Improving Differential-Neural Cryptanalysis

Liu Zhang[1,2], Zilong Wang[1,2] and Baocang Wang[3]

[1] School of Cyber Engineering, Xidian University, Xi'an, China,
liuzhang@stu.xidian.edu.cn,zlwang@xidian.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[3] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an,
China,bcwang79@aliyun.com

Abstract. In CRYPTO'19, Gohr introduced a novel cryptanalysis method by developing a differential-neural distinguisher using neural networks as the underlying distinguisher. He effectively integrated this distinguisher with classical differentials, facilitating a 12-round key recovery attack on Speck32/64 (from a total of 22 rounds). Bao *et al.* refined the concept of neutral bits, enabling key recovery attacks up to 13 rounds for Speck32/64 and 16 rounds (from a total of 32) for Simon32/64.
Our primary objective is to enhance the capabilities of differential-neural distinguishers by applying more deep-learning techniques, focusing on handling more rounds and improving accuracy. Inspired by the Inception Block in GoogLeNet, we adopted a design that uses multiple parallel convolutional layers with varying kernel sizes before the residual block to capture multi-dimensional information. Additionally, we expanded the convolutional kernels in the residual blocks, thereby enlarging the network's receptive field. In the case of Speck32/64, our efforts yield accuracy improvements in rounds 6, 7, and 8, enabling the successful training of a 9-round differential-neural distinguisher. As for Simon32/64, we developed a differential-neural distinguisher capable of effectively handling 12 rounds while achieving noteworthy accuracy enhancements in rounds 9, 10, and 11.
Additionally, we utilized neutral bits to ensure the required data distribution for launching a successful key recovery attack when using multiple-ciphertext pairs as input for the neural network. Meanwhile, we redefined the formula for time complexity based on the differences in prediction speeds of the distinguisher between a single-core CPU and a GPU. Combining these various advancements allows us to considerably reduce the time and data complexity of key recovery attacks on 13-round Speck32/64. Furthermore, we used knowledge distillation techniques to reduce the model size, thereby accelerating the distinguisher's prediction speed and reducing the time complexity. In particular, we achieved a successful 14-round key recovery attack by exhaustively guessing a 1-round subkey, marking a significant milestone in differential-neural cryptanalysis. For Simon32/64, we accomplished a groundbreaking 17-round key recovery attack for the first time and reduced the time complexity of the 16-round key recovery attack.

Keywords: Differential-Neural Distinguisher · Inception Module · Speck · Simon · Knowledge Distillation · Key Recovery Attack

1 Introduction

In CRYPTO 2019, Gohr [Goh19] proposed the idea of differential-neural cryptanalysis. The differential-neural distinguisher, trained by the 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. However, the current differential-neural distinguisher seems only effective for limited rounds of ciphertext. Therefore, a short high-probability classical differential $\Delta S \rightarrow \Delta P$ is prepended before the differential-neural distinguisher to increase the number of rounds for key recovery attacks.

Gohr [Goh19] showed that the Residual Network [HZRS16] could capture the non-randomness of the distribution of output pairs when the input pairs of round-reduced Speck32/64 meet a specific difference. As a result, 6, 7, and 8-round differential-neural distinguishers were trained, and 11, 12-round key recovery attacks for Speck32/64 were achieved by combining a 2-round classical differential. There may be two directions to improve differential-neural cryptanalysis. One is to use a longer classical differential prepended on top of the differential-neural distinguisher. Bao *et al.* [BGL$^+$22] generalized the concept of neutral bits and searched for (conditional) simultaneous neutral bit-set with a higher probability for more rounds of the classical differential. Thus, Bao *et al.* devised a new 13-round key recovery attack for Speck32/64 with the same differential-neural distinguisher proposed in [Goh19]. The other is to study the effective differential-neural distinguisher with more rounds. Chen *et al.* [CSYY21] and Benamira [BGPT21] *et al.* almost simultaneously proposed the method of using multiple-ciphertext pairs instead of single-ciphertext pairs (in Gohr's work) as input of the neural network, both improved the accuracy of the 6, 7-round differential-neural distinguisher of Speck32/64. Bao *et al.* [BGL$^+$22] used the Dense Network [HLvdMW17] and the Squeeze-and-Excitation Network [HSS18] to train differential-neural distinguisher, obtained 9, 10, and 11-round differential-neural distinguisher and devised a 16-round key recovery attack for Simon32/64. We made some improvements for differential-neural cryptanalysis, as listed below.

Our contribution. The contributions of this work include the following:

- We have enhanced the differential-neural distinguisher by revising its network architecture. This improvement entailed introducing an Inception module comprising multiple parallel convolutional layers before the Residual block. The Inception module's integration aims to capture a broader spectrum of information across various dimensions within the ciphertext pairs. Furthermore, we fine-tuned the convolutional kernel sizes to better match the cipher's round functions. These developments have resulted in increased accuracy for the 6, 7, and 8-round differential-neural distinguishers of Speck32/64, and have allowed us to develop a new 9-round differential-neural distinguisher. In a similar vein, for Simon32/64, we have enhanced the accuracy of the 9, 10, and 11-round distinguishers and successfully created a novel 12-round distinguisher. The performance results of the differential-neural distinguishers for Speck32/64 and Simon32/64 are comprehensively presented in Tables 2 and 6, respectively.

- To successfully conduct key recovery attacks, each ciphertext pair obtained through classical differentials must exhibit the same difference. Inspired by Gohr's work on the combined response of differential-neural distinguishers, we employed neutral bits with a probability of 1 to generate multiple-plaintext pairs. These plaintext pairs were then encrypted to produce multiple ciphertext pairs. Furthermore, to estimate the time complexity of differential-neural cryptanalysis, we calculated the ratio of prediction speeds between a single-core CPU and a GPU. This approach allows for a more equitable and reasonable comparison of the results of differential-neural cryptanalysis with those of classical cryptanalysis.

- We have successfully executed several key recovery attacks by combining the enhanced differential-neural distinguisher with a classical differential. This methodology has led to a reduction in both time and data complexity for the 13-round Speck32/64 key recovery attack. Additionally, we implemented knowledge distillation techniques to reduce the size of the model, which in turn accelerated the prediction speed of the distinguisher during key recovery attacks, further lowering the time complexity. Notably, we achieved a breakthrough by successfully conducting a 14-round key recovery attack on Speck32/64, marking the first instance of such an attack in differential-neural cryptanalysis. This was accomplished by exhaustively guessing a 1-round subkey. For Simon32/64, we pioneered a 17-round key recovery attack using deep learning methods for the first time. Moreover, we managed to reduce the time complexity of the 16-round key recovery attack on Simon32/64. Comprehensive experimental comparisons demonstrating these achievements are detailed in Table 1. Source codes are available in https://github.com/CryptAnalystDesigner/NeuralDistingsuisherWithInception.git.

Table 1: Summary of key recovery attacks on Speck32/64 and Simon32/64

Target	Round	Type	Configure	Time Complexity	Data Comoplexity	Success Rate	Key Space	Reference
Speck 32/64	13	$\mathcal{DD}$	1+8+4	2^{57}	2^{25}	–	2^{64}	[Din14]
		$\mathcal{DD}$	1+8+2+2	$2^{50.16}$	$2^{31.13}$	63%	2^{64}	[BCdST+23]
		$\mathcal{DD}$	2+8+3	$2^{55.58}$	$2^{24.26}$	–	2^{64}	[FLW+23]
		$\mathcal{ND}$	1+3+8+1	$2^{48.67+2.4\star}$	2^{29}	82%	2^{63}	[BGL+22]
		$\mathcal{ND}$	1+3+8+1	$2^{41.44+5.68\dagger}$	2^{27}	21%	2^{63}	Sect.4.4
		$\mathcal{ND}$	1+3+8+1	$2^{41.22+3.81\dagger}$	2^{27}	34%	2^{63}	Sect.5.2
	14	$\mathcal{DD}$	1+9+4	$2^{62.47}$	$2^{30.47}$	–	2^{64}	[SHY16]
		$\mathcal{DD}$	1+9+2+2	$2^{60.99}$	$2^{31.75}$	63%	2^{64}	[BCdST+23]
		$\mathcal{DD}$	2+9+3	$2^{60.58}$	$2^{30.26}$	76%	2^{64}	[FLW+23]
		$\mathcal{ND}$	1+3+8+1+1	$2^{57.12+3.81\dagger}$	2^{27}	34%	2^{63}	Sect.5.3
Simon 32/64	16	$\mathcal{DD}$	2+12+2	$2^{26.48}$	$2^{29.48}$	62%	2^{64}	[AL13]
		$\mathcal{ND}$	1+3+11+1	$2^{41.81+2.4\star}$	2^{21}	49%	2^{64}	[BGL+22]
		$\mathcal{ND}$	1+3+11+1	$2^{31.71+3.20\dagger}$	2^{21}	59%	2^{64}	Sect.6.5
	17	$\mathcal{ND}$	1+4+11+1	$2^{40.64+5.22\dagger}$	2^{28}	9%	2^{64}	Appendix.B
	18	$\mathcal{DD}$	1+13+4	$2^{46.00}$	$2^{31.2}$	63%	2^{64}	[ALLW14]
	19	$\mathcal{DD}$	2+13+4	$2^{34.00}$	$2^{31.5}$	-	2^{64}	[BRV14]
	21	$\mathcal{DD}$	4+13+4	$2^{55.25}$	$2^{31.0}$	-	2^{64}	[WWJZ18]

1. $\mathcal{DD}$: differential distinguisher; $\mathcal{ND}$: differential-neural distinguisher; $-$: Not available;
2. *: 2.4 is the ratio of the time required for key recovery attacks using CPU and GPU in [BGL+22];
3. †: 5,68, 3.81, 3.20 and 5.22 are the ratio of the time required for different key recovery attacks using single-core CPU and GPU in our device.

