001), (0000,0002), (0000,0004), (0000,0008), (0000,0010), (0000,0020), (0000,0040), (0000,0080), (0000,0100), (0000,0200), (0000,0400), (0000,8000), (0000,1000), (0000,2000), (0000,4000), (0000,8000),

can construct 11-round ND of Simon32/64 with an accuracy of about 0.561.

For $\mathrm{HW}(\Delta_p) = 2$, using the input difference:

(0001,0004), (0002,0008), (0004,0010), (0008,0020), (0010,0040), (0020,0080), (0040,0100), (0080,0200), (0100,0400), (0200,0800), (0400,1000), (0800,2000), (1000,4000), (2000,8000), (4000,0001), (8000,0002),

can build 11-round ND of Simon32/64 with an accuracy of about 0.560.

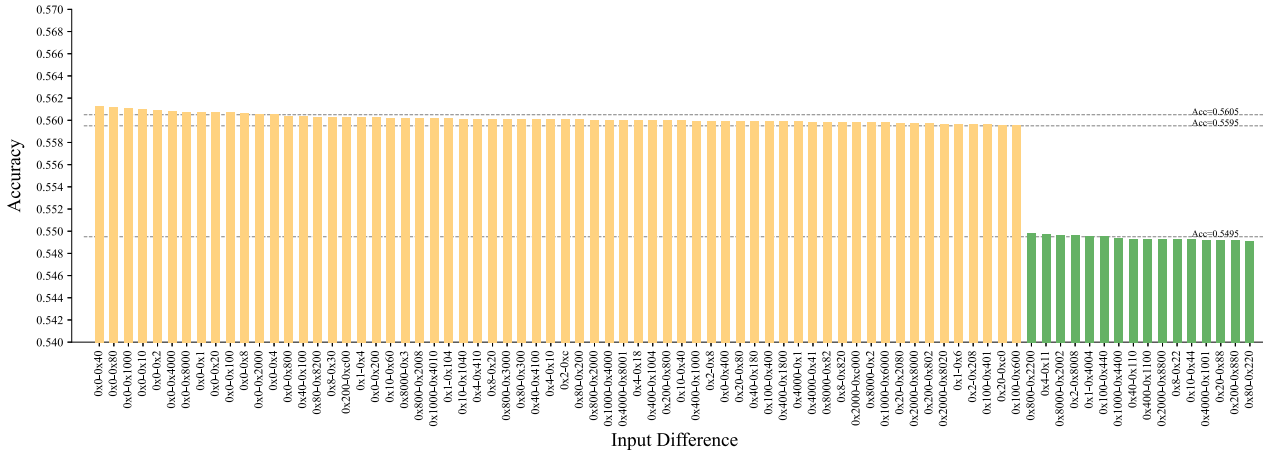

FIGURE 3: The input differences with Hamming weights of 1, 2, and 3 that can obtain a clear non-marginal advantage (accuracy above 0.52) on the 11-round ND of SIMON32/64 with 8 ciphertext pairs as a sample.

For $\mathrm{HW}(\Delta_p) = 3$, there are three sets of (a, b). Using the input difference:

(0001,0104), (0002,0208), (0004,0410), (0008,0820),
(0010,1040), (0020,2080), (0040,4100), (0080,8200),
(0100,0401), (0200,0802), (0400,1004), (0800,2008),
(1000,4010), (2000,8020), (4000,0041), (8000,0082),

can construct 11-round ND of SIMON32/64 with an accuracy of about 0.560.

Using the input difference:

(0001,0006), (0002,000c), (0004,0018), (0008,0030),
(0010,0060), (0020,00c0), (0040,0180), (0080,0300),
(0100,0600), (0200,0c00), (0400,1800), (0800,3000),
(1000,6000), (2000,c000), (4000,8001), (8000,0003),

can obtain 11-round ND of SIMON32/64 with an accuracy of about 0.560.

Using the input difference:

(0001,4004), (0002,8008), (0004,0011), (0008,0022),
(0010,0044), (0020,0088), (0040,0110), (0080,0220),
(0100,0440), (0200,0880), (0400,1100), (0800,2200),
(1000,4400), (2000,8800), (4000,1001), (8000,2002),

can get 11-round ND of SIMON32/64 with an accuracy of about 0.549.

It can be found that the effect of the 16 input differeces $(\texttt{0x0001} \lll i, \texttt{0x4004} \lll i)$ $(0 \leq i < 16)$ is slightly inferior to the other 64 $(80 - 16 = 64)$ input differeces for 11-round ND.

