ISBN: xxxxxxxx
2021 the 11th International Workshop on Computer Science and Engineering (WCSE 2021)
doi: xxxxxxxx

Neural-Linear Attack Based on Distribution Data and Its Application on DES

Rui Zhou [1,2], Ming Duan [1,2], Qi Wang[1], Qianqiong Wu[1], Sheng Guo[1], Lulu Guo[1,2] and Zheng Gong[3]

[1] Information Engineering University, Zhengzhou, China
zhourui110.love@163.com, mdscience@sina.com
[2] Henan Key Laboratory of Network Cryptography, Zhengzhou, China
[3] South China Normal University, Guangzhou, China

Abstract. The neural-differential distinguisher proposed by Gohr boosted the development of neural aided differential attack. As another significant cryptanalysis technique, linear attack has not been developing as rapidly in combination with deep learning technology as differential attack. In 2020, Hou et al. proposed the first neural-linear attack with one bit key recovery on 3, 4 and 5-round DES and restricted multiple bits recovery on 4 rounds, where the effective bits in one plain-ciphertext pair are spliced as one data sample. In this paper, we compare the neural-linear cryptanalysis with neural-differential cryptanalysis and propose a new data preprocessing algorithm depending on their similarities and differences. We call the new data structure *distribution data*. Basing on it, we mount our key recovery on round-reduced DES—first, we raise the accuracy of the neural-linear distinguisher by about 50%. Second, our distinguisher improves the effectiveness of one bit key recovery against 3, 4 and 5-round DES than the former one, and attack 6-round DES with success rate of 60.6% using 2048 plain-ciphertext pairs. Third, we propose a real multiple bit key recovery algorithm, leading neural-linear attack from theory to practice.

Keywords: Linear cryptanalysis, Neural-linear attack, Deep learning, Data preprocessing, DES

1. Introduction

As deep learning has been widely used and outstands in various tasks, its security and privacy problems becomes a concern[1], e.g., modify the logistic regression algorithm to preserve privacy[2], and use facial key templates in homomorphic encryption[3].

Meanwhile, scholars also exploit neural networks to help cryptanalysis. In 2019, Gohr used deep learning technology to build differential distinguishers, which is the first successful combination of deep learning and traditional cryptanalysis[4]. The essential of this neural distinguisher is binary classification: distinguishing the specific differential distribution from random differential distribution. On the one hand, various methods mushroomed afterwards to improve the neural distinguisher or to widen its applications — Su et al. applied neural distinguishers to polytope differential attack[5]. Bao et al. focused on neutral bits and attacked more rounds[6]. Hou et al. transformed the input data to improve the distinguisher's accuracy[7]. Chen et al. further proposed a neural aided statistical attack based on the differential cryptanalysis[8] [9]. On the other hand, scholars attempted to explain the inherent workings of this neural aided cryptanalysis and its deeper mechanism. Analysis on Gohr's distinguisher to boost simulation on non-Markov ciphers[10]. An interpretability of deep neural networks is provided by Benamira et al. from both cryptanalysis and machine learning perspectives[11]. Lu et al. conclude technically the data structure used in training neural networks[12].

With the development of the neural aided differential attacks above, how to combine linear attack with deep learning has become a natural question. Linear attack is another strong cryptanalysis technique on symmetric-key ciphers like differential attack. First proposed in 1993 by Matsui[10][11], it is still one of the most widely used attacks to test the security of primitives. Although neural aided differential attack has become a hit, researching on the neural-linear attack has been at a slow pace. Till 2020, Hou et al. first proposed a linear attack on DES using neural distinguisher[15]. Yet, neural distinguishers in Hou's paper didn't achieve better result than traditional linear attack in one bit key recovery, nor is the multiple bits key recovery close to practical use. Therefore, there remains an open question that how to use deep learning to improve traditional linear attack.

Our Contributions. Determined by the essential of linear cryptanalysis, binary classification neural network can be applied to linear attack. We show by experiments that the neural distinguisher cannot extract linear feature from direct plain-ciphertext but from a new form of data: distribution data, which is produced by a new method of data preproccessing we propose. Different from any other neural aided cryptanalysis, one sample of our data contains multiple plain-ciphertext pairs information. This new data preprocessing method is derived from comparison. Compared with former neural-linear distinguisher, our distinguisher trained with distribution data has the following advantages:

1. The accuracy of the neural distinguisher rises, which indicates that our distinguisher has a better capability of learning linear feature.

2. We mount one-bit key recovery with higher success rate and fewer plain-ciphertext pairs than the former, and even surpass the traditional method on 5 round.

3. We mount real multiple bits key recovery on 4 and 5 round DES using the same distinguisher as in one bit key recovery. Thus, we simplify the neural aided multiple bits key recovery with less space and computation time.

Still, we are going to take advantage of distribution data to improve the accuracy further and launch key recovery on longer round (e.g., train multiclass classification neural network).