Organization. Section 2 gives the preliminary on the distinguisher model, the key recovery attack process, and generalized neutral bits. Section 3 introduces the training method and the result of distinguishers for Speck32/64. We introduce data generation, complexity calculation and key recovery attack for 13-round Speck32/64 in Section 4. Section 5 presents the knowledge distillation and key recovery attack for 13, 14-round Speck32/64. Section 6 introduces the differential-neural cryptanalysis for Simon32/64. We conclude the paper in Section 7.

2 Preliminary

2.1 Brief Description of Speck32/64 and Simon32/64

Let ω be the word size (the number of bits of a word), and the block size can be denoted as L bits, where $L = 2\omega$. Let (x_i, y_i) be the left and right branches of a state after encryption of i rounds, k_i the subkey of i rounds. Denote the bitwise XOR by $\oplus$, the addition modulo 2^ω by $\boxplus$, the bitwise AND by $\cdot$, the bitwise right rotation by $\ggg$, and the bitwise left rotation by $\lll$.

Speck32/64 and Simon32/64 are members in the lightweight block cipher family Speck and Simon [BSS+13]. The round function (out of 22) of Speck32/64 takes a 16-bit subkey k_i and a 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:

$$x_{i+1} = ((x_i \ggg 7) \boxplus y_i) \oplus k_i, y_{i+1} = (y_i \lll 2) \oplus x_{i+1}.$$

The round function (out of 32) of Simon32/64 takes a 16-bit subkey k_i and a 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:

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

2.2 The Model of Differential-Neural Distinguisher for Speck32/64

The differential-neural distinguisher is a supervised model that distinguishes whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. Given m plaintext pairs $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and target cipher Speck32/64, the resulting ciphertext pairs $\{(C_{i,0}, C_{i,1}), i \in [0, m-1]\}$ are regarded as an instance. Note that $m = 1$ in [Goh19], $m \in \{1, 5, 10, 50, 100\}$ in [BGPT21], and $m \in \{2, 4, 8, 16\}$ in [CSYY21]. Each instance will be attached with a label Y:

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

where $\Delta =$ (0x0040, 0x0000). If Y is 1, this instance is sampled from the target distribution and defined as a positive example. Otherwise, this instance is sampled from a uniform distribution and defined as a negative example. A large number of instances need to be trained in neural networks. when the neural network can obtain a stable accuracy higher than 0.5 on a test set, it can effectively distinguish whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. The model of differential-neural distinguisher can be described as:

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

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

2.3 Differential-neural Cryptanalysis

Gohr [Goh19] proposed a framework for differential-neural cryptanalysis dedicated to recovering the last two rounds of subkeys for Speck. We decrypt the ciphertext using guessed subkey and use the differential-neural distinguisher to estimate the distance between the guessed subkey and the real key.

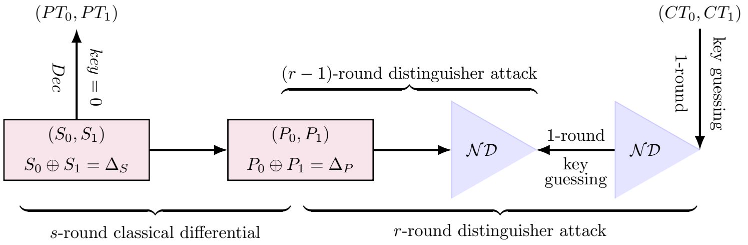

(PT_0, PT_1)
Dec
$key = 0$
(CT_0, CT_1)
1-round
key guessing
$(r-1)$-round distinguisher attack
(S_0, S_1)
$S_0 \oplus S_1 = \Delta_S$
(P_0, P_1)
$P_0 \oplus P_1 = \Delta_P$
$\mathcal{ND}$
1-round
key guessing
$\mathcal{ND}$
s-round classical differential
r-round distinguisher attack

Figure 1: $(1 + s + r + 1)$-round key recovery attack of differential-neural cryptanalysis

The overall processing of a key recovery attack based on the differential-neural distinguisher is shown in Fig. 1, where $\mathcal{ND}$ is the trained differential-neural distinguisher, (PT_0, PT_1) is plaintext pairs and (CT_0, CT_1) is ciphertext pairs. The $(1+s+r+1)$-round key recovery attack employs a r-round main and $(r-1)$-round helper differential-neural distinguisher trained using input pairs with difference Δ_P. A short s-round classical differential $(\Delta_S \rightarrow \Delta_P)$ with probability denoted by 2^{-p} is prepended on top of the differential-neural distinguisher to increase the number of the rounds of key recovery attack. To ensure the existence of data pairs satisfying the difference Δ_P after s-round encryption, about $c \cdot 2^p$ (denoted by n_{cts}) data pairs with the difference Δ_S are required according to the probability of difference propagation, where c is a small constant. Neutral bits of the s-round classical differential are used to expand each data pair to a structure of n_b data pairs. The n_{cts} structures of the data pairs are decrypted in one round with 0 as the subkey to get the plaintext structures because the nonlinear operation occurs before the addition of keys for Speck and Simon. All plaintext structures are encrypted to obtain the corresponding ciphertext structures. Each ciphertext structure is used to select a candidate of the subkey by the r-round main differential-neural distinguisher 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*. Each ciphertext structure is assigned 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 differential-neural distinguisher upon wrong-key decryption. The *wrong key response profile* is to recommend new candidate values from previous candidate values while minimizing the weighted Euclidean distance in a BayesianKeySearch Algorithm [Goh19].

2.4 Combined Response and Neutral Bits

As the number of encryption rounds increases, the accuracy of the differential-neural distinguisher decreases. To reduce the impact of the misjudgment of the single prediction of the distinguisher, Gohr used the combined response of the differential-neural distinguisher in ciphertext structure with the same distribution, which can be satisfied by neutral bits [Goh19]. The primary notion of neutral bits can be interpreted as follows.

Definition 1 (Neutral bits of a differential, NB[BC04])**.** Let $\Delta_{in} \rightarrow \Delta_{out}$ be a differential with input difference Δ_{in} and output difference Δ_{out} of a 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 differential-neural distinguisher on ciphertext pairs in the

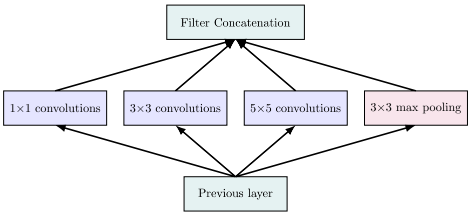


Figure 2: Inception Module

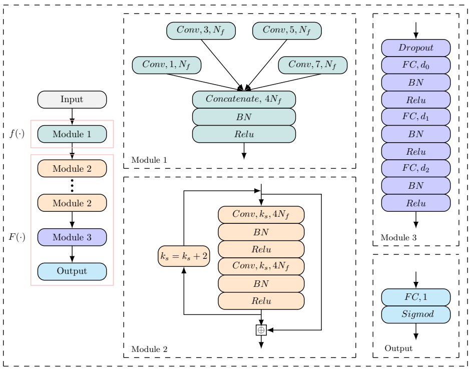


Figure 3: The network architecture of our distinguisher for Speck32/64

ciphertext structure (of size n_b) 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 s_k is used as the score of a recommended subkey. The score s_k plays a decisive role in the execution time and success rate of the attack. The number of instances with the same distribution should be sufficiently large to enhance the distinguishing ability of the low-accuracy differential-neural distinguisher. However, neutral bits of the nontrivial classical differential are scarce. Therefore, probabilistic neutral bits (PNB) are exploited in [Goh19]. Some probabilistic neutral bits, simultaneous-neutral bit-sets (SNBS), conditional (simultaneous-) neutral bit(-set)s (CSNBS), and switching bits for adjoining differentials (SBfADs) were found in [BGL+22] (refer to Appendix A).

3 Differential-Neural Distinguishers for Round-Reduced Speck32/64

3.1 Inception Module

Generally speaking, the safest way to improve network performance is to increase the width and depth of the network, which also has side effects. First, a deeper and wider network often means many parameters. When the amount of data is small, the trained network is easy to overfit, and when the depth of the network is deep, it is difficult to train and easy to cause greater errors. The two side effects of disappearance restrict the development of deep and wide convolutional neural networks, and the Inception network solves these two problems very well.

