Improving Differential-Neural Distinguisher For Simeck Family

Xue Yuan* and Qichun Wang†‡

Abstract. In CRYPTO 2019, Gohr introduced the method of differential neural cryptanalysis, utilizing neural networks as the underlying distinguishers to achieve distinguishers for (5-8)-round of the Speck32/64 cipher and subsequently recovering keys for 11 and 12 rounds. Inspired by this work, we propose an enhanced neural cryptanalysis framework that combines the Efficient Channel Attention (ECA) module with residual networks. By introducing the channel attention mechanism to emphasize key features and leveraging residual networks to facilitate efficient feature extraction and gradient flow, we achieve improved performance. Additionally, we employ a new data format that combines the ciphertext and the penultimate round ciphertext as input samples, providing the distinguisher with more useful features. Compared with the known results, our work enhance the accuracy of the neural distinguishers for Simeck32/64 (10-12)-round and achieve a new 13-round distinguisher. We also improve the accuracy of the Simeck48/96 (10-11)-round distinguishers and develop new (12-16)-round neural distinguishers. Moreover, we enhance the accuracy of the Simeck64/128 (14-18)-round distinguishers and obtain a new 19-round neural distinguisher. As a result, we achieve the highest accuracy and the longest rounds distinguishers for Simeck32/64, Simeck48/96, and Simeck64/128.

Keywords: Neural-Distinguisher, Differential Cryptanalysis, Simeck Cipher, Neural Network, ECA module

1 Introduction

Differential cryptanalysis [3] is a powerful attack technique in the field of cryptography, primarily used to analyze and break symmetric cryptographic systems. By studying the differential propagation characteristics of ciphertext pairs, this method reveals potential weaknesses in encryption algorithms, thereby effectively recovering key information. Since its disclosure in the early 1990s, differential cryptanalysis has become an essential component of cryptographic re-

*School of Computer and Electronic Information/School of Artificial Intelligence, Nanjing Normal University, Nanjing, China. Email: 232202034@njnu.edu.cn

†School of Computer and Electronic Information/School of Artificial Intelligence, Nanjing Normal University, Nanjing, China. Email: qcwang@fudan.edu.cn

‡Corresponding author.

search, exerting a profound influence on the design and evaluation of modern cryptographic algorithms. The technique was first proposed by Eli Biham and Adi Shamir in 1990. The core idea of differential cryptanalysis is to determine the possible values of keys by comparing the differences between two inputs and their corresponding output differences. The fundamental principle of differential cryptanalysis is to identify specific patterns in an encryption algorithm by analyzing the relationships between input pairs (input differences) and output pairs (output differences). With the introduction of this technique, cryptographers have paid greater attention to the resistance of encryption algorithms to differential attacks, promoting the continuous development and improvement of new cryptographic algorithms.

In resource-constrained environments, lightweight block ciphers serve as a fundamental basis for ensuring data confidentiality, owing to their capability to operate efficiently on small devices while providing adequate security. In the design and analysis of lightweight block ciphers, integrating differential cryptanalysis methods enables a more comprehensive evaluation of their resistance to attacks.

With the advancement of artificial intelligence (AI), technologies such as deep learning present new opportunities for cryptanalysis. By employing deep learning algorithms, researchers can automate the differential cryptanalysis process, enhancing both the efficiency and accuracy of attacks. Deep learning can rapidly identify and learn patterns within encryption algorithms, thereby enabling effective differential analysis on large-scale datasets. This integration not only applies to traditional symmetric cryptosystems but also introduces new perspectives for the design and security assessment of lightweight block ciphers.

In CRYPTO 2019, Gohr [5] first proposed the thought of combining neural networks with differential cryptanalysis. The differential neural distinguisher, serving as a fundamental distinguisher, can differentiate whether a ciphertext is produced by encrypting plaintext that satisfies a specific input differential or by random encryption. During the training process, the differential neural distinguisher provides an accuracy metric, which, if greater than 0.5, indicates the effectiveness of the distinguisher. Gohr utilized a residual network [6] in conjunction with differential cryptanalysis [3], applying this approach to reduced-round Speck32/64 cipher, obtaining distinguishers for (5-8)-round and achieving key recovery for 11 and 12 rounds using these distinguishers.

