A Multi-Differential Approach to Enhance Related-Key Neural Distinguishers

Xue Yuan[1] and Qichun Wang[1⋆]

School of Computer and Electronic Information
School of Artificial Intelligence, Nanjing Normal University, Nanjing, China
232202034@njnu.edu.cn, qcwang@fudan.edu.cn

Abstract. At CRYPTO 2019, Gohr pioneered the integration of differential cryptanalysis with neural networks, demonstrating significant advantages over traditional distinguishers. Subsequently, at Inscrypt 2020, Su et al. proposed the concept of constructing polyhedral differential neural distinguishers by leveraging multiple effective input differences. More recently, at FSE 2024, Bellini et al. introduced a general-purpose tool for automating the training of single-key differential neural distinguishers for various block ciphers. Inspired by this body of work, we aim to extend automated search techniques to related-key differential neural distinguishers, enabling the discovery of effective input differences and key differences for such distinguishers. To achieve this, we employ a genetic optimization algorithm to identify effective differential combinations. To validate the efficacy of our method, we apply it to the Simeck and Simon cipher families, successfully identifying effective differential combinations for the three variants of Simeck and ten variants of Simon. Furthermore, inspired by the concept of polyhedral neural distinguishers, we adopt a novel data format that leverages multiple distinct input differences and key differences to construct positive and negative samples, providing the neural network with a richer set of features. Our approach not only identify high-quality distinguishers for previously unexplored cipher variants but also achieve higher accuracy for related-key differential neural distinguishers compared to the state-of-the-art.

Keywords: Related-key · Differential neural distinguisher · Genetic optimization algorithm · Simeck cipher · Simon cipher.

1 Introduction

In resource-constrained environments, lightweight block ciphers have gained widespread application due to their ability to provide adequate security while operating efficiently on small-scale devices. The cryptanalysis of lightweight ciphers has thus emerged as a highly active area of research. Differential cryptanalysis, one of the most classical techniques in this field, was introduced by Biham and Shamir in 1991 for chosen-plaintext attacks [1]. The core concept of this method involves

⋆ Corresponding author.

analyzing how specific plaintext differences propagate to ciphertext differences. Assuming a fixed key, the process of feeding back ciphertext differences to plaintext for analysis is known as single-key differential cryptanalysis. Related-key differential cryptanalysis [2] extends the differential cryptanalysis framework by considering not only the influence of input differences on output differences but also incorporating key differences into the analytical framework. This approach is particularly suited for investigating cryptographic algorithms with key-dependent structures. The rapid advancement of artificial intelligence, particularly deep learning, has introduced transformative opportunities for cryptanalysis. By leveraging deep learning algorithms, the process of differential cryptanalysis can be automated, thereby significantly enhancing both the efficiency and precision of attacks. Deep learning excels at capturing latent patterns in encryption algorithms, demonstrating strong analytical capabilities when processing large-scale datasets. This technology not only shows extensive potential for application in traditional symmetric cryptography but also provides novel perspectives for the design and security evaluation of lightweight block ciphers.

At CRYPTO 2019, Gohr [3] proposed the use of residual networks as underlying distinguishers to analyze the Speck32/64 block cipher, successfully constructing an 8-round single-key differential neural distinguisher and performing an 11-round key recovery attack. Since this groundbreaking work, differential neural distinguishers have been widely adopted in both single-key and related-key scenarios. Subsequent research has primarily focused on explaining why neural networks are more effective at distinguishing ciphertext pairs and on enhancing the performance of neural distinguishers.At EUROCRYPT 2021, Benamira et al. [4] provided a compelling explanation of differential neural distinguishers, demonstrating that these distinguishers rely on the differential distribution of ciphertext pairs as well as the differential distributions in the penultimate and antepenultimate rounds. Numerous efforts have been made to improve the performance of neural distinguishers: studies such as [5–9] proposed structural enhancements to the neural networks, while works like [8–14] introduced improved input data formats to provide neural networks with richer features. Additionally, research in [15–17]focused on identifying optimal input differences and constructing high-quality datasets for training neural networks. We observe that most of these works concentrate on neural distinguishers in single-key scenarios, with comparatively limited research addressing related-key settings. This paper focuses on improving the performance of differential neural distinguishers in related-key scenarios, specifically for the Simon and Simeck cipher families.

Our contributions : In this study, we develop a training framework for neural distinguishers in related-key scenarios. The framework comprises five key components: differential selection, sample generation, network architecture, distinguisher training, and distinguisher evaluation. Our primary contributions focus on the differential selection and sample generation processes. For differential selection, inspired by the work of Bellini et al. [15], we enhanced the automated search tool proposed in [15] to make it applicable to related-key scenarios. This modification enables the identification of effective combinations of input and

key differences. To validate the effectiveness of these differentials, we utilize the identified differentials to generate sample datasets following the data construction methodology outlined in [14]. Additionally, we employ the same network architecture, training procedures, and evaluation metrics as in [14] and [16] for a direct performance comparison. The results demonstrate that the differentials identified by our approach provide richer and more effective features. Regarding sample generation, we further improved the data structure to enhance the performance of the distinguisher. Building on the principles of polyhedral neural distinguishers introduced in [12], we refine the data structure by using multiple distinct differentials to construct positive and negative samples. This approach minimizes the potential for confusion between positive and negative samples while providing the neural network with a broader set of useful features. Experimental results confirm that this improved data structure effectively optimizes the performance of the distinguisher, leading to increased accuracy. The detailed results are presented in Table 1.

Specifically, for Simeck32/64, the accuracy of the 14-round and 15-round distinguishers is improved from 0.730 and 0.568 to 0.802 and 0.593, respectively. Similarly, for Simeck48/96, the accuracy of the 18-round and 19-round distinguishers is increased from 0.572 and 0.523 to 0.603 and 0.533. For Simeck64/128, the accuracy of the 21-round and 22-round distinguishers is rised from 0.572 and 0.526 to 0.605 and 0.535. Additionally, for Simon32/64 (12 and 13 rounds), Simon48/96 (13 rounds), and Simon64/128 (13 and 14 rounds), the accuracy of the related-key differential neural is improved from 0.740, 0.567, 0.696, 0.916, and 0.618 to 0.788, 0.581, 0.744, 0.963, and 0.651, respectively.