One of the modules in the Inception module is as follows in Fig. 2: in the same layer, there are 1×1, 3×3, 5×5 convolution and pooling layers, respectively, and the convolution operation and the pooling layer are pooled using filters. Padding is used in all operations to ensure that the output is the same size and the output results are all integrated after these operations. The feature of this module is that in the same layer, different features of the input of the previous layer are collected by using the above-mentioned kernels of different sizes and performing pooling operations. This increases the width of the network and uses these different-sized kernels and pooling operations to extract different features from the previous layer.

3.2 Network Architecture

The overall structure of our neural network for training the differential-neural distinguisher is depicted in Fig 3. Our neural network comprises four main components: an input layer that incorporates multiple-ciphertext pairs, an initial convolutional layer consisting of four parallel convolutional layers, a residual tower that consists of multiple convolutional neural networks with two layers each, and a prediction head that comprises multiple fully connected layers.

Input Representation. When the output of the r-th round is denoted as $(C, C') = (x_r \parallel y_r, x'_r \parallel y'_r)$, it enables the direct computation of (y_{r-1}, y'_{r-1}) without the knowledge of the $(r-1)$-th subkey, utilizing the round function of Speck. Consequently, the neural network can process data in the format of $(x_r, x'_r, y_r, y'_r, y_{r-1}, y'_{r-1})$. To accommodate this data format, the input layer of the neural network comprises m ciphertext pairs, where each pair consists of $3L$ units arranged in a $[m, \omega, \frac{3L}{\omega}]$ array. For Speck32/64, the values of L and ω are set as 32 and 16 respectively.

Initial Convolution (Module 1). The initial convolutional layer, linked to the input layer, comprises four distinct convolutional layers. Each layer utilizes $N_f = 32$ channels but with varying kernel sizes. This design is inspired by the Inception module from GoogLeNet [SLJ+15], where the outputs of these convolutional layers are concatenated along the channel dimensions to integrate different feature maps. Batch normalization is then employed on these concatenated outputs to stabilize the learning process. Subsequently, a rectifier nonlinearity is applied to the normalized outputs. The resulting matrix, with dimensions $[m, \omega, 4N_f]$, is forwarded to the subsequent convolutional blocks layer, facilitating further feature extraction.

Residual Blocks (Module 2). Each residual block in our network comprises two layers, each equipped with $4N_f$ filters. The operational sequence within each block initiates with a residual layer using a kernel size of $k_s = 3$. This is followed by batch normalization, enhancing the stability and efficiency of the training process. Subsequently, a rectifier layer is applied, introducing nonlinearity into the model. A critical design element is incorporating a skip connection at the end of each block, which links the output from the final rectifier layer directly to the input of the block. This skip connection facilitates the transfer of information to the subsequent block. Furthermore, the kernel size incrementally increases by 2 following each residual block, allowing for a progressive receptive field expansion. The architecture comprises five residual blocks, a configuration empirically determined through experimental validation.

Prediction Head (Module 3 and Output). The prediction head features three hidden layers, leading to a single output unit. A dropout layer is added before the first hidden layer to reduce overfitting. The hidden layers consist of $d_0 = 512$, $d_1 = 64$, and $d_2 = 64$ units, each followed by batch normalization and a rectifier layer. The output unit employs a sigmoid activation function.

Rationale. The performance of a differential-neural distinguisher primarily depends on two aspects: data and network structure. On the one hand, we have modified the input data of the neural network according to the round function of Speck, as more data may contain more information. On the other hand, unlike Gohr, who only used convolutional layers with a kernel size of 1 to capture differential information, we think that due to the cyclic shift operations and modular addition in the round functions, adjacent bits of the ciphertext might also exhibit features. Therefore, we use the Inception module to attempt to capture features across multiple dimensions. All in all, using the Inception module and increasing the size of the convolutional kernel in the residual block are also aimed at enlarging the receptive field of the network.

3.3 The Training of Differential-Neural Distinguisher

A comprehensive training procedure was undertaken to substantiate the effectiveness of our proposed network architecture.

Data Generation. Training and test datasets were generated utilizing the Linux random number generator. This approach ensured the uniform distribution of keys K_i and the generation of multiple-plaintext pairs $\{(P_{i,j,0}, P_{i,j,1})\}, j \in [0, m-1]$ with the designated input difference $\Delta = $ (0x0040, 0x0000) and an associated vector of binary labels Y_i. For the r-round Speck32/64, the encryption process varied based on the label Y_i. If $Y_i = 1$, the multiple-plaintext pairs underwent encryption for r rounds. Conversely, if $Y_i = 0$, the second plaintext in each pair was substituted with a newly generated random plaintext before undergoing the same r-round encryption process.

Remark 1. *We use two different numbers of datasets to train differential-neural distinguisher.* In Gohr's study [Goh19], both the training and test sets consist of N and M instances, respectively, each containing a single ciphertext pair, thus resulting in a total of N and M ciphertext pairs. Contrastingly, in the work of Chen and Yu [CSYY21], the training and test sets include ${}^{N}/_{m}$ and ${}^{M}/_{m}$ instances, respectively, with each instance comprising m ciphertext pairs. This approach leads to the same total number of ciphertext pairs, N and M, being used in both sets. For a fair comparison, we adopted the latter approach of using ${}^{N}/_{m}$ and ${}^{M}/_{m}$ instances for our training and test sets. However, this methodology potentially risks overfitting as discussed in *Remark 2.* To address this issue, we also considered using N and M instances for the training and test sets, each containing m ciphertext pairs, thereby involving a total of $N \times m$ and $M \times m$ ciphertext pairs.

Basic Training Method. We conducted the training for 20 epochs in the dataset for $N = 10^7$ and $M = 10^6$. The batch size processed by the dataset is adjusted according to the parameter m to maximize GPU performance. 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 [KB15]. A cyclic learning rate schedule was applied, setting the learning rate l_i for epoch i to $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n} \cdot (\beta - \alpha)$ 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. A multi-stage pretraining approach was employed to train an 8-round differential-neural distinguisher for Speck32/64. Initially, our 7-round distinguisher was used to identify 5-round Speck32/64 with the input difference (0x8000, 0x804a), which is the most probable difference to manifest three rounds after the initial difference (0x0040, 0x0000). This training involved 10^7 instances over 20 epochs, using a learning rate of 10^{-4}. Subsequently, the distinguisher was trained to recognize 8-round Speck32/64 with the input difference (0x0040, 0x0000), processing 10^7 new instances for 10 epochs at the same learning rate. The training concluded with the learning rate being reduced to 10^{-5}, and the model processed another 10^7 instances for 10 epochs. The training methodology for the 9-round distinguisher mirrored this approach. The primary variation was the use of an 8-round distinguisher for identifying 5-round Speck32/64 with the input difference (0x850a, 0x9520), which is the most probable difference after four rounds starting from (0x0040, 0x0000).

3.4 Result

Test Accuracy. We present a comparative analysis of the accuracy of 6, 7, 8, and 9-round differential-neural distinguishers against the benchmarks set in previous studies [Goh19, CSYY21, BGPT21], as summarized in Table 2. Additionally, we report the accuracy (Acc), true positive rate (TPR), and true negative rate (TNR) of these distinguishers, as evaluated on newly generated datasets, in Table 3. The training methodology for the 6 and 7-round distinguishers adhered to the basic training method, whereas the 8 and 9-round distinguishers were developed using the staged training method. Notably, variations in accuracy under these two experimental conditions can be attributed to model overfitting, as detailed in *Remark 2*.

Table 2: Summary accuracy of distinguisher on Speck32/64 using different number of instances

r	m=2				m=4				m=5		
	[CSYY21]	[GLN22]*	$^N/_m$	N	[CSYY21]	[GLN22]*	$^N/_m$	N	[BGPT21]	$^N/_m$	N
6	0.8613	0.877	0.8771	0.8773	0.931	0.950	0.9468	0.9497	0.9541	0.9623	0.965
7	0.6393	0.663	0.6663	0.6649	0.6861	0.725	0.7194	0.7283	0.735	0.7436	0.7491
8	-	0.521	-	-	-	0.530	0.5329	0.5428	-	-	-

r	m=8				m=10			m=16			
	[CSYY21]	[GLN22]*	$^N/_m$	N	[BGPT21]	$^N/_m$	N	[CSYY21]	[GLN22]*	$^N/_m$	N
6	0.9562	0.990	0.9868	0.9895	0.99	0.9915	0.9939	0.9802	1.000	0.9969	0.9992
7	0.7074	0.801	0.7991	0.8106	0.808	0.8243	0.8341	0.6694	0.883	0.8759	0.8963
8	-	0.543	0.5347	0.5590	-		-	-	0.560	0.5566	0.5854
9		-	-	0.5024	-		-	-	-	-	0.5050

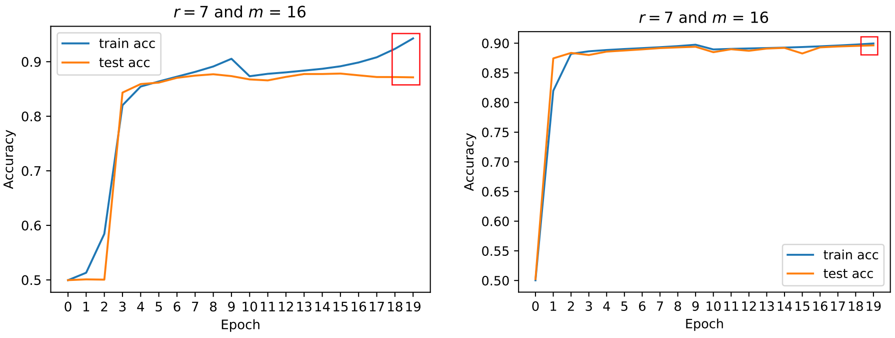
*: combining the scores of multiple distinguishers trained by using single-ciphertext pair under independence assumption.
The "$^N/_m$" column results in accuracy using $^N/_m$ and $^M/_m$ instances as training and test sets, respectively. The "N" column results from accuracy using N and M instances as the training and test sets, respectively.

