A New Neural Distinguisher Considering Features Derived from Multiple Ciphertext Pairs

Yi Chen, Yantian Shen, Hongbo Yu, Sitong Yuan

Department of Computer Science and Technology, Tsinghua University, Haidian District, Beijing, 100084, P. R. China
Email: {chenyi19, shenyt18, yst20}@mails.tsinghua.edu.cn,
yuhongbo@mail.tsinghua.edu.cn

Neural aided cryptanalysis is a challenging topic, in which the neural distinguisher (ND) is a core module. In this paper, we propose a new ND considering multiple ciphertext pairs simultaneously. To our best knowledge, this is the only ND except for the ND proposed by Gohr at CRYPTO'19. Taking Gohr's ND as the strong baseline model, we perform an in-depth analysis of our new ND. First, applications to five different ciphers show that our NDs achieve higher distinguishing accuracy. Second, we prove that our ND successfully captures features derived from multiple ciphertext pairs. Third, we further show how to perform various key recovery attacks with this new ND. More advantages of our ND are further discovered in key recovery attacks. Taking the neural aided statistical attack (NASA) as an example, we prove that the data complexity can be reduced by replacing Gohr's ND with our ND.

Keywords: Neural Distinguisher; Cryptanalysis; Speck32/64; Chaskey; DES; Present; SHA3-256; Data Reuse; False Negative Test; False Positive Test

1. INTRODUCTION

At CRYPTO'19, Gohr improved attacks on round reduced Speck32/64 using deep learning [1], which created a precedent for neural aided cryptanalysis. The neural distinguisher (ND) plays a core role in neural aided cryptanalysis.

The most important step of Gohr's key recovery attack is identifying the right plaintext structure. In [1], in order to identify the required plaintext structure, Gohr adopted a 6-round ND and a 7-round ND. The final decision is based on the output of the 6-round ND. Compared with the 7-round ND, the 6-round ND achieves a higher distinguishing accuracy. This implies that a stronger ND is more helpful for Gohr's attacks. Recently, Chen et al proposed a generic neural aided statistical attack (NASA) for cryptanalysis [2]. The data complexity of NASA is strongly related to the distinguishing accuracy of the ND. Thus, developing new NDs with better performance is an important task for neural aided cryptanalysis.

The ND proposed by Gohr [1] takes a ciphertext pair (C_0, C_1) as the input. In [3], Baksi et al changed the input of the ND to the ciphertext difference $C_0 \oplus C_1$. In [2], Chen et al suggested that the ND can be built by flexibly taking some bits of a ciphertext pair as the input. These NDs above are the same type of ND since only features hidden in a ciphertext pair are exploited.

In [4], in order to improve the NDs against round reduced Speck32/64, Benamira et al tried to take a ciphertext group as the input. However, no clear motivations or more in-depth analysis were provided.

1.1. Our Contributions

In this paper, our work contains the follwoing four aspects:

- We propose a new ND considering multiple ciphertext pairs simultaneously. If ciphertext pairs corresponding to plaintext pairs with a specific plaintext difference obey a non-uniform distribution, there are some derived features from multiple ciphertext pairs. Once neural networks capture these features, the ND would obtain some performance promotions.
- We design an verification framework for checking that derived features from multiple ciphertext pairs are learned. This framework is composed of two tests: false-negative test (FNT), false-positive test (FPT).
- We build two types of ND for five round-reduced ciphers: Speck32/64, Chaskey, Present, DES, SHA3-256. The first type of ND is the ND proposed by Gohr, and the other one is our new

ND. An in-depth analysis based on the analysis framework above proves that our new ND always captures derived features from multiple ciphertext pairs successfully.
- We show how to perform two different key recovery attacks using our new ND. To our date, there are only two public key recovery attacks based on the ND proposed by Gohr. We prove that our new ND is also applicable in these two attacks. Due to the performance promotion, the attack complexities of one attack can be reduced by using our new ND.

1.2. Outlines

This paper is organized as follows:

- In section 2, preliminaries are presented, including some important notations and five related ciphers.
- In section 3, the ND proposed by Gohr and two key recovery attacks are briefly reviewed.
- In section 4, our new ND is introduced, including the model, the neural network for implementing our new ND, and the training pipeline.
- In section 5, we describe the verification framework.
- In section 6, we build NDs for five ciphers and perform an analysis of these NDs.
- In section 7, we show how to perform key recovery attacks using our new ND. A data reuse strategy is also proposed in this section.

2. PRELIMINARIES

2.1. Notations

P, C	Plaintext, Ciphertext
α	Plaintext difference
N, M	The number of plaintext(ciphertext) pairs
$ND_{k=?}$	ND which the input is k ciphertext pairs
BD	Baseline distinguisher (Gohr's ND)
Z	The output of a ND
r	The number of reduced rounds

2.2. Five Ciphers

We choose five different ciphers for supporting our work.

- Speck32/64 [5] is a lightweight block cipher whose block size is 32 bits. Its non-linear component is the modulo addition.
- Chaskey [6] is a Message Authentication Code (MAC) algorithm whose intermediate state size is 128 bits. Its non-linear component is the modulo addition.
- Present64/80 [7] is a block cipher whose block size is 64 bits. Its non-linear component is a 4×4 Sbox.
- DES [8] is a block cipher whose block size is 64 bits. Its non-linear component is eight different 6×4 Sboxs.
- SHA3-256 [9] is a hash function whose intermediate state size is 1600 bits. Its non-linear component is a logic operation.

We refer readers to [5, 6, 7, 8, 9] for more details of these ciphers. From the perspective of the non-linear component, these ciphers almost cover all the types of ciphers that Gohr's ND can apply.

3. RELATED WORK

3.1. Gohr's Neural Distinguisher

In [1], Gohr built NDs against round reduced Speck32/64. The ND proposed by Gohr is a generic distinguisher since it only requires a plaintext difference constraint.

Consider a cipher E and a plaintext difference α. Gohr's ND aims at distinguishing two classes of ciphertext pairs

$$Y(C_0, C_1) = \begin{cases} 1, if\ P_0 \oplus P_1 = \alpha \\ 0, if\ P_0 \oplus P_1 \neq \alpha \end{cases} \tag{1}$$

where (C_0, C_1) is the ciphertext pair corresponding to the plaintext pair (P_0, P_1), and Y is the label of (C_0, C_1).

We denote ciphertext pairs corresponding to plaintext pairs with the target difference α as **positive samples**, and denote ciphertext pairs corresponding to plaintext pairs with random difference as **negative samples**.

If a neural network achieves a distinguishing accuracy higher than 0.5 over randomly selected ciphertext pairs, we know the cipher E is not a pseudorandom permutation. Then this neural network is a valid ND.