Outline. In Section 2, we introduce significant notations that will be used in the rest of this paper. And we will give a brief description of DES and linear cryptanalysis as well. In Section 3, we compare the similarity and difference between neural aided differential attack and linear attack as well as provide a new data processing method. We present the result and discussions of neural distinguisher training and key recovery attacks in Section 4 and Section 5 respectively. Finally, we conclude this paper and propose future work in Section 6.

2. Preliminaries

Table 1 presents the notations we use throughout the paper.

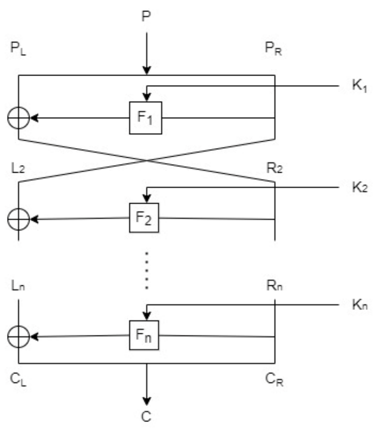
Table 1: Notations in this paper.

Notations	Description
$\oplus$	Bitwise XOR operation.
$\lll$	Left bit rotation.
$\Vert$	Concatenation operation.
$\sim$	Bitwise NOT operation.
$X = (x_{n-1}, x_{n-2}, \dots, x_0)$	An n-bit binary vector, where x_0 is the right most bit.
$X[i] = x_i$	The i-th bit position of X
$X[i, j, \dots, k]$	$x_i \oplus x_j \oplus \dots \oplus x_k$
P	The 64-bit plaintext
C	The corresponding 64-bit ciphertext
P_L	The left 32-bit of P
P_R	The right 32-bit of P
C_L	The left 32-bit of C
C_R	The right 32-bit of C
K_r	The subkey used in r-th round.
L_r	Linear approximate equation.
$P_r(L_r)$	The probability of L_r.
ND_{L_r}	Neural distinguisher trained with L_r

2.1. A Brief Description of DES

DES[16] is the first cryptographic primitive attacked by linear cryptanalysis. As a typical Feistel-structured block cipher, DES is chosen as object primitive to apply our neural-linear attack on. Its block size is 64 bits and master key is 56 bits long. In this paper, we omit the initial permutation IP and the final permutation IP^{-1}.

Fig. 1: DES cipher.



The nonlinear function $F: F_2^n \rightarrow F_2^n$ is defined as follows:

$$F(X) = P(S(E(X) \oplus K)) \tag{1}$$

where E, S, P are expansion operation, S-box operation and permutation respectively.

2.2. Linear Attack

Linear attack is one of the known-plaintext attacks. Given plaintext P and corresponding r-th round output C, we need to obtain linear approximate equation L_r, whose probability deviates from 1/2, and recover the key bits by statistical distribution. Traditional linear attack on DES proposed by Matsui includes two algorithms from paper[13]. *Algorithm 1* can recover 1 bit subkey, and *Algorithm 2* can recover multiple bits at one time. The following α, β, γ, μ and ν are all bitwise masks.

One bit key recovery: The linear approximate equation is a pure linear equation. The left side of the equation is the XOR of effective plain-ciphertext bits, the right side of the equation is the XOR among effective key bits.

$$L_r : \alpha \cdot P \oplus \beta \cdot C = \gamma \cdot K \tag{2}$$

Algorithm1 can only recover 1 bit $\gamma \cdot K$ on the right side of the equation.

Multiple bits key recovery: The chosen linear approximate equation is a nonlinear equation with a nonlinear part, usually derived from a shorter round linear equation (2).

$$L_r^{'} : \alpha \cdot P \oplus \beta \cdot C \oplus \mu \cdot F(P, K_1) \oplus \nu \cdot F(P, K_r) = \gamma \cdot K \tag{3}$$

Algorithm2 can recover not only 1 bit $\gamma \cdot K$, but also effective bits in K_1,K_r according to $\mu \cdot F(P, K_1) \oplus \nu \cdot F(P, K_r)$ on the left side.

Assuming that the equation L_r holds with the probability $P_r(L_r) = p_{L_r}$ for each K, when $\gamma \cdot K = 0$, $P_r(\alpha \cdot P \oplus \beta \cdot C = 0) = p_{L_r}$, $P_r(\alpha \cdot P \oplus \beta \cdot C = 1) = 1 - p_{L_r}$; when $\gamma \cdot K = 1$, $P_r(\alpha \cdot P \oplus \beta \cdot C = 1) = p_{L_r}$, $P_r(\alpha \cdot P \oplus \beta \cdot C = 0) = 1 - p_{L_r}$. As a result, the left side of the equation is a binomial distribution, of which linear attack take advantage.

3. Data Preprocessing

As we want to apply the binary classification distinguisher to linear attack, we first need to reduce it into a binary classification problem. According to Section 2.2, we find it naturally that the two different distributions when $\gamma \cdot K = 0$ or 1 can be chosen as our classification object. Meanwhile, basing on the supervised learning mode, the training data involve two parts—sample X and corresponding label Y.

