IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–9. DOI:XXXXXXXX

Improving Differential-Neural Distinguisher Model For DES, Chaskey and PRESENT

Liu Zhang[1] and Zilong Wang[1]

School of Cyber Engineering, Xidian University, Xi'an, China liuzhang@stu.xidian.edu.cn zlwang@xidian.edu.cn

Abstract. In CRYPTO'19, Gohr proposed a new cryptanalysis strategy using machine learning algorithms. Combining the differential-neural distinguisher with a differential path and integrating the advanced key recovery procedure, Gohr achieved a 12-round key recovery attack on SPECK32/64. Chen and Yu improved prediction accuracy of differential-neural distinguisher considering derived features from multiple-ciphertext pairs instead of single-ciphertext pairs.
By modifying the kernel size of initial convolutional layer to capture more dimensional information, the prediction accuracy of differential-neural distinguisher can be improved for for three reduced symmetric ciphers. For DES, we improve the prediction accuracy of (5-6)-round differential-neural distinguisher and train a new 7-round differential-neural distinguisher. For Chaskey, we improve the prediction accuracy of (3-4)-round differential-neural distinguisher. For PRESENT, we improve the prediction accuracy of (6-7)-round differential-neural distinguisher. [1]

Keywords: Differential-Neural Distinguisher · Inception Blocks · DES · Chaskey · PRESENT

1 Introduction

Classic differential cryptanalysis is a chosen-plaintext attack, which distinguishes ciphertext from random numbers by studying the probability propagation characteristics of specific plaintext differential values in the encryption process, then carries out the key recovery attacks based on differential distinguisher. The key to classic differential cryptanalysis is to search for a differential distinguisher with a high probability. Differential-neural cryptanalysis is proposed by Gohr [Goh19] based on classic differential cryptanalysis, using differential-neural distinguisher instead of differential distinguisher. The differential-neural distinguisher and differential distinguisher have the same role in distinguishing ciphertext from random numbers. A neural network trains the differential-neural distinguisher as the underlying differential distinguisher, and Bayesian search is used to speed up key recovery attacks. If the prediction accuracy of the differential-neural distinguisher is more significantly than 0.5, it is considered an effective distinguisher.

Gohr [Goh19] showed that the residual network (ResNet) [HZRS16] (previously applied in image recognition) could be trained to capture the non-randomness of the distribution of values of output pairs when the input pairs of round-reduced SPECK32/64 are of specific difference. As a result, (5-8)-round (effective) differential-neural distinguishers are trained successfully, and (11-12)-round key recovery attacks for SPECK32/64 were achieved by combining 2 rounds of the differential path. In order to launch more rounds of key recovery attacks, better differential-neural distinguishers were also studied recently. Chen

[1]The source codes are available in https://drive.google.com/drive/folders/1iORciZlGZsEpCyW-wQAy7zzJeOLJNWqL?usp=sharing.

and Yu [CY21] proposed multiple-ciphertext pairs instead of single-ciphertext pairs (in Gohr's work) as the input of the neural network. They improved the prediction accuracy of the (5-7)-round differential-neural distinguisher of Speck32/64 to a certain extent. Bao *et al.* [BGL+21] used Dense Network (DenseNet) [HLvdMW17] and Squeeze-and-Excitation Network (SENet) [HSS18] with existing deep architectures to train neural network, and obtained (7-11)-round differential-neural distinguisher and devised a 16-round key recovery attack for Simon32/64. Zhang et al. [zWW22] borrowed the idea of the Inception block of GoogLeNet to construct the new neural network architecture. Thus, they trained the differential-neural distinguisher for (5-9)-rounds Speck32/64 and (7-12)-rounds Simon32/64. In EUROCRYPT 2021, Benamira [BGPT21] indicated that Gohr's differential-neural distinguisher builds a good approximation of the differential distribution table of the cipher during the learning phase and learns additional information. Based on the principle that the purpose of the differential-neural distinguisher is to obtain the difference information in the ciphertext, we have done some tentative work to train a better differential-neural distinguisher on multiple ciphers in this paper. The main improvements for differential-neural distinguisher are listed as follows.