Beyond these improvements, we train new distinguishers for seven additional variants of the Simon cipher. Specifically, we identify new related-key differential neural distinguishers for 17-round Simon128/256, 12-round Simon48/72, 13-round Simon64/96, 14-round Simon96/144, 16-round Simon128/192, 13-round Simon96/96, and 15-round Simon128/128. The detailed results are presented in Table 6.

Table 1: Comparison of Related-Key Differential Neural Distinguishers for Simeck and Simon Ciphers

Ciphers	Round	Accuracy	TPR	TNR	Ciphers	Round	Accuracy	TPR	TNR	Source
Simeck32/64	14r	0.668	0.643	0.693	Simon32/64	12r	0.648	0.652	0.644	[14]
	14r	0.730	0.722	0.738		12r	0.740	0.729	0.750	[16]
	14r	0.802	0.805	0.800		12r	0.788	0.782	0.794	Sect. 4
	15r	0.547	0.517	0.576		13r	0.526	0.544	0.508	[14]
	15r	0.568	0.553	0.582		13r	0.567	0.564	0.570	[16]
	15r	0.593	0.595	0.590		13r	0.581	0.573	0.589	Sect. 4
Simeck48/96	18r	0.572	0.572	0.572	Simon48/96	12r	0.997	0.998	0.996	[16]
	18r	0.603	0.628	0.578		12r	0.999	0.999	0.999	Sect. 4
	19r	0.523	0.527	0.518		13r	0.696	0.698	0.695	[16]
	19r	0.533	0.524	0.541		13r	0.744	0.748	0.741	Sect. 4
Simeck64/128	21r	0.552	0.425	0.679	Simon64/128	13r	0.840	0.839	0.841	[14]
	21r	0.572	0.580	0.563		13r	0.916	0.910	0.922	[16]
	21r	0.605	0.590	0.619		13r	0.963	0.966	0.960	Sect. 4
	22r	0.518	0.391	0.646		14r	0.579	0.589	0.568	[14]
	22r	0.526	0.523	0.529		14r	0.618	0.596	0.639	[16]
	22r	0.535	0.526	0.544		14r	0.651	0.657	0.645	Sect. 4

2 Preliminaries

2.1 Simon Cipher Overview

In 2013, the National Security Agency (NSA) proposed the Simon cipher [18], whose core design principle relies on simple bitwise operations such as XOR, AND, and circular shifts to construct an efficient encryption structure. Based on different combinations of block length and key length, Simon provides a total of ten variants, denoted by Simon $2n/mn$, where n represents half of the block size and m denotes the number of initial key words. Examples include Simon32/64, Simon48/72, Simon48/96, Simon64/96, Simon64/128, Simon96/96, Simon96/144, Simon128/128, Simon128/192, and Simon128/256. These variants mainly differ in block length and key length, thereby influencing both the initial number of key words m and the required number of rounds.

Simon employs an unbalanced Feistel network as its core structure, with a block size of $2n$ bits. The input block is divided into two n-bit halves, denoted as L_i and R_i. During each encryption round, the left half undergoes transformation through the round function $F(\cdot)$, after which the result is combined with the round key k_i using XOR or other bitwise operations. The modified left half then interacts with the right half to complete the round transformation. This iterative encryption process is mathematically expressed as follows:

$$\begin{cases} L_{i+1} = R_i, \\ R_{i+1} = L_i \oplus F\big(R_i\big) \oplus k_i. \end{cases}$$

where k_i is the ith round key and the round function $F(x)$ is defined as

$$F(x) \;=\; \big(S^1(x) \wedge S^8(x)\big) \;\oplus\; S^2(x)$$

with $S^j(x)$ representing a j-bit circular left rotation of x. The symbols $\wedge$ and $\oplus$ stand for bitwise AND and bitwise XOR, respectively.

The key expansion algorithm in Simon is likewise simple and efficient, primarily based on circular shifts, XOR operations, and a predefined constant sequence. The initial key is composed of m n-bit words, with m taking different values in different variants (for example, $m = 4$ is commonly used in Simon32/64). Denoting the initial key by $k \;=\; \big(k_{m-1},\, \ldots,\, k_1,\, k_0\big)$, the recursive relations for the key schedule can be categorized into three cases corresponding to $m = 2, 3, 4$, generally described as follows:

$$\begin{cases} k_{i+2} = c \oplus (z_j)_i \oplus k_i \oplus \Big(S^{-3}\big(k_{i+1}\big) \oplus S^{-4}\big(k_{i+1}\big)\Big), m = 2. \\ k_{i+3} = c \oplus (z_j)_i \oplus k_i \oplus \Big(S^{-3}\big(k_{i+2}\big) \oplus S^{-4}\big(k_{i+2}\big)\Big), m = 3. \\ k_{i+4} = c \oplus (z_j)_i \oplus k_i \oplus \Big[S^{-3}\big(k_{i+3}\big) \oplus k_{i+1} \oplus S^{-4}\big(k_{i+3}\big) \oplus S^{-1}\big(k_{i+1}\big)\Big], m = 4. \end{cases}$$

where c is a fixed constant, and $(z_j)_i$ is taken from a predefined constant sequence whose selection differs among variants [18]. Here, $S^{-j}(x)$ denotes a j-bit circular right shift of x. Because both the number of initial key words m and the chosen sequence vary across different Simon variants, the specific expansion rules exhibit certain nuances among them.

2.2 Simeck Cipher Overview

Proposed in 2015 by Yang et al. [19], Simeck can be viewed as a lightweight cipher that builds upon and improves the Simon structure. It integrates the Feistel framework from Simon with a round function inspired by Speck, aiming to enhance hardware efficiency while still maintaining Simon's hardware-friendly features and providing higher security. Simeck also offers multiple variants, denoted as Simeck$2n/mn$, where n represents half the block size and m stands for the number of initial key words. Currently, there are three main variants of Simeck: Simeck32/64, Simeck48/96, and Simeck64/128.