To improve the accuracy of distinguishers, there are two main approaches. The first approach involves enhancing the network architecture by combining different networks with differential cryptanalysis. Bao et al. [1] employed DenseNet and SENet to train neural distinguishers, applying them to the Simon32/64 cipher and obtaining effective differential neural distinguishers for (7-11)-round. Inspired by GoogLeNet, Zhang et al. [14] added an Inception module, composed of multiple parallel convolutional layers with various kernel sizes, before the Residual network [6] to capture more dimensions of information. This approach was applied to the Speck32/64 and Simon32/64 ciphers, improving the accuracy of (6-8)-round distinguishers for Speck32/64, obtaining a new 9-round distinguisher, and achieving 14-round key recovery. Additionally,

it enhanced the accuracy of (9-11)-round distinguishers for Simon32/64, resulting in a new 12-round distinguisher and achieving 17-round key recovery. Lyu et al. [10] improved Gohr's framework and applied it to Simeck32/64, obtaining neural network distinguishers for (8-10)-round of Simeck32/64 and achieving key recovery attacks for (13-15)-round of Simeck32/64.

Another approach involves changing the format of the input data to the neural network. In 2023, Chen et al. [4] proposed a new data format that used multiple ciphertext pairs as input to the neural network, applying this method to the Speck32/64 cipher and improving the accuracy of its (5-7)-round neural distinguishers. Zhang et al. [13] improved the input format of neural networks for the Simeck cipher, enhancing the accuracy of (9-12)-round neural distinguishers and achieving 16-round key recovery attacks. Hou et al. [7] suggested using multiple output differential pairs as the input to the neural network rather than multiple ciphertext pairs, applying it to the Simon cipher, and enhancing the accuracy of distinguishers, achieving key recovery attacks for 13-round of Simon32/64, 14-round of Simon48/96, and 13-round of Simon64/128. Benamira et al. [2] provided an explanation for Gohr's neural distinguisher, demonstrating that the neural distinguisher's decision is based on the differences between ciphertext pairs and the differences in internal states from the penultimate round and prior rounds. Liu et al. [8] proposed a new data format utilizing characteristics from the penultimate round, experimentally verifying that the features provided to the neural network by ciphertext pairs differ from those provided by penultimate round data. This method was applied to the Speck and Simon ciphers, obtaining the highest accuracy and longest round neural distinguishers for the Speck and Simon families. Lu et al. [9] introduced a new data format for Simon and Simeck, using multiple ciphertext pairs to train SE-ResNet neural networks, resulting in (9-12)-round distinguishers for Simeck32/64.

Inspired by the aforementioned works, we conduct innovative research by modifying the input data format for the neural network and improving Gohr's network framework. Applied to the Simeck cipher, we achieve notable results.
Our contributions : The contributions are summarized as follows.

We modify Gohr's network, drawing inspiration from ECA-net [11], by incorporating an ECA layer after each residual layer. By introducing the channel attention mechanism, the model's focus on important features is enhanced, allowing it to better identify and utilize useful features, thus improving the accuracy and training efficiency of the distinguisher. We replace the ReLU activation function with the LeakyReLU activation function, which provides better feature learning capabilities during the training of differential neural network distinguishers, mitigates the vanishing gradient problem, stabilizes the training process, and enhances the robustness and generalization ability of the model.

We also modify the input data format by combining multiple sets of differential data from the penultimate round with the differential data of ciphertext pairs as a single input instance to the neural network. Applying the above modifications to the Simeck cipher, we improve the accuracy of (9-12)-round neural distinguishers for Simeck32/64, obtaining a new 13-round distinguisher. Simultaneously, we enhance the accuracy of (10-11)-round neural distinguishers

for Simeck48/96, obtaining new (12-16)-round neural distinguishers, and finally improve the accuracy of (14-18)-round neural distinguishers for Simeck64/128, obtaining a new 19-round distinguisher.

2 Preliminaries

The Simeck family of lightweight block ciphers [12] is designed to provide efficient encryption solutions for resource-constrained environments. By integrating the design principles of both the Simon and Speck ciphers, the Simeck family achieves high efficiency in both hardware and software implementations. This cipher family was introduced by Yang et al. in 2015.

Simeck ciphers [12] utilize a Feistel structure and offer three variants with different block and key sizes: 32/64, 48/96, and 64/128. The design incorporates a simple yet efficient round function primarily involving shift operations, bitwise AND operations, and addition. The number of rounds for the three variants is 32, 36, and 44. Despite the differences in block and key sizes, the encryption and decryption algorithms, as well as the key expansion algorithm, remain consistent across the variants. Below, we provide a detailed description of the encryption and key expansion algorithms.