Then, with the input differences `(0x0,0x1)`, `(0x1,0x4)`,`(0x1,0x104)`,`(0x1,0x6)`,`(0x1,0x4004)` separately, we construct 12-round ND of SIMON32/64 by using the basic training method. The results are shown in Table 3. It shows the accuracy exceeds 0.50 except for the input difference `(0x0,0x1)` (`(0x0,0x1)` can get an accuracy of 0.5142 by using the staged training method). Therefore, a total of 64 input differences can make 12-round ND obtain non-marginal advantage by using the basic training method. Meanwhile, the input differential `(0x1,0x4)` and `(0x1,0x6)` performed the best, with an accuracy of 0.5152.

TABLE 3: Experiment with Different Input Difference of 12-round ND for SIMON32/64 with 8 ciphertext pairs as a sample.

Cipher	Input Difference	Acc	TPR	TNR
SIMON 32/64	`(0x0000,0x0001)`	0.5004	0.1149	0.8857
	`(0x0001,0x0004)`	0.5152	0.4799	0.5505
	`(0x0001,0x0104)`	0.5151	0.4901	0.5401
	`(0x0001,0x0006)`	0.5152	0.4852	0.5453
	`(0x0001,0x4004)`	0.5135	0.4331	0.5940

The Second Stage

The NDs are prepended with 3 rounds of CDs in [18], so we use 3 rounds prepended CDs as a benchmark to test the performance of the input differential filtered in the first stage. An SMT solver is used to determine the probability of prepended CDs. We first decide if a differential characteristic with probability p exists, then enumerate all differential characteristics with a probability of p. The results are presented in Table 4. It can be seen that the probability of the 3 rounds prepended CDs with the input difference $(\texttt{0x0000} \lll i, \texttt{0x0001} \lll i)$, $0 \leq i < 16$ (*i.e.*, $(\texttt{0}, e_i)$) are the highest, followed by 2-bit input differential $(\texttt{0x0001} \lll i, \texttt{0x0004} \lll i)$, $0 \leq i < 16$, and the worst are $(\texttt{0x0001} \lll i, \texttt{0x4004} \lll i)$, $0 \leq i < 16$.

As a result, after these two steps of filtering, the input difference $(\texttt{0}, e_i)$ is possibly the best option for hybrid distinguishers. Meanwhile, the input difference $(\texttt{0x0001} \lll i, \texttt{0x0004} \lll i)$, $0 \leq i < 16$ is also a good choice. But we cannot yet give a clearer opinion on how much i is set.

TABLE 4: Comparing the performance of the hybrid distinguisher with different input difference for SIMON32/64. The NDs is 11-round. The number on the arrow represents the probability of the differential characteristic from the input difference to the output difference, and the number of characteristics*.

HW(Δ_p)	Δ_p	ND's Acc	ND's TPR	ND's TNR	Prepended CDs (3-round)
1-bit	(0000,0001)	0.5607	0.5407	0.5807	(0011,0040) ... $\xrightarrow{2^{-8}(\#20)}$ (0010,0146) ... $\xrightarrow{2^{-9}(\#4)}$ (0011,0040) ... $\xrightarrow{2^{-10}(\#232)}$ (0011,0040) ... $\xrightarrow{2^{-11}(\#352)}$ ... $\rightarrow$ (0000,0001)
2-bit	(0001,0004)	0.5602	0.5059	0.6145	(0040,0111) ... $\xrightarrow{2^{-8}(\#4)}$ (0140,0511) ... $\xrightarrow{2^{-9}(\#10)}$ (1040,0101) ... $\xrightarrow{2^{-10}(\#72)}$ (1060,0101) ... $\xrightarrow{2^{-11}(\#124)}$ ... $\rightarrow$ (0001,0004)
3-bit	(0001,0104)	0.5601	0.5024	0.6179	(0040,0111) ... $\xrightarrow{2^{-10}(\#4)}$ (0140,0110) ... $\xrightarrow{2^{-11}(\#48)}$ (1040,0101) ... $\xrightarrow{2^{-12}(\#80)}$ (0200,4201) ... $\xrightarrow{2^{-13}(\#620)}$ ... $\rightarrow$ (0001,0104)
	(0001,0006)	0.5597	0.4972	0.6221	(0040,0111) ... $\xrightarrow{2^{-10}(\#4)}$ (0140,0511) ... $\xrightarrow{2^{-11}(\#8)}$ (1040,0101) ... $\xrightarrow{2^{-12}(\#72)}$ (1060,0101) ... $\xrightarrow{2^{-13}(\#296)}$ ... $\rightarrow$ (0001,0006)
	(0001,4004)	0.5495	0.4433	0.6557	(4044,0101) ... $\xrightarrow{2^{-12}(\#80)}$ (4064,0101) ... $\xrightarrow{2^{-13}(\#344)}$ (0144,0140) ... $\xrightarrow{2^{-14}(\#1072)}$ ... $\rightarrow$ (0001,4004)

