Improving Differential-Neural Cryptanalysis with Inception Blocks

Liu Zhang[1 [0000−0001−6106−3767]], Zilong Wang[1 [0000−0002−1525−3356]], and Boyang Wang[2 [0000−0001−8973−2328]]

[1] School of Cyber Engineering, Xidian University, Xi'an, China
liuzhang@stu.xidian.edu.cn
zlwang@xidian.edu.cn
[2] Department of Electrical Engineering and Computer Science, University of Cincinnati, Cincinnati, USA
boyang.wang@uc.edu

Abstract. In CRYPTO'19, Gohr proposed a new cryptanalysis strategy using machine learning algorithms. Combining the neural distinguisher with a differential path and integrating the advanced key recovery procedure, Gohr achieved a 12-round key recovery attack on Speck32/64. Chen and Yu improved accuracy of neural distinguisher considering derived features from multiple-ciphertext pairs instead of single-ciphertext pairs. Bao *et al.* presented the concept of generalized neutral bits, improved 12-round, and devised a practical 13-round key recovery attack on Speck32/64 by enhancing the differential path prepended on the top of neural distinguisher.

To capture more dimensional information, inspired by the Inception Blocks in GoogLeNet, we use multiple parallel convolutional layers as the core of the network structure to train neural distinguisher. For Speck32/64, we improve the accuracy of (5-8)-round neural distinguisher and train a new 9-round neural distinguisher. For Simon32/64, we improve the accuracy of (7-11)-round neural distinguisher and train a new 12-round neural distinguisher. In addition, we extend the idea of neutral bits in Gohr's work to solve the requirement of the same distribution of multiple-plaintext pairs in key recovery attack.

Under the combined effect of multiple improvements, the time complexity of our (11-13)-round key recovery attacks for Speck32/64 has been reduced to a certain extent. Surprisingly, the success rate of our 12-round key recovery attack can reach 100%. For Simon32/64, we increase the success rate and decrease the time complexity of 16-round key recovery attack. Also, we successfully implement a 17-round key recovery attack. Sadly, because the accuracy of 9-round neural distinguisher of Speck32/64 and 12-round neural distinguisher of Simon32/64 is lower, they can't be used for more rounds key recovery attack. [3]

Keywords: Neural Distinguisher · Differential-Neural Cryptanalysis · Inception Blocks · Speck · Simon

[3] The source codes are available in https://drive.google.com/drive/folders/17AWYupor_bX2rEe000tPuppdNH4dlmHm?usp=sharing.

1 Introduction

The idea of differential-neural cryptanalysis was proposed by Gohr [8] in CRYPTO 2019, where the neural distinguisher trained by neural network is introduced as the underlying distinguisher, and Bayesian search is used to speed up key recovery attack, compared to the traditional differential cryptanalysis. The function of the neural distinguisher is to distinguish ciphertext from random numbers. If the accuracy of the neural distinguisher is greater than 0.5, it is considered to be an effective distinguisher. However, current neural distinguisher seems only effective for limited rounds for key recovery attack. In order to increase the number of rounds, a short traditional high-probability differential path $\Delta S \rightarrow \Delta P$ is prepended before the neural distinguisher. To make sure the differential path and neural distinguisher work together (i.e. essentially form a long differential path), the output difference of differential path has to match the expected input difference of input data used to train neural distinguisher.

Gohr [8] showed that the Residual Network (ResNet) [9] (previously applied in image recognition) could be trained to capture the non-randomness of the distribution of values of output pairs when the input pairs of round-reduced Speck32/64 are of specific difference. As a result, (5-8)-round (effective) neural distinguisher, and (11-12)-round key recovery attacks for Speck32/64 was achieved by combining 2 rounds of differential path. Note that neutral bits introduced in [8] group the conforming pairs of the differential path prepended on top of neural distinguisher. There may be two directions to improve the differential-neural cryptanalysis proposed by Gohr [8]. One is to improve the process of key recovery attacks with neural distinguisher by traditional differential cryptanalysis, the other is to study effective underlying neural distinguisher of more rounds.

Bao *et al.* [2] generalized the concept of neutral bits, and searched for (conditional) simultaneous neutral bit-set with higher probability for more rounds of differential path. Thus, Bao *et al.* improved the 12-round and devised a new 13-round key recovery attack for Speck32/64 with the same neural distinguisher proposed in [8]. Note that the idea [2] came from the traditional cryptanalysis.

In order to launch more rounds key recovery attack, better neural distinguishers were also studied recently. Chen and Yu [6] proposed multiple-ciphertext pairs instead of single-ciphertext pairs (in Gohr's work) as the input of neural network, and improved the accuracy of the (5-7)-round neural distinguisher of Speck32/64 to a certain extent. Bao *et al.* [2] used Dense Network (DenseNet) [11] and Squeeze-and-Excitation Network (SENet) [10] with existing deep architectures to train neural distinguisher, and obtained (7-11)-round neural distinguisher and devised a 16-round key recovery attack for Simon32/64.

In EUROCRYPT 2021, Benamira [4] indicated that Gohr's neural distinguisher builds a good approximation of the differential distribution table of the cipher during the learning phase and learns additional information. Based on the principle that the purpose of the neural distinguisher is to obtain the difference information in the ciphertext, we have done some tentative work to train a bet-

ter neural distinguisher by modifying the network structure in this paper. The main improvements for differential-neural cryptanalysis are listed as follows.

First, we proposed a better neural distinguisher. Specifically, we use multiple-ciphertext pairs as the input of the neural network, and added the Inception block composed of the multiple-parallel convolutional layers before the residual network. The propose of the inception block is to capture more dimensional information in the ciphertext pairs. Some minor modifications have also been made according to the round functions of the cipher. As a result, we improved the accuracy of (5-8)-round and trained an effective 9-round neural distinguisher for Speck32/64. In addition, we improved the accuracy of (7-11)-round and trained an effective 12-round neural distinguisher for Simon32/64. The results on neural distinguisher of Speck32/64 and Simon32/64 are presented in Table 2 and Table 7, where m is the group size of the multiple-ciphertext pairs.

Second, we use neutral bit with a probability of one to generate multiple-plaintext pairs. Similar to the single-plaintext pairs, not all plaintext pairs that satisfy the differential ΔS follow the differential ΔP after being propagated through the differential path. In order to resolve the requirement that the same difference of multiple-ciphertext parameter, inspired by the combined response of neural distinguisher in Gohr's work, we use neutral bit with probability one to generate multiple-plaintext pairs, and then encrypt multiple-plaintext pairs to get multiple-ciphertext pairs.

Third, we successfully implemented several rounds of key recovery attacks using the new neural distinguisher combined with differential path. We improved (11-13)-round key recovery attacks for Speck32/64 and reduced the time complexity to a certain extent. Surprisingly , when a 3-round differential path combined with our 7-round neural distinguisher, the success rate of the 12-round key recovery attack launched is 100%. For Simon32/64, we improved 16-round and devised a new 17-round key recovery attacks. Detailed experimental comparison results are shown in Table 1.

The rest of the paper is organized as follows. Section 2 gives the preliminary on previous neural distinguisher model, key recovery attack process, and neutral bits. Section 3 introduces the training method and accuracy of our neural distinguisher for Speck32/64. Some preparations from the distinguisher model to the key recovery attack are introduced in Section 4. Section 5 exhibits the details of our (11-13)-round key recovery attacks for Speck32/64. Section 6 introduces the training method and accuracy of our neural distinguisher for Simon32/64. Section 7 exhibits the details of our (16-17)-round key recovery attacks for Simon32/64. In Section 8 discuss our results and possible extensions of this work.

2 Preliminary

2.1 Brief Description of Speck32/64 and Simon32/64

Notations. Let n be the word size (i.e., the number of bits of a word), the state size can be denoted as $2n$ bits. Let (x_r, y_r) be the left and right branches of a

Table 1. summary of key recovery attacks on SPECK32/64 and SIMON32/64

Target	#R	Dist.	Conf.	Time	Data	Succ. Rate	Key Space	Ref.
SPECK32/64	11	$\mathcal{DD}$	1+6+4	2^{46}	2^{14}	–	2^{64}	[7]
		$\mathcal{ND}$	1+2+7+1	$2^{39.62}$	$2^{13.64}$	52%	2^{64}	[8]
		$\mathcal{ND}$	1+2+7+1	$2^{36.82}$	$2^{15.64}$	97%	2^{64}	Exp. 1
	12	$\mathcal{DD}$	1+7+4	2^{51}	2^{19}	–	2^{64}	[7]
		$\mathcal{ND}$	1+2+8+1	$2^{46.57}$	$2^{22.97}$	40%	2^{64}	[8]
		$\mathcal{ND}$	1+2+8+1	$2^{46.11\dagger}$	2^{22}	86%	2^{64}	[2]
		$\mathcal{ND}$	1+2+8+1	$2^{44.75}$	$2^{22.97}$	36%	2^{64}	Exp. 2
		$\mathcal{ND}$	1+3+7+1	$2^{44.82\dagger}$	$2^{18.58}$	81%	2^{63}	[2]
		$\mathcal{ND}$	1+3+7+1	$2^{45.38\ddagger}$	$2^{18.58}$	32%	2^{64}	[2]
		$\mathcal{ND}$	1+3+7+1	$2^{42.32}$	2^{25}	100%	2^{63}	Exp. 3
		$\mathcal{ND}$	1+3+7+1	$2^{41.32}$	2^{24}	–	2^{63}	Ideal
	13	$\mathcal{DD}$	1+8+4	2^{57}	2^{25}	–	2^{64}	[7]
		$\mathcal{ND}$	1+3+8+1	$2^{53.97\dagger}$	2^{30}	75%	2^{63}	[2]
		$\mathcal{ND}$	1+3+8+1	$2^{53.85\dagger}$	2^{29}	61%	2^{63}	[2]
		$\mathcal{ND}$	1+3+8+1	$2^{52.49\ddagger}$	2^{29}	21%	2^{64}	[2]
		$\mathcal{ND}$	1+3+8+1	$2^{52.68}$	2^{31}	49%	2^{63}	Exp. 4
		$\mathcal{ND}$	1+3+8+1	$2^{51.68}$	2^{30}	–	2^{63}	Ideal
SIMON32/64	16	$\mathcal{DD}$	2+12+2	$2^{26.48}$	$2^{29.48}$	62%	2^{64}	[1]
		$\mathcal{ND}$	1+3+11+1	$2^{46.98}$	2^{21}	49%	2^{64}	[2]
		$\mathcal{ND}$	1+3+11+1	$2^{43.19}$	2^{22}	72%	2^{64}	Exp. 5
	17	$\mathcal{ND}$	1+4+11+1	$2^{54.65}$	2^{28}	8%	2^{64}	Exp. 6

$\mathcal{DD}$: differential distinguisher; $\mathcal{ND}$: neural distinguisher.
All time complexities are calculated using the new time complexity calculation formula proposed in Section 4.2.
–: Not available.
†: The earlier version of Bao's results
‡: The latest version of Bao's results
Ideal: Assuming two classical differentials share the same output differences, the time complexity and data complexity of the attack can be halved.

state after the encryption of r rounds. Denotes by $\oplus$ the bit-wise XOR, $\boxplus$ the addition modulo 2^n,$\cdot$ the bit-wise AND, $\gg$ the bit-wise right rotation , $\ll$ the bit-wise left rotation, k_r the subkey of the encryption of r rounds.

SPECK32/64 and SIMON32/64 are small members in the lightweight block cipher family SPECKand SIMON [3]designed by researchers from the National Security Agency (NSA). The round function of SPECK32/64 takes a 16-bit subkey k_r and a cipher state consisting of two 16-bit words (x_i, y_i) as input. The state of the next round (x_{i+1}, y_{i+1}) is computed as follows:

$$x_{i+1} := ((x_i \gg 7) \boxplus y_i) \oplus k_r, y_{i+1} := (y_i \ll 2) \oplus x_{i+1}.$$

The round function is repeated 22 times for SPECK32/64. The 16-bit subkeys are generated based on a 64-bit master key by the non-linear key schedule using the same round function for SPECK32/64.