First, the encryption algorithm of the Simeck cipher is discussed. For different variants of Simeck, the encryption algorithm differs only in the length of the plaintext, ciphertext, and key, as well as the number of rounds. The encryption process and the round function remain identical. Below is a detailed description of the encryption algorithm.

Algorithm 1 Simeck Encryption

Input:Plaintext (L,R), Key (K), Number of Rounds (N)
Output:Ciphertext (L_N, R_N)
1.Intialize:
    $L_0 \leftarrow L$
    $R_0 \leftarrow R$
2.For i=0 to N-1:
    $L_{i+1} \leftarrow R_i$
    $R_{i+1} \leftarrow L_i \oplus f(R_i) \oplus K_i$
Return (L_N, R_N)
Function f(x):
    return $(x <<< 5) \odot (x <<< 0) \oplus (x <<< 1)$

Next, the decryption algorithm of the Simeck cipher is discussed. The decryption algorithm is the inverse of the encryption algorithm. Therefore, similar to the encryption algorithm, the decryption algorithm only differs in the length of the plaintext, ciphertext, and key, as well as the number of encryption rounds for different variants. Below is a detailed description of the decryption algorithm.

Algorithm 2 Simeck Decryption

Input:Ciphertext (L_N, R_N), Key (K), Number of Rounds (N)
Output:Ciphertext (L, R)
1.Intialize:
    $L_N \leftarrow L$
    $R_N \leftarrow R$
2.For i = N-1 down to 0:
    $R_i \leftarrow L_{i+1}$
    $L_i \leftarrow R_{i+1} \oplus f(R_i) \oplus K_i$
Return (L_0, R_0)

The key expansion algorithm involves the use of round constants. For different variants of the Simeck cipher, the round constant sequences vary. We define the constant $C = 2^n - 4$, where n refers to the word size. The m-sequence z is derived using a specific method outlined in [12]. The m-sequences for Simeck32/64 and Simeck48/96 are identical, while the m-sequence for Simeck64/128 is different and requires separate derivation. The key expansion algorithm process is entirely consistent across Simeck ciphers with different block lengths. Below is a detailed description of the key expansion algorithm.

Algorithm 3 Simeck Key Expansion

Input: K: the master key
        T: the number of rounds
        C: the round constants
Output: $K_{seq} = (k_1, k_2, ..., k_T)$: the round key sequence
1.Intialize:
    $K = (t_2, t_1, t_0, k_0)$
2.Round Constant Initialization:
    $C = 2^n - 4$
    z = generate_m_sequence()
3.Generate Round Keys:
    for i = 0 to T - 1 do:
        $k_i \leftarrow k_0$
        $k_{next} \leftarrow t_0$
        $t_{next} \leftarrow k_0 \oplus f(t_0) \oplus C \oplus z[i]$
        $t_2 \leftarrow t_1$
        $t_1 \leftarrow t_0$
        $t_0 \leftarrow t_2$
        $k_0 \leftarrow t_{next}$
        end for
4.Return the set of round keys
    return K_{seq}

3 Differential-Neural Distinguishers

3.1 Distinguisher Model

A neural distinguisher is a supervised model used to determine whether a ciphertext is generated by encrypting plaintexts that satisfy a specific input differential or by encrypting random numbers. Given m pairs of plaintexts $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and the target cipher algorithm Simeck, the output differentials of the generated ciphertext pairs and the differentials of the ciphertext outputs after one round of decryption from the penultimate round $\{(\Delta C_{i,0}, \Delta C_{i,1}, \Delta C'_{i,0}, \Delta C'_{i,1}), i \in [0, m-1]\}$ are considered as a single sample. Each sample is assigned a label Y.

$$Y = \begin{cases} 1, if P_{i,0} \oplus P_{i,1} = \Delta, i \in [0, m-1] \\ 0, if P_{i,0} \oplus P_{i,1} \neq \Delta, i \in [0, m-1] \end{cases}$$

When the plaintext pairs satisfy the specific input differential, the label Y is set to 1; otherwise, it is set to 0. During the training process, if the accuracy of the neural distinguisher exceeds 0.5, the distinguisher is deemed effective.