In [1], Gohr chose a residual network [10] with one output neuron. Thus, the output Z of Gohr's ND is also used as the following posterior probability

$$\begin{gathered} Pr\left(Y = 1 \,|(C_0, C_1)\,\right) = F_1\left(f\left(C_0, C_1\right)\right) \\ 0 \leqslant Pr\left(Y = 1 \,|(C_0, C_1)\,\right) \leqslant 1 \end{gathered} \tag{2}$$

where $f(C_0, C_1)$ stands for features learned by the ND from (C_0, C_1), $F_1(\cdot)$ is the posterior probability estimation function learned by the ND. If $Pr(Y = 1|(C_0, C_1)) > 0.5$, the label of (C_0, C_1) predicted by the ND is 1.

3.2. Gohr's Key Recovery Attack

Given a ND, we denote the output of ND as Z. Positive samples are expected to obtain a higher posterior probability than negative samples, which is the core idea of Gohr's key recovery attack [1].

Consider a $(r + 1)$-round cipher E and a r-round ND built over a plaintext difference α. Gohr's attack recovers the subkey of the $(r + 1) - th$ round as follows:

1. Generate m positive samples with α randomly.
2. For each possible subkey guess kg:

   (a) Decrypt m positive samples with kg.
   (b) Feed partially decrypted samples into the ND and collect the outputs $Z_i, i \in [1, m]$.
   (c) Compute the rank score V_{kg} of kg as:

$$V_{kg} = \sum_{i=1}^{m} \log_2 \left(\frac{Z_i}{1 - Z_i} \right) \tag{3}$$

   (d) If V_{kg} exceeds a threshold c_1, save kg as a subkey kandidate.

3. Return kg with the highest key rank score as the final subkey guess.

The value of c_1 and m is set experimentally.

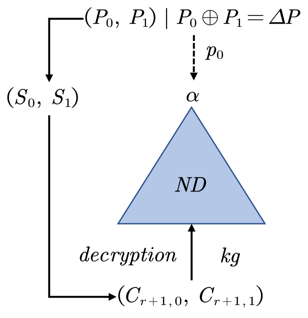

FIGURE 1. The key recovery process. The prepended differential $\Delta P \to \alpha$ is satisfied with a probability p_0. The intermediate state pair is (S_0, S_1).

A differential $\Delta P \to \alpha$ can be placed before the ND to extend the rounds covered by the attack (see Fig.1). With the help of neutral bits [11], ciphertext structures consisting of m positive samples or negative samples can be generated. Then a high rank score occurs only when the structure consisting of positive samples is decrypted by the true subkey. More details can refer to [1].

3.3. Neural Aided Statistical Attack

The neural aided statistical attack proposed by Chen et al [2] is performed as follows:

1. Randomly generate N plaintext pairs with a difference ΔP.
2. Collect the ciphertext pairs.
3. For each possible subkey guess kg:
   (a) Decrypt N ciphertext pairs with kg.
   (b) Feed partially decrypted ciphertext pairs into the ND and collect the outputs $Z_i, i \in [1, N]$.
   (c) Count the following statics T:

$$T = \sum_{i=1}^{N} \phi(Z_i), \quad \phi(Z_i) = \begin{cases} 1, if \ Z_i > c_2 \\ 0, if \ Z_i \leqslant c_2 \end{cases} \tag{4}$$

   (d) If T exceeds a decision threshold t, save kg as a subkey candidate.
4. Return all the surviving subkey candidates.

Chen et al proposed a theoretical framework to estimate N and t. The value of c_2 is set in advance, which doesn't influence the estimation of N, t.

According to Fig.1, Chen et al summarized three types of probabilities:

$$Pr(Z > c_2 | S_0 \oplus S_1 = \alpha, kg = sk) = p_1 \tag{5}$$

$$Pr(Z > c_2 | S_0 \oplus S_1 = \alpha, kg \neq sk) = p_2 \tag{6}$$

$$Pr(Z > c_2 | S_0 \oplus S_1 \neq \alpha) = p_3 \tag{7}$$

where sk is the true subkey. These three probabilities p_1, p_2, p_3 are related to the ND.

NASA returns all the possible subkey candidates. Besides, NASA allows us to set two ratios β_0, β_1 in advance. The ratio β_0 is the expected probability that the true subkey sk survives the attack. The ratio β_1 is the expected probability that wrong subkey guesses survive the attack.

Based on $p_0, p_1, p_2, p_3, \beta_0, \beta_1$, the required N is:

$$\sqrt{N} = \frac{z_{1-\beta_0} \times v_0 + z_{1-\beta_1} \times v_1}{(p_1 - p_2) \times p_0} \tag{8}$$

where

$$v_0 = \sqrt{p_0 \times p_1(1 - p_1) + (1 - p_0) p_3 (1 - p_3)}$$

$$v_1 = \sqrt{p_0 \times p_2(1 - p_2) + (1 - p_0) p_3 (1 - p_3)},$$

and $z_{1-\beta_0}$, $z_{1-\beta_1}$ are the quantiles of the standard normal distribution.

The decision threshold t is:

$$t = \mu_0 - z_{1-\beta_0} \times \sigma_0 \tag{9}$$

where

$$\mu_0 = N \times (p_0 p_1 + (1 - p_0) p_3)$$

$$\sigma_0 = \sqrt{N \times p_0 \times p_1(1 - p_1) + N(1 - p_0) p_3 (1 - p_3)}$$

If $c_2 = 0.5$, the distinguishing accuracy of the ND is $(p_1 + 1 - p_3) \times 0.5$. Thus the data complexity of NASA is strongly related to the ND. We refer readers to [2] for more details of NASA.

4. NEW NEURAL DISTINGUISHER

4.1. Motivations

The motivations of our new neural distinguisher contain two aspects.

First, in the machine learning community, providing more features is a common method to improve the accuracy of neural networks. For example, depth map estimation [12] and action recognition [13] are both tackled by feeding various features (eg. stereo knowledge [14], depth maps [15]) into neural networks simultaneously.

Second, there are some useful features among multiple samples drawn from the same non-uniform distribution. Fig.2 shows a simple example. If we

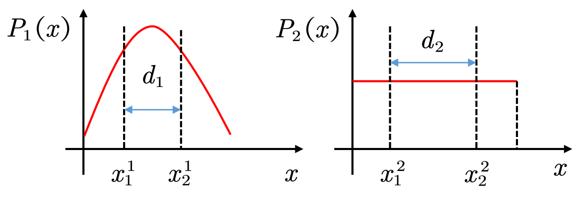

FIGURE 2. $P_1(x)$: a Gaussian distribution. $P_2(x)$: a uniform distribution.

randomly draw two samples $(x_1^1, x_2^1)/(x_1^2, x_2^2)$ from a Gaussian distribution or a uniform distribution, the average distance of two samples is d_1/d_2. Then it is expected that $d_1 < d_2$, which can be used to distinguish the two distributions.

Based on the two phenomena above, we obtain the idea of building a new neural distinguisher by considering multiple ciphertext pairs. When features hidden in a single ciphertext pair do not provide useful clues, we still can obtain useful clues from features among multiple ciphertext pairs.

