LETTER

Improving the Accuracy of Differential-Neural Distinguisher For DES, Chaskey, and PRESENT

Liu ZHANG†a, Zilong WANG††b, Nonmembers, and Yindong CHEN††‡c, Member

SUMMARY 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.

key words: Deep Learning, Cryptanalysis, Differential-Neural Distinguisher, DES, Chaskey, PRESENT

1. Introduction

In CRYPTO 2019, Gohr proposed the idea of differential-neural cryptanalysis [1]. 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. In EUROCRYPT 2021, Benamira [2] indicated that Gohr’s differential-neural distinguisher builds a good approximation of the differential distribution table of the cipher and learns additional information.

Gohr [1] showed that the residual network (ResNet) [3] 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 differential-neural distinguishers are trained successfully. Chen et al. [4] and Benamira [2] 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. [5] used Dense Network (DenseNet) [6], and Squeeze-and-Excitation Network (SENet) [7] with existing deep architectures to train a neural network, and obtained (7-11)-round differential-neural distinguisher for Simon32/64. Zhang et al. [8] borrowed the idea of the Inception block of GoogLeNet [9] 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.

Our Contribution. Inspired by Zhang’s work, we have done some work to train a better differential-neural distinguisher on three reduced symmetric ciphers by modifying the neural architecture and using staged train method. The main improvements for differential-neural distinguisher are listed in Table 1, 2 and 3. Compared to Gohr’s [1] and Chen’s distinguisher [4], we improve the accuracy of differential-neural distinguisher and obtain a more round differential-neural distinguisher for DES, Chaskey, and PRESENT.†

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 [1, 2, 4] 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 an instance. Note that \( m = 1 \) in [1], \( m \in \{2, 4, 8, 16\} \) in [4], and \( m \in \{1, 5, 10, 50, 100\} \) in [4]. 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 \( Y = 1 \), 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, it

†The author is with School of Cyber Engineering, Xidian University, Xi’an 710126, China and State Key Laboratory of Cryptology, P. O. Box 5159, Beijing, 100878, China
††The author is with the College of Engineering, Shantou University.
*The work is supported by the National Natural Science Foundation of China (No. 62172319, U19B2021).
a) E-Mail: liuzhang@stu.xidian.edu.cn
b) E-Mail: zlwang@xidian.edu.cn (corresponding author)
c) E-Mail: ydchen@stu.edu.cn

https://github.com/CryptAnalystDesigner/MultipleCipherDesChaskeyPresent

Copyright © 200x The Institute of Electronics, Information and Communication Engineers
is effective. The differential-neural distinguisher model can be described as follows:

$$\Pr(Y = 1 \mid X_0, \ldots, X_{m-1}) = F(f(X_0), \ldots, f(X_{m-1})), \quad \varphi(f(X_0), \ldots, f(X_{m-1})))$$

$$X_i = (C_i,0, C_i,1), i \in [0, m - 1]$$

$$\Pr(Y = 1 \mid X_0, \ldots, X_{m-1}) \in [0, 1]$$

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 [1] and Chen’s [4] model mainly includes an initial convolutional layer consisting of width-1 convolutional layers and multiple residual blocks. Zhang et al. [8] modified the initial convolutional layer using the Inception model and multiple-plaintext residual blocks. The neural network receives ciphertext pairs instead of single-ciphertext pairs as the input.

### 4. Model Training Process and Results

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 \(\{(P_i, f, 0, P_i, f, 1), f \in [0, m - 1]\}\) with the input difference \(\Delta\) 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 \(M\) instances. The batch size (denoted by \(B_i\)) is set to a fixed value. Optimization was performed against mean square error loss plus a small penalty based on L2 weights regularization parameter \(\lambda\) using the Adam algorithm [10]. A cyclic learning rate schedule was applied, setting the learning rate \(l_i\) for epoch \(i\) to \(l_i = \alpha + (\beta - \alpha)/(n-\ell) \mod (n+1)\). 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 [1], Gohr trained an 8-round distinguisher of Speck32/64 by using the staged train method. For more detailed method details, refer to [1].