3.2 Efficient Channel Attention (ECA) Module

In convolutional neural networks (CNNs), different channels of feature maps may contain different types of information. Effectively enhancing important features is key to improving network performance. The Squeeze-and-Excitation (SE) network was the first to propose a channel attention mechanism, using fully connected layers after global average pooling to learn the importance of each channel. However, the fully connected layers in SE networks introduce additional parameters and computational overhead. To address this issue, the Efficient Channel Attention (ECA) [11] mechanism was developed. Bao et al. [1] utilized SE-Net to train neural distinguishers and achieved promising results. Inspired by this, we attempted to combine the ECA module with residual networks to train neural distinguishers and obtained notable results.

The ECA module is an efficient channel attention mechanism that adaptively adjusts the weights of feature map channels, enhancing the network's feature representation capability and overall performance. The design philosophy of the ECA module is to avoid using fully connected layers to maintain computational efficiency while applying different weights to each channel to enhance the network's attention mechanism. In the ECA module, the input features are first subjected to global average pooling to obtain the global feature of each channel. Then, a one-dimensional convolution (Conv1D) operation is applied to the global features, with the convolution kernel size typically being an odd number (e.g., 3, 5, 7) to capture inter-channel dependencies. This step does not use fully connected layers, thereby reducing the number of parameters and computational load. Next, the Sigmoid activation function is used to convert the convolved features into channel weights. Finally, these channel weights are

multiplied element-wise with the input features on a per-channel basis to obtain the enhanced features.

The ECA module is characterized by its lightweight design, achieving efficient computation of the channel attention mechanism. Within the same layer, the ECA module collects features from different channels through global average pooling and one-dimensional convolution operations. This design not only improves the network's feature representation ability but also enhances the attention mechanism, thereby boosting overall performance.

3.3 Improving Network Architecture

In CRYPTO 2019, Gohr [5] utilized a residual network with a depth of five to capture differential characteristics between single ciphertext pairs. His network architecture employed one-dimensional convolutions with a kernel size of 1 to learn differential features at the same position within the ciphertext. Building upon Gohr's network architecture, we introduce several improvements. Specifically, we add an Efficient Channel Attention (ECA) module after the second convolution layer of each residual block to enhance the network's feature representation capability and attention mechanism, thereby improving overall performance. Additionally, we incorporate a Dropout layer after each residual block to mitigate overfitting. Training the Simeck cipher using the improved network resulted in neural distinguishers with higher accuracy. Our network architecture is illustrated in Figure 1.

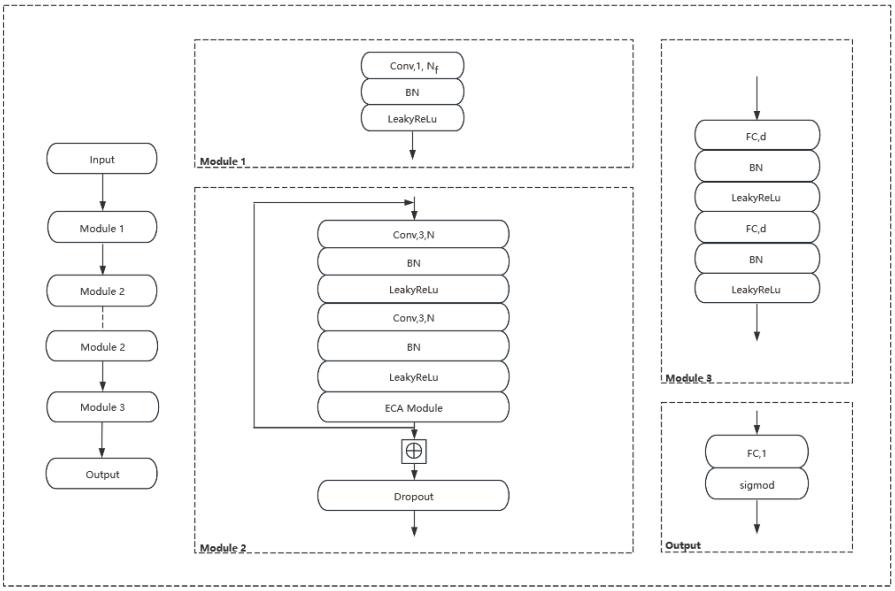

Figure 1: Improved Network Architecture

Input Representation. We use the output differentials of m ciphertext pairs and the output differentials of the penultimate round ciphertexts after one round of decryption $\{(\Delta C_{i,0}, \Delta C_{i,1}, \Delta C'_{i,0}, \Delta C'_{i,1}), i \in [0, m-1]\}$ as the input to the neural network. These sample instances are converted into a two-dimensional matrix and arranged into an $m \times 2L$ array in the input layer of the neural network, where L denotes the block size of the target cipher.