3.1. Pre-experiments

The first question we need to answer is that: Can Residual Network directly learn the linear feature from plain-ciphertext? For this purpose, we conduct two experiments: A sample is N-bit binary string, of which first $(N-4)$ bits are generated randomly and the last 4 bits are XOR of two bits chosen from the front $(N-4)$ bits. The specific data generation is shown in the following table. In this way, there are linear relations inside data.

Table 2: Accuracy of neural distinguisher trained with different data structures.

Experiment	Sample X	Label Y	Accuracy
a	64-bit 0-1 string $(x_{63}, x_{62}, \dots, x_0)$, where $x_0, \dots, x_{59}$ are random, and $x_{60} = x_0 \oplus x_1, x_{61} = x_2 \oplus x_3, x_{62} = x_4 \oplus x_5, x_{63} = x_6 \oplus x_7$	1	0.501
	64-bit 0-1 string $(x_{63}, x_{62}, \dots, x_0)$, where $x_0, \dots, x_{59}$ are random, and $x_{60} = \sim(x_0 \oplus x_1), x_{61} = \sim(x_2 \oplus x_3), x_{62} = \sim(x_4 \oplus x_5), x_{63} = \sim(x_6 \oplus x_7)$	0	
b	32-bit 0-1 string $(x_{31}, x_{30}, \dots, x_0)$, where $x_0, \dots, x_{27}$ are random, and $x_{28} = x_0 \oplus x_1, x_{29} = x_2 \oplus x_3, x_{30} = x_4 \oplus x_5, x_{31} = x_6 \oplus x_7$	1	0.496
	32-bit 0-1 string $(x_{31}, x_{30}, \dots, x_0)$, where $x_0, \dots, x_{27}$ are random, and $x_{28} = \sim(x_0 \oplus x_1), x_{29} = \sim(x_2 \oplus x_3), x_{30} = \sim(x_4 \oplus x_5), x_{31} = \sim(x_6 \oplus x_7)$	0	

Table 2 shows the average accuracy of the neural distinguisher from repeated experiments a and b, both of which and even all the accuracy we obtained is close to 0.5, indicating that the neural distinguisher fails to learn the linear features of the data. Next, we use just real plain-ciphertext, namely, do not have data prepocessing phase, to train the binary classification neural distinguisher and it fails again.

3.2. Data Structures in Neural-Aided Attacks

We recall the plain-ciphertext data prepocessing in paper[15]. For one-bit key recovery, the effective text bits were extracted as one sample ($P[i_0], P[i_1], \dots, P[i_m]$ and $C[j_0], C[j_1], \dots, C[j_n]$ are effective plain-cipher text bits in a linear approximate equation L_r in Table 3).

Table 3: Data structure for one-bit key recovery.

Sample X	Label Y
Binary string with fixed length: $(P[i_0] \parallel P[i_1] \parallel \cdots \parallel P[i_m] \parallel C[j_0] \parallel C[j_1] \parallel \cdots \parallel C[j_n])$	$\gamma \cdot K$

The accuracy of the trained distinguisher is higher than 0.5, but the success rate is lower than that of the traditional model.

For multi-bits key recovery, assuming that $\gamma \cdot K$ is known in a linear approximate equation L'_r, we extract effective bits in left-side plaintext P_L, right-side plaintext P_R, left-side ciphertext C_L, right-side ciphertext C_R, $F_1(P, K_1)$, $F_r(C, K_r)$: $P_L[i_0], P_L[i_1], \dots, P_L[i_m]$, $P_R[i'_0], P_R[i'_1], \dots, P_R[i'_u]$, $C_L[j_0], C_L[j_1], \dots, C_L[j_n]$, $C_R[j'_0]$, $C_R[j'_1], \dots, C_R[j'_v]$, $F_1(P, K_1)[k_0], \dots, F_1(P, K_1)[k_s]$ and $F_r(C, K_r)[l_0], \dots, F_r(C, K_r)[l_t]$. A sample labelled 1 is composed of 6 parts above, and 0s are padded in the end of each to keep these 6 parts in the same length, while the samples labelled 0 substitute the last 2 parts with random data.

Table 4: Data structure for multi-bits key recovery.

Sample X	Label Y
Binary string: $(P_L[i_0] \parallel P_L[i_1] \parallel \cdots \parallel P_L[i_m] \parallel P_R[i'_0] \parallel P_R[i'_1] \parallel \cdots \parallel P_R[i'_u] \parallel C_L[j_0] \parallel C_L[j_1] \parallel \cdots \parallel C_L[j_n] \parallel C_R[j'_0] \parallel C_R[j'_1] \parallel \cdots \parallel C_R[j'_v] \parallel F_1(P, K_1)[k_0] \parallel \cdots \parallel F_1(P, K_1)[k_s] \parallel F_r(C, K_r)[l_0] \parallel \cdots \parallel F_r(C, K_r)[l_t])$	1
Binary string: $(P_L[i_0] \parallel P_L[i_1] \parallel \cdots \parallel P_L[i_m] \parallel P_R[i'_0] \parallel P_R[i'_1] \parallel \cdots \parallel P_R[i'_u] \parallel C_L[j_0] \parallel C_L[j_1] \parallel \cdots \parallel C_L[j_n] \parallel C_R[j'_0] \parallel C_R[j'_1] \parallel \cdots \parallel C_R[j'_v] \parallel random \parallel random)$	0

What could be improved with the data structure above is that $\gamma \cdot K$ is known as the premise and padding 0s will also reduce the information entropy.