4.2. New Distinguisher Model

Our new ND needs to distinguish two types of ciphertext groups $(C_{1,1}, C_{1,2}, \cdots, C_{k,1}, C_{k,2})$:

$$Y = \left\{ \begin{array}{l} 1, if\ P_{j,1} \oplus P_{j,2} = \alpha, j \in [1,k] \\ 0, if\ P_{j,1} \oplus P_{j,2} \neq \alpha, j \in [1,k] \end{array} \right. \tag{10}$$

where Y is the label of ciphertext groups, and $(C_{j,1}, C_{j,2})$ is the ciphertext pair corresponding to the plaintext pair $(C_{j,1}, C_{j,2}), j \in [1,k]$.

Our new ND can be described as

$$\begin{gathered} Pr\left(Y = 1 \left| X_1, \cdots, X_k \right.\right) = \\ F_2\left(f\left(X_1\right), \cdots, f\left(X_k\right), \varphi\left(f\left(X_1\right), \cdots, f\left(X_k\right)\right)\right) \\ X_i = \left(C_{i,1}, C_{i,2}\right), i \in [1,k] \end{gathered} \tag{11}$$

where $f\left(X_i\right)$ represents the basic features extracted from the ciphertext pair X_i, $\varphi\left(\cdot\right)$ is the derived features, and $F_2\left(\cdot\right)$ is the new posterior probability estimation function.

Features $\varphi(f(X_1), \cdots, f(X_k))$ are extracted from the distribution of basic features $f(X_i), i \in [1,k]$. This is consistent with the motivations.

4.3. Residual Network

4.3.1. Network Architecture

In order to implement our new ND, we propose a residual network which is a little different from the network adopted by Gohr.

Fig.3 in next page shows the neural network architecture for implementing our new ND. The network architecture contains several modules that are described in Fig.3. The input consisting of k ciphertext pairs is arranged in a $k \times w \times \frac{2L}{w}$ array. L represents the block size of the target cipher and w is the size of a basic unit. For example, L is 32 and w is 16 for speck32/64.

Based on this input, the new ND can be implemented in two parts. First, the learning of a single ciphertext pair's basic features is accomplished by module 1. The kernel size is 1×1, which can capture basic features efficiently. Second, the learning of derived features and posterior probability estimation functions are combined in a part. The two-dimensional filters with a size of $K_s \times K_s$ can learn derived features from k ciphertext pairs. Such an architecture obeys the model of our neural distinguisher completely.

4.3.2. Training Pipeline

Our new ND is obtained by following three processes:

1. **Data Generation:** All the k ciphertext pairs belonging to a ciphertext group are collected using the same master key. If the label is $Y = 1$, the ciphertext group is denoted as a positive sample. Otherwise it's denoted as a negative sample. A training set is composed of $\frac{N}{2k}$ positive samples and $\frac{N}{2k}$ negative samples. A testing set is composed of $\frac{M}{2k}$ positive samples and $\frac{M}{2k}$ negative samples. We need to generate a training set and a testing set.
2. **Training:** Train the residual network (Fig.3) on the training dataset. If the training accuracy is not larger than 0.5, choose a different α and start from the data generation process again. Or save the trained neural network and perform the testing process.
3. **Testing:** Test the distinguishing accuracy of the trained neural network on the testing dataset. If the test accuracy is larger than 0.5, return the neural network as a valid ND. Or choose a different α and start from the data generation process again.

In the training phase, the neural network is trained for E_s epochs with a batch size of B_s. The cyclic learning rate scheme in [1] is adopted. Optimization is performed against the following loss function:

$$loss = \sum_{i=1}^{\frac{N}{k}} \left(Z_{i,p} - Y_i\right)^2 + \lambda \times \|W\| \tag{12}$$

where $Z_{i,p}$ is the output of the ND, Y_i is the true label, W is the parameters of the neural network, and λ is the penalty factor. The Adam algorithm [16] with default parameters in Keras [17] is applied to the optimization.

5. THE VERIFICATION FRAMEWORK

To check whether our ND captures derived features from multiple ciphertext pairs as expected, we design an analysis framework that is composed of two tests. In this analysis framework, the ND proposed by Gohr is used as a baseline distinguisher (BD).

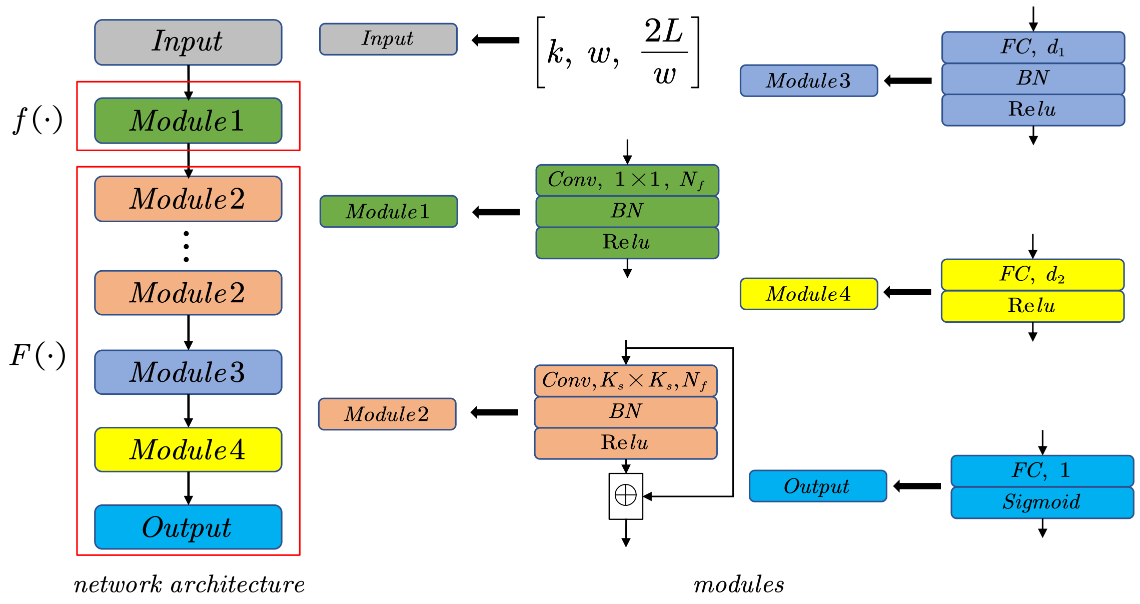

FIGURE 3. The network architecture of our new ND. *Conv* stands for a convolution layer with N_f filters. The size of each filter is $K_s \times K_s$. Module 2 also adopts the skip connection [10]. *FC* is a fully-connected layer which has d_1 or d_2 neurons. *BN* is batch normalization. *Relu* and *Sigmoid* are two different activation functions. The output of *Sigmoid* ranges from 0 to 1.

