IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–10. DOI:XXXXXXXX

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

Liu Zhang[1], Zilong Wang[1] and Yindong Chen[2]

[1] School of Cyber Engineering, Xidian University, Xi'an, China liuzhang@stu.xidian.edu.cn
zlwang@xidian.edu.cn
[2] College of Engineering, Shantou University, Shantou, China
ydchen@stu.edu.cn

Abstract. In CRYPTO 2019, Gohr first introduced the deep learning method to cryptanalysis for Speck32/64. A differential-neural distinguisher was obtained using ResNet neural network. Zhang *et al.* used multiple parallel convolutional layers with different kernel sizes to capture information from multiple dimensions, thus improving the accuracy or obtaining a more round of distinguisher for Speck32/64 and Simon32/64. Inspired by Zhang's work, we apply the network structure to other ciphers. We not only improve the accuracy of the distinguisher, but also increase the number of rounds of the distinguisher,that is, distinguish more rounds of ciphertext and random number for DES, Chaskey and PRESENT.

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

1 Introduction

In CRYPTO 2019, Gohr proposed the idea of differential-neural cryptanalysis [Goh19]. The differential-neural distinguisher model, a trained neural network, is introduced as the underlying distinguisher. The differential-neural distinguisher can distinguish whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. If the accuracy of the differential-neural distinguisher is greater than 0.5, it is an effective distinguisher. In EUROCRYPT 2021, Benamira [BGPT21] indicated that Gohr's differential-neural distinguisher builds a good approximation of the differential distribution table of the cipher and learns additional information.

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. Chen *et al.* [CY21] and Benamira [BGPT21] *et al.* almost simultaneously the method of using multiple-ciphertext pairs instead of single-ciphertext pairs (in Gohr's work) as the input of the neural network, both improved the accuracy of the 6, 7-round differential-neural distinguisher of Speck32/64. Bao *et al.* [BGL+21] used Dense Network (DenseNet) [HLvdMW17], and Squeeze-and-Excitation Network (SENet) [HSS18] with existing deep architectures to train a neural network, and obtained (7-11)-round differential-neural distinguisher for Simon32/64. Zhang et al. [zWW22] borrowed the idea of the Inception block of GoogLeNet [SLJ+15] to construct the new neural network architecture. Thus, they trained the differential-neural distinguisher for (5-8)-rounds Speck32/64 and (7-12)-rounds Simon32/64. Inspired by Zhang's work, we have done some tentative work to train a better differential-neural distinguisher on three reduced symmetric ciphers. The

main improvements for differential-neural distinguisher are listed as follows. Compared to Gohr's [Goh19] and Chen's distinguisher [CY21], we improve the accuracy of differential-neural distinguisher and obtain a more round differential-neural distinguisher for DES, Chaskey, and PRESENT.[1]

The rest of the letter is organized as follows. Section 3 introduces the network architecture. Section 4 exhibits the model training process and result for three reduced symmetric ciphers. Our work is summarized in Section 5.

2 Differential-Neural Distinguisher Model

The differential-neural distinguisher is a supervised model which distinguishes whether ciphertexts are encrypted by plaintexts that satisfies a specific input difference or by random numbers. The differential-neural distinguisher model in [Goh19, CY21, BGPT21] is almost identical except for the input format. Thus, we introduce these three models collectively. Given m plaintext pairs $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and target cipher SPECK32/64, the resulting ciphertext pairs $\{(C_{i,0}, C_{i,1}), i \in [0, m-1]\}$ is regarded as a instance. Note that $m = 1$ in [Goh19], $m \in \{2, 4, 8, 16\}$ in [CY21], and $m \in \{1, 5, 10, 50, 100\}$ in [CY21]. Each instance will be attached with a label Y:

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

If Y is 1, this instance is sampled from the target distribution and defined as a positive example. Otherwise, this instance is sampled from a uniform distribution and defined as a negative example. A large number of instances need to be put into neural network training. Suppose the neural network can obtain a stable accuracy higher than 0.5 on a test set. In that case, it can effectively distinguish whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. The differential-neural distinguisher model can be described as follows:

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

where $f(X_i)$ represents the basic features of a ciphertext pair X_i, and $\varphi(\cdot)$ is the derived features, and $F(\cdot)$ is the new posterior probability estimation function.