* For example: (0011,0040) ... $\xrightarrow{2^{-8}(\#20)}$ (0000,0001) means that when the input differences are (0011,0040) etc. and the output difference is (0000,0001), there are 20 characteristics with a probability of 2^{-8} for 3-round SIMON32/64. And these input differences in the prepended CDs are the smallest Hamming weight in these characteristics.

5. (RELATED-KEY) DIFFERENTIAL-NEURAL DISTINGUISHERS FOR ROUND-REDUCED SIMON32/64 AND SIMON64/128

In this section, the NDs are trained using the basic training method and the staged training method. The training model is based on Section 3.

5.1. Differential-Neural Distinguishers

Simon32/64

Training using the basic scheme. Using the input difference `(0x0000,0x0040)`, we build NDs against SIMON32/64 cover to 9-, 10-, and 11-round with 0.9176, 0.6975, and 0.5609 accuracy, respectively. Using the input difference `(0x0001,0x0004)`, we build 12-round ND with 0.5152 accuracy. Table 1 presents the results.

Note that for NDs fed with single ciphertext pairs, with multiple ciphertext pairs with the same label, one can directly obtain a combine-response distinguisher (CRD) using the formula (3) in [16]. Similar to the NDs fed with multiple ciphertext pairs, the CRDs' accuracy improves quickly with increasing the number of ciphertext pairs. Therefore, we compare the accuracy of NDs with CRDs under the number of ciphertext pairs with the same label. Compared with [18], the accuracy of our NDs are improved.

Training using the Staged Training Method. We also use several stages of pre-training to train a 12-round differential-neural distinguisher for SIMON32/64. In the first stage, the best 10-round distinguisher is retained to recognize 9-round SIMON32/64 with the input difference `(0x0440,0x0100)`. The number of samples for training and for testing are 2^{25} and 2^{23}, respectively. The number of epochs is 30 and the learning rate is 10^{-4}.

In the second stage, the best network of the first stage is retained to recognize 12-round SIMON32/64 with the input difference `(0x0000,0x0040)`. For this stage, 2^{25} and 2^{23} examples are freshly generated for training and testing, respectively. The learning rate is 10^{-4} for 30 epochs.

Cyclical learning rates are also used for these training stages, the first and second stage both use a minimum learning rate of 0.0001 and a maximum of 0.001. All cycle lengths in these stages are set to 30 epochs. Eventually, the resulting ND achieves an accuracy of 0.5142.

Simon64/128

Training using the basic scheme. Based on the input difference `(0x00000000,0x00000040)`, the NDs reach 0.9181, 0.7117, 0.5722, and 0.5148 accuracy for 11-, 12-, 13-, and 14-round, respectively. As shown in Table 1, the results are summarized.

Training using the Staged Training Method. The best 14-round distinguisher for SIMON64/128 is trained using the staged training method.

In the first stage, the retained best 12-round distinguisher is trained and tested with 11-round 2^{25} and 2^{23} samples of SIMON64/128 with the input difference `(0x00000440,0x00000100)`. The number of epochs is 30 and the learning rate is 10^{-4}. The learning rate scheduler used in this stage is cyclic_lr$(30, 0.001, 0.0001)$.

Then the best network from the first stage is trained in the second stage. The number of examples for training and for testing are 2^{25} and 2^{23}, using 14-round SIMON64/128 data with the input difference `(0x00000000,0x00000040)`. This stage is done in 30 epochs with learning rate of 10^{-4}. The learning rate scheduler used in this stage is cyclic_lr$(30, 0.001, 0.0001)$. Finally, the accuracy of the resulting ND is 0.5185.