Simeck adopts a Feistel structure in the same manner as Simon, with a round function $F(\cdot)$ closely resembling that of Simon. Specifically, $F(x)$ is defined by

$$F(x) \;=\; \big(S^5(x) \wedge x\big) \;\oplus\; S^1(x)$$

Its key schedule is further simplified, aiming to reduce hardware implementation complexity; the round keys are updated through minimal operations, including XOR, circular shifts, and constant addition. Denoting the initial key by $k = (t_2, t_1, t_0, k_0)$, the iterative relations in the key expansion can be summarized as

$$\begin{cases} k_{i+1} = t_i, \\ t_{i+3} = k_i \oplus t_i \oplus S^5(t_i) \oplus S^1(t_i) \oplus c \oplus (z_j)_i. \end{cases}$$

where c is a constant, $(z_j)_i$ is a predefined sequence (see [19] for details), and $S^j(x)$ denotes a j-bit circular left rotation of x. Because all three variants of Simeck employ four n-bit words as their initial key, they share the same key expansion rules as illustrated above.

2.3 Related-Key Differential Cryptanalysis

Differential cryptanalysis [1] has been widely applied to the security evaluation of block ciphers since its inception. In differential cryptanalysis, the attacker typically considers a known plaintext difference δ_p and studies its effect on the ciphertext difference δ_c. On the other hand, related-key differential cryptanalysis [2] extends traditional differential analysis by incorporating key differences, thus enabling a more in-depth investigation of the algorithm's security. In related-key differential cryptanalysis, the attacker assumes the existence of a pair of related keys k_1 and k_2, which satisfy a specific differential relationship, i.e., $k_1 \oplus k_2 = \delta_k$. Based on this assumption, the attacker analyzes the relationship between the ciphertext differences obtained from encrypting the plaintext pair p_1 and p_2 using the keys k_1 and k_2, respectively.

Definition 1 (Related-Key Differential Characteristic) Consider a block cipher with r rounds, and let the plaintext pair (p, p') and key pair (k, k') satisfy $\delta_p = p \oplus p', \delta_k = k \oplus k'$. During the encryption process, define the output ciphertext pair of the i-th round as (c_i, c_i'), where $\delta_{c_i} = c_i \oplus c_i', 1 \leq i \leq r$. This sequence $(\delta_p, \delta_{c_1}, \delta_{c_2}, \ldots, \delta_{c_r})$ is referred to as the related-key differential characteristic of the block cipher for r rounds. Here, δ_c is the final ciphertext difference, which can be derived from $\delta_c = c \oplus c', c = E_k(p), c' = E_{k'}(p')$. Thus, the characteristic describes the propagation of plaintext and key differences to the final ciphertext difference.

Definition 2 (Related-Key Differential Probability) Given a plaintext difference δ_p, a key difference δ_k, and a target ciphertext difference δ_c, the related-key differential probability is defined as

$$\mathrm{RDP}(\delta_p, \delta_k, \delta_c) = \frac{\left|(x,k) \mid E_{k\oplus\delta_k}(x \oplus \delta_p) \oplus E_k(x) = \delta_c\right|}{2^{|p|+|k|}},$$

where $|p|$ and $|k|$ represent the bit lengths of the plaintext and key, respectively, and x runs over all possible plaintexts, while k runs over all possible keys. This probability quantifies the likelihood that, when the difference δ_p in the plaintext and the difference δ_k in the key both exist simultaneously, a ciphertext difference δ_c will result. The higher this probability, the more likely it is that the ciphertext difference δ_c will appear, enabling an attacker to perform a related-key differential cryptanalysis attack.

3 Finding Effective Input and Key Differences for New Ciphers

Currently, there are two primary approaches to selecting differentials for neural distinguishers. The first involves utilizing existing optimal differential character-

istics, while the second selects optimal differential characteristics from $n-1$ or $n-2$ rounds for an n-round neural distinguisher. Both approaches require manually identifying input differentials, yet neither effectively enhances the accuracy of the distinguisher. In [15], Bellini observed that the input differential corresponding to the best n-round differential path does not necessarily yield optimal results for neural distinguishers. Through experimental analysis, Bellini discovered that well-chosen input differentials enable high biases to persist into later rounds. Based on this observation, the author hypothesized that effective input differentials for neural distinguishers exhibit high bias scores. This hypothesis was experimentally validated, generalizing the findings of Benamira [4] to other ciphers. Using an evolutionary optimizer, Bellini [15] automated the selection of effective input differentials for various ciphers and identified favorable input differentials in related-key scenarios. However, these results do not surpass the latest or achieve superior performance.

Inspired by Bellini's work, we seek to improve the optimizer to make it applicable to related-key neural distinguishers. Our primary focus is on identifying effective combinations of input and key differentials that can improve the accuracy of neural distinguishers. According to Bellini's findings, effective differentials tend to exhibit high bias scores. We hypothesize that while Bellini's approach successfully identifies good input differentials, it fails to find optimal combinations of input and key differentials because the heuristic search process is limited to a single directional optimization, neglecting the interplay between the two types of differentials. Thus, we aim to investigate whether a relationship exists between input and key differentials. If such a relationship can be established, heuristic search methods could be employed to identify effective differential combinations. These combinations, like their individual counterparts, should also exhibit high bias scores. The definition of the bias score is presented below.

Definition 3 (Exact bias score for related-key scenario). Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ be a block cipher, $\delta_p \in \mathbb{F}_2^n$ an input difference, and $\delta_k \in \mathbb{F}_2^k$ a key difference. The exact bias score for the pair (δ_p, δ_k), denoted as $b(\delta_p, \delta_k)$, is the sum of the biases of each bit position j in the output difference, i.e.,

$$b(\delta_p, \delta_k) = \frac{1}{n} \sum_{j=0}^{n-1} \left| 0.5 - \frac{1}{2^{n+k}} \sum_{\substack{X \in \mathbb{F}_2^n \\ K \in \mathbb{F}_2^k}} \left(E_K(X) \oplus E_{K \oplus \delta_k}(X \oplus \delta_p) \right)_j \right|.$$

The computation of the exact bias score is infeasible for practical ciphers, as it involves exhaustive enumeration of all possible keys and plaintexts. Instead, an approximation can be employed, which is derived from a finite set of samples t.

Definition 4 (Bias score for related-key scenario). Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ be a block cipher, $\delta_p \in \mathbb{F}_2^n$ an input difference, and $\delta_k \in \mathbb{F}_2^k$ a key difference. The bias score for the pair (δ_p, δ_k), denoted as $\tilde{b}^t(\delta_p, \delta_k)$, is the sum of the biases of each bit position j in the output difference, computed for t samples (X_i, K_i),

i.e.,

$$\tilde{b}^t(\delta_p, \delta_k) = \frac{1}{n}\sum_{j=0}^{n-1}\left|0.5 - \frac{1}{t}\sum_{i=0}^{t-1}\left(E_{K_i}(X_i) \oplus E_{K_i \oplus \delta_k}(X_i \oplus \delta_p)\right)_j\right|.$$

Conjecture 1. Input and key difference pairs (δ_p, δ_k) that reach the most rounds with a neural distinguisher have a high bias score $b(\delta_p, \delta_k)$. We further assume that $\tilde{b}^t(\delta_p, \delta_k)$ is a good estimation of $b(\delta_p, \delta_k)$.