Initial Convolution Layer (Module 1). The input layer is connected to an initial convolution layer with N_f filters and a kernel size of 1. The outputs are then batch normalized to stabilize the learning process. Subsequently, the LeakyReLU nonlinear activation function (with $\alpha = 0.1$) is applied to the normalized outputs. The resulting matrix, of size $[m, 2L]$, is then passed to subsequent residual block layers to facilitate further feature extraction.

Residual Blocks (Module 2). Each residual block in the network consists of two layers, each equipped with N_f filters. The operations within each block commence with a convolution layer having a kernel size of 3. This is followed by batch normalization to enhance training stability and efficiency. The LeakyReLU layer is then applied to introduce nonlinearity. Additionally, an Efficient Channel Attention (ECA) layer is added after the second convolution layer to emphasize significant inter-channel features. At the end of each block, a skip connection directly links the ECA layer's output to the block's input, facilitating information transfer to subsequent blocks. A Dropout layer, with a dropout rate of 0.3, is added after each residual block to reduce overfitting. The network architecture includes a configurable number of residual blocks, determined by the depth parameter. In our experiments, we select a depth of 3.

Prediction Head (Module 3 and Output). The prediction head comprises two hidden layers, ultimately outputting a single unit. Each hidden layer consists of d units, followed by batch normalization and the LeakyReLU activation function. The output unit employs the Sigmoid activation function to generate the final prediction.

4 Model Training Process and Results

4.1 Training Process

Data Generation. The training and testing datasets are obtained using the Linux random number generator. This method ensures the uniform distribution of the keys K_i and the random generation of ciphertext pairs, with specified input differentials Δ and binary labels Y_i. For the r-round Simeck encryption process, when $Y_i = 1$, multiple plaintext pairs undergo r rounds of encryption. When $Y_i = 0$, the second plaintext in each pair is replaced with a newly generated random plaintext before undergoing r rounds of encryption. A subkey for

one round is then randomly generated, and the ciphertext pairs are decrypted for one round to obtain a set of penultimate round ciphertexts. Our data samples combine the output differentials of multiple ciphertext pairs and the output differentials of multiple sets of penultimate round ciphertext pairs into a single instance. Below is a detailed description of the data generation process using Simeck32/64 as an example.

Algorithm 4 Data Generation for Neural-Distinguisher

Input:

m: The number of difference pairs contained in a sample
n: Number of samples
nr: Number of encryption rounds
diff: Input differential

Output:

X: Ciphertext differences in binary format
Y: Labels (0 or 1)
ks: The one-round subkey

X ← [ ]
Y← $urandom(n) \odot 1$
keys ← urandom(8 * n).reshape(4, -1)
ks ← urandom(8).reshape(4, -1)
ks ← expand_key(ks, 1)
for i from 0 to m-1 do
    p_0 ← urandom(4 * n)
    if Y[i] == 1 then
        $p_1 \leftarrow p_0 \oplus diff$
    else
        p_1 ← urandom(4 * n)
    end if
    ks_expanded ← expand_key(keys[:, i], nr)
    c_0 ← encrypt(p_0, ks_expanded)
    c_1 ← encrypt(p_1, ks_expanded)
    c_0_prime ← dec_one_round(c_0, ks)
    c_1_prime ← dec_one_round(c_1, ks)
    $ct\_diff \leftarrow c_0 \oplus c_1$
    $ct\_prime\_diff \leftarrow c_0\_prime \oplus c_1\_prime$
    X.append($[ct\_diff, ct\_prime\_diff]$)
end for
X ← convert_to_binary(X)
return (X, Y, ks)

The generation of the validation dataset uses the same one-round subkey returned by the aforementioned algorithm, and the data generation process is similar, thus will not be repeated.

Basic Training Method. We train the neural distinguisher for 20 epochs using a training dataset of size N and a validation dataset of size M, with a batch size of bs. For Simeck32/64, Simeck48/96, and Simeck64/128, we use input differences of $(0x0000, 0x0040)$, $(0x000000, 0x000040)$, and $(0x00000000, 0x00000040)$, respectively, to train the distinguisher. We employ the adaptive optimization algorithm Adam, with L2 weight regularization parameter $c = 10^{-5}$, to optimize the mean square error loss with a small penalty. A learning rate scheduler is used to implement cyclical learning rate adjustments, setting the learning rate l_i for epoch i as $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n} \cdot (\beta - \alpha)$, where $\alpha = 10^{-4}$, $\beta = 2 \times 10^{-3}$, and $n = 9$. After training, the model file is saved for subsequent evaluation.