Our Contributions. We modify the network architecture to train differential-neural distinguisher for three reduced symmetric ciphers in this paper. Compared to Gohr's [Goh19] and Chen's distinguisher [CY21], we improve the prediction accuracy of differential-neural distinguisher for DES, Chaskey, and PRESENT and obtain a more round differential-neural distinguisher for DES under different group size m.

The rest of the paper is organized as follows. Section 2 introduces the network architecture and train process of our differential-neural distinguisher. Section 3 exhibits the prediction accuracy of our differential-neural distinguisher for three reduced symmetric ciphers. Our work is summarized in Section 4.

2 Our Differential-Neural Distinguisher Model

Gohr [Goh19] proposed the method of differential-neural cryptanalysis based on classic differential cryptanalysis, where a differential-neural distinguisher is trained using a neural network. The differential-neural distinguisher is a one-to-many differential path compared to classic differential cryptanalysis. The input difference of plaintext is identical, but the output difference of ciphertext is different in differential-neural cryptanalysis. The role of the differential-neural distinguisher is to learn the differential information in the ciphertext.

2.1 Theoretical Model Function

The differential-neural distinguisher is a supervised model to distinguish ciphertext and random numbers. Therefore, it is necessary to construct ciphertext and random numbers artificially, thereby assigning corresponding labels. Gohr [Goh19] took a single-ciphertext pair as input to the model function, and Chen et al. [CY21]generalized it, using multiple-ciphertext pairs as input to the model function. For brevity in the description, two model functions are represented by an expression.

Given m plaintext pairs $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and target cipher, the resulting ciphertext pairs $\{(C_{i,0}, C_{i,1}), i \in [0, m-1]\}$ is regarded as a sample. Each sample will be attached with a label Y:

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

If Y is 1, this sample is sampled from the target distribution and defined as a positive example. Otherwise, this sample is sampled from a uniform distribution and defined as a negative example. To guarantee the prediction accuracy of differential-neural distinguisher, a large number of samples need to be put into neural network training. If the neural network can obtain a stable prediction accuracy higher than 0.5 on a test set, it can effectively distinguish ciphertext and random numbers. The theoretical model function can be described as:

$$\Pr(Y = 1 \mid X_0, \dots, X_{m-1}) = F(f(X_0), \dots, f(X_{m-1}), \varphi(f(X_0), \dots, f(X_{m-1})))$$
$$X_i = (C_{i,0}, C_{i,1}), i \in [0, m-1]$$
$$\Pr(Y = 1 \mid X_0, \dots, X_{m-1}) \in [0, 1]$$

where $f(X_i)$ represents the basic features of a ciphertext pair X_i and $\varphi(\cdot)$ is the derived features obtained from $f(X_i)$ and $F(\cdot)$ is the new posterior probability estimation function. In Gohr's model, the value of m is 1 [Goh19]. In Chen's model, the value of m is $\{2, 4, 8, 16\}$ [CY21].

2.2 Design the Network Architecture

The differential-neural distinguisher is a posterior probability estimation function that evaluates the quality of the distinguisher with prediction accuracy. Training a differential-neural distinguisher using a neural network is to capture differential information in the ciphertext and unknown information between multiple-ciphertext pairs. The network architecture of Gohr's [Goh19] and Chen's [CY21] model mainly includes an initial convolutional layer consisting of width-1 convolutional layers and multiple residual blocks. Zhang et al. [zWW22] modified the initial convolutional layer using the Inception block instead of the width-1 convolutional layer. According to theoretical derivation and experiment findings, the following network architecture can ensure the prediction accuracy of the distinguisher to the greatest extent. The network architecture contains several modules that are described in Figure 1.