5.2. Related-key Differential-Neural Distinguishers

We use the basic training method to train the related-key differential-neural distinguishers. Based on the plaintext difference `(0x0000,0x0040)` and the key difference `(0x0000,0x0000,0x0000,0x0040)`, we enjoy 1, 0.9604, 0.6477, and 0.5262 accuracy for 10-, 11-, 12-, and 13-round RKNDs against SIMON32/64, respectively.

Based on the plaintext difference `(0x00000000,0x00000040)` and the key difference `(0x00000000,0x00000000,0x00000000,0x00000040)`, we build RKNDs cover to 12-, 13-, and 14-round with 0.9880, 0.8398, and 0.5788 accuracy for SIMON64/128, respectively. To the best of our knowledge, this is the first successful application of the RKNDs against SIMON-like ciphers.

5.3. Experiment with Different Data Format

In order to improve the accuracy of the ND, we introduce a new data format $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$ suitable for the network architecture in this paper. Here, we explain the reason for choosing this data format. We mainly compare the effect of the different data format on the performance of the network based on the experiment of 9-, 10-, and 11-round NDs for SIMON32/64.

We use the basic method to train the 9-, 10-, and 11-round NDs based the input difference `(0x0000,0x0040)`, batch size 30000, and cyclic_lr$(30, 0.003, 0.0001)$. The results are presented in Table 5.

It shows that the NDs using data formats of $(C_r, C_r', \Delta_R^{r-1})$, $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1})$, $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$ can achieve

11-round, and the accuracy with data format $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$ is greater than others. This is the primary cause for using this data format in the paper.

Meanwhile, it is noted that the accuracy dropped when the $p\Delta_R^{r-2}$ component was deleted from the data format $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$, *i.e.*, the neural network benefits from providing data $p\Delta_R^{r-2}$. In fact, $p\Delta_R^{r-2}$ denotes the partial Δ_R^{r-2}, and it can be determined without the round key when the ciphertext pair is given.

It is important to note that this comparison is only to show that the data format used in this paper better matches the current network for better performance. Different results may occur when the network is changed.

6. (RELATED-KEY) DIFFERENTIAL-NEURAL DISTINGUISHERS FOR ROUND-REDUCED SIMECK32/64 AND SIMECK64/128

SIMECK is a lightweight block cipher family that combines the good design components of SIMON and SPECK to make it even more compact and efficient. In this section, we build NDs and RKNDs for round-reduced SIMECK32/64 and SIMECK64/128.

6.1. Differential-Neural Distinguishers

Simeck32/64

Training using the basic scheme. Using the input difference `(0x0000,0x0040)`, we build NDs against SIMECK32/64 cover to 9-, 10-, and 11-round with 0.9952, 0.7354, and 0.5646 accuracy, respectively. The results are presented in Table 1.

Training using the Staged Training Method. A 12-round differential-neural distinguisher for SIMECK32/64 is also obtained by utilizing several stages of pre-training.

The first stage selects the best 10-round distinguisher to recognize 9-round SIMECK32/64 with the input difference `(0x0140,0x0080)`. Note that the most likely difference to appear three rounds after the input difference `(0x0000,0x0040)` is `(0x0140,0x0080)`, and the probability is about 2^{-4}.

It freshly generates 2^{25} and 2^{23} samples to train and test the distinguisher, respectively. This stage has 30 epochs and a learning rate of 10^{-4}. The learning rate scheduler used in this stage is cyclic_lr$(30, 0.001, 0.0001)$.

The best network obtained from the first stage is retained to recognize 12-round SIMECK32/64 with the input difference `(0x0000,0x0040)`. The number of examples for training and for testing are 2^{25} and 2^{23}, respectively. The number of epochs is 30 and the learning rate is 10^{-4}. The learning rate scheduler used in this stage is cyclic_lr$(30, 0.001, 0.0001)$. Lastly, the ND produced has an accuracy of 0.5146.

Simeck64/128

Training using the basic scheme. Similarly, based on the input difference `(0x00000000,0x00000040)`, the NDs reach accuracies of 0.9142, 0.7663, 0.6356, 0.5577, and 0.5202 for 14-, 15-, 16-, 17-, and 18-round, respectively. The results are shown in Table 1.

Training using the Staged Training Method. We use the staged training method to obtain the best 18-round distinguisher for SIMECK64/128.

In the first stage, the retained best 16-round distinguisher is trained and tested with 15-round 2^{25} and 2^{23} samples of SIMECK64/128 with the input difference `(0x0000140,0x00000080)`. The number of epochs is 30 and the learning rate is 10^{-4}.