5.1. False Negative Test (FNT)

If k ciphertext pairs with label 1 are all wrongly classified by the BD

$$\begin{array}{c} p\left(Y=1\left|X_1\right.\right)=F_1\left(f\left(X_1\right)\right)<0.5 \\ \vdots \\ p\left(Y=1\left|X_k\right.\right)=F_1\left(f\left(X_k\right)\right)<0.5, \end{array} \tag{13}$$

such ciphertext pairs are false negative samples. It means that the features of a single ciphertext pair $f\left(X_i\right)$ can't help correct classification. These k samples are combined into a ciphertext group and fed into our ND.

Generate a large number of such ciphertext groups and feed them to our ND. What we care about is the following pass ratio

$$F_2\left(f\left(X_1\right),\cdots,f\left(X_k\right),\varphi\left(f\left(X_1\right),\cdots,f\left(X_k\right)\right)\right)\geqslant 0.5 \tag{14}$$

The classification is totally determined by $\varphi\left(f\left(X_1\right),\cdots,f\left(X_k\right)\right)$ now. The final pass ratio under such a setting can show whether derived features have been learned and their effects. If our ND can obtain a non-negligible pass ratio, then $\varphi\left(f\left(X_1\right),\cdots,f\left(X_k\right)\right)$ can offset the negative influence of $f\left(X_i\right), i\in[1,k]$. If the pass ratio is high, derived features from k ciphertext pairs play a vital role in classification for this kind of ciphertext pairs.

TABLE 1. Parameters for constructing our new ND

N_f	d_1	d_2	K_s	B_s
32	64	64	3	500
λ	L_r	E_s	N	M
10^{-5}	$0.02 \to 0.001$	10	10^7	10^6

5.2. False Positive Test (FPT)

Similarly, if k ciphertext pairs with label 0 are wrongly classified by the BD

$$\begin{array}{c} p\left(Y=1\left|X_1\right.\right)=F_1\left(f\left(X_1\right)\right)\geqslant 0.5 \\ \vdots \\ p\left(Y=1\left|X_k\right.\right)=F_1\left(f\left(X_k\right)\right)\geqslant 0.5, \end{array} \tag{15}$$

such ciphertext pairs are false positive samples. These k samples are combined into a ciphertext group and fed into our ND. Now what we care about is the following pass ration

$$F_2\left(f\left(X_1\right),\cdots,f\left(X_k\right),\varphi\left(f\left(X_1\right),\cdots,f\left(X_k\right)\right)\right)<0.5 \tag{16}$$

6. APPLICATIONS TO FIVE CIPHERS

We build two types of NDs for five ciphers introduced in section 2.2. The first one is Gohr's ND that is used as the BD. The second one is our new ND.

The training pipeline of Gohr's ND is presented in [1]. Table 1 summarizes the parameters that are related to the residual network and training pipeline that are introduced in Section 4.3.

TABLE 2. Distinguishing accuracy of NDs against Speck32/64.

r	BD	Our neural distinguishers			
		$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.929	0.9738	0.991	0.9992	0.9999
6	0.788	0.8613	0.931	0.9562	0.9802
7	0.616	0.6393	0.6861	0.7074	0.6694

TABLE 3. Pass ratios of **FPT** and **FNT** of NDs against Speck32/64.

r	False Negative Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.0112	0.0013	0.0001	0
6	0.0331	0.0143	0.0081	0.0048
7	0.0511	0.0212	0.0283	0.0917
r	False Positive Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.3068	0.6268	0.6748	0.7228
6	0.1519	0.1432	0.3723	0.4375
7	0.0659	0.0233	0.0157	0.0691

6.1. Experiments on Speck32/64

The plaintext differential is $\alpha = (0x0040, 0)$ introduced in [18]. NDs against Speck32/64 reduced to 5, 6, and 7 rounds are built. The accuracy comparison is presented in Table 2. It's clear that our NDs achieve higher distinguishing accuracy than the BD.

We further perform the **FPT** and **FNT**. Corresponding pass ratios are presented in Table 3. For each new ND, there is at least one type of pass ratios higher than 0. This fully proves that our NDs capture derived features from k ciphertext pairs.

6.2. Experiemnts on Chaskey

Based on the plainext difference $\alpha = (0x8400, 0x0400, 0, 0)$ [6], we build NDs against Chaskey reduced to 3, 4 rounds. The accuracy comparison is presented in Table 4. Our NDs achieve higher distinguishing accuracy than the BD.

The **FPT** and **FNT** are performed over our NDs. Corresponding pass ratios are summarized in Table 5, which also proves that derived features from k ciphertext pairs are captured by our NDs.

TABLE 4. Distinguishing accuracy of NDs against Chaskey

r	BD	Our neural distinguishers			
		$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.8608	0.8958	0.9583	0.9887	0.9986
4	0.6161	0.6589	0.6981	0.7603	0.7712

TABLE 5. Pass ratios of **FPT** and **FNT** of NDs against Chaskey.

r	False Negative Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.1156	0.0635	0.0373	0.0087
4	0.1412	0.1749	0.1481	0.1675
r	False Positive Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.4027	0.4032	0.3976	0.4705
4	0.8369	0.7439	0.7298	0.5591

TABLE 6. Distinguishing accuracy of NDs against Present64/80

r	BD	Our neural distinguishers			
		$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
6	0.6584	0.7198	0.7953	0.8308	0.8259
7	0.5486	0.5503	0.5853	0.5786	0.5818

6.3. Experiments on Present64/80

Based on the plaintext difference $\alpha = (0, 0, 0, 0x9)$ provided in [19], we have built NDs against Present64/80 reduced up to 6, 7 rounds respectively. The penalty factor is 10^{-4} and other related parameters are the same with Table 1. The distinguishing accuracy comparisons are presented in Table 6.

The pass ratios of the **FPT** and **FNT** of our NDs are presented in Table 7. For each new ND, two related pass ratios are both higher than 0. Thus, our NDs capture derived features from k ciphrtext pairs.

6.4. Experiments on DES

Based on the analysis of DES in [20], the plaintext difference adopted in this paper is $\alpha = (0x40080000, 0x04000000)$. We have built baseline dis-

TABLE 7. Pass ratios of **FNT** and **FPT** of NDs against Present64/80.

r	False Negative Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
6	0.0277	0.0097	0.0258	0.0751
7	0.1796	0.0587	0.1214	0.1488
r	False Positive Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
6	0.0147	0.0046	0.0068	0.0183
7	0.0533	0.0126	0.0324	0.0302

TABLE 8. Distinguishing accuracy of NDs against DES.

r	BD	Our neural distinguishers			
		$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.6261	0.7209	0.8382	0.9318	0.9585
6	0.5493	0.5653	0.5568	0.5507	0.5532

TABLE 9. Pass ratios of **FNT** and **FPT** of NDs against DES.