Based on our experiments a, b and Hou's experiments, we believe that a more suitable method of data preprocessing should be designed according to the characteristic of linear attacks.

In section 2.2, we state that linear attack focus on binomial distribution over F_2. While in Gohr's neural-differential attack, the differential distribution is a multinomial distribution. The classification target of Gohr's neural distinguisher is to determine whether a difference belongs to an output differential distribution with a fixed input difference or a random distribution. Experiments A and B from paper[11] indicates that Gohr's neural distinguisher generally relies on the probability of (ciphertext, penultimate and antepenultimate-round) differences in the full space F_2^{32} — the higher the probability of its difference is, the higher score it got from the distinguisher, and the more likely it is to be judged as a real ciphertext pair. And vice versa, some differences' probability is 0.

In comparison, it is infeasible for the neural distinguisher to identify whether it belongs to a specific binomial distribution based on the probability of one piece of plain-ciphertext, like what is done in neural-differential attack, because there are only two events in a binomial distribution with respective probabilities: p_{L_r} and $1 - p_{L_r}$, which have negligible deviation with each other. If one sample only contains one-bit value from F_2, there is a great chance that the distinguisher will misjudge. Hence, we state the assumption that *samples containing the information from multiple plain-ciphertext pairs work better than those from one plain-ciphertext pair in neural aided linear attack*, which we will verify with later experiments.

3.3. Distribution Data

Next, in what form does a sample contain multiple plain-ciphertext pairs? Gohr directly use the ciphertext pairs denoted as a vector: (C_L, C_R, C'_L, C'_R), and many researches have been done to figure out the effectiveness that different methods of data preprocessing have on neural aided cryptanalysis. These attempts are described as matrix multiplication with ciphertext-pair vector (C_L, C_R, C'_L, C'_R) in paper[12]. Another way to improve the accuracy of neural distinguishers is add more text pairs in one sample.

Similarly, We also put the XOR operation of effective text bits forward in data preprocessing: $(\alpha, \beta) \cdot (P, C)^T = \alpha \cdot P \oplus \beta \cdot C$, and the dot product of mask and vector can be seen as matrix multiplication reversely. The following is the new data preprocessing algorithm:

```
Algorithm 1: A new data preprocessing algorithm for neural-linear attack
Input: Master key $K$
       $N$ pieces of plaintext $P$: $P_1, P_2, \dots, P_N$
Output: A sample $X$ with corresponding label $Y$
1. Compute $\gamma \cdot K$ as the label Y depending on the linear approximate equation $L$.
2. Encrypt $P$ with $K$ and obtain $N$ pieces of cipertext $C$: $C_1, C_2, \dots, C_N$.
3. For the $N$ pairs of plain-ciphertext, compute the value (0 or 1) of $\alpha \cdot P \oplus \beta \cdot C$ according to the linear
   equation $L$. Splice the $N$ values into an $N$-bit string as sample $X$.
```

The data we obtain after preprocessing are in the form: a N-bit string as a sample with 1-bit label, and the data structure is shown in table 5. One sample contains information from N pairs of plain-ciphertext, which is a partial binomial distribution itself. Thus, we call it distribution data. The former training data forms a distribution, whereas we take distributions as data. And in rest of the paper, we will generate our training data through Algorithm 1 and mount a key recovery attack on round-reduced DES with trained linear distinguisher.

Table 5: Distribution data.

Sample X	Label Y
N-bit binary string: $(\alpha \cdot P_1 \oplus \beta \cdot C_1 \parallel \alpha \cdot P_2 \oplus \beta \cdot C_2 \parallel \cdots \parallel \alpha \cdot P_N \oplus \beta \cdot C_N)$	$\gamma \cdot K$

4. Training the Neural Linear Distinguisher

4.1. Network Structure

The neural network used in differential distinguisher[4] and linear distinguisher[15] is Residual Network, whose main component is iterated Residual Towers. The convolution blocks can reduce the computational cost when deepening the network, which is a highly effective deep learning model. The main structure of the network we use remains unchanged, and the input layer is modified according to data structure (Fig. 2). We change the reshape layer from a 4×16 matrix to an 8-column matrix. The input strings are N-bit in length, and passed through the reshape layer changed into $(N/8) \times 8$ matrices. The last two dense layers contain 32 neurons each. And the output layer contains 1 neuron.