The round function of SIMON32/64 takes a 16-bit subkey k_r and a cipher state consisting of two 16-bit words (x_i, y_i) as input. The next round state (x_{i+1}, y_{i+1}) is computed as follows:

$$x_{i+1} := (x_i \ll 1) \cdot (x_i \ll 8) \oplus (x_i \ll 2) \oplus y_i \oplus k_r, y_{i+1} := x_i.$$

The round function is repeated 32 times for SIMON32/64. The subkeys of 16-bit for each round are generated from a master key of 64-bit by linear functions of simple rotation and XOR for SIMON32/64.

2.2 Overview of Gohr's and Chen's Neural Distinguisher Model For SPECK32/64

The neural network used by Gohr and Chen to train the neural distinguisher is identical except for the input. Chen's model is more like a generalization of Gohr's model on the input. Thus, we introduce the two models uniformly.

The neural distinguisher model is a supervised model. The purpose of a neural distinguisher is to distinguish ciphertext and random numbers. Therefore, it is necessary to construct ciphertext and random numbers artificially, thereby assigning corresponding labels. Given m plaintext pairs $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and target cipher SPECK32/64, the resulting ciphertext pairs $\{(C_{i,0}, C_{i,1}), i \in [0, m-1]\}$ is regarded as a sample. Each sample will be attached a label Y:

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

where $\Delta = (0x0040, 0x0000)$. If Y is 1, it means this sample is sampled from the target distribution and defined as a positive example. Otherwise, this sample is sampled from a uniform distribution and defined as a negative example. To guarantee the accuracy of distinguisher prediction, a large amount of samples need to be put into neural network training. If the neural network can obtain a stable distinguishing accuracy higher than 0.5 on a test set, it can effectively

distinguish ciphertext and random numbers. The neural distinguisher model can be described as:

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

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. In Gohr's model, the value of m is 1 [8]. In Chen's model, the value of m is $\{2, 4, 8, 16\}$ [6].

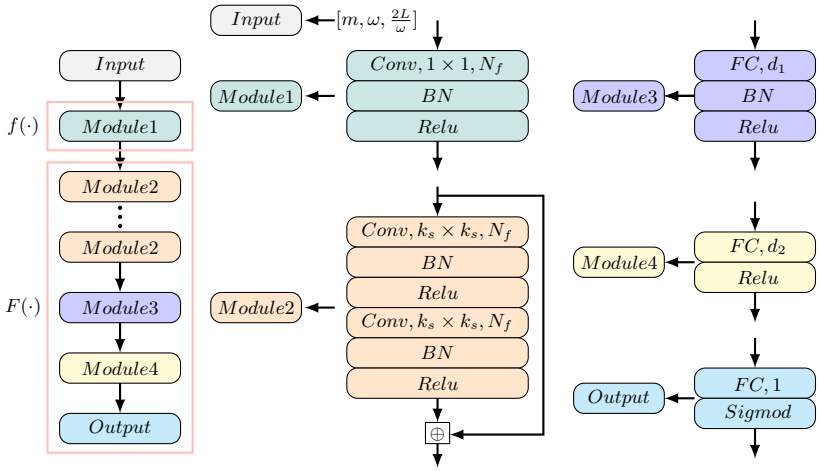


Fig. 1. The network architecture of Gohr's and Chen's model

The network architecture contains several modules that are described in Figure 1 (This Figure is from [6]). The input layer of neutral network consisting of multiple-ciphertext pairs is arranged in a $m \times \omega \times \frac{2L}{\omega}$ array, where L represents the block size of the target cipher, and ω is the size of a basic unit. For example, L is 32 and ω is 16 for SPECK32/64. The module 1 is the initial with-1 convolution layer that intend to make the learning of simple bit-sliced functions such as bitwise addition easier. The module 2 is the structure of ResNet. *Conv* stands for a convolutional layer with N_f filters. The size of each filter is $k_s \times k_s$. The amount of module 2 is determined by experiment. The prediction head consists of module 3, module 4, and output layer. *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.

2.3 Differential-neural cryptanalysis

The differential-neural cryptanalysis, which is a chosen-plaintext attack, aims at recovering the subkey of the last two rounds. The idea of differential-neural cryptanalysis is similar to the traditional differential key recovery attack. By guessing the subkey of the last round, we decrypt the ciphertext obtained by encrypting the specific plaintext structure and use the neural distinguisher to determine whether the decrypted ciphertext is correct for the differential characteristic. If the score of neural distinguisher exceeds a certain threshold, the guessed key is used as the candidate key. Repeat the above steps for the candidate key to obtaining another round of candidate keys, the key recovery attack is successful. Figure 2 shows the overall flow of a key recovery attack based on neural distinguisher, where $\mathcal{ND}$ is the trained neural distinguishers.

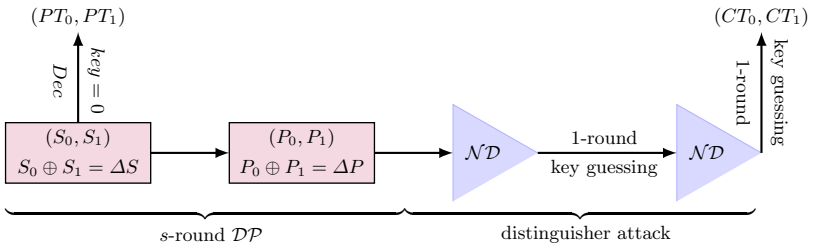


Fig. 2. $(1 + s + r + 1)$-round key recovery attack based on neural distinguisher

The $(1+s+r+1)$-round key recovery attack employs an r-round main and $(r-1)$-round helper neural distinguisher trained by input pairs with difference ΔP. When using the guessed subkey to decrypt the $(1 + s + r + 1)$-round ciphertext, if the subkey guessed correctly, the $(1+s+r)$-round ciphertext will be obtained, so the score of neural distinguisher will be greater than 0.5. If the guess of the subkey is wrong, it is equivalent to encrypting the ciphertext for another round. The randomness of the data will be enhanced, resulting in the score of neural distinguisher being less than 0.5. If the Hamming distance between the guessed subkey and the correct subkey is small, we put the data obtained after decrypting the ciphertext with the guessed subkey into neural distinguisher, and the score of neural distinguisher will have a greater probability of being greater than 0.5, otherwise it will be less than 0.5.

A short s-round differential path $(\Delta S \rightarrow \Delta P)$ with probability denoted by 2^{-p} is prepend-ed on top of the neural distinguisher to increase the number of rounds of key recovery attack. In order to ensure the existence of data pairs satisfying difference ΔP after s-round encryption, about $c \cdot 2^p$ (denoted by n_{cts}) data pairs with difference ΔS are required according to difference propagation probability, where c is a small constant. The n_{cts} structures of data pairs are decrypted by one round with 0 as the subkey to get plaintext structures because

of nonlinear operation before key addition for SPECK and SIMON. All plaintext structures are queried to obtain the corresponding ciphertext structures.

Each ciphertext structure is used to select a candidate of the last subkey by the r-round main neural distinguisher with a highly selective key search policy based on a variant of Bayesian optimization. The usage of ciphertext structures is also highly selective by using a standard exploration-exploitation technique, namely *Upper Confidence Bounds* (UCB). Each ciphertext structure is assigned with a priority according to the score of the recommended subkeys and the visited times. Without exhaustively performing trail decryption, the key search policy depends on the expected response of the neural distinguisher upon wrong-key decryption. This *wrong key response profile* is used to recommend new candidate values for the key from previous candidate values with minimizing the weighted Euclidean distance as the criteria in a BAYESIANKEYSEARCH Algorithm [8].

In general, the more rounds the neural distinguisher covers, the lower its accuracy will be. A neural distinguisher with an accuracy higher than 0.5 means some distinguisher advantage over a random distinguisher. However, if the accuracy is marginally higher than 0.5, it can be hardly used in practical key recovery attack since the single prediction of the neural distinguisher may misjudge. Thus, Gohr in [8] used the combined response of the neural distinguisher over large amounts of samples of the same distribution, which can be satisfied by neutral bits. The primary notion of neutral bits can be interpreted as follows.

Definition 1 (Neutral bits of a differential, NB[5]). *Let $\Delta_{in} \rightarrow \Delta_{out}$ be a differential with input difference Δ_{in} and output difference Δ_{out} of an r-round encryption F^r. Let (P, P') be the input pair, and $(C, C' \mid C = F^r(P), C' = F^r(P'))$ be the output pair, where $P \oplus P' = \Delta_{in}$. If $C \oplus C' = \Delta_{out}$, (P, P') is said to be conforming the differential $\Delta_{in} \rightarrow \Delta_{out}$. Let $e_0, e_1, \ldots, e_{n-1}$ be the standard basis of $\mathbb{F}_2^n$. Let i be an index of a bit (starting from 0). The i-th bit is a neutral bit for the differential $\Delta_{in} \rightarrow \Delta_{out}$, if $(P \oplus e_i, P' \oplus e_i)$ is also a confirming pair for any confirming pair (P, P').*

The responses $v_{i,k}$ from the neural distinguisher on ciphertext pairs in the ciphertext structure (of size n_b, i.e., the number of neutral bits) are combined using the Formula $s_k = \sum_{i=0}^{n_b-1} \log_2 \left(\frac{v_{i,k}}{1-v_{i,k}} \right)$ and used as the score s_k of the recommended subkey. The score plays a decisive role in the execution time and success rate of the attack. Bao *et al.* presented the following conjecture[2] that the required number of samples is closely related to the bias of the accuracy of the neural distinguisher.

Conjecture 1 ([2]). For a combined-score-distinguisher derived from a neural distinguisher with accuracy lager than 0.5 and of bias ϵ to be successfully used in the improved attack using *Upper Confidence Bound* and BAYESIANKEYSEARCH Algorithm, the size of samples from the same distribution should be about $c \times 1/(2\epsilon)^2$, where c is a small constant.

In general, neutral bits of non-trivial differential path are scarce. In [8], probabilistic neutral bits (PNB) are exploited. Bao *et al.* [2] found some probabilistic

neutral bits with a higher probability and simultaneous-neutral bit-sets (SNBS). Also, they proposed the concept of conditional (simultaneous-) neutral bit(-set)s (CSNBS). The explicit definitions of PNB, SNBS, and CSNBS (collectively referred to as generalized neutral bits) refer to [2]. For details about the generalized neutral bits and key recovery attack procedure, see [8] and [6].

3 Neural Distinguishers for Reduced-Round Speck32/64

3.1 Network Structure

The overall framework of our neural network for training the neural distinguisher is similar as that of Gohr's and Chen's model shown in Figure 1. Our best neural network consists of four parts: an input layer consisting of multiple-ciphertext pairs, an initial convolutional layer consisting of four parallel convolutional layers, a residual tower of multiple two-layer convolutional neural networks, and a prediction head consisting of multiple fully connected layers.

Input Representation. Once the output of the r-th round $(C, C') = (x_r||y_r, x'_r||y'_r)$ is known, one can directly compute (y_{r-1}, y'_{r-1}) without knowing the $(r-1)$-th subkey according to the round function of Speck. Thus, the neural network accept data of the form $(x_r, x'_r, y_r, y'_r, y_{r-1}, y'_{r-1})$. The input layer has m group ciphertext pairs consisting of $3L$ units likewise arranged in a $[m, \omega, \frac{3L}{\omega}]$ array, where $L = 32, \omega = 16$ for Speck32/64. The detail data structure of the input layer is shown in Figure 3, where $C_{i,0} = (x_r, y_r), C_{i,1} = (x'_r, y'_r), C_{i,2} = (y_{r-1}, y'_{r-1}), i \in (0, m)$.

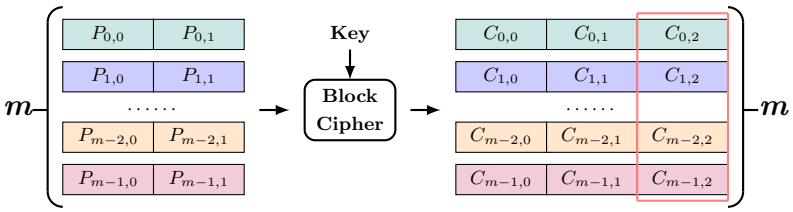


Fig. 3. The format of input data

Initial Convolution. The input layer is connected to the initial convolutional layer, which comprises four convolution layers with N_f channels of different kernel sizes. The four convolution layers are concatenated at the channel dimension, similar to the Inception module [13] in GoogLeNet. Batch normalization is applied to the output of concatenate layers. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[m, \omega, 4N_f]$ matrix is passed to the convolutional blocks layer. The structure of the initial convolution layer can be seen in Figure 4.