Table 3: Acc, TPR, TNR of distinguisher on Speck32/64 using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	6	0.8768	0.8448	0.9087	4	6	0.9495	0.9429	0.9559
	7	0.6635	0.6240	0.7029		7	0.7291	0.7048	0.7532
	8	-	-	-		8	0.5428	0.5318	0.5537
8	6	0.9897	0.9862	0.9931	16	6	0.9992	0.9988	0.9996
	7	0.8103	0.8024	0.8184		7	0.8958	0.8906	0.9009
	8	0.5562	0.5434	0.5690		8	0.5853	0.5704	0.6002
	9	0.5016	0.2122	0.7915		9	0.5045	0.5175	0.4915

In the study by Benamira et al. [BGPT21], the value of m varies as $\{1, 5, 10, 50, 100\}$. However, for the ease of key recovery attack implementations, we chose to set m to powers of 2, specifically $\{2, 4, 8, 16\}$. It is noteworthy that when $m = 1$, the approach of Benamira et al. aligns with that of Gohr. Benamira et al. employed two distinct methodologies to train and evaluate the differential-neural distinguisher with multiple-ciphertext pairs. The first method, known as the Averaging Method, entails evaluating the neural distinguisher score for each element of a ciphertext pair, followed by calculating the median of these scores. This method is essentially the combined score method proposed by Gohr [GLN22]. The second, known as the 2D-CNN Method, treats the entire set of multiple-ciphertext pairs as a singular input for the neural network. While these methods are theoretically equivalent, the latter approach yields lower accuracy for the trained differential-neural distinguisher. For instance, in the case of the 6-round distinguisher, accuracies of 0.9541 and 0.9327 were reported for the Averaging and 2D-CNN Methods, respectively, when $m = 5$; and 0.99 and 0.977 when $m = 10$, as per Benamira et al.'s findings. This trend indicates a consistently lower accuracy for the distinguisher trained using the 2D-CNN Method. However, the accuracy of the 7-round distinguisher, as reported by Benamira et al., was obtained using the Averaging Method, suggesting that direct neural network training might result in even lower accuracy.

Remark 2. *Why do we train a differential-neural distinguisher with two different numbers of data?* To ensure a fair comparison, we utilized $^N/_m$ and $^M/_m$ instances for our training and test sets, as discussed in *Remark 1*. However, as indicated by Fig. 4a, a notable discrepancy exists between the training and test accuracies, suggesting a tendency towards overfitting when using N/m and M/m instances, particularly for larger values of r and m. In contrast, Fig. 4b shows nearly equal training and test accuracies, indicating that using N and M instances for training and testing can mitigate overfitting, expedite model convergence, and enhance accuracy. The observed overfitting not only diminishes the accuracy of the distinguisher but may also impede the training of distinguishers for more rounds.



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

Figure 4: Overfitting with the different number of instances.

Wrong Key Response Profile. We analyze scenarios where the distinguisher's response to incorrect key decryption depends on the bitwise difference between the guessed and real subkey. Utilizing the *wrong key response profile*, as introduced in [Goh19], we precompute this profile using 3000 ciphertext pairs per subkey guess. This precomputation informs the BayesianKeySearch Algorithm, which iteratively refines subkey candidates by minimizing their weighted Euclidean distance, thereby recommending probable subkeys without exhaustive decryption. The empirical mean μ and standard deviation δ of the distinguisher score are critical for the algorithm's decision-making, enhancing the efficiency of the key search process.

In Fig. 5, the abscissa represents the difference between the guessed and real subkey, while the ordinate reflects the score from the differential-neural distinguisher using ciphertext pairs decrypted with the guessed subkey. A smaller Hamming weight of the key difference correlates with higher distinguisher scores and vice versa. The distinguisher's score is notably higher when the difference between the guessed and real subkey is among $\{16384, 32768, 49152\}$, as indicated by the '$\star$' markers on the vertical axis in the figures. This observation suggests that incorrect guesses of the 14th and 15th bits of the subkey minimally impact the distinguisher's score. Consequently, excluding these two bits from the key guessing process can effectively reduce the key space, a strategy also employed in Gohr's study [Goh19] to expedite key recovery attacks.

4 From Differential-Neural Distinguisher to Key Recovery Attack

4.1 Generation of Same Distributed Data

During the training of the differential-neural distinguisher, we treat multiple-ciphertext pairs as a single instance for input into the neural network. For positive examples, these

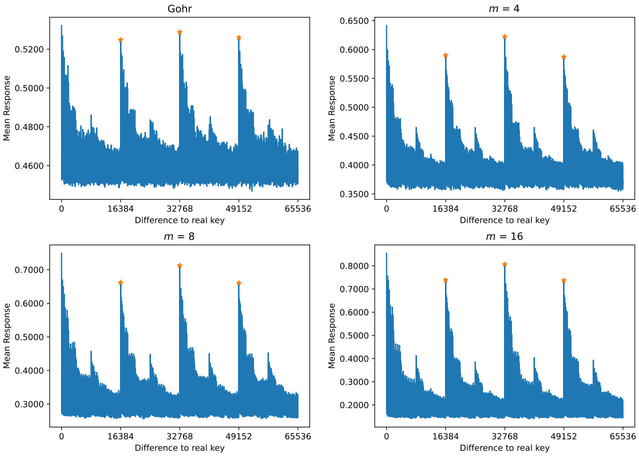


Figure 5: Wrong key response profile for 7-round Speck32/64 using N instances

ciphertext pairs are derived from encrypting multiple-plaintext pairs that share the same difference Δ_P.

After developing the differential-neural distinguisher, we prepend a classical differential $\Delta_S \to \Delta_P$ to extend the attack's reach across more rounds. However, since classical differentials are probabilistic, ensuring the same difference in multiple-ciphertext pairs post-classical differential propagation is challenging.

To address this, we utilize neutral bits, which facilitate obtaining multiple-ciphertext pairs with a consistent difference after classical differential propagation, which is crucial for effective key recovery attacks. We start with randomly generated plaintext pairs with the initial difference Δ_S and manipulate the plaintext bits corresponding to $\log_2 m$ neutral bits to create m plaintext pairs. This approach ensures that the m ciphertext pairs produced by processing these plaintext pairs through the classical differential share the same difference.

4.2 Experimental Environment and Data Complexity

Our approach follows the framework of previous studies [Goh19, BGL⁺22], with the primary distinction being our use of a differential-neural distinguisher trained on N instances. For detailed key recovery attack procedures, please refer to Appendix C. Consistent with the criteria in [Goh19], a key guess is considered successful if it accurately predicts the last round key and the second round key is within a Hamming distance of two from the real subkey. The experiments were conducted using Python 3.7.15 and Tensorflow 2.5.0 on Ubuntu 20.04. Our hardware setup included Intel(R) Xeon(R) Gold 6226R processors (2.90GHz), 256GB RAM, and five NVIDIA RTX2080Ti 12GB GPUs. The specific parameters used in the key recovery attacks are outlined below.

Parameter	Definition
n_{kg}	the number of times to guess the subkey k_r involved in the condition.
n_{cts}	The number of ciphertext structures.
n_b	The number of ciphertext pairs in each ciphertext structure, i.e., $2^{\lvert \mathrm{NB} \rvert}$.
n_{it}	The total number of iterations in the ciphertext structures.
c_1, c_2	The cutoffs with respect to the scores of the recommended last subkey and second to last subkey, respectively.
$n_{byit1/2}$	The number of iterations, the default value is 5.
$n_{cand1/2}$	The number of key candidates within each iteration, default value is 32.

Theoretical Data Complexity. During key recovery attacks, the classic differentials or neutral bits may require specific conditions, thereby dictating particular data needs. As a result, the data complexity formula needs to include the factor n_{kg}. The data complexity D_c of our experiment is calculated with the formula $n_{kg} \times n_b \times n_{ct} \times m \times 2$. This complexity is a theoretical estimation. When the differential-neural distinguisher achieves high accuracy, key recovery can be expedited, often using less data. Thus, the actual data complexity in the experiment tends to be lower than the theoretical estimation.