**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 as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_f)</td>
<td>32</td>
</tr>
<tr>
<td>(k_x)</td>
<td>3</td>
</tr>
<tr>
<td>(B_i)</td>
<td>1000</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>(10^{-5})</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>0.002</td>
</tr>
<tr>
<td>(\beta)</td>
<td>0.0001</td>
</tr>
<tr>
<td>(N)</td>
<td>(10^7)</td>
</tr>
<tr>
<td>(M)</td>
<td>(10^6)</td>
</tr>
</tbody>
</table>

The baseline distinguisher, abbreviated as \(ND_{bas}\), is reproduced by Chen et al. [4] according to the network architecture of Gohr [1]. The differential-neural distinguisher of Chen et al., named \(ND_{mc}\), is trained by using multiple-ciphertext pairs instead of single-ciphertext pairs as the input of the neural network in [4]. 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 \(ND_{C_1}\). 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 \( ND_{C_2} \).

4.2 Experiments on DES

DES [11] is a block cipher that is built on a \( 6 \times 4 \) Sbox. Based on the analysis of DES in [12], the plaintext difference adopted is \( \alpha = (0x40080000, 0x04000000) \) and the baseline distinguishers were built for reduced DES [1]. 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 \((0x40080000, 0x04000000)\) (the most likely difference to appear three rounds after the input difference \((0x40080000, 0x04000000)\). The training was done on \( N(N/\alpha) \) 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 [1, 4] in Table 1. From Table 1, the accuracy of our differential-neural distinguisher was improved both \( C_1 \) and \( C_2 \) compared to [1, 4]. 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.

4.3 Experiments on Chaskey

Based on the best differential path searched in [13], baseline distinguishers are built for reduced Chaskey [1]. Given the plaintext difference \( \alpha = (0x8400, 0x0400, 0x0, 0) \), 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.

4.4 Experiments on Present

Present [15] is a block cipher based on a \( 4 \times 4 \) Sbox. Based on the plaintext difference \( \alpha = (0, 0, 0, 0x9) \) provide in [16], the baseline distinguisher were built for Present64/80 reduced up to 7 rounds [1]. 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 6-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 [1,4] in Table 3. From Table 3, the accuracy of our differential-neural distinguisher was improved both \( C_1 \) and \( C_2 \) compared to [1,4]. Also, we trained the differential-neural distinguisher for one more round.

<table>
<thead>
<tr>
<th>Table 3: Accuracy of distinguisher for Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

4.5 Overfitting and Fluctuation.

In [1], the training set and test set include \( N \) and \( M \) instances, which consist of a ciphertext pair in each instance, that is, total \( N \) and \( M \) ciphertext pairs, respectively. To ensure a fair comparison, the training set and test set include \( N/m \) and \( M/m \) instances, and each instance includes \( m \) ciphertext pairs in [4], i.e., the total number of ciphertext pairs used is also \( N \) and \( M \). Thus, we used the same amount of data as [4] in \( C_1 \). However, Using \( N/m \) and \( M/m \) instances as training and test sets may lead to overfitting or fluctuation, see Fig.1a. In order to overcome this problem, we also use \( N \) and \( M \) instances as training and test set in \( C_2 \), which consists of \( m \) ciphertext pair in each instance.

From Fig.1a, 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.1b. 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. Essentially, it is due to the insufficient number of training and test sets. Also, using \( N \) and \( M \) as training and test sets to train a distinguisher can avoid speed up the model convergence and improve the model accuracy to a certain extent. A similar phenomenon exists in the training process of PRESENT and Chaskey.

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-neural distinguisher and obtained more rounds of differential-neural distinguisher for DES, Chaskey, and PRESENT.

**References**