Then the best network from the first stage is trained in the second stage. The number of freshly generated examples for training and for testing are 2^{25} and 2^{23}, using 18-round SIMECK64/128 data with the input difference `(0x00000000,0x00000040)`. This stage is done in 30 epochs with learning rate of 10^{-4}.

Cyclical learning rates are used for these training stages, the first and second stage both use a minimum learning rate of 0.0001 and a maximum of 0.001. All cycle lengths in these stages are set to 30 epochs. As a final result, the ND produced has an accuracy of 0.5218.

6.2. Related-key Differential-Neural Distinguishers

For related-key differential-neural distinguishers, based on the input difference `(0x0000,0x0040)` and the key difference `(0x0000,0x0000,0x0000,0x0040)`, it covers to 13-, 14-, and 15-round with 0.9950, 0.6679 and 0.5467 accuracy for SIMECK32/64, respectively.

For SIMECK64/128, based on the input difference `(0x00000000,0x00000040)` and the key difference `(0x00000000,0x00000000,0x00000000,0x00000040)`, it cover to 18-, 19-, 20-, 21-, and 22-round with 0.9066, 0.7558, 0.6229, 0.5519, and 0.5180 accuracy for SIMECK64/128, respectively. It can be seen the gap of RKNDs for SIMON and SIMECK is obvious, and SIMON's key-expansion algorithm offers better resistance. This is consistent with the conclusion that Lu *et al.* get using rotational-XOR cryptanalysis in [30].

7. CONCLUSION

In this paper, we provide an in-depth analysis of the (related-key) differential-neural distinguishers for SIMON and SIMECK ciphers. We adopt the multiple ciphertext pairs with data of the form $(\Delta_L^r, \Delta_R^r, C_l, C_r, C_l', C_r', \Delta_R^{r-1}, p\Delta_R^{r-2})$ fed to the neural network to improve the accuracy of the neural

TABLE 5: Experiment with different data format of 9-, 10-, and 11-round NDs for SIMON32/64. The best NDs for 9-, 10-, and 11-round are shown shaded.

Cipher	Round	Data Format	Acc	TPR	TNR	Source
SIMON 32/64	9	(C_l, C_r, C'_l, C'_r)	0.7524	0.7304	0.7743	[16]
		(Δ^r_L, Δ^r_R)	0.6895	0.6613	0.7176	[28]
		$(C_r, C'_r, \Delta^{r-1}_R)$	0.8908	0.8786	0.9031	[18]
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R)$	0.8945	0.8834	0.9057	This Paper.
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R, p\Delta^{r-2}_R)$	0.9176	0.9052	0.9299	This Paper.
	10	(C_l, C_r, C'_l, C'_r)	0.5007	0.7015	0.2989	[16]
		(Δ^r_L, Δ^r_R)	0.5605	0.5402	0.5809	[28]
		$(C_r, C'_r, \Delta^{r-1}_R)$	0.6856	0.6610	0.7102	[18]
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R)$	0.6889	0.6639	0.7139	This Paper.
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R, p\Delta^{r-2}_R)$	0.6975	0.6662	0.7287	This Paper.
	11	(C_l, C_r, C'_l, C'_r)	0.5006	0.4148	0.5863	[16]
		(Δ^r_L, Δ^r_R)	0.5007	0.8110	0.1898	[28]
		$(C_r, C'_r, \Delta^{r-1}_R)$	0.5555	0.5437	0.5673	[18]
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R)$	0.5578	0.5455	0.5700	This Paper.
		$(\Delta^r_L, \Delta^r_R, C_l, C_r, C'_l, C'_r, \Delta^{r-1}_R, p\Delta^{r-2}_R)$	0.5609	0.5366	0.5852	This Paper.

distinguisher. Meanwhile, we investigate the impact of input difference on the performance of the hybrid distinguishers to select the appropriate input difference. For SIMON32/64, SIMON64/128, SIMECK32/64 and SIMECK64/128, we construct the (related-key) differential-neural distinguishers with higher accuracy.

It is undeniable that there are many factors that can affect the performance of neural distinguishers. This paper explores its impact on the performance of neural distinguishers from the perspective of data format and input difference. In the future, we plan to further explore ways that can improve the performance of neural networks from multiple dimensions, such as using methods of feature engineering to extract more essential features of the training data and so on.