To investigate whether a relationship exists between input differentials and key differentials, we employ an exhaustive search algorithm to enumerate all combinations of input and key differentials with Hamming weights of 1 and 2 for the Simon32/64 cipher, resulting in a total of $(C_{32}^1 \cdot C_{64}^1 + C_{32}^1 \cdot C_{64}^2 + C_{32}^2 \cdot C_{64}^1 + C_{32}^2 \cdot C_{64}^2)$ combinations. Our analysis reveals that for differential combinations with high bias scores, the values of the input differential and the key differential are identical. Based on this observation, we hypothesize that effective differential combinations are characterized by equal input and key differentials. To validate this hypothesis, we conduct experiments to test whether constructing datasets using differential combinations where the input and key differentials are identical leads to higher accuracy in neural network training. If such combinations result in improved accuracy, our hypothesis would be supported. Using the same network architecture and evaluation methods as Lu et al. [14], we perform experiments and confirmed that datasets constructed with equal input and key differentials indeed enhance the accuracy of the distinguishers. This improvement suggests that these combinations provide the neural network with more transferable features, demonstrating the effectiveness of high-bias differential combinations where the input and key differentials are equal. Given that exhaustive search requires substantial memory and is computationally time-consuming, we propose utilizing heuristic search methods to identify effective differential combinations more efficiently.

We improve the automated search tool proposed in [15]. Since well-performing input differentials typically have a relatively small Hamming weight, we aim to generate an initial population with Hamming weights constrained within a specific range to guide the subsequent search process. During the search, the key differential is enforced to remain identical to the input differential. Following this approach, we adopt the heuristic search framework based on the genetic optimization algorithm in [15]. The algorithm proceeds as follows:

We filter the obtained results, selecting differential combinations whose bias scores deviate by no more than 0.06 from the highest bias score for subsequent experiments. In the following, we present the number of candidate differentials identified for each cipher. The detailed results are presented in Table 2.

Algorithm 1: Evolutionary optimizer

Input: L^2: length of the initial population, plain_bits: plaintext length, min_hw: minimum hamming weight, max_hw: maximum hamming weight;
plain_population ← GeneratePopulation(L^2, plain_bits, min_hw, max_hw);
key_population ← plain_population;
starting_population ← key_population and plain_population;
Sort starting_population by $\tilde{b}(\cdot)$ (descending order);
current_population ← first 32 elements of starting_population;
for *iterations* ← 0 *to* 50 **do**
    candidates ← [];
    for $i \leftarrow 0$ *to* 32 **do**
        for $j \leftarrow i + 1$ *to* 32 **do**
            $m \leftarrow 1$;
            Add $current\_population_i \oplus current\_population_j \oplus (m \ll$ RandomInt$(0, n-1))$ to candidates;
    Sort candidates by $b(\cdot)$ (descending order);
    current_population ← first 32 elements of candidates;
return *candidates*

Table 2: Total Number of Searched Differences and Effective Differences for Each Cipher

Cipher	Total	0.06-close
Simeck32/64	244	16
Simeck48/96	235	24
Simeck64/128	219	26
Simon32/64	229	32
Simon48/96	218	24
Simon64/128	217	25
Simon128/256	227	28
Simon48/72	198	23
Simon64/96	212	24
Simon96/144	178	27
Simon128/192	194	31
Simon96/96	150	28
Simon128/128	161	29

4 Related-Key Differential-Neural Distinguishers

4.1 Distinguisher Model

Differential neural distinguishers are a type of supervised model designed for the binary classification of ciphertext pairs. As such, their datasets consist of positive and negative samples, with positive samples labeled as 1 and negative

samples labeled as 0. In most scenarios, these distinguishers are used to determine whether ciphertext pairs are generated by encrypting plaintext pairs that satisfy specific input and key differentials or by random encryption. In such cases, there is a possibility of confusion between positive and negative samples. In this study, the distinguisher is primarily used to differentiate whether ciphertexts are generated by encryption with one set of input and key differential combinations or another distinct set of differential combinations. To ensure fairness in experimental comparisons, we follow the methodology of Lu et al. [14] and Wang et al. [16], using ciphertext pairs in the data format proposed in [14] to construct a dataset. Specifically, eight ciphertext pairs in the specified format are used to train the related-key differential neural distinguisher.

Basic Distinguisher Model For the target ciphers Simon and Simeck, the i-th $(1 \leq i \leq 8)$ plaintext pair $(P, P')_i$ is encrypted for r rounds to obtain the ciphertext pair $(C_l, C_r, C'_l, C'_r)_i$. Using partial decryption as described in [14], the enhanced data structure $(\Delta L^r, \Delta R^r, C_l, C_r, C'_l, C'_r, \Delta R^{r-1}, p\Delta R^{r-2})_i$ is derived and regarded as a single sample. Then each sample is assigned a label Y.

$$Y = \begin{cases} 1, & \text{if } P_i \oplus P'_i = \delta_p \text{ and } K_i \oplus K'_i = \delta_k. \\ 0, & \text{if } P_i \oplus P'_i = \mathit{Random} \text{ and } K_i \oplus K'_i = \mathit{Random}. \end{cases}$$

During model training, a related-key differential neural distinguisher is considered effective if its accuracy exceeds 0.5.

Enhanced Distinguisher Model From the basic model description, it can be observed that the enhanced data structure proposed by Lu et al. [14] already incorporates all features from the last three encryption rounds. To further improve the performance of the distinguisher, we draw inspiration from the polyhedral distinguisher construction proposed by Su et al. [12] and utilized t combinations of input and key differentials to construct positive and negative samples. This approach eliminates the possibility of positive and negative samples satisfying the same differential, while simultaneously providing the neural network with a greater number of usable features. The labels for the enhanced samples are defined as follows:

$$Y = \begin{cases} 1, & \text{if } P_i \oplus P'_i = \delta_{p_j} \text{ and } K_i \oplus K'_i = \delta_{k_j}, j \in [0, t-1]. \\ 0, & \text{if } P_i \oplus P'_i = \delta_{p'_j} \text{ and } K_i \oplus K'_i = \delta_{k'_j}, j \in [0, t-1]. \end{cases}$$

In this study, we set $t = 2$. Consequently, constructing the sample dataset requires four combinations of input and key differentials.