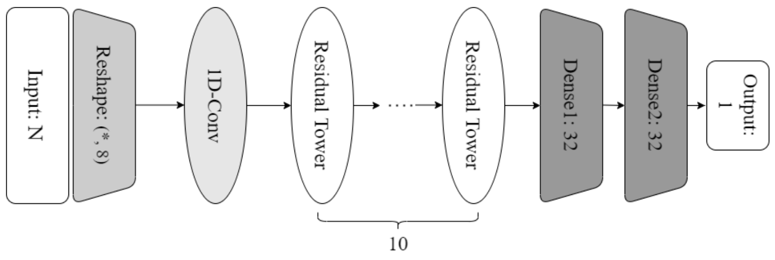


Fig. 2: An overview on the network structure.

4.2. Training Process

We generate training data depending on 3-round, 4-round, 5-round and 6-round linear equations: L_3, L_4, L_5, L_6. And their respective probabilities are $P_r(L_3) = 0.7$, $P_r(L_4) = 0.439$, $P_r(L_5) = 0.519$, $P_r(L_6) = 0.4962$. The training data are divided into training data set and validation data set, whose sizes are shown in the following table. When executing the experiments, we find that the convergence speed of training is moderate when the training epoch is 50 and the batch size is 500. Table 6 presents the validation accuracy on the 4 rounds. In addition, we take round 3 and round 4 for example, to show the accuracy varying with different value of input strings' length N (Fig. 3).

$$L_3 : P_L[7,18,24,29] \oplus P_R[15] \oplus C_L[7,18,24,29] \oplus C_R[15] = K_1[22] \oplus K_3[22] \tag{4}$$

$$\begin{aligned} L_4 : & P_L[7,18,24,29] \oplus P_R[15] \oplus C_L[15] \oplus C_R[7,18,24,27,28,29,30,31] \\ & = K_1[22] \oplus K_3[22] \oplus K_4[42,43,45,46] \end{aligned} \tag{5}$$

$$\begin{aligned} L_5 : & P_L[15] \oplus P_R[7,18,24,27,28,29,30,31] \oplus C_L[15] \oplus C_R[7,18,24,27,28,29,30,31] \\ & = K_1[42,43,45,46] \oplus K_2[22] \oplus K_4[22] \oplus K_5[42,43,45,56] \end{aligned} \tag{6}$$

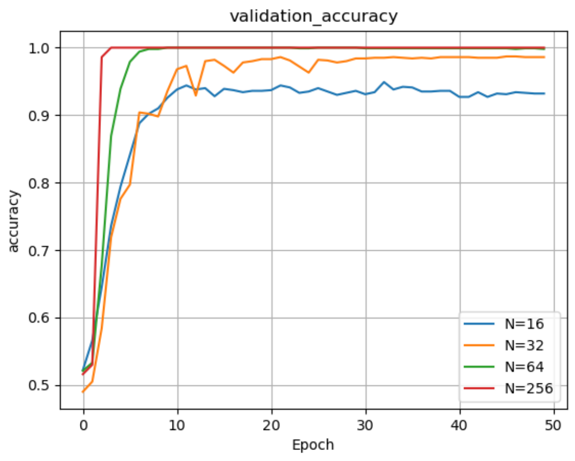
$$L_6 : P_R[7,18,24] \oplus C_L[7,18,24,29] \oplus C_R[15] = K_2[22] \oplus K_3[44] \oplus K_4[22] \oplus K_6[22] \tag{7}$$

Table 6: Accuracy of two neural distinguishers on four linear equations. We show the accuracy range of 2000 times test.

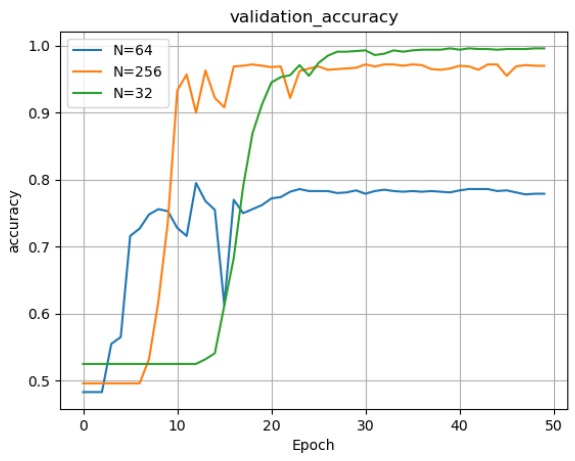
Round	Source	N[1]	Train data[2] (training data set + validation data set)	Accuracy
3	This paper	32	$10^4 + 10^3$	0.99 ± 0.009
		64	$10^4 + 10^3$	1.000
		256	$10^4 + 10^3$	1.000
	[15]	-	10^5	0.6723
4	This paper	64	$10^4 + 10^3$	0.815 ± 0.02
		256	$10^4 + 10^3$	0.970 ± 0.01
		512	$10^4 + 10^3$	0.996 ± 0.02
	[15]	-	10^6	0.5375
5	This paper	256	$10^4 + 10^3$	0.658 ± 0.02
		512	$10^4 + 10^3$	0.749 ± 0.02
		1024	$10^5 + 2 \times 10^3$	0.856 ± 0.01
	[15]	-	10^6	0.5128
6	This paper	512	$10^4 + 10^3$	0.546 ± 0.01
		1024	$10^5 + 2 \times 10^3$	0.57 ± 0.015

[1] N is the length of input samples.