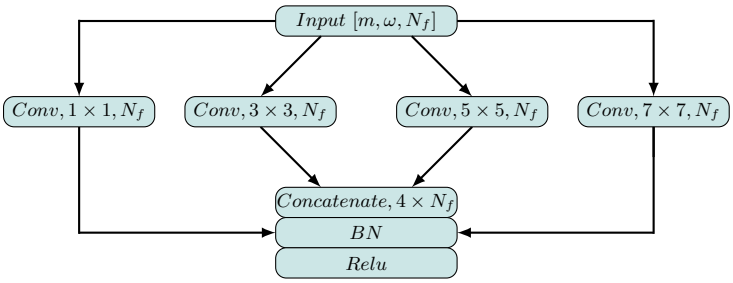
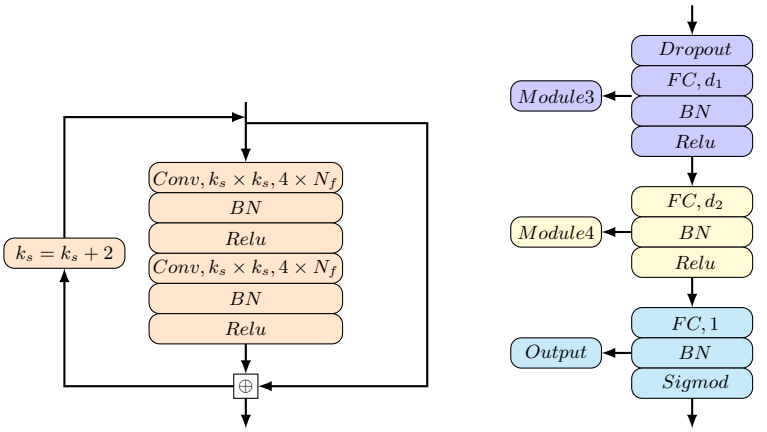


Fig. 4. The initial convolution layer

Convolutional Blocks. Each convolutional block consists of two layers of $4N_f$ filters. Each block applies first the convolution of kernel size k_s, then a batch normalization, and finally a rectifier layer. At the end of the convolutional block, a skip connection is added to the output of the final rectifier layer of the block to the input of the convolutional block, and passes the result to the next block. After each convolutional block, the kernel size increases by 2. The amount of convolutional blocks is determined by experiment. The structure of convolutional blocks layer can be seen in Figure 5(a).

Prediction Head. The prediction head consists of two hidden layers and one output unit. To prevent model overfitting, we add a dropout layer before the first hidden layer. The three fully connected layers comprise 512, 64, and 1 units. The three fully connected layers are followed by the batch normalization layer and rectifier layer. The final layers consist of a single output unit using a *Sigmoid* activation function instead of *Relu* function. The structure of the prediction head is shown in Figure 5(b).

Improvement Ideas. First, the input data of Gohr's model only contains the ciphertext pair of the last round. Given the ciphertext of the last round, the right half of the ciphertext pair of the penultimate round can be calculated according to the round function of SPECK32/64 without knowing the (r-1)-th subkey. We increase the amount of input data to the model to obtain more information about the cryptographic algorithm. Second, according to Chen's method, using multiple-ciphertext pairs with the same distribution as the input of the neural network can significantly reduce the influence of a single misjudgment on the whole result. Third, in Gohr's model, using the initial width-1 convolutional layer is intended to make learning simple bit-sliced functions easier such as bitwise addition. However, we think this is not enough to capture the features of the internal structure of the cryptographic algorithm. Thus, we add the convolution operation of widths 3, 5, and 7 to capture features under different dimensions, considering the circular shift operation in the round function of SPECK and some relationship in the adjacent bits generated by the modular addition operation. This improvement was mainly inspired by the Inception



(a) The convolutional blocks (b) The prediction head

module [13] in GoogLeNet to capture more dimensional information. Also, to capture information in a larger dimension, we modify the size of the residual network convolution kernel in Gohr's model to keep it incrementing 2. Other components in the neural network are basically the same as those in Gohr's neural network. The essence of using a deep residual network to construct a differential distinguisher is to treat the cipher with a nonlinear round function as a complex function and use multiple residual blocks to fit the function.

3.2 The Training of Neural Distinguisher

The distinguisher accuracy is the most critical indicator which reflects the performance of neural distinguisher. The following training was carried out to verify the performance of our neural distinguisher.

Data Generation. Training and test data were generated by using the Linux random number generator to obtain uniformly distributed keys K_i and plaintext pairs P_i with the input difference $\Delta = (0x0040, 0x0000)$ as well as a vector of binary-valued real/random labels Y_i. During produce training or test data for r-round SPECK32/64, the plaintext pair P_i was then encrypted for r rounds if $Y_i = 1$, while otherwise the second plaintext of the pairs was replaced with a freshly generated random plaintext and then encrypted for r rounds. A key parameter of our neural distinguisher is number of ciphertext pairs: the group size m, which has four options $\{2, 4, 8, 16\}$. We used two different numbers of training and test sets to train the neural network. The first set of experiments is to use N and M data as the training and test sets of the neural network. The second set of experiments is to use $N \times m$ and $M \times m$ data as the training set and test set of the neural network.

Basic Training Scheme. We run the training for 20 epochs on the dataset of size 10^7. To maximize GPU performance, the batch size processed by the dataset is adjusted according to the parameter m. The last 10^6 sample was withheld for the test. Optimization was performed against mean square error loss plus a small penalty based on L2 weights regularization parameter $c = 10^{-5}$ using the Adam algorithm [12]. A cyclic learning rate schedule was used, setting the learning rate l_i for epoch i to $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n}.(\beta - \alpha)$ with $\alpha = 10^{-4}, \beta = 2 \times 10^{-3}$ and $n = 9$. The networks obtained at the end of each epoch were stored, and the best network by validation loss was evaluated against a test set.

Training (8-9)-round Distinguishers Using Staged Train Method. For 8 rounds, the basic training scheme described above fails, i.e., the model does not learn to approximate any helpful function. We still succeeded in training an 8-round neural distinguisher superior to the 8-round neural distinguisher of SPECK32/64 in [8] by using several stages of pre-training. First, we use our 7-round distinguisher to recognize 5-round SPECK32/64 with the input difference (0x8000,0x804a) (the most likely difference to appear three rounds after the input difference (0x0040,0x0000)). The training was done on $10^7 \times m$ samples for 20 epochs with cyclic learning rates. Then we trained the distinguisher so obtained to recognize 8-round SPECK32/64 with the input difference $(0x0040, 0x0000)$ by processing $10^7 \times m$ freshly generated samples for 10 epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another $10^7 \times m$ fresh samples for 10 epochs. During the staged training, the batch size was changed according to the value of parameter m, to match the computing performance of the GPU equipment. For the 9-round distinguisher, the overall training method is the same. The only difference is the use of an 8-round distinguisher to identify 5-round SPECK32/64 with the input difference (0x850a,0x9520) (the most likely difference to appear four rounds after the input difference (0x0040,0x0000).

3.3 Result

Test Set Accuracy. We summarize the experimental results of (5-9)-round of the neural distinguisher. The (5-7)-round distinguishers was trained using the basic training method. An 8-round distinguisher was derived from a 7-round distinguisher using the staged training method. The training method of the 9-round distinguisher is the same as that of the 8-round distinguisher.

The function of the neural distinguisher is to distinguish between ciphertext and random numbers, so the accuracy rate is greater than 0.5, that is, it is considered effective. However, with the increase of the number of encryption rounds, the randomness of the ciphertext is getting better and better, and it is more difficult to distinguish it from the random number, so the accuracy of the distinguisher decreases with the increase of the number of rounds. The comparison results of our neural distinguisher accuracy with Gohr's and Chen's are shown Table 2. As the number of ciphertext pairs m increases, the accuracy

of the neural distinguisher also increases. Compared to Gohr's and Chen's, the accuracy of our neural distinguisher was significantly improved when the group size m takes different values.

Table 2. accuracy of neural distinguisher for (5-9)-round SPECK32/64

R	Gohr [8]	m=2			m=4		
		Chen [6]	N	$N \times m$	Chen [6]	N	$N \times m$
5	0.929	0.9738	0.9813	0.9803	0.991	0.9975	0.9977
6	0.788	0.8613	0.8771	0.8773	0.931	0.9468	0.9497
7	0.614	0.6393	0.6663	0.6649	0.6861	0.7194	0.7283
8	0.514	-	-	-	-	-	0.5428
R	Gohr [8]	m=8			m=16		
		Chen [6]	N	$N \times m$	Chen [6]	N	$N \times m$
5	0.929	0.9992	0.9994	0.9997	0.9999	0.9998	0.9999
6	0.788	0.9562	0.9868	0.9895	0.9802	0.9969	0.9992
7	0.614	0.7074	0.7879	0.8106	0.6694	0.8460	0.8963
8	0.514	-	-	0.5590	-	-	0.5854
9	-	-	-	0.5024	-	-	0.5050

Overfitting. The number of training samples for Gohr's neural distinguisher is N. In Chen's model, N/m samples are generated as training data sets, and each sample contains m ciphertext pairs. Therefore, it uses N data as training data. However, we used two different numbers of training and test sets to train the neural network. The first set of experiments is to use N and M data as the training and test sets of the neural network. The second set of experiments is to use $N \times m$ and $M \times m$ data as the training set and test set of the neural network. In order to ensure the fairness of the experimental results, we must use the same amount of data to train the neural network as the predecessors. However, using N data to train a neural network will suffer from overfitting, especially when the number of rounds R is small and the group size m is large. To solve this problem, we use $N \times m$ data to train the neural network, which not only solves the problem of overfitting, but also improves the accuracy of the neural distinguisher. We believe that the small amount of training data leads to overfitting and does not realize the full potential of our network structure. The comparative experimental results of different numbers of training and test sets are shown in Appendix B. It can be clearly seen that when m is large and R is small, using N and M data as training set and test set to train the neural network has obvious overfitting phenomenon.

Wrong Key Response Profile. Gohr presented an improved key recovery attack using Upper Confidence Bound and Bayesian Optimization [8]. More specifically, the key search policy depends on an important observation that the

expected response of the neural distinguisher upon wrong-key decryption will rely on the bitwise difference between the trial key and the real key. This wrong key response profile, which can be precomputed, is used to recommend new candidate values for the key from previous candidate values by minimizing the weighted Euclidean distance as the criteria in a BayesianKeySearch Algorithm. It recommends a set of subkeys and provides their scores without exhaustively performing trial decryption. We expect a higher score for neural distinguisher when the hamming distance between the correct and wrong keys is small, and a lower score for neural distinguisher when the Hamming distance is large. With the increase of Hamming distance, the more the score decreases, the better. From Figure 5 and Figure 6, we know the score of our neural distinguisher is higher than that of Gohr's distinguisher when the Hamming distance of key difference is small. When the Hamming distance is large, we can obtain a lower score. Our distinguisher makes the difference between the score of the correct key and the score of the wrong key even more significant, i.e., our neural distinguisher is better.