Staged Training Method. The staged pre-training method is used to train differential neural distinguishers for (12-13)-round for Simeck32/64. Initially, we use a 10-round distinguisher to identify 7-round Simeck32/64 with input differences $(0x0140, 0x0080)$, which are the most likely three-round differences after the initial difference $(0x0040, 0x0000)$. Subsequently, we train a distinguisher to identify 12-round Simeck32/64 with input differences $(0x0040, 0x0000)$, processing 10^7 new instances over 10 epochs using the same cyclical learning rate schedule as in basic training. After the second stage, the model continue to process another 10^7 instances over 10 epochs. The 13-round distinguisher is trained similarly, with the primary change being the use of an 11-round distinguisher to identify 8-round Simeck32/64 with input differences $(0x0040, 0x0000)$.

The staged pre-training method is also applied to train (14-16)-round differential neural distinguishers for Simeck48/96. Initially, we use a 13-round distinguisher to identify 10-round Simeck48/96 with input differences $(0x000140, 0x000080)$. Subsequently, we train a distinguisher to identify 14-round Simeck48/96 with input differences $(0x000040, 0x000000)$, processing 10^7 new instances over 10 epochs using the same cyclical learning rate schedule as in basic training. After the second stage, the model continue to process another 10^7 instances over 10 epochs. The (15-16)-round distinguishers are trained similarly.

The staged pre-training method is further applied to train (18-19)-round differential neural distinguishers for Simeck64/128. Initially, we use a 16-round distinguisher to identify 13-round Simeck48/96 with input differences $(0x00000140, 0x00000080)$. Subsequently, we train a distinguisher to identify 18-round Simeck48/96 with input differences $(0x00000040, 0x00000000)$, processing 10^7 new instances over 10 epochs using the same cyclical learning rate schedule as in basic training. After the second stage, the model continue to process another 10^7 instances over 10 epochs. The 19-round distinguisher is trained similarly.

Model and Training Parameter. An important parameter for differential neural distinguishers is the number m of ciphertext pair differences and penultimate round ciphertext pair differences contained in a sample. For different variants of Simeck, we select different values for m. Table 1 lists the parameters related to the network architecture and the training process of the differential neural distinguishers.

Table 1: Related parameter for differential-neural distinguishers

Simeck32/64	m=64	$N_f = 64$	d=128			
Simeck48/96	m=48	$N_f = 128$	d=256	bs=5000	$N = 10^7$	$M = 10^6$
Simeck64/128	m=32	$N_f = 128$	d=256			

4.2 Results

Below are our experimental results. First, for the Simeck32/64 cipher, we compare our results with those of Lu et al. [9] and Zhang et al. [13]. Our training results are superior, enhancing the accuracy of the neural distinguishers for rounds (10-12) of Simeck32/64. Additionally, we obtain a new 13-round distinguisher using the staged training method.

Table 2: Accuracy of distinguisher for Simeck32/64

Ciphers	Attack Model	Round	Accuracy	TPR	TNR	Source
Simeck32/64	ND	10	0.7354	0.7207	0.7501	[9]
		10	0.7371	0.7165	0.7525	[13]
		10	0.9319	0.8846	0.9676	Sect.4
		11	0.5609	0.5366	0.5852	[9]
		11	0.5666*	0.5441	0.5895	[13]
		11	0.6676	0.6195	0.7157	Sect.4
		12	0.5152*	0.4799	0.5505	[9]
		12	0.5161*	0.4807	0.5504	[13]
		12	0.5429	0.4771	0.6042	Sect.4
		13	0.5074*	0.3457	0.6631	Sect.4

Next, for the Simeck48/96 cipher, we compare our results with those of Hou et al. [7]. We improve the accuracy of the distinguishers for (10-11)-round and obtain new neural distinguishers for (12-16)-round.