4.3 Experimental Time Complexity

It has always been challenging to accurately calculate the time complexity and conduct a fair comparison with previous work.

- In classical differential cryptanalysis, researchers often quantify the time complexity of a key recovery attack in terms of the number of encryptions (the number of attack rounds) required [Din14, ALLW14, BRV14, FLW+23, BCdST+23].

- In differential-neural cryptanalysis, Gohr [Goh19] utilized a GPU for training the differential-neural distinguisher but implemented key recovery attacks on a single-core CPU. He estimated that an optimized, SIMD-parallelized implementation of Speck32/64 could achieve brute force key search at roughly 2^{28} keys per second per core. Subsequently, Bao et al. [BGL+22] employed a GPU to expedite key recovery attacks for 13-round Speck32/64. For a fair comparison with earlier studies, they adopted 2^{28} keys per second as the benchmark for encryption speed and calculated the ratio ($Ratio_{cpu/gpu}$) of the time required for key recovery attacks on CPU versus GPU, where $Ratio_{cpu/gpu} = 2^{2.4}$ in their work.

To ensure a fair comparison, we conduct multiple key recovery attacks using our differential-neural distinguisher and calculate the average running time (rt) for each experiment. This rt serves as the representative time for the experiment. Additionally, we evaluate the success rate (sr) of the key recovery attack by calculating the ratio of successful experiments to the total number conducted. Moreover, classical differential cryptanalysis typically utilize a single-core CPU for key recovery attack execution. However, part of the differential-neural cryptanalysis key recovery attack process requires GPU execution. To achieve a fairer comparison, we performed several experiments to determine the time ratio required for key recovery attacks on a single-core CPU versus a GPU on our equipment. The detailed results are provided in Table 4.

Table 4: Calculate the ratio of time required for key recovery attack using CPU and GPU

R	Conf.	m	c_1	c_2	n_{cts}	n_{it}	n_b	N_e	Device	rt	$Ratio_{cpu/gpu}$	sr
12	1+3+7+1	8	7	10	2^{11}	2^{12}	2^6	40	CPU	26729.89	$2^{5.565}$	100%
									GPU	564.67		100%

In Table 4, when performing a key recovery attack on 12-round Speck32/64 using our trained distinguisher, we determined the ratio (Ratio$_{cpu/gpu}$) of the time required on a single-core CPU versus a GPU to be $2^{5.56}$. Additionally, we utilized the Python package *Cprofile* to record the time taken for neural network prediction during the key recovery attack on a single-core CPU. Our observations revealed that the neural network's scoring process for ciphertexts accounted for 99.77% of the total attack time. This significant time disparity between CPU and GPU in key recovery attacks can be attributed to the CPU's relative inefficiency in handling the neural network's extensive matrix computations.

Executing key recovery attacks on a single-core CPU is notably time-consuming. However, utilizing GPU acceleration can significantly speed up the process. Therefore, we propose a novel formula for calculating time complexity, which estimates the complexity of key recovery attacks on CPU devices based on the time required to perform these attacks on GPU devices.

- **Prediction Speed P_s of Distinguisher on Single-core CPU and GPU**: When utilizing a GPU for key recovery attacks, optimizing its performance is crucial, as the GPU may not be fully utilized. Setting GPU memory usage thresholds and adjusting batch sizes are essential to control GPU utilization during distinguisher prediction, thereby ensuring consistency in key recovery attacks. For instance, in a 12-round key recovery attack, the GPU memory utilization was 3379MB with around 67% GPU utilization. Therefore, controlling these parameters is vital when evaluating the GPU's predictive speed to ensure fairness. In contrast, predictions on a single-core CPU typically reach 100% CPU utilization. Predictions on 10^7 instances were conducted using both a GPU and a single-core CPU, and the required time was recorded. The GPU completed the task in 729.749 seconds, while the single-core CPU took 33761 seconds. Thus, the prediction speed ratio P_s between the single-core CPU and GPU is $2^{5.53}$. Given that P_s closely aligns with the $Ratio_{cpu/gpu}$ from Table 4, we use P_s instead of $Ratio_{cpu/gpu}$ to calculate the time complexity.

- **Encryption Speed E_s of Cryptographic Algorithms on Single-core CPU**: We utilize a 2^{32} plaintext to evaluate our device's encryption speed E_s. For Speck32/64, each core can execute approximately $2^{27.09}$ 1-round encryption per second per core, i.e., $E_s = 2^{27.09}$; For Simon32/64, each core can execute approximately $2^{25.48}$ 1-round encryption per second per core, i.e., $E_s = 2^{25.48}$.

- **Time Comlexity**: Then the calculation formula of time complexity T_c is $n_{kg} \times rt \times \frac{E_s}{R} \times P_s$, where R is the number of rounds for key recovery attack.

4.4 Using Two Classical Differentials and SBfADs for 13-round Speck32/64

Bao et al. observed that the output of the classical differential plays a significant role in the performance of differential-neural distinguishers, whereas the input difference does not. This insight allows for the use of multiple classical differentials with the same output difference to be prepended to a differential-neural distinguisher. Such classical differentials can share some neutral bits, potentially enabling data reuse and slightly reducing data complexity. Additionally, Bao et al. employed the SBfADs, which save one condition, thereby halving the time and data complexity compared to the CSNBS. Leveraging these insights, we implemented a 13-round key recovery attack using two classical differentials and SBfADs to harness the potential of our differential-neural distinguisher further.

Experiment 1: The components of key recovery attack $\mathcal{A}_I^{\textsc{Speck}13R}$ of 13-round Speck32/64 are shown as follows.

1. 3-round classical differentials (0x8020, 0x4101) → (0x0040, 0x0000) and (0x8060, 0x4101) → (0x0040, 0x0000);

2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[20], [13], [12, 19]\}$; generalized neutral bits of combined response of differential-neural distinguisher: $\{[22], [14, 21], [6, 29], [30], [0, 8, 31], [5, 28], [15, 24], [4, 27, 29]\}$ (refer to Table 12 and one SBfADs [21]);

3. 8(7)-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Speck}_{8(7)r}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\textsc{Speck}_{8(7)r}} \cdot \mu$ and $\mathcal{ND}^{\textsc{Speck}_{8(7)r}} \cdot \delta$.

At the beginning of the 13-round Speck32/64 key recovery attack, we initially guess only two bits of k_0: $k_0[7]$ and $k_0[15] \oplus k_0[8]$, influenced by two 3-round classical differentials, along with two additional key bits $k_0[12] \oplus k_0[5]$, $k_0[1]$ for employing two CSNBS, as per Table 12. Due to the use of two classical differentials (0x8020, 0x4101) → (0x0040, 0x0000) and (0x8060, 0x4101) → (0x0040, 0x0000), the condition $k_0[5] \oplus k_0[14]$ is unnecessary, as indicated in Table 11. Additionally, by using SBfADs instead of CSNBS, the key bit $k_0[2] \oplus k_0[11]$ is not required to be guessed. It is important to note that although we used SNBS, we did not use the key condition $k_0[11] \oplus k_0[4]$ to increase its probability since 0.672 is already sufficiently high. Thus, $n_{kg} = 2^4$. However, for the economic feasibility of experimental verification, we tested the core of the attack, ensuring these four conditions were met. The specific parameters used in our 13-round key recovery attack $\mathcal{A}_I^{\text{Speck13}R}$ are listed below.

$n_{kg} = 2^4$	$m = 8$	$n_b = 2^{8+1}$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 8, c_2 = -500$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

For the 13-round key recovery attack $\mathcal{A}_I^{\text{Speck13}R}$, the GPU memory utilization was 9011MB, with GPU utilization approximately 91%. Predictions on 10^6 instances were conducted using both a GPU and a single-core CPU, and the required time was recorded. The GPU completed the task in 47.072 seconds, while the single-core CPU took 2413.396 seconds. Thus, the prediction speed ratio P_s between the single-core CPU and GPU is $2^{5.68}$. Also, the classical differential prepended to the attack is valid when keys satisfy $k_2[12] \neq k_2[11]$, so the attack was tested only for these valid keys, covering 2^{63} keys. The data complexity D_c is calculated as $2^4 \times 2^9 \times 2^{11} \times 8 \times 2/2 = 2^{27}$ plaintexts, halved due to the use of two classical differentials, which generate half of the required data pairs. In 100 trials, 21 were successful, yielding a success rate (sr) of 21%. The average runtime per trial on our server was 17011.22 seconds. Therefore, the time complexity T_c is given by $n_{kg} \times rt \times \frac{E_s}{R} \times P_s = 2^4 \times 17011.22 \times \frac{2^{27.09}}{13} \times 2^{5.68} = 2^{41.44+5.68}$.