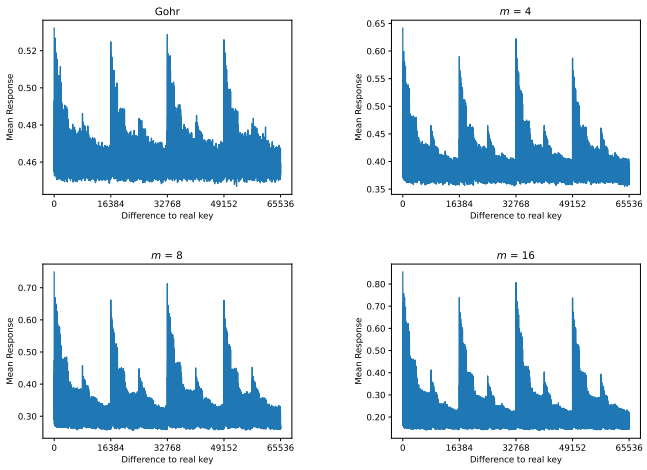


Fig. 5. Wrong key response profile for 7-round SPECK32/64

4 From Neural Distinguisher to Key Recovery Attack

4.1 Generation of Same Distributed Data

Like Chen's neural distinguisher, the data complexity of our neural distinguisher is also m times that of Gohr's model. To solve this problem, Chen and Yu [6] proposed a data reuse strategy that randomly selects m ciphertext pairs from

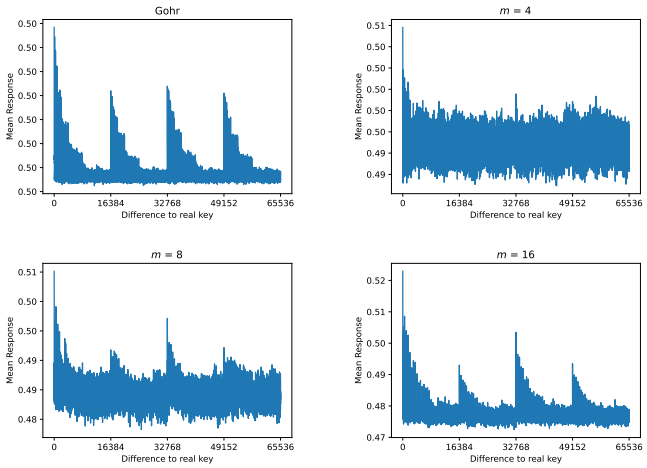


Fig. 6. Wrong key response profile for 8-round SPECK32/64

N ciphertext pairs to generate N ciphertext pair group. There are a total of C_N^m options which is much larger than N. However, Chen did not obtain good results about the key recovery attack. We think that the main reason is that the demand for the same distribution of data has not been met.

The input of the neural network is obtained by encrypting plaintext pairs that satisfy the same difference. In the process of key recovery, the initial plaintext propagated through the differential path also needs to have the same difference when entering the neural distinguisher. But the differential path is probabilistic. Even if multiple initial inputs have the same difference, after propagation through the differential path, the output difference is not the same. Therefore, the initial input during a key recovery attack cannot be randomly generated.

In the work of Gohr, the accuracy of the neural distinguisher is low, so the single prediction of the neural distinguisher is prone to misjudgment. To solve this problem, Gohr combined the multiple predictions of the neural distinguisher as the final score. This also requires the data entering the neural distinguisher to have the same distribution, and Gohr solved this problem using neutral bits. Inspired by Gohr's method, we randomly generate a plaintext pair that meets the initial difference and use $\log_{2^m}$ neutral bits to obtain m plaintext pairs. And to choose neutral bits with high probability, the optimal choice is to use neutral bits with probability 1. Therefore, multiple-plaintext pairs entering to neural network have the same difference, i.e., multiple-ciphertext pairs entering the neural distinguisher have the same distribution.

4.2 Experimental Environment and Complexity Calculation

The Core of the key recovery attack was examined using a server with multiple GeForce GTX 2080-Ti GPUs. When a fast graphics card is used, the performance

of our implementation is not limited by the speed of neural network evaluation but by the total number of iterations on the ciphertext structures. We count a key guess as successful if the subkey of last round was guessed correctly and if the second round key is at the hamming distance at most two of the real key.

The data complexity of the experiment is calculated by the formula $n_{kg} \times n_b \times n_{ct} \times m \times 2$. During a key recovery attack, the differential path used may need to satisfy some conditions involving the key. n_{kg} indicates the number of times to guess the key involved in the condition The data complexity is calculated as theoretical values. $n_b = 2^{\mathcal{NB}}$, where $\mathcal{NB}$ instead of the number of neutral bit. n_{ct} represents the number of ciphertext structures used, and m is the number of multiple-ciphertext pairs. In the actual experiment, when the accuracy of the neural distinguisher is high, the key can be recovered quickly and successfully, and not all the data is used, so the actual data complexity is lower than the theoretical data complexity.

In [8], Gohr estimated that a highly optimized implementation of SPECK32/64 could perform brute force key search at a speed of about 2^{28} key per second per core. A correct key guess takes an average of 500 seconds, and the single success rate of a key recovery attack of 11-round SPECK32/64 is 52.1%. Adjusting for the empirically measured success rate of key recovery attack, Gohr expected to need about 1000 seconds average to execute the key recovery attack on a single core. This yields an estimated computational attack complexity of 2^{38} SPECK32/64 encryptions until a solution is found. Gohr didn't give a formula for calculating the time complexity. In order to better evaluate the effect of key recovery attack, we give a new calculation method and update Gohr's result accordingly. To reduce the experimental error, we conduct multiple key recovery recovery attacks, take the average running time as the running time of an experiment, and divide the number of successful key recovery experiments by the total number of experiments as the success rate of a single key recovery attack. In order to calculate the time complexity of the experiment, it is necessary to consider two indicators, the average run time and the success rate of key recovery attack. Assuming that the success rate of single experiment is sr, calculate how many experiments need to be performed to ensure that at least one experiment is successful. When the overall success rate is 99%, we consider the experiment to be successful, and the calculation formula for the number of experiments ne is: $1 - (1 - sr)^{ne} = 0.99$, i.e., $ne = \log_{1-sr} 0.01$. This yields an estimated computational attack complexity of $2^{28} \times 500 \times \log_{1-0.52} 0.01 \approx 2^{39.6}$ for Gohr's key recovery attack of 11-round SPECK32/64. In our experiment to reproduce Gohr's experiment, the average running time is 225s in 300 trials. To facilitate comparison, the time complexity of Gohr's experiment is taken as the benchmark for conversion. The new time complexity calculation formula: $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$, where rt is the average run time of key recovery attack.

5 Key Recovery Attack on Reduced-Round Speck32/64

This section shows that the capabilities of neural distinguisher for key recovery attack can be improved. All distinguishers used in key recovery attacks are trained using $N \times m$ data. In the following, we improve (11-13)-round key recovery attacks by employing improved neural distinguisher. Our attacks follow the same framework in [8] The key guessing procedure applies a simple reinforcement learning procedure UCB. The last subkey and the second to last subkey are to be recovered without exhaustively using all candidate values to do one-round decryption. Instead, a Bayesian key search employing the wrong key response profile is used. We count a key guess as successful if the sum of the hamming weights of the differences between the returned last two subkeys and the real two subkeys is at most two. Since Chen's attack has a little improvement compared to the result of Gohr, we don't compare it with Chen's result. To better compare with the results of Gohr and Bao *et al.*, we conducted multiple comparison experiments.

5.1 Key Recovery Attack on 11-round Speck32/64

To demonstrate the superiority of our neural distinguisher, we conducted a comparative experiment with Gohr's neural distinguisher. To ensure the fairness of the comparative experiment, except for the difference of neural distinguisher, other parameters should be the same as far as possible.

Experiment 1: The components of key recovery attack $\mathcal{A}^{\text{Speck11}R}$ of 11-round Speck32/64 are as follows.

1. a 2-round differential path $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$;
2. generalized neutral bits of generating multiple-ciphertext pairs; generalized neutral bits of combined response of neural distinguisher: $\{[20], [22], [21], [9, 16], [2, 11, 25], [14]\}$ (refer to Appendix A.1 Table 11).
3. a 7-round neural distinguisher $\mathcal{ND}^{\text{Speck}_{7R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{7R}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{7R}} \cdot \delta$.
4. a 6-round neural distinguisher $\mathcal{ND}^{\text{Speck}_{6R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{6R}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{6R}} \cdot \delta$.

The parameters for recovery the last two subkeys are denoted as follows.

1. n_{kg}: the number of possible values for the bits of k_0, on which the conditions depend.
2. n_{cts}: the number of ciphertext structures.
3. n_b: the number of ciphertext pairs in each ciphertext structures, i.e., $2^{|NB|}$.
4. n_{it}: the total number of iterations on the ciphertext structures.
5. c_1 and c_2: the cutoffs with respect to the scores of the recommended last subkey and second last subkey, respectively.
6. n_{byit1}, n_{cand1} and n_{byit2}, n_{cand2}: the number of iterations and number of key candidates within each iteration in the BayesianKeySearch algorithm for guessing each of the last and the second last subkeys, respectively.

In the experimental verification of the attack $\mathcal{A}^{\text{SPECK11}R}$, the 7-round and 6-round neural distinguisher provided in Sect. 3. Due to too many control experiments, refer to the source code for the specific use value of generalized neutral bits. Concrete parameters and the complexity that our 11-round key recovery attack $\mathcal{A}^{\text{SPECK11}R}$ used are listed.

$$\begin{array}{lll} n_b = 2^6 & n_{ct} = 100/200 & n_{it} = 500 \\ c_1 = 8, c_2 = 10 & n_{byit1} = n_{byit2} = 5 & n_{cand1} = n_{cand2} = 32 \end{array}$$

In Experiment 1, several experiments were done for different values of m and n_{ct}, and the specific experimental results are shown in Table 3. The data complexity is $n_b \times n_{ct} \times m \times 2$. The core of these attacks were examined in 35 trials. The time complexity is $2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$.

Table 3. experiment result of 11-round key recovery attack

	m	Conf.	$n_{ct} = 100$				$n_{ct} = 200$			
			Succ. Rate	Run Time	Data	Time	Succ. Rate	Run Time	Data	Time
Gohr [8]			52.1%	225	$2^{13.64}$	$2^{39.61}$	-	-	-	-
Our	2	1+2+7+1	57.14%	178	$2^{14.64}$	$2^{39.07}$	97.14%	158	$2^{15.64}$	$2^{36.82}$
	4		77.14%	165	$2^{15.64}$	$2^{38.16}$	94.28%	137	$2^{16.64}$	$2^{36.93}$
	8		85.71%	234	$2^{16.64}$	$2^{38.26}$	91.43%	243	$2^{17.64}$	$2^{37.98}$
	16		77.14%	464	$2^{17.64}$	$2^{39.65}$	94.29%	137	$2^{18.64}$	$2^{36.93}$

From Table 3, we know the success rate of our key recovery attacks is significantly improved, and the time complexity is decrease. So, our neural distinguisher has better performance than Gohr's neural distinguisher. To improve performance, we pay the price of increased data complexity. When using different group size m, the data complexity is corresponding m times that of Gohr. In addition, when n_{ct} is 200, the success rate of the key recovery attack can still be significantly improved compared with that when n_{ct} is 100. Although the data complexity of key recovery attacks increases in this case, the time complexity is generally lower than that of Gohr.

Because the probability of 2-round difference path $(0x0211, 0x0a04) \rightarrow (0x00\ 40, 0x0000)$ is 2^{-6}, the average number of correct ciphertext structures only are 1.56 when n_{ct} is 100. Therefore, no correct ciphertext structures may exist, so the key cannot be recovered successfully, and the success rate of the key recovery attack is low. Thus, two aspects affect the success rate of key recovery attack. First, there should be enough plaintext pairs to ensure that after differential propagation, a certain amount of data still meets the difference of input data of neural network. On the other hand, the performance of neural distinguisher is strong enough.

5.2 Key Recovery Attack on 12-round Speck32/64

To verify the performance of our 8-round neural distinguisher, the following experiments were carried out.

Experiment 2: The components of key recovery attack $\mathcal{A}_I^{\text{Speck12}R}$ of 12-round Speck32/64 are as follows.

1. a 2-round differential path $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[9, 16], [2, 11, 25], [6, 29]\}$; generalized neutral bits of combined response of neural distinguisher:$\{[20], [21], [22], [14], [15], [23], [30], [7], [0]\}$ (refer to Appendix A.1 Table 11).
3. a 8-round neural distinguisher $\mathcal{ND}^{\text{Speck}_{8R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{8R}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{8R}} \cdot \delta$.
4. a 7-round neural distinguisher $\mathcal{ND}^{\text{Speck}_{7R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{7R}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{7R}} \cdot \delta$.

In the experimental verification of the attack $\mathcal{A}_I^{\text{Speck12}R}$, the 8-round and 7-round neural distinguisher provided in Sect. 3. Concrete parameters and the complexity that our 12-round key recovery attack $\mathcal{A}_I^{\text{Speck12}R}$ used are listed.

$$\begin{array}{llll} m = 8 & n_b = 2^9 & n_{cts} = 500 & n_{it} = 2000 \\ c_1 = 3 & c_2 = 8 & n_{byit1} = n_{byit2} = 5 & n_{cand1} = n_{cand2} = 32 \end{array}$$

The data complexity is $n_b \times n_{ct} \times m \times 2$, that is, $2^9 \times 500 \times 8 \times 2$, i.e., $2^{22.97}$ plaintexts. The core of the attack was examined in 70 trials. There are 25 succeeded trials, the average run times of every trails in our server is 4785s. Thus, we count the success rate is $^{25}/_{70}$, which is 0.357. The time complexity is $2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$, that is $2^{28} \times 4785 \times {}^{500}/_{225} \times \log_{1-0.357} 0.01$, i.e., $2^{44.75}$. For more detailed comparison results, see Table 4.