4.2 Description of Neural Network

We evaluate the neural network models proposed by Gohr [3], Bao [5], and Zhang [8] and determine that the following architecture achieves the highest accuracy while maintaining a reasonable balance between computational efficiency and memory overhead.

This network architecture is composed of three fundamental modules (Module 1, Module 2, and Module 3) along with an attention mechanism module (SENet Module). The input data first undergoes preprocessing in the input layer, where it is reshaped before sequentially passing through each module for feature extraction and transformation. Ultimately, the model generates classification or regression outputs. The overall architecture incorporates one-dimensional convolutional layers (Conv), batch normalization (BN), ReLU activation functions, fully connected layers (Dense), and the SENet attention mechanism (Squeeze-and-Excitation Networks) to enhance feature representation and improve the model's generalization capability.

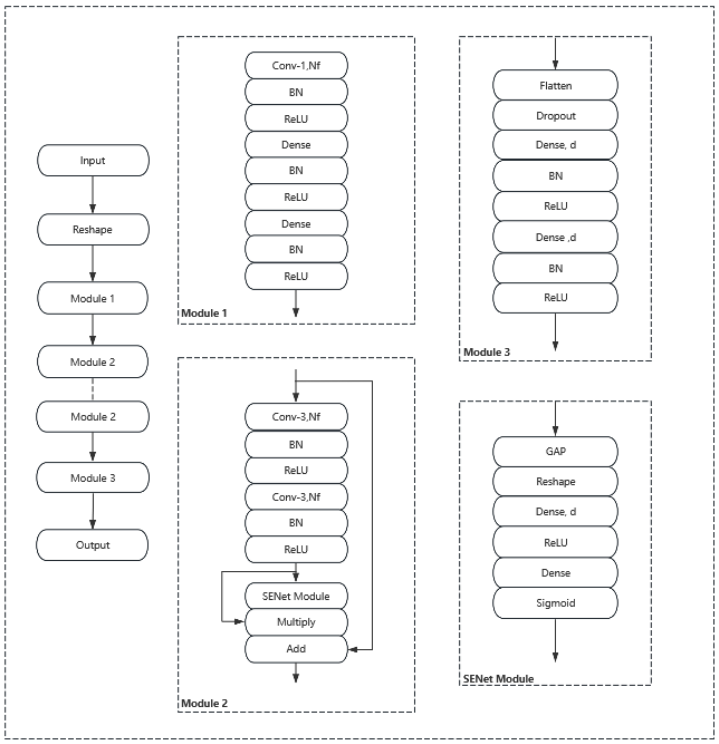

Fig. 1: Network Architecture

Input Representation. We employ eight partially decrypted and enhanced data structures as inputs to the neural network. These inputs are reshaped in the network's input layer into an $8 \times L$ matrix, where L denotes the size of the enhanced structure corresponding to the target cipher.

Initial Convolution Layer (Module 1). This module reformats the input for one-dimensional convolutional processing. A 1D convolutional layer with 64 filters ($Nf = 64$) extracts local features, followed by batch normalization and ReLU activation to stabilize training and introduce nonlinearity. Two fully connected layers, each containing d neurons, are subsequently applied, with batch normalization and ReLU activation after each layer. This module establishes a foundational feature representation, facilitating deeper extraction in later stages.

Residual Blocks (Module 2). This module stacks two 1D convolution layers of size 3×3, each with 64 filters ($Nf = 64$). Batch normalization and ReLU activation immediately follow each convolutional operation, ensuring stable training and favorable convergence in a deep architecture. A SENet module is introduced as an attention mechanism to compute channel-wise weights and perform channel-wise multiplication along with residual addition, thereby recalibrating the features. The SENet module obtains importance scores for each channel and multiplies these scores with the corresponding channels of the input feature map, emphasizing critical information. The resulting feature maps are subsequently added back to the original input, ensuring that gradients are effectively propagated as network depth increases and preventing performance degradation. By integrating residual connections with channel attention, this module adaptively focuses on pertinent channels, thereby enhancing the network's ability to capture and express key information.

Prediction Head (Module 3). This module processes high-level features for classification or regression. The feature maps are first flattened into a 1D vector before passing through fully connected layers. A dropout mechanism is introduced to mitigate overfitting, followed by two Dense layers with 128 neurons ($d = 128$), each paired with batch normalization and ReLU activation to ensure numerical stability and nonlinearity. The final output layer applies an appropriate activation function (e.g., Sigmoid or Softmax) depending on the task.

SE-Net Module. As the central attention mechanism, the SENet module adaptively reweights the importance of each channel. Global average pooling (GAP) across spatial dimensions yields a single descriptor per channel. This descriptor is fed into a Dense layer with $d = 128$ neurons and a ReLU activation, capturing channel interdependencies. A second Dense layer followed by a Sigmoid activation then maps these interdependencies into weights in $[0, 1]$. Finally, these weights are multiplied element-wise with the original feature map, and a residual connection fuses the reweighted features back to the network flow. This adaptive recalibration enhances the model's focus on critical information while mitigating irrelevant noise.

4.3 Model Training Process

The experiment consists of training both the basic differential neural distinguisher and the enhanced differential neural distinguisher. The training process

is divided into two phases: offline training and online evaluation. In the offline phase, the attacker generates the training set and validation set based on predefined plaintext differentials and key differentials, then trains the neural network. If the accuracy on the validation set exceeds 0.5, the distinguisher is considered effective in distinguishing between positive and negative samples. In the online phase, the trained distinguisher is used to analyze ciphertext data. If more than half of the samples yield a score higher than 0.5, the ciphertext is classified as originating from the block cipher; otherwise, it is considered the output of a random function. For the enhanced distinguisher training, the ciphertext is classified as satisfying the first set of differentials; otherwise, it is considered to satisfy the second set of differentials.