5 Knowledge distillation for Differential-neural Cryptanalysis

As analyzed in Section 4.3, it is evident that the majority of the time spent on key recovery attacks on a single-core CPU is allocated to executing the distinguisher's prediction. At CRYPTO 2019, Gohr reported that the time ratio for executing key recovery attacks on a single-core CPU compared to a GPU was only 1.447. This relatively small speed ratio can be attributed to the small scale of Gohr's neural network, resulting in lower computational requirements. On the other hand, Gohr also implemented knowledge distillation, reducing the depth of the residual tower in the neural network from 10 to 1, thereby further minimizing the network size. Therefore, in this section, we also apply knowledge distillation to reduce the model's parameters and computational overhead, aiming to accelerate key recovery attacks.

5.1 Design and Train of Student Distinguisher

Simple Form of Knowledge Distillation Used by Gohr: Gohr [Goh19] utilized a depth-10 residual tower to train a differential-neural distinguisher as the teacher network. He then reduced the number of residual towers from 10 to 1 in the student network. A training dataset of 9 million instances was generated, with each instance predicted using the teacher distinguisher. These predictions were then used as training targets for the student network. The training spanned over 30 epochs, with the learning rate reduced from

0.001 to 0.0001 within 20 epochs. For the 7-round scenario, the student distinguisher's performance was comparable to that of the teacher distinguisher.

Data Generation of Student network: We adopted Gohr's method for data generation, randomly generating 10^7 instances for the training set and 10^6 instances for the test set. The distinguisher, trained as outlined in Section 3.3, was employed as the teacher distinguisher. It was used to make predictions on the dataset, with the outcomes of these predictions serving as the labels for the corresponding data points.

Network Architecture of Student Network: In our teacher network, we utilized four convolutional layers with various kernel sizes to capture features across multiple dimensions. Given the diverse receptive fields these kernel sizes offer, the kernel size in the student network was set to $k_I = 7$. To reduce the parameter count typically associated with fully connected layers, we implemented GlobalAveragePooling as a substitute. Our objective, as we scaled down the network's size, was to maintain as close a resemblance as possible between the student and teacher networks, illustrated in Fig 6.

Accuracy of Student Distinguisher Trained Using Knowledge Distillation: The training of the student network was conducted over 40 epochs, with the learning rate being reduced from 0.001 to 0.0001 within 20 epochs. As indicated in Table 5, the parameter of the student network is merely 1/283 of that of the teacher network. Despite this drastic parameter reduction, the number of convolutional operations within the student network remains significant. Therefore, using multiple-ciphertext pairs as inputs to the network yields only a 6.55-fold increase in GPU prediction speed for the distinguisher. While there is a noticeable decline in accuracy compared to the teacher distinguisher, the student distinguisher shows an improvement in the True Positive Rate (TPR). Employing the student network directly for training the differential-neural distinguisher for 7 rounds results in an accuracy of 0.8013.

Table 5: Parameters, Acc, TPR, TNR of Student Distinguishers

r	Type	Parameters	Acc	TPR	TNR
7	$\mathcal{DD}_T$	9585729	0.8128	0.7941	0.8306
	$\mathcal{DD}_S$	33857	0.8053	0.8119	0.7989
8	$\mathcal{DD}_T$	9585729	0.5532	0.5596	0.5453
	$\mathcal{DD}_S$	33857	0.5403	0.5607	0.5212

$\mathcal{DD}_T$: Teacher distinguisher; $\mathcal{DD}_S$: Student distinguisher

5.2 Key Recovery attack of 13-round Speck32/64 Using Student Distinguisher

We implement key recovery attacks using student distinguishers and examine whether knowledge distillation can accelerate key recovery attacks.

Experiment 2: The overall procedure for conducting the key recovery attack $\mathcal{A}_{II}^{\textsc{Speck}13R}$ using the student distinguisher, trained via knowledge distillation, closely follows the method outlined in Section 4.4. The procedure includes only minor modifications to certain parameters, which are detailed below.

$$n_{it} = 2^{13} \qquad c_1 = 25 \qquad c_2 = -650$$

Compared to Experiment 1, we replaced the distinguisher, necessitating corresponding adjustments to the thresholds, c_1 and c_2. Due to the relatively small scale and the faster

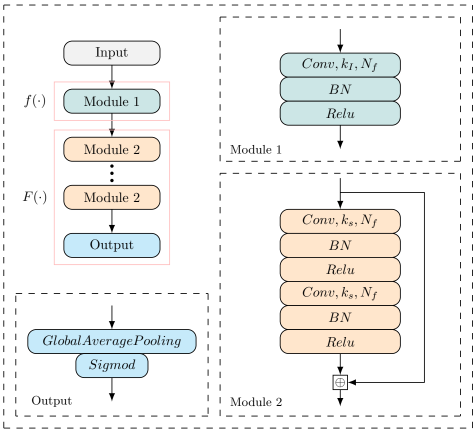


Figure 6: The network architecture of student distinguisher for Speck32/64 and Simon32/64

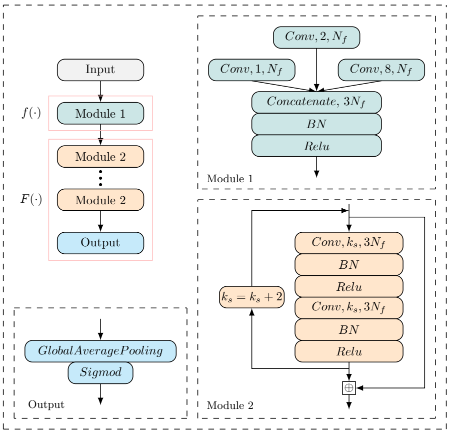


Figure 7: The network architecture of our distinguisher for Simon32/64

prediction speed of the student distinguisher, we increased the number of iterations, thereby enhancing the success rate of the key recovery attack.

For the 13-round key recovery attack $\mathcal{A}_{II}^{\text{Speck}13R}$, the GPU memory utilization was 5239MB, with GPU utilization around 48%. Predictions on 10^6 instances were conducted using both a GPU and a single-core CPU, and the required time was recorded. The GPU completed the task in 15.582 seconds, while the single-core CPU took 218.638 seconds. Thus, the prediction speed ratio P_s between the single-core CPU and GPU is $2^{3.81}$. The classical differential prepended to the attack is valid when keys satisfy $k2[12] \neq k_2[11]$, so the attack was tested only for these valid keys, covering 2^{63} keys. The data complexity D_c is calculated as $2^4 \times 2^9 \times 2^{11} \times 8 \times 2/2 = 2^{27}$ plaintexts, halved due to the use of two classical differentials, which generate half of the required data pairs. In 100 trials, 34 were successful, yielding a success rate (sr) of 34%. The average runtime per trial on our server was 14587.229 seconds. Therefore, the time complexity T_c is given by $n_{kg} \times rt \times \frac{E_s}{R} \times P_s = 2^4 \times 14587.229 \times \frac{2^{27.09}}{13} \times 2^{3.81} = 2^{41.22+3.81}$.

5.3 Brute Force Guessing of 1-round Subkey for 14-round Speck32/64

Within the current framework, there are intuitively two approaches to increase the round of key recovery attacks. The first involves extending the length of the classical differential, and the second involves increasing the rounds of the differential-neural distinguisher. However, based on our current results, neither of these paths appears to be feasible.

- **Increase the Length of the Classical Differential.** In the 13-round key recovery attack $\mathcal{A}_{II}^{\text{Speck}13R}$, we employed 3-round sub-optimal classical differentials with a probability of 2^{-12}, leading to an estimated complexity of the core attack around $2^{41.22}$. We utilized a MILP model to search for an optimal 4-round classical differential path $(0x1488, 0x1008) \rightarrow (0x0040, 0x0000)$, which has a probability of 2^{-17}. Implementing a 14-round key recovery attack using a 4-round classical differential would result in the time complexity of the core attack exceeding 2^{46} under ideal conditions, potentially surpassing the capabilities of our current computing resources. Additionally, the 4-round classical differential lacks sufficient neutral bits for effective use in key recovery attacks.

- **Increase the Round of the Differential-neural Distinguisher.** Although

> we successfully trained a 9-round differential-neural distinguisher, its accuracy was insufficiently high, precluding its use for a 14-round key recovery attack. Our calculations indicate that the accuracy of the Difference Distribution Table (DDT) should be 0.5089 and 0.5138 for $m = 8$ and $m = 16$ respectively. Meanwhile, a differential-neural distinguisher can learn more features than DDT, suggesting that its accuracy should surpass that of DDT. Therefore, if we can improve the accuracy of the 9-roun distinguisher, it may become feasible to implement a 14-round key recovery attack directly. This possibility is an ongoing area of our research.

Fortunately, we can estimate the time complexity of a 14-round key recovery attack. Given the relatively low time complexity of our 13-round key recovery attack, we can approach a 14-round attack by employing a brute force method to guess an additional round of the subkey. The time complexity of the 14-round attack is then calculated by multiplying the time complexity of the 13-round attack by 2^{16}. Therefore, the time complexity T_c of the 14-round key recovery attack $\mathcal{A}^{\textsc{Speck}14R}$ is given by $2^{16} \times n_{kg} \times rt \times \frac{E_s}{R} \times P_s = 2^{16} \times 2^4 \times 14587.229 \times \frac{2^{27.09}}{14} \times 2^{3.81} = 2^{57.12+3.81}$.