Table 4. experiment result of 12-round key recovery attack using 2-round differential path

	Conf.	Succ. Rate	Run Time	Data	Time
Gohr [8]		40%	-	$2^{22.97}$	$2^{43.4}$
Bao [2]	1+2+8+1	86%	-	2^{22}	$2^{46.11}$
Our		35.7%	4785	$2^{22.97}$	$2^{44.75}$

When group size m is greater than 8, we do not get better results. The possible reason is that when m increases, the probability of the newly used neutral bit is not high enough, or the performance of neural distinguisher is not good enough.

In pursuit of better key recovery attack, we considered combining the 3-round differential path and the more powerful 7-round neural distinguisher. In

this case, unconditional SNBS are enough for our 7-round neural distinguisher. Bao found 12 CSNBS for 3-round differential path $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$. But in order for the 3-round differential path to hold, 3 conditions need to be met (refer to Appendix A.1 Table 12). Because of the extended one round on top of these 3-round differentials, these conditions cannot be fulfilled by chosen data without guessing corresponding bits of k_0. To make the experimental verification economic, we tested the core of the attack with the three conditions being fulfilled only. Because the prepended differential path are valid when the keys fulfilling $k_2[12] \neq k_2[11]$, thus we tested for these valid keys only, and the presented attack works for 2^{63} keys.

Experiment 3: The components of key recovery attack $\mathcal{A}_{II}^{\text{SPECK12}R}$ of 12-round SPECK32/64 are as follows.

1. a 3-round differential path $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[22], [20], [13]\}$; generalized neutral bits of combined response of neural distinguisher: $\{[12, 19], [14, 21], [6, 29], [30], [0, 8, 31], [5, 28]\}$ (refer to Appendix A.1 Table 13).
3. a 7-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{7R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{7R}} \cdot \mu$ and $\mathcal{ND}^{\text{SPECK}_{7R}} \cdot \delta$.
4. a 6-round neural distinguisher $\mathcal{ND}^{\text{SPECK}_{6R}}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{6R}} \cdot \mu$ and $\mathcal{ND}^{\text{SPECK}_{6R}} \cdot \delta$.

In the experimental verification of the attack $\mathcal{A}_{II}^{\text{SPECK12}R}$, the 8-round and 7-round neural distinguisher provided in Sect. 3. Concrete parameters and the complexity that our 12-round key recovery attack $\mathcal{A}_{II}^{\text{SPECK12}R}$ used are listed.

$n_{kg} = 2^4$	$m = 8$	$n_b = 2^6$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 7, c_2 = 10$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

The data complexity is $n_{kg} \times n_b \times n_{ct} \times m \times 2$, that is, $2^4 \times 2^6 \times 2^{11} \times 8 \times 2$, i.e., 2^{25} plaintexts. The core of the attack was examined in 35 trials. There are 35 succeeded trials, the average run times of every trails in our server is 577s. Thus, we count the success rate is 1. The time complexity is $2^{28} \times rt \times {}^{500}/_{225}$, that is $2^{28} \times 577 \times {}^{500}/_{225}$, i.e., $2^{42.32}$. For more detailed comparison results, see Table 5.

Table 5. experiment result of 12-round key recovery attack using 3-round differential path

	Conf.	Succ. Rate	Run Time	Data	Time
Bao [2]	1+3+7+1	81%	-	$2^{18.58}$	$2^{44.82}$
Our		100%	577	2^{25}	$2^{42.32}$

It is the first time that a key recovery attack has a 100 percent success rate. When we don't use the weak key and modify the ciphertext structure to 2^{12}, the

success rate of key recovery attack is only 51%. The main reason for the reduced success rate is that no correct ciphertext structures exists. Notably if we exclude the absence of the correct ciphertext structure, our success rate is still 100%. This shows that our neural distinguisher has a very strong ability to distinguish between ciphertext and random numbers.

5.3 Key Recovery Attack on 13-round Speck32/64

Combining 3-round differential path with an 8-round neural distinguisher, we examine how far a practical attack can go on 13-round Speck32/64.

Experiment 4: The components of key recovery attack $\mathcal{A}^{\text{Speck13}R}$ of 13-round Speck32/64 are as follows.

1. a 3-round differential path $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[22], [20], [13]\}$; generalized neutral bits of combined response of neural distinguisher: $\{[5, 28], [15, 24], [12, 19], [6, 29], [6, 11, 12, 18], [4, 27, 29], [14, 21], [0, 8, 31], [30]\}$ (refer to Appendix A.1 Table 13).
3. a 8-round neural distinguisher $\mathcal{ND}^{\text{Speck}8R}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}8R} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}8R} \cdot \delta$.
4. a 7-round neural distinguisher $\mathcal{ND}^{\text{Speck}7R}$ under difference $(0x0040, 0x0000)$ and its wrong key response profiles $\mathcal{ND}^{\text{Speck}7R} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}7R} \cdot \delta$.

In the experimental verification of the attack $\mathcal{A}^{\text{Speck13}R}$, the 8-round and 7-round neural distinguisher provided in Sect. 3. Concrete parameters and the complexity that our 13-round key recovery attack $\mathcal{A}^{\text{Speck13}R}$ used are listed.

$n_{kg} = 2^7$	$m = 8$	$n_b = 2^9$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 5, c_2 = -86$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

To make the experimental verification economic, we tested the core of the attack with the seven conditions being fulfilled only, including three conditions for confirming the 3-round differential path (refer to Appendix A.1 Table 12) and four conditions for increasing probability of neutral bits (refer to Appendix A.1 Table 13). Because the prepended differential path are valid when the keys fulfilling $k_2[12] \neq k_2[11]$, we tested for these valid keys only, and the presented attack works for 2^{63} keys.

The data complexity is $n_{kg} \times n_b \times n_{ct} \times m \times 2$, that is, $2^7 \times 2^9 \times 2^{11} \times 8 \times 2$, i.e., 2^{31} plaintexts. The core of the attack was examined in 35 trials. There are 17 succeeded trials, the average run times of every trails in our server is 13690s. Thus, we count the success rate is $^{17}/_{35}$, which is 0.4857. The time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$, that is $2^7 \times 2^{28} \times 13690 \times {}^{500}/_{225} \times \log_{1-0.4857} 0.01$, i.e., $2^{52.68}$. For more detailed comparison results, see Table 6.

Table 6. experiment result of 13-round key recovery attack

	Conf.	Succ. Rate	Run Time	Data	Time
Bao [2]	1+3+8+1	61%	-	2^{29}	$2^{53.85}$
Our		48.57%	13690	2^{31}	$2^{52.68}$

5.4 Using Multiple Differential paths

Bao *et al.* found that the output difference path matters to neural distinguisher, but not the input difference. Hence, more than one differential paths can be prepended to neural distinguisher, as long as they share the same output difference. Multiple such differential paths can share some neutral bits. Using such differential paths might enable data reuse, thus slightly reducing data complexity. Also, there are three sufficient (linear) conditions to conform to the 3-round differentials (refer to Appendix A.1 Table 12). Bao launched the $R_{1+3+7+1}$-round key recovery attack using four sub-optimal differential paths and the $R_{1+3+8+1}$-round key recovery attack using two sub-optimal differential paths. However, we only use one differential path in our experiment when we launch the key recovery attack by the prepended 3-round differential path on top of neural distinguisher. When using two sub-optimal differential path to generate multiple-ciphertext pairs, we only need to use two conditions to conform to the two 3-round sub-optimal differential path. So, in the ideal situation, we are able to halve both the data complexity and the time complexity of the 12-round and 13-round key recovery attacks.

6 Neural Distinguishers on Reduced-Round Simon32/64

In CRYPTO'19, Gohr used the Residual Network (ResNet) to train neural distinguisher for Speck32/64. Considering that the Dense Network (DenseNet) [11] and the Squeeze-and-Excitation Network (SENet) [10] show advantages in specific tasks than ResNet, Bao *et al.* modified the structure of Gohr's neural network using DenseNet and SeNet and obtained (7-11)-round neural distinguishers for Simon32/64. This section presents our neural distinguisher on Simon32/64. Compared to Bao's results in [2], we improve the accuracy of the 11-round neural distinguisher and train a new 12-round neural distinguisher for Simon32/64.

6.1 Network Structure

In Sect. 3, we detailed the network structure of neural distinguisher for the Speck32/64, but found that the trained neural distinguisher model had a deficiency. In the network structure of the neural distinguisher, the prediction head uses three fully connected layers, resulting in too many model parameters, which makes the model training time too long and the model file too large. When designing the network structure of the neural distribution for Simon32/64, we

solved this problem by modifying the structure of the prediction head and replacing the fully connected layer with a GlobalAveragePooling layer. The network structure of neural distinguisher for SIMON32/64 is similar to SPECK32/64 overall, with only minor modifications based on round function of SIMON32/64.

Input Representation. In the round function of SPECK32/64, we can directly compute (y_{r-1}, y'_{r-1}) without knowing the $(r-1)$-th subkey. However, we only get $(y_{r-1} \oplus y'_{r-1})$ without knowing the $(r-1)$-th subkey for SIMON32/64 because of the different of round function. Thus, the neural network accept data of the form $(x_r, x'_r, y_r, y'_r, y_{r-1} \oplus y'_{r-1})$. So, the input layer has m group ciphertext pairs consisting of $2.5L$ units likewise arranged in a $[m, \omega, \frac{2.5L}{\omega}]$ array, where $L = 32, \omega = 16$ for SIMON32/64.

Initial Convolution. In the initial convolution of SPECK32/64, the Inception consists of four convolutional layers whose convolutional kernel size is set according to the cyclic shift operation in the round function of SPECK32/64. There are three convolutional layers in the Inception for SIMON32/64, and the size of the convolutional kernels is 1, 2, and 8. The input layer is connected to the initial convolutional layer, which comprises three convolution layers with N_f channels of different kernel sizes. The three convolution layers are concatenated at the channel dimension. Batch normalization is applied to the output of concatenate layers. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[m, \omega, 3N_f]$ matrix is passed to the Convolutional Blocks layer. The structure of the initial convolutional layer for SIMON32/64 can be seen in Figure 7.

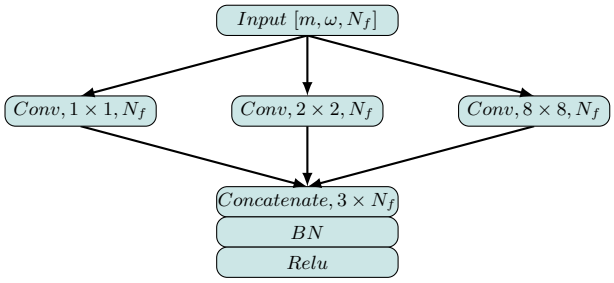
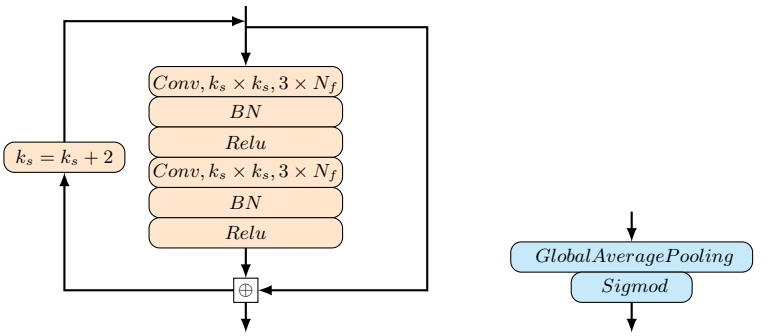


Fig. 7. The initial convolution layer

Convolutional Blocks. For SIMON32/64, the convolutional blocks layer of the neural distinguisher model are the same as SPECK32/64, except for the shape of the input is different. The structure of convolutional blocks layer can be seen in Figure 8(a).