3 Network Architecture

The differential-neural distinguisher is a posterior probability estimation function that evaluates the quality of the distinguisher with 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, described in Figure 1.

Input Represents. 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

[1] The source codes are available in https://github.com/CryptAnalystDesigner/MutipleCipherDesChaskeyPresent.git.

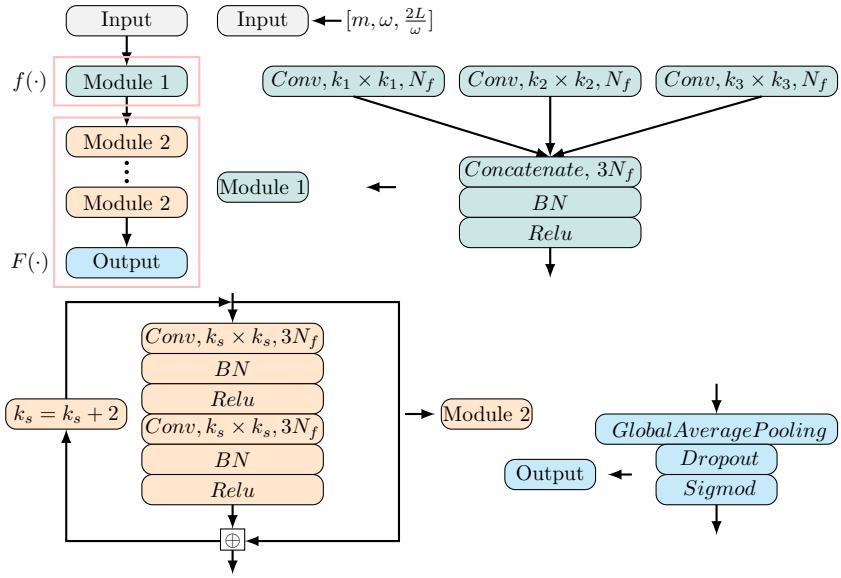


Figure 1: The network architecture of the differential-neural distinguisher model.

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.

Initial Convolution (Module 1). 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 [SLJ+15]. 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.

Convolutional Blocks (Module 2). 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.

Prediction Head (Output). The prediction head consists of a GlobalAveragePooling layer and an output unit using a *Sigmoid* activation function.

4 Model Training Process and Results

4.1 Model Training Process

Data Generation. Training and test sets were generated by using the Linux random number generator to obtain uniformly distributed keys K_i and multiple-plaintext pairs

$\{(P_{i,j,0}, P_{i,j,1}), j \in [0, m-1]\}$ with the input difference Δ as well as a vector of binary-valued labels Y_i. During producing the training or test sets for the target cipher, the multiple-plaintext pairs were 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.

Basic Training Scheme. We run the training for 20 epochs on the dataset for N and Minstances. The batch size (denoted by B_s) is set to a fixed value. Optimization was performed against mean square error loss plus a small penalty based on L2 weights regularization parameter λ using the Adam algorithm [KB15]. A cyclic learning rate schedule was applied, setting the learning rate l_i for epoch i to $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n}.(\beta - \alpha)$ 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].

Model and Training Parameter. A key parameter of our differential-neural distinguisher is the number of ciphertext pairs m, which has four options $\{2, 4, 8, 16\}$. Other parameters related to the network architecture and the training process of our differential-neural distinguisher are listed in Table 1.

Table 1: Related parameter for training differential-neural distinguishers

$N_f = 32$	$k_s = 3$	$B_s = 1000$	$\lambda = 10^{-5}$
$\alpha = 0.002$	$\beta = 0.0001$	$N = 10^7$	$M = 10^6$

The baseline distinguisher, abbreviated as $\mathcal{ND}_{bd}$, is reproduced by Chen et al. [CY21] according to the network architecture of Gohr [Goh19]. The differential-neural distinguisher of Chen *et al.*, named $\mathcal{ND}_{mc}$, 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 3, we carried out two sets of experiments. The case C_1 is an experiment using N/m and M/m instances as training and test sets, where each instance includes m ciphertext pairs. The differential-neural distinguisher obtained in C_1 named $\mathcal{ND}_{C_1}$. Also, the case C_2 is an experiment using N and M instances as training and test sets, where each instance includes m ciphertext pairs. The differential-neural distinguisher obtained in C_2 named $\mathcal{ND}_{C_2}$.