Table 3: Accuracy of distinguisher for Simeck48/96

Ciphers	Attack Model	Round	Accuracy	TPR	TNR	Source
Simeck48/96	ND	10	0.5789	—	—	[7]
		10	0.9987	0.9964	0.9999	Sect.4
		11	0.8140	—	—	[7]
		11	0.9834	0.9676	0.9999	Sect.4
		12	0.8955	0.8320	0.9551	Sect.4
		13	0.7138	0.5167	0.8859	Sect.4
		14	0.5921*	0.4727	0.7132	Sect.4
		15	0.5316*	0.3971	0.6622	Sect.4
		16	0.5074*	0.1359	0.8689	Sect.4

Finally, for the Simeck64/128 cipher, we compare our results with those of Lu et al. [9]. We improve the accuracy of the distinguishers for (14-18)-round and obtain a new 19-round neural distinguisher.

Table 4: Accuracy of distinguisher for Simeck64/128

Ciphers	Attack Model	Round	Accuracy	TPR	TNR	Source
Simeck64/128	ND	14	0.9142	0.8914	0.9371	[9]
		14	0.9916	0.9928	0.9914	Sect.4
		15	0.7663	0.6981	0.8345	[9]
		15	0.9176	0.8736	0.9590	Sect.4
		16	0.6356	0.5245	0.7467	[9]
		16	0.7548	0.6595	0.8452	Sect.4
		17	0.5577	0.4301	0.6853	[9]
		17	0.6095	0.5111	0.7058	Sect.4
		18	0.5202	0.3917	0.6486	[9]
		18	0.5256	0.4254	0.6607	Sect.4
		18	0.5218*	0.3927	0.6510	[9]
		18	0.5476*	0.4534	0.6384	Sect.4
		19	0.5160*	0.3201	0.7055	Sect.4

We obtain favorable training results with an error margin of± 0.005 . The results presented are the best from multiple training sessions. *: The staged training method is used to train ND.

5 Conclusion

In this paper, we employed an improved network architecture to train the neural distinguishers, integrating the ECA module with a residual network and introducing a channel attention mechanism. This enhancement improved the model's feature recognition and generalization capabilities. Moreover, we adopted a new data format that combines the ciphertext with the penultimate round ciphertext as sample inputs, providing the distinguisher with more usable features. Consequently, we improved the accuracy of the distinguishers and developed longer round distinguishers. We achieved the highest accuracy and the longest rounds distinguishers for Simeck32/64, Simeck48/96, and Simeck64/128 ciphers.

Funding

This work was supported by the National Natural Science Foundation of China (Grant 62172230) and Natural Science Foundation of Jiangsu Province (No. BK20201369).

Conflict of Interest

The authors declare that there are no conflict of interests, we do not have any possible conflicts of interest.

Data availability statement

The data for this research will be available upon request.

References

[1] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Enhancing differential-neural cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 318–347. Springer, 2022.

[2] Adrien Benamira, David Gerault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In *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*, pages 805–835. Springer, 2021.

[3] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4:3–72, 1991.

[4] Yi Chen, Yantian Shen, Hongbo Yu, and Sitong Yuan. A new neural distinguisher considering features derived from multiple ciphertext pairs. *The Computer Journal*, 66(6):1419–1433, 2023.

[5] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 150–179. Springer, 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Zezhou Hou, Jiongjiong Ren, and Shaozhen Chen. Improve neural distinguisher for cryptanalysis. *Cryptology ePrint Archive*, 2021.

[8] JiaShuo Liu, JiongJiong Ren, ShaoZhen Chen, and ManMan Li. Improved neural distinguishers with multi-round and multi-splicing construction. *Journal of Information Security and Applications*, 74:103461, 2023.

[9] Jinyu Lu, Guoqiang Liu, Bing Sun, Chao Li, and Li Liu. Improved (related-key) differential-based neural distinguishers for simon and simeck block ciphers. *The Computer Journal*, 67(2):537–547, 2024.

[10] Lijun Lyu, Yi Tu, and Yingjie Zhang. Deep learning assisted key recovery attack for round-reduced simeck32/64. In *International Conference on Information Security*, pages 443–463. Springer, 2022.

[11] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11534–11542, 2020.

[12] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In *International workshop on cryptographic hardware and embedded systems*, pages 307–329. Springer, 2015.

[13] Liu Zhang, Jinyu Lu, Zilong Wang, and Chao Li. Improved differential-neural cryptanalysis for round-reduced simeck32/64. *Frontiers of Computer Science*, 17(6):176817, 2023.

[14] Liu Zhang, Zilong Wang, et al. Improving differential-neural cryptanalysis. *Cryptology ePrint Archive*, 2022.