R	False Negative Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.0046	0.0034	0.0132	0.0131
6	0.0802	0.2348	0.2526	0.3207
R	False Positive Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
5	0.0594	0.0627	0.0566	0.0518
6	0.0462	0.0598	0.0921	0.0809

tinguishers for DES reduced to 5, 6 rounds.

The batch size is adjusted to 5000. The penalty factor is increased to 8×10^{-4}. Other related parameters are the same as Table 1. Corresponding distinguishing accuracy comparisons are presented in Table 8. The pass ratios of the **FPT** and **FNT** of our NDs are presented in Table 9.

6.5. Experiments on SHA3-256

SHA3-256 is a hash function. When one message block is fed into reduced SHA3-256, we collect the first 32 bytes of the output process after r-rounds permutation is applied to this message block. Given a message difference $\alpha = 1$, we have built NDs for SHA3-256 reduced up to $3, 4$ rounds.

Limited by the computer memory, the number of ciphertext pairs is adjusted to $N = 2 \times 10^6$. The batch size is 500, and the penalty factor is 10^{-5}. The distinguishing accuracy comparisons are presented in Table 10. The pass ratios of the **FPT** and **FNT** of our NDs are presented in Table 11.

7. KEY RECOVERY ATTACKS

In this section, we propose a data reuse strategy for reducing data complexity. Then we prove that our ND can be applied to the two key recovery attacks introduced in section 3. Since the data complexity of NASA is directly related to the performance of NDs, the NASA is first performed to highlight the extra superiority of our NDs.

TABLE 10. Distinguishing accuracy of NDs against SHA3-256.

r	Gohr	Our neural distinguishers			
		$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.7228	0.8149	0.9241	0.971	0.9904
4	0.5844	0.6409	0.8441	0.8748	0.8775

TABLE 11. Pass ratios of **FNT** and **FPT** of NDs against SHA3-256.

r	False Negative Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.2249	0.2347	0.3336	0.2711
4	0.8834	0.1425	0.1406	0.8646
r	False Positive Test			
	$ND_{k=2}$	$ND_{k=4}$	$ND_{k=8}$	$ND_{k=16}$
3	0.1045	0.0961	0.0171	0.0088
4	0.4619	0.4933	0.4918	0.5044

7.1. Data Reuse Strategy for Reducing Data Complexity

There is a potential problem when we directly apply our new ND to key recovery attacks.

Assuming the BD and our ND have the same performance, and a certain attack requires M random inputs. If we directly reshape $M \times k$ ciphertext pairs into M ciphertext groups, the data complexity of our ND is k times as much as the data complexity of the BD.

Given M ciphertext pairs $X_i = (C_{i,0}, C_{i,1}), i \in [1, M]$, there are a total of C_M^k options for composing a ciphertext group, which is much larger than $\frac{M}{k}$. Thus we can randomly select M ciphertext groups from C_M^k options. Such a strategy can help reduce the data complexity. In fact, it is equivalent to attach more importance to derived features from k ciphertexts.

However, the subsequent key recovery attacks using this naive strategy do not obtain good results. The main reason is that the sampling randomness of M ciphertext groups is greatly destroyed. Two new concepts are proposed for overcoming this problem.

Maximum Reuse Frequency: During the generation of M ciphertext groups, a ciphertext pair is likely to be reused several times. Let's denote the reuse frequency of the i_{th} ciphertext pair as $RF_i, i \in [1, M]$. *Maximum Reuse Frequency* (MRF) is defined as the maximum value of RF_i:

$$MRF = \max RF_i, i \in [1, M] \tag{17}$$

Sample Similarity Degree: For any two ciphertext groups G_i, G_j, the similarity of these two ciphertext groups is defined as the number of the same ciphertext pairs. As for M ciphertext groups, *Sample Similarity Degree* (SSD) is defined as the maximum of any two ciphertext groups' similarity:

$$\begin{gathered} SSD = \max \left| G_i \bigcap G_j \right|, i, j \in [1, M] \\ G_i = \{X_{i1}, \cdots, X_{ik}\} \\ G_j = \{X_{j1}, \cdots, X_{jk}\} \\ i1, \cdots, ik, j1, \cdots, jk \in [1, M] \end{gathered} \tag{18}$$

MRF can ensure that the contribution of each ciphertext pair is similar. SSD can increase the distribution uniformity of M ciphertext groups as much as possible. Based on the above two concepts, we propose the following ***Data Reuse Strategy*** (see Algorithm 1) that can reduce data complexity and maintain sampling randomness.

Algorithm 1 Data Reuse Strategy

```
Require: $MRF$; $SSD$; $k$; $M$.
Ensure: $M$ ciphertext groups with a size of $k$.
1: Randomly select $k$ ciphertext pairs from $M$ ciphertext pairs to form a ciphertext group.
2: Repeat step 2 for $M$ times to obtain $M$ ciphertext groups.
3: Compute $MRF$ and $SSD$. If two values are both smaller than the threshold we set, return the $M$ ciphertext groups. Or start from step 1 again.
```

7.2. Application to NASA

When we replace the BD with our new ND, the process of NASA does not change. The only difference is the data collection.

7.2.1. Data Collection

Consider the attack process as shown in Fig.1. Assuming that our new ND is built with α. Now, we need to generate ciphertext groups.

Generate k plaintext pairs $(P_0^i, P_1^i), i \in [1, k]$ with the difference ΔP. Collect corresponding ciphertexts $(C_0^i, C_1^i), i \in [1, k]$. The intermediate states are $(S_0^i, S_1^i), i \in [1, k]$.

According to introduction in Section 4.2, these k ciphertext pairs should satisfy

$$S_0^i \oplus S_1^i = \alpha, \quad or \quad S_0^i \oplus S_1^i \neq \alpha, i \in [1, k]$$

simultaneously. We use neutral bits [11] to generate such k ciphertext pairs.

Here we briefly review the definition of neutral bits. Let E denote the encryption function. We focus on the following conforming pairs

$$P_0 \oplus P_1 = \Delta P, \quad E(P_0) \oplus E(P_1) = \alpha.$$

If the condition $E(P_0 \oplus e^j) \oplus E(P_1 \oplus e^j) = \alpha$ always holds where $e^j = 1 << j$, the j-th bit is a neutral bit.

Thus, we can generate $2^m \geqslant k$ ciphertext pairs using m neutral bits. The probability that these k ciphertext pairs satisfy the difference transition $\Delta P \to \alpha$ simultaneously is still p_0. Then N ciphertext groups with a size of k can be generated as

1. Ranomly generate N plaintext pairs with ΔP.
2. Generate N plaintext structures using m neutral bits.
3. Randomly pick k plaintext pairs from a structure and collect the ciphertext pairs.

The total data complexity is $N \times k$.

It is worth noticing that the data reuse strategy is still applicable here. More exactly, the data collection is performed as

1. Ranomly generate $\frac{N}{M}$ plaintext pairs with ΔP.
2. Generate $\frac{N}{M}$ plaintext structures using m neutral bits.
3. Randomly pick M plaintext pairs from a structure, and generate M ciphertext groups using the data reuse strategy (Algorithm 1).