4.2 Experiments on 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 is $\alpha =$ (0x40080000, 0x04000000) and the baseline distinguishers were built for reduced DES [Goh19]. Our differential-neural distinguishers are obtained for DES reduced to 5 and 6-round using a basic training scheme. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 4, 6)$.

Training 7-round Distinguisher. We use several stages of pre-training to train a 7-round differential-neural distinguisher for DES. 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 N instances 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 N freshly generated instances for ten epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another N new instances.

Test Accuracy. We summarize the accuracy of 5, 6, and 7-round differential-neural distinguisher compared to [Goh19, CY21] in Table 2. Also, we list the accuracy (Acc), true positive rate (TPR), and true negative rate (TNR) tested on the newly generated N instances in Table 3. From Table 2, the accuracy of our differential-neural distinguisher was improved both C_1 and C_2 compared to [Goh19, CY21]. Under the two experiments with different numbers of datasets, the difference in the accuracy is relatively small except for $r = 6$ and $m = 16$. Also, we trained the differential-neural distinguisher for one more round.

Table 2: Accuracy of distinguisher for DES

r	$\mathcal{ND}_{bd}$	m=2			m=4		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
5	0.6261	0.7209	0.7232	0.7246	0.8382	0.8427	0.8442
6	0.5493	0.5653	0.5764	0.5776	0.5568	0.6128	0.6270
7	-	-	-	-	-	-	-
r	$\mathcal{ND}_{bd}$	m=8			m=16		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
5	0.6261	0.9318	0.9469	0.9496	0.9585	0.9911	0.9941
6	0.5493	0.5507	0.6634	0.6900	0.5532	0.7073	0.7693
7	-	-	-	-	-	-	0.5114

Table 3: Acc, TPR, TNR on DES using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	5	0.7244	0.4749	0.9738	4	5	0.8434	0.6935	0.9931
	6	0.5782	0.3415	0.815		6	0.6265	0.4679	0.7852
	7	-	-	-		7	-	-	-
8	5	0.9492	0.9023	0.9962	16	5	0.9940	0.9892	0.9988
	6	0.6901	0.5843	0.7959		6	0.7692	0.7352	0.8031
	7	-	-	-		7	0.5102	0.2932	0.7271

4.3 Experiments on Chaskey

Based on the best differential path searched in [MMH+14], baseline distinguishers are built for reduced Chaskey [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 using a basic training

scheme. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 5, 8)$.

Training 5-round Distinguisher. We use several stages of pre-training to train a 5-round differential-neural distinguisher for Chaskey. First, we use our 3-round distinguisher to recognize 3-round Chaskey with the input difference $(0x80000000, 0x0, 0x0, 0x80000000)$. The training was done on N/m instances for twenty epochs with cyclic learning rates. Then we trained the 3-round distinguisher to recognize 4-round Chaskey with the input difference $(0x8400, 0x0400, 0, 0)$ by processing N/m freshly generated instances for ten epochs with a learning rate of 10^{-4}, then get a 4-round distinguisher. Finally, we trained the 4-round distinguisher to recognize 5-round Chaskey with the input difference $(0x8400, 0x0400, 0, 0)$ by processing N/m freshly generated instances for ten epochs with a learning rate of 10^{-5}, then get a 5-round distinguisher.

Test Accuracy. We summarize the accuracy of 3, 4, and 5-round differential-neural distinguisher compared to [Goh19, CY21] in Table 4. Also, we list Acc, TPR, and TNR tested on the newly generated N/m instances in Table 5. From Table 4, the accuracy of our differential-neural distinguisher was improved both C_1 and C_2 compared to [Goh19, CY21]. Also, we trained the differential-neural distinguisher for one more round.