6 Differential-Neural Distinguishers on Round-Reduced Simon32/64

In ASIACRYPT 2022, Bao et al. [BGL+22] employed Dense Network and Squeeze-and-Excitation Network to train 9, 10, and 11-round differential-neural distinguishers for Simon32/64. We present the accuracy of the 9, 10, 11, and 12-round differential-neural distinguishers for Simon32/64, employing a modified network architecture in the case of multiple-ciphertext pairs.

6.1 Network Architecture

The network architecture used for training the differential-neural distinguisher for Simon32/64 closely resembles that of Speck32/64. Meanwhile, considering the round function of Simon32/64, we modified the convolutional layers and the convolutional kernel size in the Inception module. Additionally, we have replaced the three fully connected layers with a GlobalAveragePooling layer. The overall architecture is depicted in Fig 7.

Input Representation. Considering the round function of Simon32/64, it is possible to compute $(y_{r-1} \oplus y'_{r-1})$ without knowledge of the $(r-1)$-th subkey. Consequently, the neural network is designed to accept data in the format $(x_r, x'_r, y_r, y'_r, y_{r-1} \oplus y'_{r-1})$. The input layer accommodates m ciphertext pairs, each comprising $2.5L$ units, which are organized in a $[m, \omega, \frac{2.5L}{\omega}]$ array configuration. For Simon32/64, the values of L and ω are set as 32 and 16, respectively.

Initial Convolution (Module 1). The convolutional layer consists of three separate convolution layers, each with $N_f = 32$ channels but differing kernel sizes of 1, 2, and 8. The outputs from these three layers are concatenated along the channel dimension. Following this concatenation, batch normalization is applied to the combined output. The final step involves applying rectifier nonlinearity to the batch-normalized output, resulting in a $[m, \omega, 3N_f]$ matrix. This matrix is then forwarded to the residual blocks layer.

Residual Blocks (Module 2). The residual blocks layer in the netowork architecture mirrors that of Speck32/64. The primary difference lies in the shape of the input, which is adapted to accommodate the specific requirements of the network.

Prediction Head (Output). The prediction head of the network comprises a GlobalAveragePooling layer followed by an output unit. The output unit employs a *Sigmoid* activation function to facilitate the final prediction.

Rationale. Besides the prediction head, the network architectures used for training the differential-neural distinguishers for Speck32/64 and Simon32/64 are fundamentally the same. However, the use of multiple fully connected layers in the Speck32/64 network architecture led to an excessive number of model parameters, increasing the training time. Therefore, in designing the network structure for Simon32/64, we attempted to replace these fully connected layers with GlobalAveragePooling layers. Another modification was adapting the convolutional kernel sizes in the Inception module according to the round function of Simon32/64.

6.2 The Training of Differential-Neural Distinguisher

Training Using the Basic Training Method. The training parameters were based on the basic training method for Speck32/64 as detailed in Sect. 3. By modifying the network architecture, we successfully trained the differential-neural distinguisher to recognize output pairs from 9, 10, and 11-round Simon32/64, given the input difference (0x0000, 0x0040).

Training Using the Staged Training Method. A 12-round differential-neural distinguisher for Simon32/64 was trained through a series of pre-training stages. Initially, we utilized our 11-round distinguisher to identify 9-round Simon32/64 instances with the input difference (0x0440, 0x0100), which is the most likely difference to emerge three rounds after the initial difference (0x0000, 0x0040). This training phase involved 10^7 instances across 20 epochs, with a learning rate set at 10^{-4}. Subsequently, the distinguisher was further trained to recognize 12-round Simon32/64 with the input difference (0x0000, 0x0040). This involved processing another set of 10^7 freshly generated instances over 10 epochs, maintaining the learning rate at 10^{-4}. Finally, we reduced the learning rate to 10^{-5} for processing an additional 10^7 new instances each.

6.3 Result

Test Accuracy. We present the accuracy of 9, 10, 11, and 12-round differential-neural distinguishers for Simon32/64 in Table 6. Additionally, Table 7 lists the Accuracy (Acc), True Positive Rate (TPR), and True Negative Rate (TNR) as tested on newly generated data. Compared to Gohr's work [GLN22], although the accuracy of the differential-neural distinguisher was not improved, the length of the distinguisher was increased by 1 round.

Table 6: Summary accuracy of the distinguisher on Simon32/64

r	m=1		m=2			m=4		
	[BGL+22]	[GLN22]*	[GLN22]*	N/m	N	[GLN22]*	N/m	N
9	0.6532	0.661	0.730	0.7251	0.7240	0.811	0.7991	0.8095
10	0.5629	0.567	0.598	0.5917	0.5907	0.637	0.6239	0.6339
11	0.5173	0.520	0.529	0.5193	0.5240	0.543	0.5343	0.5387
12	-	-	-	-	-	-	-	-

r	m=8			m=16		
	[GLN22]*	N/m	N	[GLN22]*	N/m	N
9	0.896	0.8774	0.8958	0.963	0.9344	0.9630
10	0.692	0.6716	0.6900	0.761	0.7230	0.7608
11	0.563	0.5441	0.5591	0.589	0.5339	0.5878
12	-	-	0.5152	-	-	0.5225

1. *: combining scores of multiple distinguishers trained by using single-ciphertext pair under independence assumption.
2. The combination method used in [GLN22] is the same as that in [BGPT21], which evaluates the distinguisher score of each element in multiple ciphertext pairs, and then takes the median of the results. When the distinguisher is directly trained with multiple-ciphertext pairs, the accuracy of the distinguisher will be reduced.

Table 7: Acc, TPR, TNR of distinguisher on SIMON32/64 using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	9	0.7234	0.7022	0.7446	4	9	0.8101	0.7821	0.8382
	10	0.5902	0.4779	0.7022		10	0.6347	0.5570	0.7124
	11	0.5155	0.8618	0.1663		11	0.5344	0.7139	0.3550
	12	-	-	-		12	-	-	-
8	9	0.8956	0.8811	0.9101	16	9	0.9630	0.9595	0.9664
	10	0.6908	0.6846	0.6953		10	0.7619	0.7207	0.8030
	11	0.5592	0.5946	0.5239		11	0.5871	0.5338	0.6406
	12	0.5159	0.5324	0.4995		12	0.5218	0.5445	0.4991

Wrong Key Response Profile. The *wrong key response profile* for our 9, 10, 11, and 12-round differential-neural distinguishers is illustrated in Fig. 8. It is noteworthy that the distinguisher's score is higher when the difference between the guessed and the real key falls within the set $\{8192, 16384, 24576, 32768, 40960, 49152, 57344\}$, particularly for $r = 10$ and 11, as indicated by the vertical coordinates marked with '$\star$'. This observation suggests that incorrect guesses of the 13th, 14th, and 15th bits of the subkey have a minimal impact on the distinguisher's score. Consequently, it is feasible to reduce the key guessing space by excluding these bits, thereby potentially accelerating the key recovery attack.

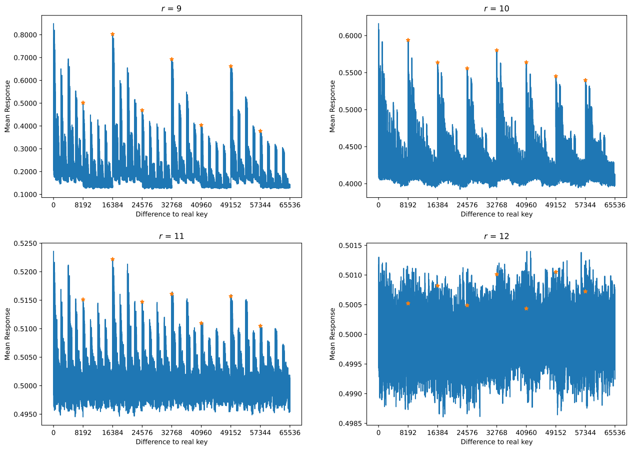


Figure 8: Wrong key response profile for SIMON32/64 using N instances where $m = 8$

6.4 Knowledge Distillation for Simon32/64

When training a student distinguisher for SIMON32/64 using knowledge distillation, the training process and the student network's architecture closely follow the approach outlined in Section 5.1. The sole modification entails adjusting the k_I value in Module 1 of Fig 6 to 8, reflecting the maximum convolution kernel size of 8 in Module 1 of the teacher network.

As indicated in Table 8, employing knowledge distillation leads to a significant reduction in the model's parameters, with only a minor decline in accuracy. When the student network is directly used to train the distinguisher for 10 rounds, the Accuracy (Acc) achieves 0.6810, the True Positive Rate (TPR) is 0.6356, and the True Negative Rate (TNR) is 0.7265. Compared to distinguishers trained directly using the student network, those trained via knowledge distillation, while maintaining similar accuracy levels, demonstrate a more balanced TPR and TNR, indicating smaller variations.