[2] The data size in this paper is described in the form of `size of the training data set' + `size of the validation data set'.



(a): Validation accuracy on L_3.



(b): Validation accuracy on L_4.

Fig. 3: Training accuracy with different N.

Discussions. With the new data preprocessing method, we successfully improve the accuracy of the distinguisher comparing to Hou's result on L_3, L_4, L_5 and add the round that the neural distinguisher can identify to 6. Furthermore, we find that the accuracy declines with the decrease of the probability p_L according to table 6. And figure 3 demonstrates that on the same linear equation, the longer the input string is, the better the distinguisher learns. These experiment phenomena are consistent with our expectations as we assume that feature of the binomial distribution is what the distinguisher learns.

5. Key Recovery

5.1. One Bit Key Recovery

As the linear distinguishers have been trained in section 4 with fixed input length N, we need at least N pairs of plain-ciphertext when mounting key recovery. Similar to the data preprocessing, we compute the values on the left side of linear approximate equation to obtain an N-bit string. Feed it to the trained distinguisher and we get a score Z ranging from 0 to 1.

Algorithm 2: One bit key recovery algorithm

```
Input: $N$ pairs of plain-ciphertext encrypted by a same master key
       Neural distinguisher $ND_{L_r}$ trained by $L_r$
Output: one-bit value of $\gamma \cdot K$
1. For the $N$ pairs of plain-ciphertext, compute the value (0 or 1) of $\alpha \cdot P \oplus \beta \cdot C$ according to the linear
   approximate equation $L_r$. Splice the $N$ values into an $N$-bit string $S$.
2. $Z \leftarrow ND_{L_r}(S)$
3. If $Z \geq 0.5$ then
4.     Return $\gamma \cdot K = 0$
5. Else
6.     Return $\gamma \cdot K = 1$
7. End if
```

In case that we have more than N pairs of known plain-ciphertext, subtle changes can be made on Algorithm 2: we take N pairs as a group and get multiple strings S in step 1. In step 2, we get multiple scores and take their average as the final score Z.

In practical experiment, we randomly generate the master key and the plaintext. Table 7 shows the comparison among traditional way, neural-linear attack in paper [15] and our approach in one bit key recovery. The success rate of traditional way is the theoretical result calculated by expression (9) from paper [13]. And we put the average success rate in 200 times tests in table 7.

Table 7: Success rate of one bit key recovery on 3, 4, 5 and 6-round DES

Round	Source	Tarin data	Number of plain-ciphertext pairs[1]	Success rate
3	[13]	-	30	98.5%
	[15]	$10^4 + 10^3$	32	95%
	This paper	$10^4 + 10^3$	32×1	98.5%
4	[13]	-	448	99.5%
	[15]	$10^4 + 10^3$	633	99%
	This paper	$10^4 + 10^3$	256×1	96.6%
	This paper	$10^4 + 10^3$	512×1	99.5%
5	[13]	-	4617	90%
	[15]	$10^6 + 5 \times 10^4$	8631	90%
	This paper	$10^4 + 10^3$	1024×1	84.28%
	This paper	$10^4 + 10^3$	1024× 2	91.15%
6	[13]	-	34627	92.1%
	This paper	$10^5 + 2 \times 10^3$	1024×2	60.6%

[1] The number of pairs in this paper is described in the form of 'N' × 'number of the groups'.

Discussions. The neural-linear distinguisher trained in this paper shows its advantages in one bit key recovery. In the situation that we have N pairs of plain-ciphertext (i.e., the number of the group is 1). the success rate is slightly lower than the accuracy of training, and the difference is within 0.02. The success rate will increase slightly when we add more groups of plain-ciphertext—Hence, in key recovery of 5 and 6 round, we increased the number of the groups instead of training new distinguisher. Compared with Hou's linear distinguisher, our new distinguisher not only has higher accuracy, but also get higher success rate with fewer plain-ciphertext pairs. Besides, it also surpasses the traditional method in 5-th round.