Prediction Head. For SIMON32/64, we replaced the fully connected layer with a GlobalAveragePooling layer, which effectively reduced the number of parameters in the model training process, thereby reducing the training time and model file size. Sadly, the accuracy of the model has decreased slightly. The prediction head consists of a GlobalAveragePooling layer and an output unit using a *Sigmoid* activation function. The structure of the prediction head is shown in Figure 8(b).



(a) The convolutional blocks (b) The prediction head

6.2 The Training of Neural Distinguisher

Training using the basic scheme. We used two different numbers of training and test sets to train the neural network. The first set of experiments is to use N and M data as the training and test sets of the neural network. The second set of experiments is to use $N \times m$ and $M \times m$ data as the training set and test set of the neural network. In [2], Bao *et al.* obtained the (7-9)-round neural distinguisher using the basic training scheme and trained the (10-11)-round neural distinguisher using the KeyAveraing algorithm and the staged training method separately. Also, the accuracy of the 11-round neural distinguisher trained by Bao is 51.73%. For training parameters, refer to basic training scheme of SPECK32/64 in Sect. 3. By modifying the network structure and using the basic training scheme, we trained the neural distinguisher to recognize output pairs of (7-11)-round SIMON32/64 with the input difference $(0x0000, 0x0040)$. The accuracy of our 11-round neural distinguisher is significantly higher than Bao's when the group size m takes a different value. Summaries are presented in Table 7 for detailed result.

Training using the Staged Training Method. For 12 rounds, the training scheme described above fails, i.e., the model does not learn to approximate any helpful function. We still succeeded in training a 12-round neural distinguisher

of SIMON32/64 by using several stages of pre-training. First, we use our 11-round distinguisher to recognize 9-round SPECK32/64 with the input difference (0x0440,0x0100) (the most likely difference to appear three rounds after the input difference (0x0000,0x0040). The training was done on $10^7 \times m$ samples for twenty epochs with cyclic learning rates. Then we trained the distinguisher so obtained to recognize 12-round SPECK32/64 with the input difference $(0x0000, 0x0040)$ by processing $10^7 \times m$ freshly generated samples for ten epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another $10^7 \times m$ fresh samples each. During the staged training, the batch size was changed according to the value of parameter m, to match the computing performance of the equipment.

6.3 Result

Test Set Accuracy and Wrong Key Response Profile. We summarize the experimental results of (7-12)-round of the neural distinguisher model. The (7-11)-round distinguisher was trained using the basic training method. Using the staged training method, a 12-round distinguisher was derived from an 11-round distinguisher.

The comparison results with the accuracy of the neural distinguisher of Bao are presented in in Table 7. It shown the accuracy of (7-12)-round neural distinguisher. As the number of ciphertext pairs m increases, the accuracy of the neural distinguisher also increases. However, when the value of m is too large, the demand for neutral bits will increase, thereby increasing the data complexity of key recovery attack. Therefore, an appropriate value of m should be selected for key recovery attack. Compared to Bao's, the accuracy of our neural distinguisher was significantly improved when the group size m takes different values. Figure 8 shows the wrong key response profile for our (9-12)-round neural distinguisher. As we can see from the figure, when the Hamming Distance between the correct key and the wrong key is smaller, the score of the neural distinguisher is higher, and vice versa, it is smaller. This makes it easier to judge how far the guess key deviates from the correct key, which shows that our neural distinguisher can effectively distinguish between ciphertext and random numbers.

Table 7. accuracy of neural distinguisher for (7-12)-round SIMON32/64

R	Bao [2]	m=2		m=4		m=8		m=16	
		N	$N \times m$	N	$N \times m$	N	$N \times m$	N	$N \times m$
7	0.9802	0.9987	0.9995	0.9999	0.9999	1.0000	0.9999	1.0000	1.0000
8	0.8148	0.9306	0.9295	0.9815	0.9842	0.9978	0.9989	0.9981	0.9999
9	0.6532	0.7251	0.7240	0.7991	0.8095	0.8774	0.8958	0.9344	0.9630
10	0.5629	0.5917	0.5907	0.6239	0.6339	0.6716	0.6900	0.7230	0.7608
11	0.5173	0.5193	0.5240	0.5343	0.5387	0.5441	0.5591	0.5339	0.5878
12	-	-	-	-	-	-	0.5152	-	0.5225

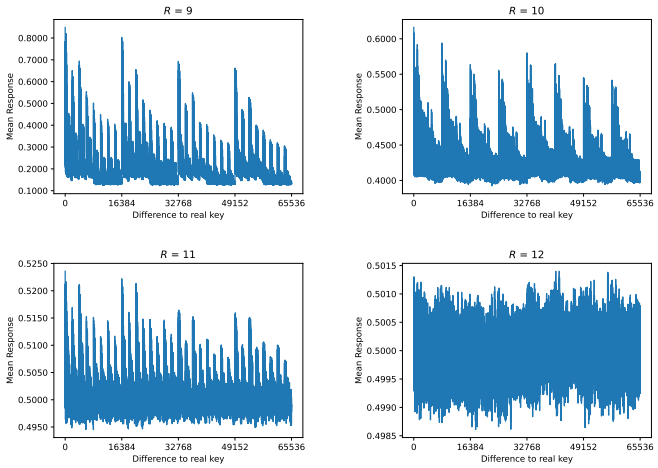


Fig. 8. Wrong key response profile for (9-12)-round SIMON32/64 where $m = 8$

7 Key Recovery Attack on Reduced-Round SIMON32/64

Under a similar framework to the key recovery attack on SPECK32/64, the trained neural distinguishers can be prepended with a differential path to perform key recovery attack. This article only list the experimental procedure and results of a key recovery attack where group size $m = 8$. We improved 16-round and devised the first 17-round neural-distinguisher-based key recovery attacks on SIMON32/64. Sadly, due to the low accuracy of the 12-round neural distinguisher and the lack of a sufficient number of generalized neutral bits, we were unable to successfully carry out the 18-round key recovery attack for SIMON32/64.

7.1 Key Recovery Attack on 16-round SIMON32/64

Experiment 5: The components of key recovery attack $\mathcal{A}^{\text{SIMON16}R}$ of 16-round SIMON32/64 are as follows.

1. a 3-round differential path $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[2], [3], [4]\}$; generalized neutral bits of combined response of neural distinguisher:$\{[6], [8], [9], [10], [18], [22], [0, 24], [12, 26]\}$ (refer to Appendix A.2 Table 14).
3. a 11-round neural distinguisher $\mathcal{ND}^{\text{SIMON}_{11R}}$ under difference $(0x0000, 0x0040)$ and its wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{11R}} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{11R}} \cdot \delta$.
4. a 10-round neural distinguisher $\mathcal{ND}^{\text{SIMON}_{10R}}$ under difference $(0x0000, 0x0040)$ and its wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \delta$.

In the experimental verification of the attack $\mathcal{A}^{\text{SIMON16}R}$, the 11-round and 10-round neural distinguisher provided in Sect. 6. The goal is to recover the last two subkeys k_{15} and k_{14}. At the beginning, we guess two key bits of k_0, that is $k_0[1]$ and $k_0[3]$, because for the 3-round differential, the conditions for correct pairs are $x_1[1] = x_1'[1] = 0$ and $x_1[3] = x_1'[3] = 0$. Thus, n_{kg} is 2, and there are 2^2 loops. The concrete parameter of the attack are as follows. The accuracy of $\mathcal{ND}^{\text{SIMON}_{11R}}$ is 0.5591 (note that $(2 \times (0.5591 - 0.5))^{-2} \approx 2^{-6.13}$). Thus, by Conjecture 1, $n_b = 2^8$ should be enough.

$n_{kg} = 2^2$	$m = 8$	$n_b = 2^8$	$n_{cts} = 2^8$
$n_{it} = 2^9$	$c_1 = 35, c_2 = 70$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

The data complexity is $n_{kg} \times m \times n_{cts} \times n_b \times 2$, that is, 2^{22} plaintexts. To examine the performance of the attack, experiments are done using 5 threads on the same GPU server. In total 25 trials are run, there are 18 success trials, for which the returned last two subkeys have a Hamming distance to real subkeys of at most two. Thus, the success rate is computed as $^{18}/_{25}$, i.e., 0.72. The average running time of the experiment is 1168.7s in 25 trials. The time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$, that is $2^2 \times 2^{28} \times 1168 \times {}^{500}/_{225} \times \log_{1-0.72} 0.01$, i.e., $2^{43.19}$. For more detailed comparison results, see Table 8.

Table 8. experiment result of 16-round key recovery attack

	Conf.	Succ. Rate	Run Time	Data	Time
Bao [2]	1+3+11+1	49%	-	2^{21}	$2^{46.98}$
Our		72%	1168	2^{22}	$2^{43.19}$

7.2 Key Recovery Attack on 17-round SIMON32/64

Experiment 6: The components of key recovery attack $\mathcal{A}^{\text{SIMON17}R}$ of 17-round SIMON32/64 are as follows.

1. a 4-round differential path $(0x1000, 0x4440) \rightarrow (0x0000, 0x0040)$;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[2], [6], [12, 26]\}$; generalized neutral bits of combined response of neural distinguisher: $\{[10, 14, 28]\}$ and five neutral bits conditioned on $x[1, 15], x[15, 13], x[13, 11], x[11, 9], x[9, 7]$ (refer to Appendix A.2 Table 15).
3. a 11-round neural distinguisher $\mathcal{ND}^{\text{SIMON}_{11R}}$ under difference $(0x0000, 0x0040)$ and its wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{11R}} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{11R}} \cdot \delta$.
4. a 10-round neural distinguisher $\mathcal{ND}^{\text{SIMON}_{10R}}$ under difference $(0x0000, 0x0040)$ and its wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \delta$.

In the experimental verification of the attack $\mathcal{A}^{\text{SIMON17}R}$, the 11-round and 10-round neural distinguisher provided in Sect. 6. The goal is to recover the last

two subkeys k_{16} and k_{15}. At the beginning, we guess two key bits of k_0, that is $k_0[3]$ and $k_0[5]$, because for the 4-round differential, the conditions for correct pairs are $x_1[5] = x_1'[5] = 0$ and $x_1[3] = x_1'[3] = 0$; and six key bits $k_0[1], k_0[15], k_0[13], k_0[11], k_0[9], k_0[7]$ for employing five conditional neutral bits (refer to Appendix A.2 Table 15). For different values of the chosen data pairs, with guessed values of the six key bits, we choose different neutral bit-sets to generate the structure. Thus, n_{kg} is 2+6, and there are 2^8 loops. The concrete parameter of the attack are as follows. The accuracy of $\mathcal{ND}^{\text{SIMON}_{11R}}$ is 0.5591 (note that $(2 \times (0.5591 - 0.5))^{-2} \approx 2^{-6.13}$). However, we only have 6 generalized neutral bits, which makes the success rate of key recovery attacks low.

$n_{kg} = 2^8$	$m = 8$	$n_b = 2^6$	$n_{cts} = 2^{10}$
$n_{it} = 2^{11}$	$c_1 = 15, c_2 = 65$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

The data complexity is $n_{kg} \times m \times n_{cts} \times n_b \times 2$, that is, 2^{28} plaintexts. To examine the performance of the attack, experiments are done using 5 threads on the same GPU server. In total 50 trials are running, there are 4 success trials, for which the returned last two subkeys have a Hamming distance to real subkeys of at most two. Thus, the success rate is computed as $4/50$, i.e., 0.08. The average running time of the experiment is 3356.4s in 50 trials. The time complexity is $n_{kg} \times 2^{28} \times rt \times 500/225 \times \log_{1-sr} 0.01$, that is $2^8 \times 2^{28} \times 3356 \times 500/225 \times \log_{1-0.08} 0.01$, i.e., $2^{54.65}$. For more detailed comparison results, see Table 9.

Table 9. experiment result of 17-round key recovery attack

	Conf.	Succ. Rate	Run Time	Data	Time
our	1+4+11+1	0.08%	3356.4	2^{28}	$2^{54.65}$

8 Conclusion and Discussions

In this article, we present a new network structure to train neural distinguisher, which takes multiple-ciphertext pairs as the input of neural distinguisher model and uses multiple-parallel convolution layers to capture the features of different dimensions of cryptographic algorithms, thus improving the accuracy of neural distinguisher and obtained more rounds neutral distinguisher. Focusing on the problem of the same distribution of data left by Chen's work, we propose a solution using neutral bits to generate multiple-plaintext pairs, thus significantly improve the success rate and decrease the time complexity of key recovery attacks.