Parameter setting In this experiment, we use a training set of 2×10^7 samples and a validation set of 2×10^6 samples. The training process spans 30 epochs with a batch size of 30,000. To accelerate computation and reduce training time, we utilize four A800-80GB GPUs in parallel. The network is optimized using Adam [20], with Mean Squared Error (MSE) as the loss function and accuracy as the evaluation metric. To prevent overfitting, we apply L_2 regularization with a coefficient of 10^{-5}. Additionally, we adopt a cyclic learning rate strategy, which dynamically adjusts the learning rate within a specified range over training epochs. For the i-th iteration, the learning rate ℓ_i is computed as $\ell_i = a + \frac{(n-i) \bmod (n+1)}{n} \times (b-a)$, where $a = 0.0001$ is the lower bound, $b = 0.002$ is the upper bound, and $n = 10$ represents the cycle period.

Training of the Basic Differential Neural Distinguishers In the experiments, we use the data format of the Basic Distinguisher Model described in Section 4.1, and generate training and validation samples using the Linux random number generator, ensuring the randomness of the data. The data format of the Basic Distinguisher Model includes information from the last three rounds of the target cipher, providing the neural network with rich feature representations. We construct the sample dataset using the effective differentials obtained from the search in Section 3, and train the neural network. The results demonstrate that the differential neural distinguisher we develop can match or even outperform the latest results in some cases. This further validates the correctness of our hypothesis and the effectiveness of the differentials discovered. We present the superior results obtained from the training in Table 3.

Table 3: The basic related-key differential neural distinguishers for Simeck32/64, Simeck48/96, Simeck64/128, Simon32/64, Simon48/96 and Simon64/128

Ciphers	Round	Difference	Accuracy	TPR	TNR	Source
Simeck32/64	14r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.6679	0.6425	0.6933	[14]
	14r	(0x0,0x8000),(0x0,0x0,0x0,0x8000)	0.6683	0.6585	0.6875	Sect. 4
	15r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5467	0.5173	0.5762	[14]
	15r	(0x0,0x8000),(0x0,0x0,0x0,0x8000)	0.5471	0.5124	0.5818	Sect. 4
Simeck48/96	15r	(0x0,0x2000),(0x0,0x0,0x0,0x2000)	0.9362	0.9202	0.9522	Sect. 4
	16r	(0x0,0x2000),(0x0,0x0,0x0,0x2000)	0.7812	0.7288	0.8335	Sect. 4
	17r	(0x0,0x2000),(0x0,0x0,0x0,0x2000)	0.6256	0.5317	0.7192	Sect. 4
	18r	(0x0,0x2000),(0x0,0x0,0x0,0x2000)	0.5503	0.4591	0.6418	Sect. 4
	19r	(0x0,0x2000),(0x0,0x0,0x0,0x2000)	0.5158	0.4220	0.6094	Sect. 4
Simeck64/128	21r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5519	0.4248	0.6790	[14]
	21r	(0x0,0x8),(0x0,0x0,0x0,0x8)	0.5521	0.4099	0.6944	Sect. 4
	22r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5180	0.3906	0.6455	[14]
	22r	(0x0,0x8),(0x0,0x0,0x0,0x8)	0.5181	0.3875	0.6484	Sect. 4
Simon32/64	12r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.6477	0.6518	0.6435	[14]
	12r	(0x0,0x8010),(0x0,0x0,0x0,0x8010)	0.6766	0.6793	0.6738	Sect. 4
	13r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5262	0.5437	0.5081	[14]
	13r	(0x0,0x8010),(0x0,0x0,0x0,0x8010)	0.5444	0.5145	0.5742	Sect. 4
Simon48/96	11r	(0x0,0x8),(0x0,0x0,0x0,0x8)	0.9999	1	0.9999	Sect. 4
	12r	(0x0,0x8),(0x0,0x0,0x0,0x8)	0.9924	0.9993	0.9854	Sect. 4
	13r	(0x0,0x8),(0x0,0x0,0x0,0x8)	0.6435	0.6627	0.6242	Sect. 4
Simon64/128	13r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.8398	0.8398	0.8408	[14]
	13r	(0x0,0x1000),(0x0,0x0,0x0,0x1000)	0.8399	0.8412	0.8467	Sect. 4
	14r	(0x0,0x40),(0x0,0x0,0x0,0x40)	0.5788	0.5894	0.5682	[14]
	14r	(0x0,0x1000),(0x0,0x0,0x0,0x1000)	0.5789	0.5788	0.5790	Sect. 4

As shown in the table, the performance of the distinguisher trained with the new differentials and a smaller dataset for fewer rounds is comparable to that of Lu et al. [14]. Additionally, we have identify new distinguishers for the remaining seven unstudied variants of the Simon cipher. The training results are presented in Table 4.

Table 4: The basic related-key differential neural distinguishers for Simon128/256, Simon48/72, Simon64/96, Simon96/144, Simon128/192, Simon96/96 and Simon128/128

Ciphers	Round	Difference	Accuracy	TPR	TNR
Simon128/256	15r	(0x0,0x10000000),(0x0,0x0,0x0,0x10000000)	0.9942	0.9943	0.9942
	16r	(0x0,0x10000000),(0x0,0x0,0x0,0x10000000)	0.8243	0.8060	0.8427
	17r	(0x0,0x10000000),(0x0,0x0,0x0,0x10000000)	0.6009	0.5574	0.6443
Simon48/72	11r	(0x0,0x100000),(0x0,0x0,0x100000)	0.9926	0.9990	0.9863
	12r	(0x0,0x100000),(0x0,0x0,0x100000)	0.6565	0.6669	0.6460
Simon64/96	11r	(0x0,0x180),(0x0,0x0,0x180)	0.9976	0.9996	0.9956
	12r	(0x0,0x2000000),(0x0,0x0,0x2000000)	0.8381	0.8326	0.8435
	13r	(0x0,0x2000000),(0x0,0x0,0x2000000)	0.5775	0.5885	0.5666
Simon96/144	12r	(0x0,0x100),(0x0,0x0,0x100)	0.9998	0.9999	0.9998
	13r	(0x0,0x100),(0x0,0x0,0x100)	0.9361	0.9398	0.9324
	14r	(0x0,0x100),(0x0,0x0,0x100)	0.6733	0.6701	0.6764
Simon128/192	14r	(0x0,0x8000000),(0x0,0x0,0x8000000)	0.9953	0.9945	0.9960
	15r	(0x0,0x8000000),(0x0,0x0,0x8000000)	0.8239	0.8000	0.8477
	16r	(0x0,0x8000000),(0x0,0x0,0x8000000)	0.6062	0.5714	0.6411
Simon96/96	11r	(0x0,0x20),(0x0,0x20)	0.9999	0.9999	0.9998
	12r	(0x0,0x20),(0x0,0x20)	0.9709	0.9673	0.9745
	13r	(0x0,0x20),(0x0,0x20)	0.6753	0.6771	0.6735
Simon128/128	13r	(0x0,0x2),(0x0,0x2)	0.9956	0.9949	0.9963
	14r	(0x0,0x2),(0x0,0x2)	0.8389	0.8160	0.8616
	15r	(0x0,0x2),(0x0,0x2)	0.6148	0.6009	0.6286