5.2. Multiple Bits Key Recovery

We still use the trained distinguisher in section 4 for multiple bits key recovery. Linear approximate equation L'_4, L'_5 are derived from L_3, L_4, keeping the probability, i.e., $P_r(L'_4) = P_r(L_3)$, $P_r(L'_5) = P_r(L_4)$. The nonlinear part of both is $F(P_R, K_1)[15]$, effecting 6 bits: $K_1[42]$, $K_1[43]$, $K_1[44]$, $K_1[45]$, $K_1[46]$, $K_1[47]$:

$$\begin{aligned} L'_4 &: P_L[15] \oplus P_R[7,18,24,29] \oplus C_L[7,18,24,29] \oplus C_R[15] \oplus F(P_R, K_1)[15] \\ &= K_2[22] \oplus K_4[22] \end{aligned} \tag{8}$$

$$\begin{aligned} L'_5 &: P_L[15] \oplus P_R[7,18,24,29] \oplus C_L[15] \oplus C_R[7,18,24,27,28,29,30,31] \oplus F(P_R, K_1)[15] \\ &= K_2[22] \oplus K_4[22] \oplus K_5[42,43,45,46] \end{aligned} \tag{9}$$

From the result of one bit key recovery, the distinguisher we trained can classify two binomial distributions: $B(N, p_L)$ and $B(N, 1 - p_L)$. The multi-bit linear approximate equation holds the same probability with one-bit ones, therefore we consider the classification object keeps the same in multiple key recovery. So we did not design new distinguisher. In fact, we trained new distinguisher depending on L'_4 and L'_5, the accuracy and its performance in key recovery remains similar, which also demonstrates it is the binomial distribution that the distinguisher classifies.

In multiple key recovery, we recover not only one-bit $\gamma \cdot K$ on the right side, but also six effective key bits in nonlinear function on the left side. Our multiple key recovery algorithm contains two steps accordingly.

When recovering left-side effective key bits, we consider the distinguisher can classify the two real binomial distributions but cannot identify the distribution when the key is wrong. So, we assume that the more the score deviates from 0.5, the more possible the key is real.

When recovering $\gamma \cdot K$, according to the way we set the label, we believe that if the score is close to 0, then $\gamma \cdot K$ is 0; otherwise, $\gamma \cdot K$ is 1.

Algorithm 3: Left-side multiple bits key rank algorithm

```
Input: r-round linear approximate equation $L'_r: \alpha \cdot P \oplus \beta \cdot C \oplus \mu \cdot F(P, K_1) \oplus \nu \cdot F(P, K_r) = \gamma \cdot K$
       $r$-round $n \times N$ pairs of plain-ciphertext encrypted by a same master key
       Neural distinguisher $ND_{L_{r-1}}$ trained by $L_{r-1}$
Output: $Rank_{key}$
1.  Divide the plain-ciphertext pairs into $n$ groups $G_1, G_2, \dots, G_n$, $N$ pairs a group.
2.  $l \leftarrow$ length of left-side effective key bits
3.  for $key$ in $2^l$ do
4.    for plain-ciphertext pairs in $\{G_1, G_2, \dots, G_n\}$ do
5.      Compute the value (0 or 1) of left side of $L'_r$
6.    end for
7.    Splice the $N$ values into an $N$-bit string $S$
8.    $Z_i \leftarrow ND_{L_r}(S)$
9.  end for
10.     $|C_{key}| = |\frac{\sum_{i=1}^{n} Z_i}{n} - \frac{1}{2}|$
11. Return $Rank_{key} \leftarrow sort\,|C_{key}|$ by descending value order
```

Algorithm 4: $\gamma \cdot K$ recovery algorithm

Algorithm A

```
Input: $L'_r$: $r$-round linear approximate equation
       $C_{K_{real}}$: score of real left-side effective key $K_{real}$
Output: one-bit value of $\gamma \cdot K$
1. If $C_{K_{real}} \geq 0.5$ then
2.     Return $\gamma \cdot K = 0$
3. Else
4.     Return $\gamma \cdot K = 1$
5. End if
```

Algorithm B

```
Input: $L'_r$: $r$-round linear approximate equation
       $C_{K_{max}}$: the max score $max|C_{K_i}|$ which corresponds to the first key $K\_max$ in $Rank_{key}$
Output: one-bit value of $\gamma \cdot K$
1. If $C_{K_{real}} \geq 0.5$ then
2.     Return $\gamma \cdot K = 0$
3. Else
4.     Return $\gamma \cdot K = 1$
5. End if
```

Unlike one bit key recovery, the group number n has negligible effect on the multiple bits key recovery through the experiment. We repeated 200 times and the average real key rank is around 30—slightly higher than random search. Algorithm A and B recover $\gamma \cdot K$ under different circumstances. If we have determined the real effective key on the left side, we can recover $\gamma \cdot K$ with high possibility using Algorithm A. And Algorithm B make use of the key rank obtained through Algorithm 3 (table 8).

Table 8: Multiple bits key recovery on 4 and 5-round DES.