Limit to More Rounds of Key Recovery Attack . The differential-neural cryptanalysis mainly consists of two parts: differential path and neural distinguisher. Bao *et al.* use a higher probabilistic neutral bits and the same neural

distinguisher used in the 11-round and 12-round attacks on SPECK32/64 in [8], getting a significantly result than Gohr. Now, we have greatly improved the performance of neural distinguisher and enhanced the key recovery attack. However, we are not satisfied with the current results because it does not reach the full potential of neural distinguisher. On the one hand, for key recovery attacks of 13 rounds SPECK32/64, we only used neural distinguisher with group size $m = 8$, where the accuracy of neural distinguisher only is 55.9%. When the group size m is equal to 16, the accuracy of our 8-round neural distinguisher reaches 58.53%. However, we need more generalized neutral bits when $m = 16$ to launch a key recovery attack. Thus, the number of generalized neutral bits limits the ability of our neural distinguisher to reach their full potential. On the other hand, the accuracy of our 12-round neural distinguisher of SIMON32/64 is lower, we cannot use it for key recovery attack. Therefore, we need to make in-depth use of the nature of the structure of the cryptographic algorithm and the more effective structure of the neural network to improve the accuracy of 12-round neural distinguisher for SIMON32/64. We need to search for more generalized neutral bits and improve the accuracy of neural distinguisher to launch more rounds of key recovery attack with the joint efforts of the two components.

References

1. AlKhzaimi, H., Lauridsen, M.M.: Cryptanalysis of the SIMON family of block ciphers. IACR Cryptol. ePrint Arch. p. 543 (2013), http://eprint.iacr.org/2013/543
2. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Conditional differential-neural cryptanalysis. IACR Cryptol. ePrint Arch. p. 719 (2021)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. p. 404 (2013), http://eprint.iacr.org/2013/404
4. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_28
5. Biham, E., Chen, R.: Near-collisions of SHA-0. In: CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 290–305. Springer (2004)
6. Chen, Y., Yu, H.: A new neural distinguisher model considering derived features from multiple ciphertext pairs. IACR Cryptol. ePrint Arch. p. 310 (2021), https://eprint.iacr.org/2021/310
7. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 147–164. Springer (2014)
8. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer (2019)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778. IEEE Computer Society (2016)
10. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City,

UT, USA, June 18-22, 2018. pp. 7132–7141. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00745
11. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 2261–2269. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.243
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (Poster) (2015)
13. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1–9. IEEE Computer Society (2015). https://doi.org/10.1109/CVPR.2015.7298594

A Used Generalized Neural Bit-sets for Key Recovery Attack

A.1 Generalized Neural Bit-sets for Speck32/64

Neutral bit-sets (NB) Used in [8] for 2-round Classical Differential: The signal from distinguisher will rather weak. Gohr boosts it by using $|NB|$ probabilistic neutral bits to create from each plaintext pair. A plaintext structure consisting of $2^{|NB|}$ plaintext pairs that are expected to pass the initial 2 round classical differential together. Concretely, neutral bits are probabilistically neutral are summarized as follows.

Table 10. (Probabilistic) single-bit neutral bit for 2-round Classical Differential $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$ of Speck32/64 [8]

NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.
[20]	1	[21]	1	[22]	1	[14]	0.965	[15]	0.938	[23]	0.812	[7]	0.806
[30]	0.809	[0]	0.763	[8]	0.664	[24]	0.649	[31]	0.644	[1]	0.574		

Simultaneous-neutral bit-sets (SNBS) used in [2] for 2-round classical differential: for the prepended 2-round differential on top of neural distinguisher, Bao *et al.* [2] can experimentally obtain 3 complete NB and 2 SNBS using exhaustive search. Concretely, for the 2-round differential$(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$, bit and bit-sets that are (probabilistically) (simultaneous-)neutral are summarized in Table 11.

Table 11. (Probabilistic) SNBS for 2-round Classical Differential $(0x0211, 0x0a04) \rightarrow (0x0040, 0x0000)$ of Speck32/64 [2]

NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.
[20]	1	[21]	1	[22]	1	[9,16]	1	[2,11,25]	1	[14]	0.965	[15]	0.938
[6,29]	0.91	[23]	0.812	[30]	0.809	[7]	0.806	[0]	0.754	[11,27]	0.736	[8]	0.664

Conditional simultaneous-neutral bit-sets (CSNBS) used in [2] for 3-round classical differential: Bao *et al.* found that there are the three sufficient conditions for a pair $(x, y), (x', y')$ to conform the 3-round differential $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$, summarized in Table 12. Concretely, for the 3-round four sub-optimal differential, bit and bit-sets that are (probabilistically) conditional simultaneous-neutral are summarized in Table 13.

Table 12. Three sufficient conditions conform the 3-round sub-optimal differential [2]

$(0x8020, 0x4101)$ $\downarrow$ $(0x0040, 0x0000)$	$(0x8060, 0x4101)$ $\downarrow$ $(0x0040, 0x0000)$	$(0x8021, 0x4101)$ $\downarrow$ $(0x0040, 0x0000)$	$(0x8061, 0x4101)$ $\downarrow$ $(0x0040, 0x0000)$
$x[7] = 0$	$x[7] = 0$	$x[7] = 0$	$x[7] = 0$
$x[5] \oplus y[14] = 1$	$x[5] \oplus y[14] = 0$	$x[5] \oplus y[14] = 1$	$x[5] \oplus y[14] = 0$
$x[15] \oplus y[8] = 0$	$x[15] \oplus y[8] = 0$	$x[15] \oplus y[8] = 1$	$x[15] \oplus y[8] = 1$

Table 13. (Probabilistic) (simultaneous-)neutral bit-sets for 3-round differential $(0x8020, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8060, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8021, 0x4101) \rightarrow (0x0040, 0x0000)$, $(0x8061, 0x4101) \rightarrow (0x0040, 0x0000)$ of SPECK32/ 64 [2]

	(8020,4101)		(8060,4101)		(8021,4101)		(8061,4101)		
Bit-set	Pre.	Post.	Pre.	Post.	Pre.	Post.	Pre.	Post.	Condition
[22]	0.995	1.000	0.995	1.000	0.996	1.000	0.997	1.000	-
[20]	0.986	1.000	0.997	1.000	0.996	1.000	0.995	1.000	-
[13]	0.986	1.000	0.989	1.000	0.988	1.000	0.992	1.000	-
[12,19]	0.986	1.000	0.995	1.000	0.993	1.000	0.986	1.000	-
[14,21]	0.855	0.860	0.874	0.871	0.881	0.873	0.881	0.876	-
[6,29]	0.901	0.902	0.898	0.893	0.721	0.706	0.721	0.723	-
[30]	0.803	0.818	0.818	0.860	0.442	0.442	0.412	0.407	-
[0,8,31]	0.855	0.859	0.858	0.881	0.000	0.000	0.000	0.000	-
[5,28]	0.495	1.000	0.495	1.000	0.481	1.000	0.469	1.000	$x[12] \oplus y[5] = 1$
[15,24]	0.482	1.000	0.542	1.000	0.498	1.000	0.496	1.000	$y[1] = 0$
[6,11,12,18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$
[4,27,29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$

A.2 Generalized Neural Bit-sets for Simon32/64

For an input pair $((x, y), (x', y'))$ to conform the 3-round differential $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$, one has conditions that

$$\begin{cases} x[1] = x'[1] = 0 \\ x[3] = x'[3] = 0 \end{cases}$$

Table 14. NB and SNBS for 3-round Classical Differential $(0x0440, 0x1000) \rightarrow (0x0000, 0x0040)$ of SIMON32/64 [2]

[2]	[3]	[4]	[6]	[8]	[9]
[10]	[18]	[22]	[0,24]	[12,26]	

For an input pair $((x, y), (x', y'))$ to conform the 4-round differential $(0x1000, 0x4440) \to (0x0000, 0x0040)$, one has conditions that

$$\begin{cases} x[5] = x'[5] = 0 \\ x[3] = x'[3] = 0 \end{cases}$$

Table 15. CSNBS for 4-round Classical Differential $(0x1000, 0x4440) \to (0x0040, 0x0000)$ of SIMON32/64 [2]

Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.
$x[1, 15]$		$x[15, 13]$		$x[13, 11]$		$x[11, 9]$		$x[9, 7]$	
[24,10]	00	[22,8]	00	[20]	00	[18,4]	00	[16,8]	00
[24,10,9]	10	[22,8,7]	10	[20,5]	10	[18,4,3]	10	[16,8,1]	10
[24,10,0]	01	[22,8,14]	01	[20,12]	01	[18,4,10]	01	[16]	01
[24,10,9,0]	11	[22,8,7,14]	11	[20,12,5]	11	[18,4,10]	11	[16,1]	11

C.: Condition on $x[i, j]$, e.g., $x[i, j] = 10$ means $x[i] = 1$ and $x[j] = 0$.

B train neural distinguisher with N and $N \times m$ data

B.1 train neural distinguisher for 5-round Speck32/64

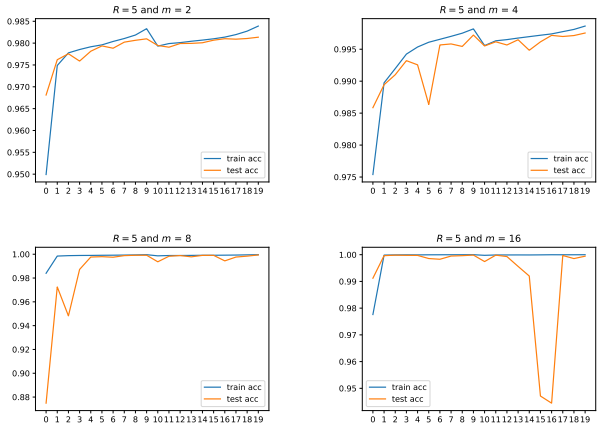


Fig. 9. Train neural distinguisher with N data for 5-round Speck32/64

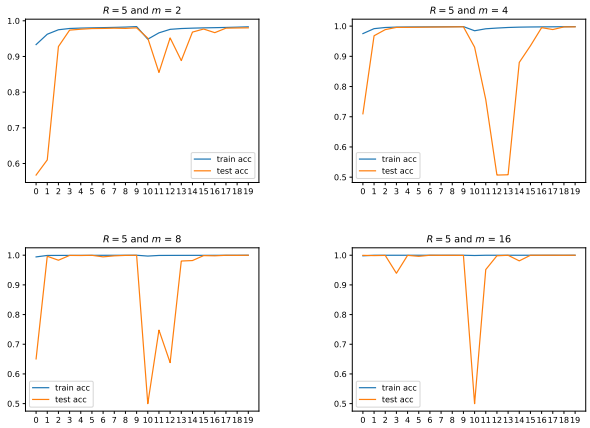


Fig. 10. Train neural distinguisher with $N \times m$ data for 5-round Speck32/64

B.2 train neural distinguisher for 6-round Speck32/64

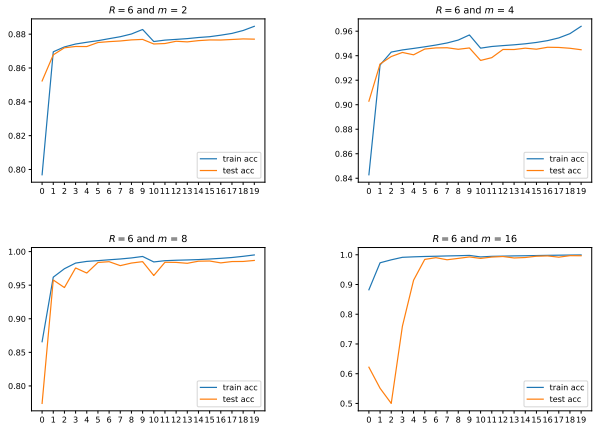


Fig. 11. Train neural distinguisher with N data for 6-round SPECK32/64

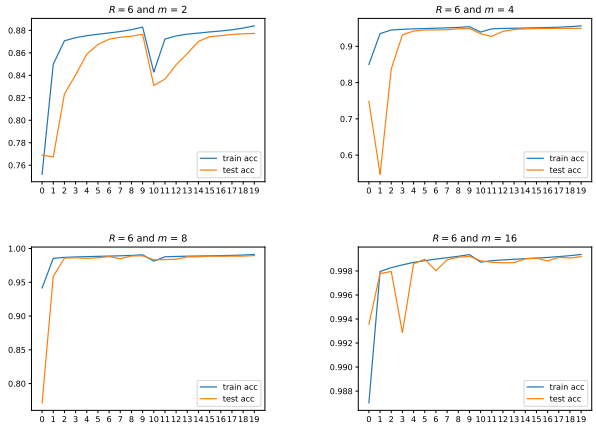


Fig. 12. Train neural distinguisher with $N \times m$ data for 6-round SPECK32/64