The total data complexity is N now.

7.2.2. Experiments on Speck32/64

To prove that our ND is applicable to the NASA, we perform experiments on Speck32/64.

Our new ND achieves higher performance than the ND proposed by Gohr. Thus it is also expected that the data complexity of NASA can be reduced by adopting our new ND.

Experiment settings. More exactly, we adopt a 2-round differential $\Delta P = 0x211/0xa04 \xrightarrow{p_0=2^{-6}} \alpha = 0x40/0x0$ as the prepended differential. Let $\beta_0 = 0.005, \beta_1 = 2^{-16}, c_2 = 0.5$. The concrete meaning of these parameters can refer to Section 3.3. Since c_2 is set, the values of p_1, p_3 are experimentally estimated based on NDs.

The estimation of p_2 is a little complex. Let $p_{2|d}$ denote the estimated value of p_2 where d is the Hamming distance between the correct key tk and wrong keys kg. According to the introduction in [2], when d increases, $p_{2|d}$ will decrease. Moreover, when p_2 increases, the data complexity of NASA also increases. Thus, if we hope the Hamming distance between tk and surviving kg does not exceed d, the value of p_2 is

$$p_2 = \max\{p_{2|i} | i \in [d+1, 16]\}. \tag{19}$$

In this paper, we choose two different settings: $d = 2$, $d = 1$.

Comparison of data complexity. Table 12 and Table 13 show the comparison of data complexity under two experiment settings respectively.

The second row corresponds to the data complexity when Gohr's ND is adopted. These results are also used as the baseline. When a r-round ND with a group size of k is adopted, the corresponding data complexity is displayed in bold if it is smaller than the baseline.

We test 12 new NDs in total. Table 12 and Table 13 show that the data complexity is reduced in most cases. There is only one case in which the data complexity is not reduced.

Analysis of the data complexity. There are two questions to be explained: (1) why does the accuracy promotion of NDs bring the reduction of the data

TABLE 12. Data complexity comparisons when $p_0 = 2^{-6}, d = 2, \beta_0 = 0.005, \beta_1 = 2^{-16}, c_2 = 0.5$. The prepended differential is a 3-round differential that is extended from $0x211/0xa04 \xrightarrow{p_0} 0x40/0x0$ without loss of transition probability.

Distinguisher	Data Complexity ($log_2 N$)		
	$r = 5$	$r = 6$	$r = 7$
BD	14.211	16.907	20.515
$ND_{k=2}$	**13.184**	**15.869**	**19.792**
$ND_{k=4}$	**12.908**	**14.757**	**18.881**
$ND_{k=8}$	**12.016**	**14.609**	**18.573**
$ND_{k=16}$	**13.171**	**14.427**	**19.684**

TABLE 13. Data complexity comparisons when $p_0 = 2^{-6}, \boldsymbol{d = 1}, \beta_0 = 0.005, \beta_1 = 2^{-16}, c_2 = 0.5$. The prepended differential is a 3-round differential that is extended from $0x211/0xa04 \xrightarrow{p_0} 0x40/0x0$ without loss of transition probability.

Distinguisher	Data Complexity ($log_2 N$)		
	$r = 5$	$r = 6$	$r = 7$
BD	14.72	17.158	21.085
$ND_{k=2}$	**13.971**	**16.517**	**20.386**
$ND_{k=4}$	**14.189**	**15.566**	**19.483**
$ND_{k=8}$	**13.863**	**15.688**	**19.214**
$ND_{k=16}$	15.560	**16.065**	**20.433**

complexity? (2) why does the data complexity is not reduced in the only failed case shown in Table 13?

To answer the **first** question, we need to analyze how the data complexity is influenced by p_1, p_3. Based on Equation 8 in Section 3.3, we get two following conclusions:

- when $p_1 | p_1 \geqslant 0.5$ increases, the data complexity N decreases.
- when $p_3 | p_3 \leqslant 0.5$ decreases, the data complexity N decreases.

During the training of NDs, the accuracy can be formulated as

$$acc = 0.5 \times (TPR + TNR)$$

where TPR is the ratio that samples with a label $Y = 1$ are correctly classified by NDs, and TNR is the ratio that samples with a label $Y = 0$ are correctly classified by NDs.

If we set $c_2 = 0.5$, the following conclusions also hold

$$\begin{gathered} TPR = p_1, \quad TNR = 1 - p_3 \\ acc = 0.5 \times (p_1 + 1 - p_3). \end{gathered} \tag{20}$$

Thus, when the accuracy acc of NDs increases, there are three phenomena: p_1 increases, or p_3 decreases, or the former two phenomena both occur.

TABLE 14. The value of p_1, p_2, p_3 related to the 5-round NDs when $c_2 = 0.5, \boldsymbol{d = 1}, r = 5$. $p_0 = 2^{-6}$.

Distinguisher	p_1	p_2	p_3	$log_2 N$
BD	0.8975	0.3334	0.0462	14.72
$ND_{k=2}$	0.9677	0.468	0.01964	**13.971**
$ND_{k=4}$	0.9894	0.6908	0.00704	**14.189**
$ND_{k=8}$	0.9988	0.8539	0.00063	**13.863**
$ND_{k=16}$	0.9999	0.9693	2.4×10^{-5}	15.56

TABLE 15. The value of p_1, p_2, p_3 related to the 5-round NDs when $c_2 = 0.5, d = 1, r = 5$. $\boldsymbol{p_0 = 2^{-12}}$.

Distinguisher	p_1	p_2	p_3	$log_2 N$
BD	0.8975	0.3334	0.0462	26.66
$ND_{k=2}$	0.9677	0.468	0.01964	**25.81**
$ND_{k=4}$	0.9894	0.6908	0.00704	**25.84**
$ND_{k=8}$	0.9988	0.8539	0.00063	**24.49**
$ND_{k=16}$	0.9999	0.9693	2.4×10^{-5}	**24.46**

No matter which phenomenon occurs, it is helpful for reducing the data complexity. This is why the data complexity is reduced in most cases shown in Table 12 and Table 13.

To answer the **second** question, we must consider the impact of p_2. For convenience, we summarized the values of p_1, p_2, p_3 related to the 5-round NDs in Table 14.

The value of p_2 also increases as shown in Table 14. Chen et al presented that the impact of p_1, p_2 on N is $\mathcal{O}((p_1 - p_2)^{-2})$ [2]. Therefore, the promotion of p_2 has a negative impact on the data complexity. If p_2 is very close to p_1, the positive impact of the accuracy increase may be offset. This is why the data complexity is not reduced when the 5-round $ND_{k=16}$ is adopted.

Although p_2 also increases when we improve the distinguishing accuracy of NDs, its negative influence seldom offsets the positive influence resulted from the accuracy promotion. Table 12 and Table 13 have already proved it. Actually, when p_0 becomes smaller, the reduction of data complexity is more significant. Table 15 shows an example.