ACKNOWLEDGEMENTS

This work was supported in part by the National Key Research and Development Program of China [No.2021YFB3100800]; and the State Key Laboratory of Information Security [2020-MS-02]; and the National Natural Science Foundation of China [grant numbers 62172427, 61872379, 61702537]; and the Academy of Finland [grant number 331883]; and Postgraduate Scientific Research Innovation Project of Hunan Province [grant number CX20220016].

DATA AVAILABILITY

The data underlying this article are available in the article and in its online supplementary material.

REFERENCES

[1] Biham, E. and Shamir, A. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, **4**, 3–72.

[2] Matsui, M. Linear cryptanalysis method for des cipher. *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 386–397. Springer.

[3] Knudsen, L. and Wagner, D. Integral cryptanalysis. *International Workshop on Fast Software Encryption*, pp. 112–127. Springer.

[4] Bogdanov, A. and Rijmen, V. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Designs, codes and cryptography*, **70**, 369–383.

[5] Mouha, N., Wang, Q., Gu, D., and Preneel, B. Differential and linear cryptanalysis using mixed-integer linear programming. *International Conference on Information Security and Cryptology*, pp. 57–76. Springer.

[6] Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., and Song, L. Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des (l) and other bit-oriented block ciphers. *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 158–178. Springer.

[7] Mouha, N. and Preneel, B. A proof that the arx cipher salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, **2013**, 328.

[8] Kölbl, S., Leander, G., and Tiessen, T. Observations on the simon block cipher family. *Annual Cryptology Conference*, pp. 161–185. Springer.

[9] Minier, M., Solnon, C., and Reboul, J. Solving a symmetric key cryptographic problem with constraint programming. *ModRef 2014, Workshop of the CP 2014 Conference* 13.

[10] Gerault, D., Minier, M., and Solnon, C. Constraint programming models for chosen key differential

cryptanalysis. *International Conference on Principles and Practice of Constraint Programming*, pp. 584–601. Springer.

[11] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, **521**, 436–444.

[12] Bengio, Y., Lecun, Y., and Hinton, G. Deep learning for ai. *Communications of the ACM*, **64**, 58–65.

[13] Rivest, R. L. Cryptography and machine learning. *International Conference on the Theory and Application of Cryptology*, pp. 427–439. Springer.

[14] Maghrebi, H., Portigliatti, T., and Prouff, E. Breaking cryptographic implementations using deep learning techniques. *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pp. 3–26. Springer.

[15] Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., and Vandewalle, J. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, **1**, 293.

[16] Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. *Annual International Cryptology Conference*, pp. 150–179. Springer.

[17] Benamira, A., Gerault, D., Peyrin, T., and Tan, Q. Q. A deeper look at machine learning-based cryptanalysis. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 805–835. Springer.

[18] Bao, Z., Guo, J., Liu, M., Ma, L., and Tu, Y. Enhancing differential-neural cryptanalysis. *International Conference on the Theory and Application of Cryptology and Information Security*. Springer.

[19] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The simon and speck lightweight block ciphers. *Proceedings of the 52nd Annual Design Automation Conference*, pp. 1–6.

[20] Yang, G., Zhu, B., Suder, V., Aagaard, M. D., and Gong, G. The simeck family of lightweight block ciphers. *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 307–329. Springer.

[21] Biham, E. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, **7**, 229–246.

[22] Jakimoski, G. and Desmedt, Y. Related-key differential cryptanalysis of 192-bit key aes variants. *International Workshop on Selected Areas in Cryptography*, pp. 208–221. Springer.

[23] Ko, Y., Hong, S., Lee, W., Lee, S., and Kang, J.-S. Related key differential attacks on 27 rounds of xtea and full-round gost. *International Workshop on Fast Software Encryption*, pp. 299–316. Springer.

[24] Biryukov, A. and Nikolić, I. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 322–344. Springer.

[25] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

[26] Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141.

[27] Chen, Y., Shen, Y., Yu, H., and Yuan, S. A new neural distinguisher considering features derived from multiple ciphertext pairs. bxac019.

[28] Hou, Z., Ren, J., and Chen, S. Improve neural distinguishers of simon and speck. *Security and Communication Networks*, **2021**.

[29] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization.

[30] Lu, J., Liu, Y., Ashur, T., and Li, C. On the effect of the key-expansion algorithm in simon-like ciphers. *The Computer Journal*, **65**, 2454–2469.