Table 4: Accuracy of distinguisher for Chaskey

r	$\mathcal{ND}_{bd}$	m=2			m=4		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
3	0.8608	0.8958	0.9583	0.9364	0.9583	0.9918	0.9854
4	0.6161	0.6589	0.7150	0.7129	0.6981	0.8390	0.8292
5	-	-	-	-	-	-	-
r	$\mathcal{ND}_{bd}$	m=8			m=16		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
3	0.8608	0.9887	0.9983	0.9974	0.9986	0.9999	0.9999
4	0.6161	0.7603	0.9316	0.9319	0.7712	0.9904	0.9850
5	-	-	0.5181	-	-	-	-

Table 5: Acc, TPR, TNR on Chaskey using N/m instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
	3	0.9340	0.9094	0.9586		3	0.9839	0.9792	0.9886
2	4	0.7122	0.4420	0.9825	4	4	0.8286	0.6808	0.9764
	5	-	-	-		5	-	-	-
	3	0.9970	0.9957	0.9982		3	0.9998	0.9997	0.998
8	4	0.9316	0.8868	0.9764	16	4	0.9842	0.9768	0.9917
	5	0.4998	0.0319	0.9678		5	-	-	-

4.4 Experiments on Present

Present [BKL+07] is a block cipher 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 and 7 rounds using a basic training scheme. The parameter (k_1, k_2, k_3) in the initial convolutional layer are $(1, 2, 4)$.

Training 8-round Distinguisher. We use several stages of pre-training to train a 8-round differential-neural distinguisher for PRESENT. First, we use our 7-round distinguisher to recognize 6-round PRESENT with the input difference $(0x0, 0x0, 0x0100, 0x0100)$ (the most likely difference to appear two rounds after the input difference $0x9$. The training was done on $N(N/m)$ instances for twenty epochs with cyclic learning rates. Then we trained the distinguisher so obtained to recognize 8-round PRESENT with the input difference $0x9$ by processing $N(N/m)$ freshly generated instances for ten epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another $N(N/m)$ new instances.

Test Accuracy. We summarize the accuracy of 6, 7, and 8-round differential-neural distinguisher compared to [Goh19, CY21] in Table 6. Also, we list Acc, TPR, and TNR tested on the newly generated N instances in Table 7. From Table 6, the accuracy of our differential-neural distinguisher was improved both C_1 and C_2 compared to [Goh19, CY21]. Also, we trained the differential-neural distinguisher for one more round.

Table 6: Accuracy of distinguisher for Present

r	$\mathcal{ND}_{bd}$	m=2			m=4		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
6	0.6584	0.7198	0.7354	0.7353	0.7953	0.8177	0.8218
7	0.5486	0.5503	0.5733	0.5741	0.5853	0.6049	0.6092
8	-	-	0.5125	0.5136	-	0.5183	0.5202
r	$\mathcal{ND}_{bd}$	m=8			m=16		
		$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$	$\mathcal{ND}_{mc}$	$\mathcal{ND}_{C_1}$	$\mathcal{ND}_{C_2}$
6	0.6584	0.8308	0.8984	0.9091	0.8259	0.9603	0.9713
7	0.5486	0.5786	0.6429	0.6584	0.5818	0.7017	0.7225
8	-	-	0.5271	0.5292	-	0.5341	0.5416

Table 7: Acc, TPR, TNR on PRESENT using N instances

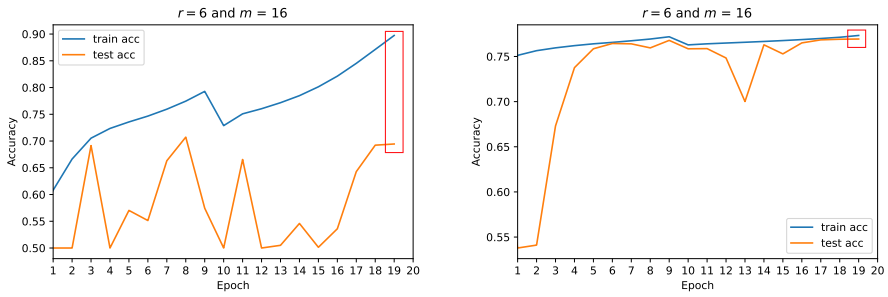
m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	6	0.7347	0.6489	0.8204	4	6	0.8219	0.7849	0.8589
	7	0.5737	0.4933	0.6540		7	0.6101	0.5163	0.7038
	8	0.5140	0.4308	0.5972		8	0.5202	0.4957	0.5447
8	6	0.9088	0.8946	0.9230	16	6	0.9715	0.9637	0.9793
	7	0.6586	0.5996	0.7176		7	0.7220	0.7052	0.7384
	8	0.5293	0.6032	0.2485		8	0.5407	0.4964	0.5851