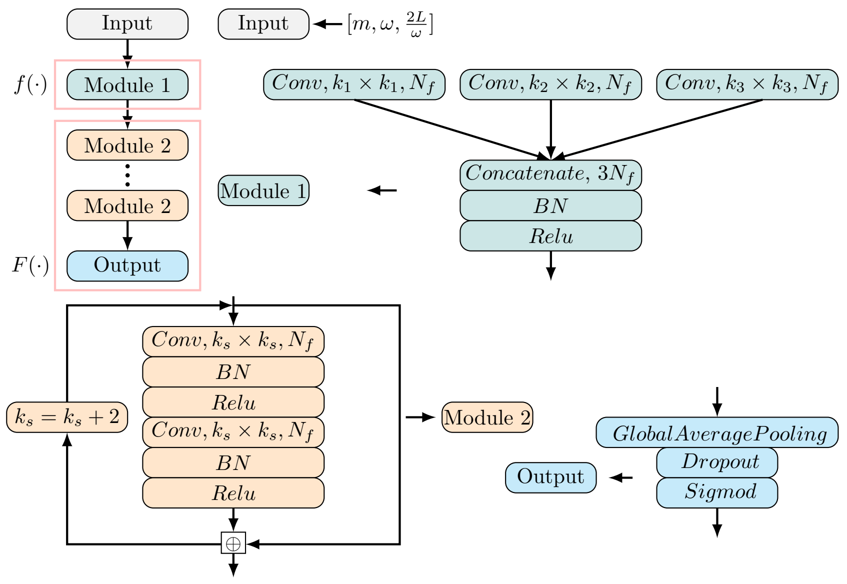


Figure 1: The network architecture of our differential-neural distinguisher model.

Input Module: Data Format. The neural network receives m ciphertext pairs $\{(C_{i,0}, C_{i,1}) \mid i \in (0, m)\}$ as input data. We convert a ciphertext pair into a two-dimensional matrix based on the word size of the target cipher. The input layer of the neural network consisting of multiple-ciphertext pairs is arranged in a $m \times \omega \times \frac{2L}{\omega}$ array, where L represents the block size of the target cipher, and ω is the size of a basic unit. If the target cipher belongs to the Feistel structure, ω is usually 4. The generation method and arrangement structure of the input data are shown in Figure 2.

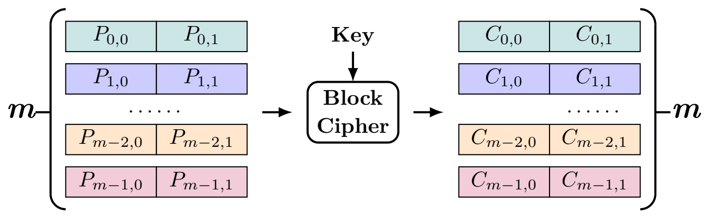
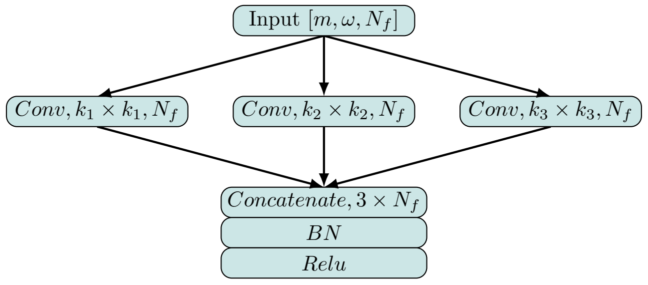


Figure 2: The arrangement structure of input data

Module 1: Initial Convolution. After converting the initial ciphertext data to a specific format, the train data enters the initial convolutional layer. The input layer is connected to the initial convolutional layer, which comprises three convolution layers with N_f channels of different kernel sizes (k_1, k_2, k_3), where ideas come from the Inception block of GoogLeNet. The three convolution layers are concatenated at the channel dimension. Batch normalization is applied to the output of concatenate layers. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[m, \omega, 3 \times N_f]$ matrix is passed to the Convolutional Blocks layer. The architecture of the initial convolutional layer can be seen in Figure 3.



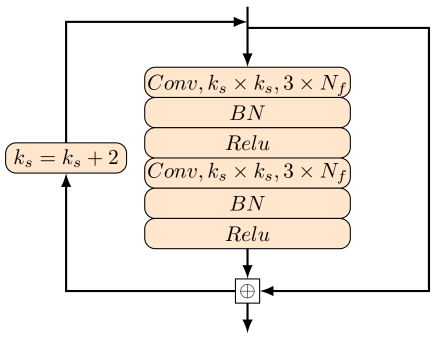
Input $[m, \omega, N_f]$
$Conv, k_1 \times k_1, N_f$
$Conv, k_2 \times k_2, N_f$
$Conv, k_3 \times k_3, N_f$
$Concatenate, 3 \times N_f$
BN
$Relu$