Training of the Enhanced Differential Neural Distinguishers To further enhance the accuracy of differential neural distinguishers, we focus on optimizing the input data format to enrich the feature information that the neural network can extract, thereby improving the model's discriminative capability. Inspired by the Polyhedral Distinguisher method [12], we utilize multiple effective differentials identified in Section 3 to construct positive and negative samples in the dataset separately. This approach enables the neural network to learn a broader set of distinguishing features, thereby enhancing its adaptability to different data patterns. Specifically, we first generate the sample dataset using the effective differentials obtained through the search process and train a series of neural distinguishers. During this process, we select the top four differentials with the highest classification accuracy as the target differentials and further analyze their similarity in the feature space. We then use the two most similar differentials to construct positive samples, while the remaining two differentials are used to construct negative samples. This ensures that the feature distributions between positive and negative samples exhibit greater divergence, thereby enhancing the model's distinguishing capability. Experimental results demonstrate that this optimization strategy significantly improves the classification accuracy of the distinguisher. Compared to the traditional approach of constructing samples using a single differential, our method provides a more di-

verse set of training features, further enhancing the neural network's ability to recognize cryptographic data structures. The detailed experimental results are presented in Table 5, validating the effectiveness of our approach and offering a novel optimization strategy for neural network-based differential analysis.

Table 5: The enhanced related-key differential neural distinguishers for Simeck32/64, Simeck48/96, Simeck64/128, Simon32/64, Simon48/96 and Simon64/128

Ciphers	Round	Input Difference	Key Difference	Accuracy	TPR	TNR
Simeck32/64	14r	(0x0,0x10)	(0x0,0x0,0x0,0x0,0x10)	0.802	0.805	0.800
		(0x0,0x80)	(0x0,0x0,0x0,0x0,0x80)			
	15r	(0x0,0x2000)	(0x0,0x0,0x0,0x0,0x2000)	0.593	0.595	0.590
		(0x0,0x8000)	(0x0,0x0,0x0,0x0,0x8000)			
Simeck48/96	18r	(0x0,0x2)	(0x0,0x0,0x0,0x0,0x2)	0.603	0.628	0.578
		(0x0,0x200)	(0x0,0x0,0x0,0x0,0x200)			
	19r	(0x0,0x80000)	(0x0,0x0,0x0,0x0,0x80000)	0.533	0.524	0.541
		(0x0,0x800000)	(0x0,0x0,0x0,0x0,0x800000)			
Simeck64/128	21r	(0x0,0x8)	(0x0,0x0,0x0,0x0,0x8)	0.605	0.590	0.619
		(0x0,0x20)	(0x0,0x0,0x0,0x0,0x20)			
	22r	(0x0,0x80000)	(0x0,0x0,0x0,0x0,0x80000)	0.535	0.526	0.544
		(0x0,0x200000)	(0x0,0x0,0x0,0x0,0x200000)			
Simon32/64	12r	(0x0,0x8010)	(0x0,0x0,0x0,0x0,0x8010)	0.788	0.782	0.794
		(0x0,0x801)	(0x0,0x0,0x0,0x0,0x801)			
	13r	(0x0,0x2004)	(0x0,0x0,0x0,0x0,0x2004)	0.581	0.573	0.589
		(0x0,0x1002)	(0x0,0x0,0x0,0x0,0x1002)			
Simon48/96	12r	(0x0,0x1)	(0x0,0x0,0x0,0x0,0x1)	0.999	0.999	0.999
		(0x0,0x8)	(0x0,0x0,0x0,0x0,0x8)			
	13r	(0x0,0x200000)	(0x0,0x0,0x0,0x0,0x200000)	0.744	0.748	0.741
		(0x0,0x400000)	(0x0,0x0,0x0,0x0,0x400000)			
Simon64/128	13r	(0x0,0x1000)	(0x0,0x0,0x0,0x0,0x1000)	0.963	0.966	0.960
		(0x0,0x100000)	(0x0,0x0,0x0,0x0,0x100000)			
	14r	(0x0,0x200000)	(0x0,0x0,0x0,0x0,0x200000)	0.651	0.657	0.645
		(0x0,0x400000)	(0x0,0x0,0x0,0x0,0x400000)			

It can be observed that for the Simeck32/64 cipher, using the first two differentials from Table 5 to construct positive samples and the last two differentials to construct negative samples resulted in a 7% improvement in the accuracy of the 14-round distinguisher and a 2.5% improvement in the accuracy of the 15-round distinguisher. For the other ciphers, the related-key distinguisher accuracy also show significant improvement following this construction method. Furthermore, we achieve substantial performance enhancements for the distinguishers of the remaining seven variants of the Simon cipher. Compared to the training approach using the basic differential neural distinguisher, the accuracy of the highest-round distinguisher for each variant is improved by approximately 10%. The detailed results are presented in Table 6.