4.5 Overfitting and Fluctuation.

Why do we train the differential-neural distinguisher with two different numbers of datasets? In [Goh19], the training set and test set include N and M instances, which consist of a ciphertext pair, that is, total N and M ciphertext pairs, respectively. In [CY21], the

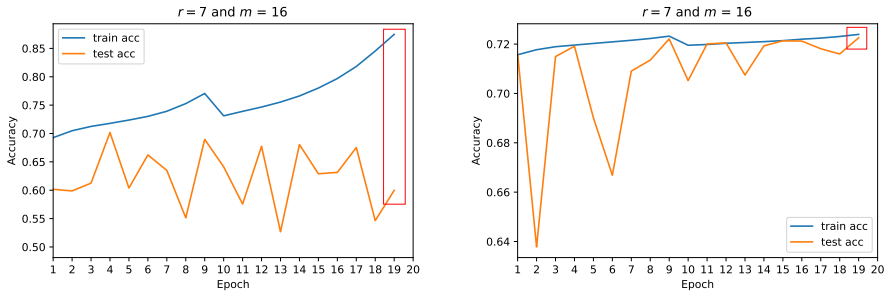
training set and test set include N/m and M/m instances, and each instance includes m ciphertext pairs; that is, the total numbers of ciphertext pairs used are N and M. To ensure a fair comparison, we used the same amount of data as [CY21] in C_1. However, Using N/m and M/m instances as training and test sets may lead to overfitting or fluctuation. In order to overcome this problem, we also use N and M instances as training test and test set in C_2, which consists of m ciphertext pair, that is, total $N \times m$ and $M \times m$ ciphertext pairs, respectively.

From Fig.2a, we can see that the difference between train accuracy and test accuracy is relatively significant for DES. However, train and test accuracy are almost equal in Fig.2b. Therefore, using N/m and M/m as training and test sets to train a distinguisher will suffer from overfitting, especially when the number of rounds r and m is large. Also, using N and M as training and test sets to train a distinguisher can avoid overfitting, speed up the model convergence and improve the model accuracy to a certain extent. For Present, we also found a similar phenomenon of overfitting in Fig.3. From Fig.4, there is no overfitting for Chaskey when we use N and M instances as training and test sets to train a distinguisher. However, the accuracy fluctuates relatively large, and the model converges slowly.



(a) Using N/m instances (b) Using N instances

Figure 2: Overfitting and accuracy fluctuation for DES.

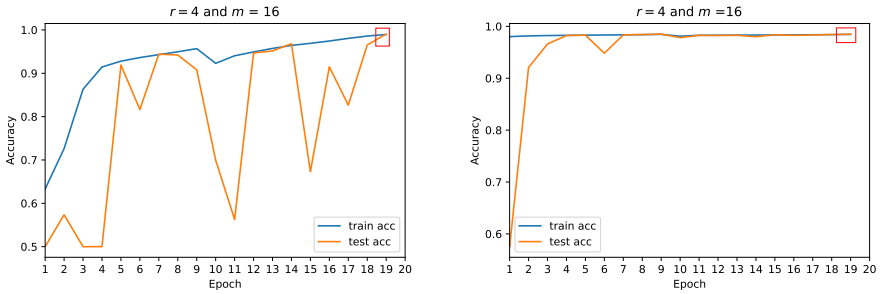


(a) Using N/m instances (b) Using N instances

Figure 3: Overfitting and accuracy fluctuation for Present.

5 Conclusions

In this letter, we use the neural network to train differential-neural distinguisher for three reduced symmetric ciphers. As a result, we improved the accuracy of the differential-



(a) Using N/m instances (b) Using N instances

Figure 4: Accuracy fluctuation for Chaskey

neural distinguisher and obtained more rounds of 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.