Figure 3: The initial convolution layer

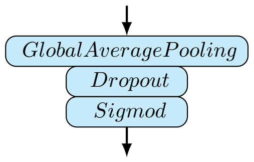
Module 2: Convolutional Blocks. Each convolutional block consists of two convolutional layers of $3 \times N_f$ filters. Each block applies first the convolution of kernel size k_s, then a batch normalization, and finally a rectifier layer. At the end of the convolutional block, a skip connection is added to the output of the final rectifier layer of the block to the input of the convolutional block and passes the result to the next block. After each convolutional block, the kernel size increases by 2. The amount of convolutional blocks is determined by experiment. The architecture of convolutional blocks layer can be seen in Figure 4(a).

Output Module: Prediction Head. The prediction head consists of a GlobalAveragePooling layer and an output unit using a *Sigmoid* activation function. We add a dropout layer (the drop rate is set to 0.8) before the *Sigmoid* activation function to prevent model overfitting. The structure of the prediction head is shown in Figure 4(b).

Rationale. First, to make it easier for the neural network to capture the differential information of the ciphertext pair, we convert the ciphertext vector into matrices in the input module. Using multiple-ciphertext pairs with the same distribution as the input of the neural network can significantly reduce the influence of a single misjudgment on the whole result. Second, the design idea for the initial convolutional layer comes mainly from


$Conv, k_s \times k_s, 3 \times N_f$
BN
$Relu$
$Conv, k_s \times k_s, 3 \times N_f$
BN
$Relu$
$k_s = k_s + 2$
$\oplus$

(a) The convolutional blocks



(b) The prediction head

the Inception block [SLJ+15] in GoogLeNet to capture more dimensional information. In the initial convolutional layer, the size of kernel is k_1, k_2, k_3 separately. In general, we set k_1 to 1. Using the initial width-1 convolutional layer is intended to make learning simple bit-sliced functions easier, such as bitwise addition. To capture the features of the internal architecture of the cryptographic algorithm, we add the convolution operation of widths k_2 and k_3 to capture features under different dimensions, such as the circular shift operation, the modular addition operation. Third, to ensure that the cryptographic algorithm encrypts different rounds without modifying the network architecture, we use a residual network to let the network automatically adjust the model parameters. In order to capture information in a larger dimension, we modify the size of the residual network convolution kernel in Gohr's model to keep it incrementing 2. Finally, to prevent the problem of overfitting caused by too few sample sizes, we added a dropout layer. The essence of using a deep residual network to construct a differential distinguisher is to treat the cipher with a nonlinear round function as a complex function and use multiple residual blocks to fit the function.

2.3 Model Training Process

Data Generation. Training and test data were generated by using the Linux random number generator to obtain uniformly distributed keys K_i and plaintext pairs P_i with the input difference Δ as well as a vector of binary-valued real/random labels Y_i. During producing training or test data for the target cipher, the plaintext pair P_i was then encrypted for R rounds if $Y_i = 1$, while otherwise, the second plaintext of the pairs was replaced with a freshly generated random plaintext and then encrypted for R rounds. In this way, training data set $N = 10^7$ and test data set $M = 10^6$ samples were generated for training and testing.

Basic Training Scheme. We run the training for 20 epochs (denoted by B_s) on the dataset of size 10^7. In order to maximize GPU performance, the batch size (denoted by E_s) processed by the dataset is adjusted according to the parameter m. The last 10^6 sample was withheld for the test. Optimization was performed against mean square error loss plus a small penalty based on L2 weights regularization parameter $\lambda = 10^{-5}$ using the Adam algorithm [KB15]. A cyclic learning rate schedule was used, setting the learning rate l_i for epoch i to $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n}.(\beta - \alpha)$ with $\alpha = 10^{-4}, \beta = 2 \times 10^{-3}$ and $n = 9$. The networks obtained at the end of each epoch were stored, and the best network by validation loss was evaluated against a test set.

Staged Train Method. When the number of encryption rounds is large, the basic training scheme described above fails, i.e., the model does not learn to approximate any helpful function. The staged train method divides the training process of the differential-neural distinguisher into multiple stages. In [Goh19], Gohr trained an 8-round distinguisher of Speck32/64 by using the staged train method. For more detailed method details, refer to [Goh19].