Practical experiments. Based on the attack settings shown in Table 12, we perform NASA against 10-round Speck32/64 based on the BD and $ND_{k=2}$ ($r = 6$) respectively. The target is to recover sk_{10}. Since $d = 2$, the number of surviving subkey guess should not exceed $137 \times (1 - \beta_0) + (2^{16} - 137) \times \beta_1 = 137.31$.

When we perform NASA with Gohr's 6-round ND (BD) 100 times ($N = 2^{16.907}$), the results are

1. the true subkey sk_{10} survives in 99 trails.
2. the average number of surviving subkey guesses is 18.63.
3. In all the 100 trails, the number of surviving subkey guess is lower than 137.31.

When we perform NASA with our 6-round $ND_{k=2}$ 100 times ($N = 2^{15.869}$), the results are

1. the true subkey sk_{10} survives in 94 trails.
2. the average number of surviving subkey guesses is 29.44.
3. In all the 100 trails, the number of surviving subkey guess is lower than 137.31.

The practical experiments further prove that our new NDs can be applied to NASA. Besides, with smaller data complexity, the NASA based on our ND achieves a competitive result.

7.3. Application to Gohr's Attack

Gohr's attack is not directly related to the distinguishing accuracy of NDs. Thus, we mainly verify whether our new ND is applicable to Gohr's attack.

In [1], Gohr performed a key recovery attack on 11-round Speck32/64. In this section, we first perform the same attack using our new ND_k with $k = 2$. Then we present a deeper discuss.

7.3.1. Key Recovery Attack on 11-round Speck32/64

The target of this attack is to recover the last two subkeys (sk_{11}, sk_{10}). This attack returns a pair of subkey guesses (kg_{11}, kg_{10}). If $kg_{11} = sk_{11}$ and kg_{10} is different from sk_{10} at most 2 bits, this attack is viewed as a success [1].

Experiment settings. A 6-round and 7-round $ND_{k=2}$ are built over $\alpha = (0x40, 0x0)$. A prepended 3-round differential is extended from a 2-round differential $\Delta P = (0x211, 0xa04) \xrightarrow{p_0=2^{-6}} \alpha = (0x40, 0x0)$. Six neutral bits $\{14, 15, 20, 21, 22, 23\}$ are used to generate plaintext structures consisting of 64 plaintext pairs. The data reuse strategy is also adopted by letting $MRF = 2$ and $SSD = 1$.

The whole attack is performed as

1. Randomly generate 100 plaintext pairs with a difference ΔP.
2. Generate 100 plaintext structures using 6 neutral bits above, and collect corresponding ciphertext structures.
3. For each ciphertext structure:
   (a) collect possible kg_{11} using the method introduced in Section 3.2.
   (b) For each possible kg_{11}:
      i. Decrypt the current ciphertext structure with kg_{11}.
      ii. Collect possible subkey guess pairs (kg_{11}, kg_{10}) using the method introduced in Section 3.2.
4. Return surviving (kg_{11}, kg_{10}) with the highest rank score as the final subkey guess.

In Section 3.2, we have reviewed how Gohr's attack recovers the subkey sk_{r+1} with an r-round ND. This method needs a rank score threshold. In step 3a and 3(b)ii, we need a threshold c_3, c_4 respectively. In this paper, let $c_3 = 18$ and $c_4 = 150$.

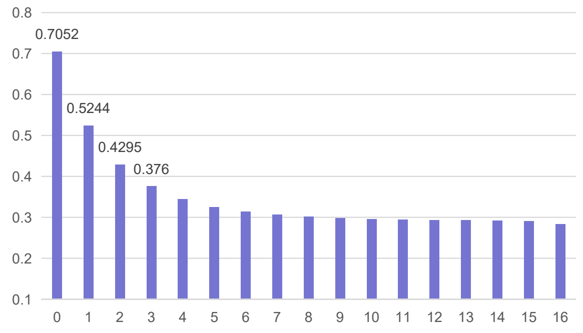

FIGURE 4. The expectations of the conditional posterior probability (Equation 21) of Gohr's 6-round ND against Speck32/64.

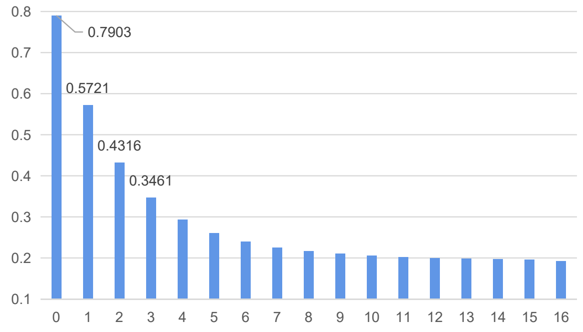

FIGURE 5. The expectations of the conditional posterior probability (Equation 21) of our 6-round $ND_{k=2}$ against Speck32/64.

Experiment results. The key recovery attack was performed 1000 times. Under the settings above, this attack was successful in 527 out of 1000 trials. **Using the same ciphertexts**, the attack based on Gohr's NDs was also performed 1000 times. As a comparison, this attack was successful in 521 out of 1000 trials.

7.3.2. Posterior Probability Comparison

We have proved that our ND is applicable to Gohr's attack. Moreover, in the 1000 trials above, the attack based on our NDs achieved a higher success rate.

To check whether the higher success rate is necessarily obtained using our new NDs, we perform a deeper analysis from the perspective of the key rank score.

Consider a $(r + 1)$-round cipher E. We first build a r-round ND based on a difference α. Then we collect numerous ciphertext pairs corresponding to plaintext pairs with a difference α. We decrypt these ciphertext pairs with a subkey guess kg, and feed the partially decrypted ciphertext pairs into the ND.

Let tk denote the true subkey of the $(r + 1)$-round. Besides, the Hamming distance between tk and kg is d. We focus on the expectation of the following conditional

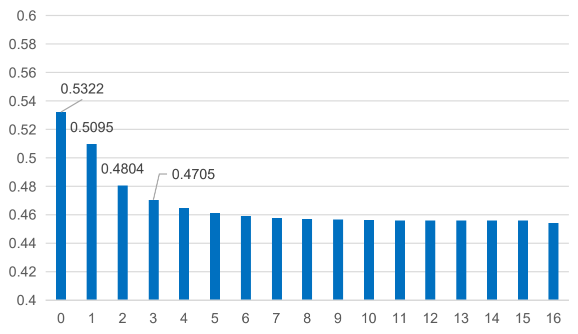

FIGURE 6. The expectations of the conditional posterior probability (Equation 21) of Gohr's 7-round ND against Speck32/64.