B.3 train neural distinguisher for 7-round Speck32/64

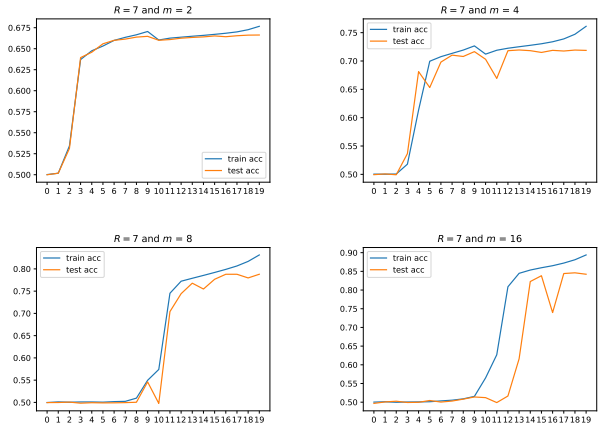


Fig. 13. Train neural distinguisher with N data for 7-round SPECK32/64

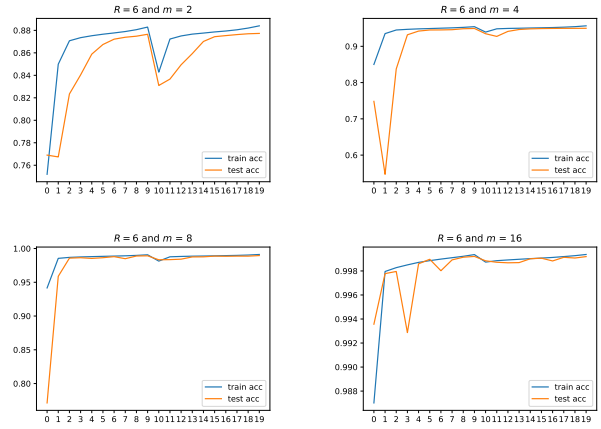


Fig. 14. Train neural distinguisher with $N \times m$ data for 7-round SPECK32/64

B.4 train neural distinguisher for 7-round Simon32/64

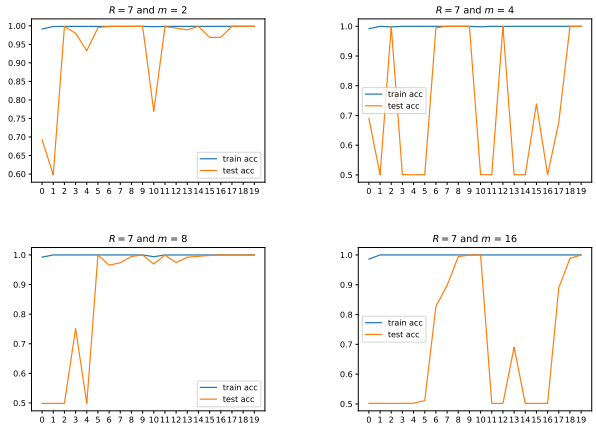


Fig. 15. Train neural distinguisher with N data for 7-round SIMON32/64

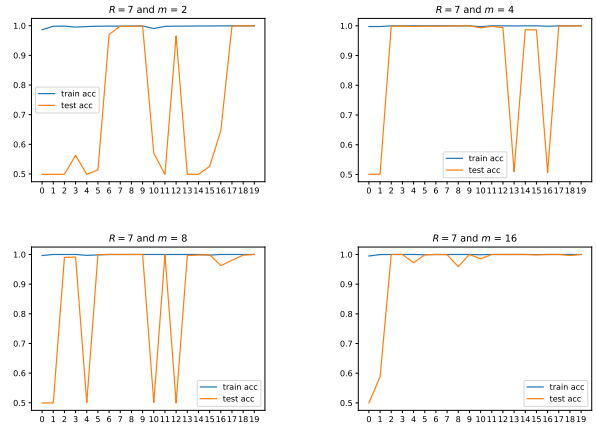


Fig. 16. Train neural distinguisher with $N \times m$ data for 7-round SIMON32/64

B.5 train neural distinguisher for 8-round Simon32/64

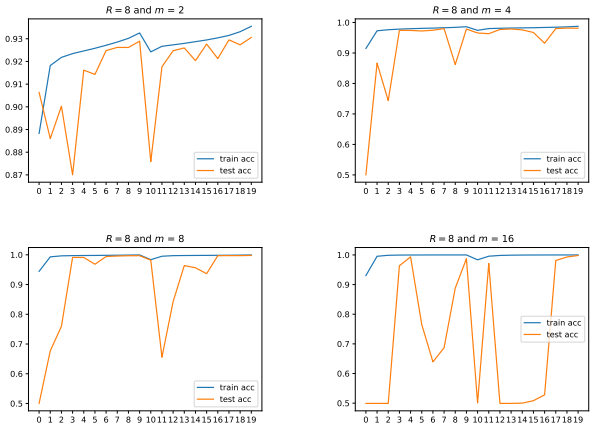


Fig. 17. Train neural distinguisher with N data for 8-round SIMON32/64

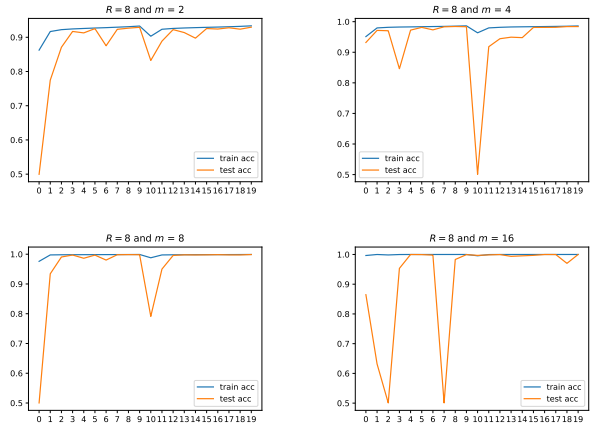


Fig. 18. Train neural distinguisher with $N \times m$ data for 8-round SIMON32/64

B.6 train neural distinguisher for 9-round Simon32/64

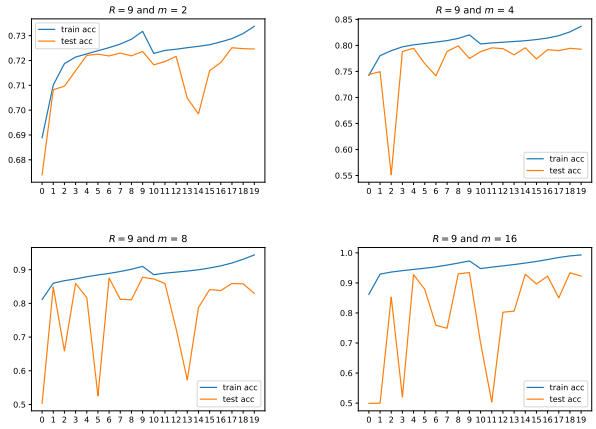


Fig. 19. Train neural distinguisher with N data for 9-round SIMON32/64

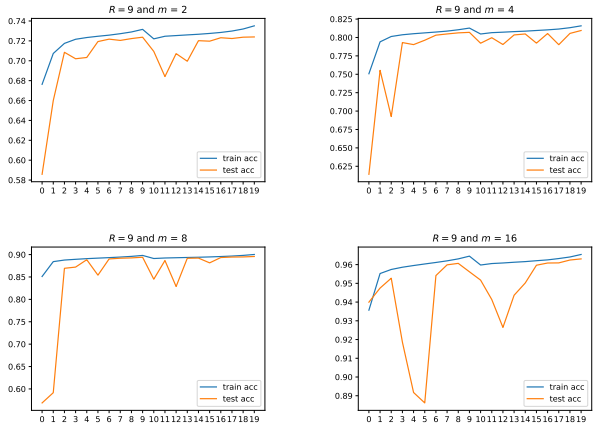


Fig. 20. Train neural distinguisher with $N \times m$ data for 9-round SIMON32/64

B.7 train neural distinguisher for 10-round Simon32/64

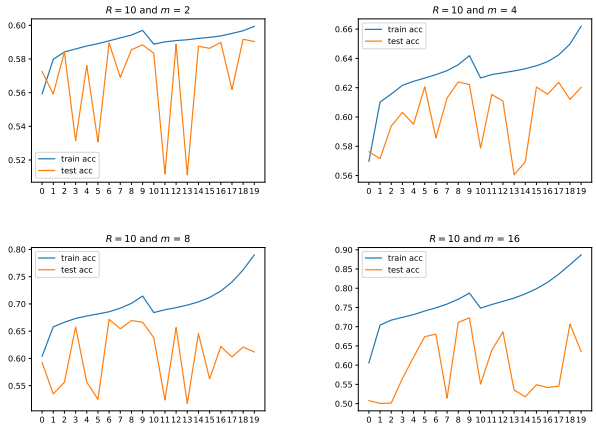


Fig. 21. Train neural distinguisher with N data for 10-round SIMON32/64

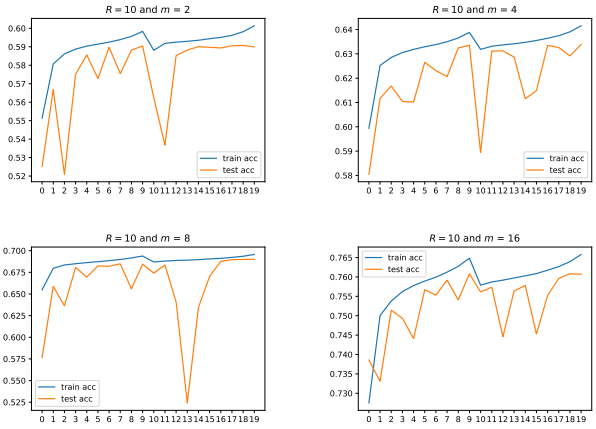


Fig. 22. Train neural distinguisher with $N \times m$ data for 10-round SIMON32/64

B.8 train neural distinguisher for 11-round Simon32/64

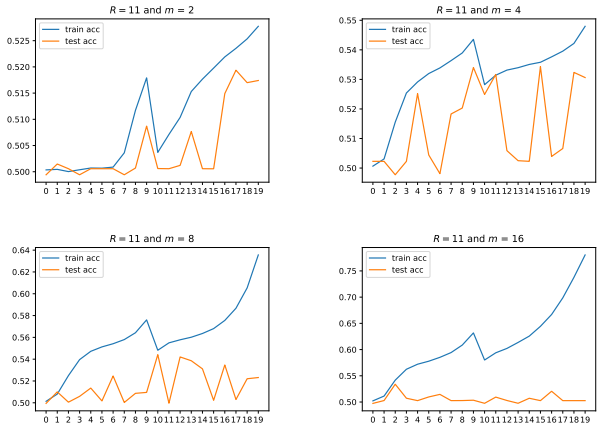


Fig. 23. Train neural distinguisher with N data for 11-round SIMON32/64

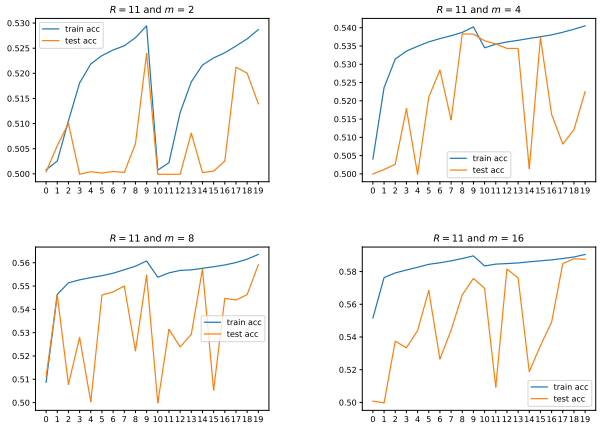


Fig. 24. Train neural distinguisher with $N \times m$ data for 11-round SIMON32/64