3 The Experiment Result

The prediction accuracy is the essential indicator that reflects the performance of the differential-neural distinguisher. For a cipher reduced to R rounds, a specific plaintext differential is set firstly. Then a training set and test set are randomly generated. After sufficient training, we will obtain the testing accuracy of the obtained new differential-neural distinguisher. A key parameter of our differential-neural distinguisher is the number of ciphertext pairs: the group size m, which has four options $\{2, 4, 8, 16\}$. Other parameters related to the training and the network architecture of our differential-neural distinguisher are listed in Table 1.

Table 1: Related parameter for training differential-neural distinguishers

$N_f = 32$	$k_s = 3$	$B_s = 1000$	$\lambda = 10^{-5}$
$l_i \in [0.02, 0.001]$	$E_s = 20$	$N = 10^7$	$M = 10^6$

The baseline distinguisher, abbreviated as BD, is reproduced by Chen et al. [CY21] according to the network architecture of Gohr [Goh19]. The differential-neural distinguisher of Chen, abbreviated as $MCND$, is trained by using multiple-ciphertext pairs instead of single-ciphertext pairs as the input of the neural network in [CY21]. According to the network architecture in Section 2, we carried out two sets of experiments. The $Case_1$ is an experiment using $N = 10^7$ samples to train and $M = 10^6$ samples to test differential-neural distinguisher. Also, the $Case_2$ is an experiment using $N = 10^7 \times m$ samples to train and $M = 10^6 \times m$ samples to test differential-neural distinguisher. Meanwhile, we removed the Dropout layer in the $Case_2$. According to the structure of the input module in the neural network, the number of multiple-ciphertext pairs in the training set ans test is $\frac{N}{m}$ and $\frac{M}{m}$, respectively. In $Case_1$, when the value of m is relatively large, the number of samples in the test set will be relatively small, resulting in overfitting, which is why experiment 2 is carried out.

3.1 Experiments on DES

Differential-Neural Distinguishers for Reduced DES: DES [How87] is a block cipher that is built on a 6×4 Sbox. Based on the analysis of DES in [BS93], the plaintext difference adopted in this paper is $\alpha = (0x40080000, 0x04000000)$ and the baseline distinguishers were built for reduced DES firstly [Goh19]. Our differential-neural distinguishers are built for DES reduced to 5, 6, and 7 rounds. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 4, 6)$. The penalty factor is increased to 8×10^{-4}. Other related parameters are the same as Tabel 1. Corresponding distinguisher prediction accuracy is shown in Table 2.

In Table 2, we can see that the distinguisher BD has effectively distinguished ciphertext and random number when the reduced rounds $R = 5$. Compared to the prediction accuracy of the distinguisher $MCND$ [CY21], we cannot significantly improve the accuracy of the differential-neural distinguisher where $R = 5$. When $R = 6$, the distinguisher $MCND$ cannot significantly improve the prediction accuracy, even if the group size m is constantly increased. Both in the case of $Case_1$ and $Case_2$, our differential-neural distinguisher significantly increase the prediction accuracy. Meanwhile, we trained successfully the

Table 2: Accuracy of differential-neural distinguisher for DES

	R	BD [Goh19]	m=2	m=4	m=8	m=16
$MCND$[CY21]			0.7209	0.8382	0.9318	0.9585
$Case_1$	5	0.6261	0.7206	0.8419	0.9422	0.9831
$Case_2$			0.7224	0.8424	0.9490	0.9939
$MCND$[CY21]			0.5653	0.5568	0.5507	0.5532
$Case_1$	6	0.5493	–	0.6135	0.6734	0.7287
$Case_2$			0.5728	0.6213	0.6842	0.7603
$Case_2$	7	–	–	–	0.5050	0.5106

7-round distinguisher for DES when the group size $m = 8$ and 16.

Training 7-round Distinguisher Using the Staged Training Method. For 7 rounds, the training scheme described above fails, i.e., the model does not learn to approximate any helpful function. We still succeeded in training a 7-round neural distinguisher of DES by using several stages of pre-training. First, we use our 6-round distinguisher to recognize 4-round DES with the input difference (0x04000000, 0x40080000) (the most likely difference to appear three rounds after the input difference (0x40080000,0x04000000). The training was done on $10^7 \times m$ samples for twenty epochs with cyclic learning rates. Then we trained the distinguisher so obtained to recognize 7-round DES with the input difference $(0x40080000, 0x04000000)$ by processing $10^7 \times m$ freshly generated samples for ten epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another $10^7 \times m$ fresh samples each.