Round	Distinguisher	Left-side multiple bits key rank	success rate of $\gamma \cdot K$ recovery	
			Algorithm A	Algorithm B
4	ND_{L_3}	30	100%	99%
5	ND_{L_4}	31	99%	63%

Discussions. We believe that our multiple bits key recovery surpasses Hou's in three aspects. Firstly, we add one more round in the recovery on DES than in paper [15]. Secondly, we did not retrain any new distinguisher but use the same one in one bit key recovery. And experiments had been executed to prove that both have the same effect. Therefore, our approach simplifies the recovery process and take up less space and precomputation time. Last but not least, we can get key rank by Algorithm 3 and $\gamma \cdot K$ by Algorithm B in sequence without any precondition, Hou's multiple bits key recovery attack is actually theoretical because there is a precondition that $\gamma \cdot K = 0$. Although Algorithm A is not feasible in practical for $\gamma \cdot K$ recovery, it proves that the distinguisher can identify $\gamma \cdot K$ according to real distribution data. To the best of our knowledge, this work is the first neural aided multiple bits key recovery that can be applied to practical linear attack. Future work will aim at improving Algorithm 3 to raise real key rankings.

6. Conclusion

In this paper, we mounted a new neural-linear attack on round-reduced DES. We compared the similarities and differences between neural aided differential and linear attack and confirmed our findings with experiments. With a new data preprocessing: we obtained distribution data as our training data and successfully improved the neural distinguisher's accuracy. Moreover, our distinguisher performs better than the state-of-art one in one bit key recovery: reaching higher success rate with fewer plain-ciphertext pairs. It is also capable in multiple bits key recovery, for which we proposed a practical but not theoretical key recovery algorithm.

Our results indicate that especial approaches are necessary to suit in linear attack when deep learning techniques are used. The work we've done reflects the comparison between linear attack and differential attack from a machine learning perspective, and more researches are needed to explain its deeper mechanism theoretically and propose even better methods. Our neural-linear attack can be applied to other cryptographic primitives, and we are going to modify the binary classification distinguisher into multiclass ones for multiple bits key recovery for better performance in the future.

7. References

[1] D. Harinath, P. Satyanarayana, M.V.R. Murthy. A Review on Security Issues and Attacks in Distributed Systems. *Journal of Advances in Information Technology*. (2017) 1–9.

[2] K. Chaudhuri, and C. Monteleoni. *Privacy-preserving logistic regression.* In: NIPS, 2008: pp. 289–296.

[3] T. Chandrasekhar, and S. Kumar. A Noval Method for Cloud Security and Privacy Using Homomorphic Encryption Based on Facial Key Templates. *Journal of Advances in Information Technology*. 13 (2022).

[4] A. Gohr. Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. In: A. Boldyreva, D. Micciancio (eds.). *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing, Cham, 2019: pp. 150–179.

[5] H.-C. Su, X.-Y. Zhu, D. Ming. Polytopic Attack on Round-Reduced Simon32/64 Using Deep Learning. In: Y. Wu, M. Yung (eds.). *Information Security and Cryptology*. Springer International Publishing, Cham, 2021: pp. 3–20.

[6] Z. Bao, J. Guo, M. Liu, L. Ma, Y. Tu. Conditional Differential-Neural Cryptanalysis. *Cryptology ePrint Archive*. Report 2021/719 (2021).

[7] Z. Hou, J. Ren, S. Chen. Improve Neural Distinguisher for Cryptanalysis. *Cryptology ePrint Archive*. Report 2021/1017 (2021).

[8] Y. Chen, H. Yu. Improved Neural Aided Statistical Attack for Cryptanalysis. *Cryptology ePrint Archive.* Report 2021/311 (2021).

[9] Y. Chen, Y. Shen, and H. Yu. Neural-Aided Statistical Attack for Cryptanalysis. *The Computer Journal*. 2022.

[10] A. Baksi, J. Breier, Y. Chen, X. Dong. Machine Learning Assisted Differential Distinguishers For Lightweight Ciphers (Extended Version). *Cryptology ePrint Archive*. Report 2020/571 (2020).

[11] A. Benamira, D. Gerault, T. Peyrin, Q.Q. Tan. A Deeper Look at Machine Learning-Based Cryptanalysis. In: A. Canteaut, F.-X. Standaert (eds.). *Advances in Cryptology – EUROCRYPT 2021.* Springer International Publishing, Cham, 2021: pp. 805–835.

[12] J. Lu, G. Liu, Y. Liu, B. Sun, C. Li, L. Liu. Improved Neural Distinguishers with (Related-key) Differentials: Applications in SIMON and SIMECK. *Cryptology ePrint Archive*. Report 2022/030 (2022).

[13] M. Matsui, Linear Cryptanalysis Method for DES Cipher, In: eurocrypt93ed (ed.). *Eurocrypt93*, 1994: pp. 386–397.

[14] M. Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. *Crypto94*. 1994: pp. 1–11.

[15] B. Hou, Y. Li, H. Zhao, B. Wu. Linear Attack on Round-Reduced DES Using Deep Learning. In: L. Chen, N. Li, K. Liang, S. Schneider (eds.). *Computer Security – ESORICS 2020*, Springer International Publishing, Cham, 2020: pp. 131–145.

[16] Data Encryption Standard. *National Bureau of Standards, NBS FIPS PUB 46*. U.S. Department of Commerce. 1977.