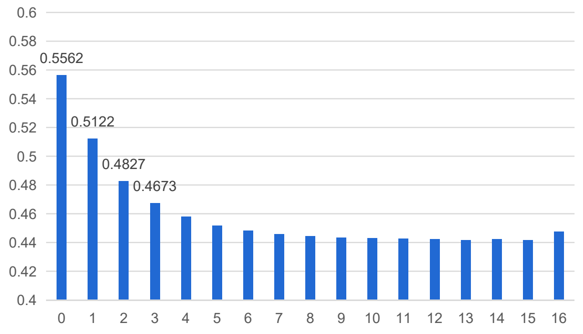

FIGURE 7. The expectations of the conditional posterior probability (Equation 21) of our 7-round $ND_{k=2}$ against Speck32/64.

posterior probability

$$Z = Pr\left(Y = 1\left|X, d\right.\right) = F\left(X\right) \tag{21}$$

where X is the input of the ND, and F is the ND. More exactly, if the ND is Gohr's ND, X is a decrypted ciphertext pair. If the ND is our ND (eg. ND_k), X is a ciphertext group consisting of k decrypted ciphertext pairs.

Taking $ND_{k=2}$ against Speck32/64 reduced to $6, 7$ rounds as examples, we estimate the expectations of the above conditional posterior probability. As a comparison, we also estimate the expectations based on Gohr's NDs. The final estimation results are shown in Figure 4, Figure 5, Figure 6, Figure 7.

There are two important phenomena. First, compared with Gohr's NDs, our NDs brings higher expectations $Pr(Y = 1|X, d = 0)$. Second, the value of $Pr(Y = 1|X, d = 0) - Pr(Y = 1|X, d = i), i \in [1, 3]$ increases.

The first phenomenon makes that a high key rank score ($c_3 = 18, c_4 = 150$) threshold is applicable. The second phenomenon makes the gap between the rank score of the true key and that of wrong keys increase. By setting a high key rank score threshold, wrong keys are less likely to obtain a key rank score higher than the threshold. Thus, a higher success rate is more likely to be obtained by replacing Gohr's NDs with our NDs.

8. CONCLUSIONS

In this paper, we focus on the neural distinguisher which is the core module in neural aided cryptanalysis. Inspired by works in the machine learning community, we propose a new neural distinguisher considering multiple ciphertext pairs simultaneously. We perform a deep exploring of this new neural distinguisher, and show its superiorities.

First, Compared with the neural distinguisher considering a single ciphertext pair, this new neural distinguisher achieves higher distinguishing accuracy. This advantage is verified through applications to five different ciphers. Second, except for features hidden in a single ciphertext pair, this new neural distinguisher can capture derived features from multiple ciphertext pairs, which is the root reason for the accuracy promotion. Derived features are detected by a newly proposed verification framework consisting of two tests: False Negative Test, False Positive Test.

Our new neural distinguisher can be applied to neural aided key recovery atacks. Due to the accuracy promotion, it can be used to reduce the data complexity of the neural aided statistical attack. A data reuse strategy is proposed to strengthen this advantage. As for the key recovery attack proposed by Gohr, compared with Gohr's neural distinguisher, our new neural distinguisher can bring a success rate promotion, which is verified by the attack against 11-round Speck32/64.

Our new neural distinguisher is full of potential. In the future, as long as neural aided key recovery attacks are related to the performance of neural distinguishers, our new neural distinguisher is a priority choice. Besides, our neural distinguisher also introduces a novel cryptanalysis direction by considering multiple ciphertext pairs simultaneously.

9. DATA AVAILABILITY

The data underlying this article are available in the article.

ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (2018YFB0803405, 2017YFA0303903).

REFERENCES

[1] Gohr, A. (2019) Improving attacks on round-reduced speck32/64 using deep learning. *international cryptology conference*, **11693**, 150–179.

[2] Chen, Y. and Yu, H. (2020). Neural aided statistical attack for cryptanalysis. Cryptology ePrint Archive, Report 2020/1620.

[3] Baksi, A., Breier, J., Dong, X., and Chen, Y. (2020) Machine learning assisted differential distinguishers for lightweight ciphers. *IACR Cryptol. ePrint Arch.*, **2020**, 571.

[4] Benamira, A., Gerault, D., Peyrin, T., and Tan, Q. Q. (2021) A deeper look at machine learning-based cryptanalysis. *IACR Cryptol. ePrint Arch*, **287**, 2021.

[5] Beaulieu, R., Shors, D., Smith, J., Treatmanclark, S., Weeks, B., and Wingers, L. (2015) The simon and speck lightweight block ciphers. *design automation conference*, **175**, 1–6.

[6] Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., and Verbauwhede, I. (2014) Chaskey: An efficient mac algorithm for 32-bit microcontrollers. *selected areas in cryptography*, **8781**, 306–323.

[7] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., and Vikkelsoe, C. (2007) Present: An ultra-lightweight block cipher. *cryptographic hardware and embedded systems*, **4727**, 450–466.

[8] Howard, R. (1987) Data encryption standard. *Information Age archive*, **9**, 204–210.

[9] Huang, S., Wang, X., Xu, G., Wang, M., and Zhao, J. (2017) Conditional cube attack on reduced-round keccak sponge function. *theory and application of cryptographic techniques*, **10211**, 259–288.

[10] He, K., Zhang, X., Ren, S., and Sun, J. (2016) Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

[11] Biham, E. and Chen, R. (2004) Near-collisions of SHA-0. In Franklin, M. K. (ed.), *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, Lecture Notes in Computer Science, **3152**, pp. 290–305. Springer.

[12] Lee, J., Heo, M., Kim, K., and Kim, C. (2018) Single-image depth estimation based on fourier domain analysis. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 330–339.

[13] Schuldt, C., Laptev, I., and Caputo, B. (2004) Recognizing human actions: a local svm approach. *international conference on pattern recognition*, **3**, 32–36.

[14] Tosi, F., Aleotti, F., Poggi, M., and Mattoccia, S. (2019) Learning monocular depth estimation infusing traditional stereo knowledge. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9799–9809.

[15] Chen, Y., Yu, L., Ota, K., and Dong, M. (2019) Hierarchical posture representation for robust action recognition. *IEEE Transactions on Computational Social Systems*, **6**, 1115–1125.

[16] Kingma, D. P. and Ba, J. (2015) Adam: A method for stochastic optimization.

[17] Francois, C. e. a. (2015). Keras.

[18] Abed, F., List, E., Lucks, S., and Wenzel, J. (2014) Differential cryptanalysis of round-reduced simon and speck. *International Workshop on Fast Software Encryption*, pp. 525–545. Springer.

[19] Meiqin, W. (2009) Differential cryptanalysis of present. *IACR eprint.*

[20] Biham, E. and Shamir, A. (1993) *Differential Cryptanalysis of the Data Encryption Standard.* Springer.

[21] Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., and Verbauwhede, I. (2014) Chaskey: an efficient mac algorithm for 32-bit microcontrollers. *International Conference on Selected Areas in Cryptography*, pp. 306–323.

[22] Biham, E. and Shamir, A. (1991) Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, **4**, 3–72.