3.2 Experiments on Chaskey

Differential-Neural Distinguishers for Reduced Chaskey: Based on the best differential path searched in [MMH+14], baseline distinguishers are built for reduced Chaskey firstly [Goh19]. Given the plaintext difference α = (0x8400,0x0400,0,0), the baseline distinguisher can distinguish Chaskey up to 4 rounds. Our differential-neural distinguishers are also built for Chaskey reduced to 3, 4 rounds. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 5, 8)$. All related parameters are the same with Table 1, except for the penalty factor λ is increased to 10^{-4}. Corresponding distinguisher accuracy is present in Table 3.

Table 3: Accuracy of differential-neural distinguisher for Chaskey

	R	BD [Goh19]	m=2	m=4	m=8	m=16
$MCND$[CY21]			0.8958	0.9583	0.9887	0.9986
$Case_1$	3	0.8608	0.9254	0.9678	0.9892	0.9885
$Case_2$			0.9294	0.9820	0.9970	0.9992
$MCND$[CY21]			0.6589	0.6981	0.7603	0.7712
$Case_1$	4	0.6161	0.6755	0.7651	0.9259	0.9403
$Case_2$			0.6784	0.8145	0.9309	0.9830

3.3 Experiments on Present

Differential-Neural Distinguishers for Reduced Present64/80: Present [BKL+07] is a block cipher that is based on a 4×4 Sbox. Based on the plaintext difference α = (0,0,0,0x9) provide in [Wan08], the baseline distinguisher were built for Present64/80 reduced up to 7 rounds [Goh19]. Our neural distinguishers are also built for Present64/80 reduced to 6, 7 rounds. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 2, 4)$. The concrete parameter of constructing neural distinguisher for Present64/80 are as follows. Corresponding distinguisher accuracy is present in Table 4 .

Table 4: Accuracy of differential-neural distinguisher for PRESENT

	R	BD [Goh19]	m=2	m=4	m=8	m=16
$MCND$[CY21]			0.7198	0.7953	0.8308	0.8259
$Case_1$	6	0.6584	0.7304	0.8116	0.8810	0.9391
$Case_2$			0.7326	0.8204	0.9066	0.9699
$MCND$[CY21]			0.5503	0.5853	0.5786	0.5818
$Case_1$	7	0.5486	0.5717	0.6054	0.6510	0.7070
$Case_2$			0.5717	0.6070	0.6559	0.7205

4 Conclusions

In this article, we modify the network architecture to train differential-neural distinguisher for three reduced symmetric ciphers, which modify the size of three convolutional kernel in the initial convolutional layer depending on the round function of cipher. Thus, we improve the prediction accuracy of differential-neural distinguisher and obtained more rounds differential-neural distinguisher for DES, Chaskey, and PRESENT.

References

[BGL$^+$21] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Conditional differential-neural cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 719, 2021.

[BGPT21] Adrien Benamira, David Gérault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. volume 12696 of *Lecture Notes in Computer Science*, pages 805–835. Springer, 2021.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer, 1993.

[CY21] Yi Chen and Hongbo Yu. A new neural distinguisher model considering derived features from multiple ciphertext pairs. *IACR Cryptol. ePrint Arch.*, page 310, 2021.

[Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. pages 2261–2269. IEEE Computer Society, 2017.

[How87] Ralph Howard. Data encryption standard. *Information age*, 9(4):204–210, 1987.

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. pages 7132–7141. Computer Vision Foundation / IEEE Computer Society, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[MMH$^+$14] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. volume 8781 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2014.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. pages 1–9. IEEE Computer Society, 2015.

[Wan08] Meiqin Wang. Differential cryptanalysis of reduced-round PRESENT. volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.

[zWW22] Liu zhang, Zilong Wang, and Boyang Wang. Improving differential-neural cryptanalysis with inception blocks. *IACR Cryptol. ePrint Arch.*, page 183, 2022.