Table 6: The enganced related-key differential neural distinguishers for Simon128/256, Simon48/72, Simon64/96, Simon96/144, Simon128/192, Simon96/96 and Simon128/128

Ciphers	Round	Input Difference	Key Difference	Accuracy	TPR	TNR
Simon128/256	16r	(0x0,0x2) (0x0,0x4)	(0x0,0x0,0x0,0x0,0x2) (0x0,0x0,0x0,0x0,0x4)	0.936	0.931	0.942
	17r	(0x0,0x10000000) (0x0,0x1000000000000000)	(0x0,0x0,0x0,0x0,0x10000000) (0x0,0x0,0x0,0x0,0x1000000000000000)	0.690	0.696	0.684
Simon48/72	11r	(0x0,0x20) (0x0,0x40)	(0x0,0x0,0x20) (0x0,0x0,0x40)	0.999	0.999	0.999
	12r	(0x0,0x100000) (0x0,0x400000)	(0x0,0x0,0x100000) (0x0,0x0,0x400000)	0.755	0.759	0.751
Simon64/96	12r	(0x0,0x10000) (0x0,0x20000)	(0x0,0x0,0x10000) (0x0,0x0,0x20000)	0.956	0.954	0.957
	13r	(0x0,0x2000000) (0x0,0x8000000)	(0x0,0x0,0x2000000) (0x0,0x0,0x8000000)	0.648	0.643	0.654
Simon96/144	13r	(0x0,0x100) (0x0,0x200)	(0x0,0x0,0x100) (0x0,0x0,0x200)	0.993	0.991	0.995
	14r	(0x0,0x10000000) (0x0,0x100000000)	(0x0,0x0,0x10000000) (0x0,0x0,0x100000000)	0.776	0.751	0.800
Simon128/192	15r	(0x0,0x100) (0x0,0x400)	(0x0,0x0,0x100) (0x0,0x0,0x400)	0.928	0.913	0.943
	16r	(0x0,0x8000000) (0x0,0x40000000)	(0x0,0x0,0x8000000) (0x0,0x0,0x400000000)	0.684	0.706	0.662
Simon96/96	12r	(0x0,0x20) (0x0,0x40)	(0x0,0x20) (0x0,0x40)	0.996	0.997	0.996
	13r	(0x0,0x100000) (0x0,0x400000)	(0x0,0x100000) (0x0,0x400000)	0.770	0.802	0.738
Simon128/128	14r	(0x0,0x2) (0x0,0x8)	(0x0,0x2) (0x0,0x8)	0.951	0.943	0.958
	15r	(0x0,0x100) (0x0,0x200)	(0x0,0x100) (0x0,0x200)	0.705	0.719	0.689

5 Conclusion

In this paper, we first experimentally determined the relationship between plaintext differences and key differences necessary for constructing related-key differential neural distinguishers. Subsequently, by applying a heuristic search, we identified multiple effective differentials for all ten variants of the Simon cipher and three variants of the Simeck cipher. We then trained new distinguishers for the cipher variants that had not been previously investigated, as summarized in Table 4. Furthermore, to enhance the performance of these distinguishers, we proposed a new data construction method that employed multiple differentials to generate positive and negative samples, thereby providing the neural network with more substantial features. Compared with the results reported by Lu [14] and Wang [16], our approach significantly improved the

accuracy of related-key differential neural distinguishers, as shown in Table 1. Moreover, for several unstudied cipher variants, we further refined the distinguishers and presented the resulting performance in Table 6. Consequently, we achieved higher accuracy and extended-round distinguishers for the following ciphers: Simeck32/64, Simeck48/96, Simeck64/128, Simon32/64, Simon48/96, Simon64/128, Simon128/256, Simon48/72, Simon64/96, Simon96/144, Simon128/192, Simon96/96, and Simon128/128.

References

1. Biham E, Shamir A. Differential cryptanalysis of DES-like cryptosystems. Journal of CRYPTOLOGY, 1991, 4: 3-72.
2. Biham E. New types of cryptanalytic attacks using related keys. Journal of Cryptology, 1994, 7: 229-246.
3. Gohr A. Improving attacks on round-reduced speck32/64 using deep learning//Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39. Springer International Publishing, 2019: 150-179.
4. Benamira A, Gerault D, Peyrin T, et al. A deeper look at machine learning-based cryptanalysis//Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40. Springer International Publishing, 2021: 805-835.
5. Bao Z, Guo J, Liu M, et al. Enhancing differential-neural cryptanalysis//International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer Nature Switzerland, 2022: 318-347.
6. Lyu L, Tu Y, Zhang Y. Deep learning assisted key recovery attack for round-reduced simeck32/64//International Conference on Information Security. Cham: Springer International Publishing, 2022: 443-463.
7. Yuan X, Wang Q. Improving Differential-Neural Distinguisher For Simeck Family. Cryptology ePrint Archive, 2024.
8. Zhang L, Wang Z. Improving differential-neural cryptanalysis. Cryptology ePrint Archive, 2022.
9. Zhang L, Lu J, Wang Z, et al. Improved differential-neural cryptanalysis for round-reduced simeck32/64. Frontiers of Computer Science, 2023, 17(6): 176817.
10. Chen Y, Shen Y, Yu H, et al. A new neural distinguisher considering features derived from multiple ciphertext pairs. The Computer Journal, 2023, 66(6): 1419-1433.
11. Hou Z Z, Ren J J, Chen S Z. Improve neural distinguishers of simon and speck. Security and Communication Networks, 2021, 2021(1): 9288229.
12. Su H C, Zhu X Y, Ming D. Polytopic attack on round-reduced simon32/64 using deep learning//International Conference on Information Security and Cryptology. Cham: Springer International Publishing, 2020: 3-20.
13. Liu J S, Ren J J, Chen S Z, et al. Improved neural distinguishers with multi-round and multi-splicing construction. Journal of Information Security and Applications, 2023, 74: 103461.
14. Lu J, Liu G, Sun B, et al. Improved (related-key) differential-based neural distinguishers for SIMON and SIMECK block ciphers. The Computer Journal, 2024, 67(2): 537-547.

15. Bellini E, Gerault D, Hambitzer A, et al. A cipher-agnostic neural training pipeline with automated finding of good input differences. IACR Transactions on Symmetric Cryptology, 2023, 2023(3): 184-212.
16. Wang G, Wang G. Enhanced related-key differential neural distinguishers for SIMON and SIMECK block ciphers. PeerJ Computer Science, 2024, 10: e2566.
17. Seok B, Chang D, Lee C. A novel approach to construct a good dataset for differential-neural cryptanalysis. IEEE Transactions on Dependable and Secure Computing, 2024.
18. Beaulieu R, Shors D, Smith J, et al. The SIMON and SPECK lightweight block ciphers//Proceedings of the 52nd annual design automation conference. 2015: 1-6.
19. Yang G, Zhu B, Suder V, et al. The simeck family of lightweight block ciphers//International workshop on cryptographic hardware and embedded systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015: 307-329.
20. Kingma D P. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.