Table 8: Parameters, Acc, TPR, TNR of Student Distinguishers

r	Type	Parameters	Acc	TPR	TNR
10	$\mathcal{DD}_T$	65227	0.6877	0.6932	0.6822
	$\mathcal{DD}_S$	33793	0.6809	0.6750	0.6867
11	$\mathcal{DD}_T$	65227	0.5595	0.5346	0.5843
	$\mathcal{DD}_S$	33793	0.5594	0.5448	0.5739

$\mathcal{DD}_T$: Teacher distinguisher; $\mathcal{DD}_S$: Student distinguisher;

6.5 Key Recovery Attack on 16-round Simon32/64

Following a procedure akin to the key recovery attack on Speck32/64, trained student distinguishers can be effectively utilized in conjunction with a classical differential for conducting key recovery attacks on Simon32/64. It is important to note that, based on insights gained from the *wrong key response profile*, we opted not to guess the 14th and 15th bits of the subkey during the 16-round key recovery attack on Simon32/64.

Experiment 3: The components of the key recovery attack $\mathcal{A}^{\text{Simon}16R}$ of the 16-round Simon32/64 are shown below.

1. 3-round classical differential (0x0440, 0x1000)→(0x0000, 0x0040);
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[2],[3],[4]\}$; generalized neutral bits of combined response of differential-neural distinguis-her:$\{[6],[8],[9],[10],[18],[22],[0,24]\}$ (refer to Table 13);
3. 11(10)-round differential-neural distinguisher $\mathcal{ND}^{\text{Simon}_{11(10)r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\text{Simon}_{11(10)r}} \cdot \mu$ and $\mathcal{ND}^{\text{Simon}_{11(10)r}} \cdot \delta$.

At the outset of our 16-round key recovery attack $\mathcal{A}^{\text{Simon}16R}$ on Simon32/64, we start by guessing two key bits of k_0, namely $k_0[1]$ and $k_0[3]$. This guessing is necessitated by the conditions of the 3-round differential, which require $x_1[1] = x_1'[1] = 0$ and $x_1[3] = x_1'[3] = 0$. Consequently, the number of key guesses, n_{kg}, is 2^2. However, to ensure economic viability in experimental verification, we focused the core of our attack on instances where only these two conditions are met. The specific parameters employed in our 16-round key recovery attack $\mathcal{A}^{\text{Simon}16R}$ are outlined below.

$n_{kg} = 2^2$	$m = 8$	$n_b = 2^7$	$n_{cts} = 2^8$
$n_{it} = 2^9$	$c_1 = 90, c_2 = 100$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

For the 16-round key recovery attack $\mathcal{A}^{\text{Simon}16R}$, the GPU memory utilization was 4923MB, with GPU utilization around 34%. Predictions on 10^6 instances were conducted using both a GPU and a single-core CPU, and the required time was recorded. The time on the GPU was 23.5 seconds, while it was 215.625 seconds on the single-core CPU. This results in a prediction speed ratio P_s between the single-core CPU and GPU of $2^{3.20}$. The data complexity D_c is calculated as $2^2 \times 2^7 \times 2^8 \times 8 \times 2 = 2^{21}$ plaintexts. A total of 100 trials were conducted, with 59 being successful, yielding a success rate (sr) of 59%. The average runtime of the experiment was 300.31 seconds. Therefore, the time complexity T_c is given by $n_{kg} \times rt \times \frac{E_s}{R} \times P_s = 2^2 \times 300.31 \times \frac{2^{25.48}}{16} \times 2^{3.20} = 2^{31.71+3.20}$.

7 Conclusion

In this paper, we designed a novel network architecture to train differential-neural distinguishers, employing multiple parallel convolution layers to capture features of cryptographic algorithms across various dimensions. As a result, we achieved enhanced accuracy and developed distinguishers for more rounds. Addressing the issue of identical data distribution, we proposed a solution that utilizes neutral bits with a probability of 1 to generate multiple-plaintext pairs. Moreover, we implemented knowledge distillation techniques to reduce the size of the distinguisher, thereby accelerating its prediction speed. The combination of these improvements not only reduced the time complexity but also increased the number of rounds in key recovery attacks.

References

[AL13] Hoda AlKhzaimi and Martin M. Lauridsen. Cryptanalysis of the SIMON family of block ciphers. *IACR Cryptol. ePrint Arch.*, page 543, 2013.

[ALLW14] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential cryptanalysis of round-reduced simon and speck. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 525–545. Springer, 2014.

[BC04] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[BCdST⁺23] Alex Biryukov, Luan Cardoso dos Santos, Je Sen Teh, Aleksei Udovenko, and Vesselin Velichkov. Meet-in-the-filter and dynamic counting with applications to speck. In *International Conference on Applied Cryptography and Network Security*, pages 149–177. Springer, 2023.

[BGL⁺22] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Enhancing differential-neural cryptanalysis. In *ASIACRYPT (1)*, volume 13791 of *Lecture Notes in Computer Science*, pages 318–347. Springer, 2022.

[BGPT21] Adrien Benamira, David Gérault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 805–835. Springer, 2021.

[BRV14] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential analysis of block ciphers SIMON and SPECK. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 546–570. Springer, 2014.

[BSS⁺13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, page 404, 2013.

[CSYY21] Yi Chen, Yantian Shen, Hongbo Yu, and Sitong Yuan. A new neural distinguisher considering features derived from multiple ciphertext pairs, 2021.

[Din14] Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In *Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

[FLW⁺23] Zhuohui Feng, Ye Luo, Chao Wang, Qianqian Yang, Zhiquan Liu, and Ling Song. Improved differential cryptanalysis on speck using plaintext structures. *Cryptology ePrint Archive*, 2023.

[GLN22] Aron Gohr, Gregor Leander, and Patrick Neumann. An assessment of differential-neural distinguishers. *Cryptology ePrint Archive*, 2022.

[Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269. IEEE Computer Society, 2017.

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141. Computer Vision Foundation / IEEE Computer Society, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In *ACISP (2)*, volume 9723 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2016.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9. IEEE Computer Society, 2015.

[WWJZ18] Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. *Sci. China Inf. Sci.*, 61(9):098103:1–098103:3, 2018.

A Used Generalized Neutral Bit-sets for Key Recovery Attack

A.1 Generalized Neural Bit-sets for Speck32/64 [BGL+22]

Neutral bit-sets (NB) Used in [Goh19] for 2-round Classical Differential: The signal from the distinguisher will rather be 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 that are probabilistically neutral are summarized as follows.

Table 9: (Probabilistic) single-bit neutral bit for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [Goh19]

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 [BGL+22] for 2-round classical differential: for the prepended 2-round differential on top of differential-neural distinguisher, Bao *et al.* [BGL+22] can experimentally obtain 3 complete NB and 2 SNBS using an exhaustive search. Concretely, for the 2-round differential (0x0211, 0x0a04)→ (0x0040, 0x0000), bit and bit-sets that are (probabilistically) (simultaneous-)neutral are summarized in Table 10.

Table 10: (Probabilistic) SNBS for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [BGL+22]

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 [BGL+22] for 3-round classical differential: Bao *et al.* found that there are three sufficient conditions for a pair $(x, y), (x', y')$ to conform to the 3-round differential (0x8020, 0x4101)→ (0x0040, 0x0000), summarized in Table 11. Concretely, for the 3-round four sub-optimal differential, bit and bit-sets that are (probabilistically) conditional simul-taneous-neutral are summarized in Table 12.

Table 11: Three sufficient conditions conform the 3-round sub-optimal differential [BGL+22]

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

Table 12: (Probabilistic) (simultaneous-)neutral bit-sets for 3-round differential (0x8020, 0x4101) → (0x0040, 0x0000), (0x8060, 0x4101) → (0x0040, 0x0000), (0x8021, 0x4101) → (0x0040, 0x0000), (0x8061, 0x4101) → (0x0040, 0x0000) of Speck32/64 [BGL+22]

	(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	$y[1] = 0$
[4,27,29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$
[6,11,12,18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$

Among the two 3-round differentials (0x8020,0x4101) → (0x0040,0x0000) and (0x8060,0x4101) → (0x0040,0x0000) are adjoining differentials. The bit 5 of x (the bit 21 of $x \| y$) is the SBfADs of both pairs. An SBfADs plays the same role as a deterministic unconditional NB, thus is better to be used than probabilistic and conditional NBs. Specifically, employing SBfADs saves one guessed key bit and reduces both time and data complexity by half compared to employing the CSNBS.

A.2 Generalized Neural Bit-sets for Simon32/64 [BGL+22]

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

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

Table 13: NB and SNBS for 3-round Classical Differential (0x0440, 0x1000)→ (0x0000, 0x0040) of Simon32/64 [BGL+22]

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

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

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

Table 14: NB and SNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of Simon32/64 [BGL+22]

[2]	[6]	[12,26]	[10,14,28]

Table 15: CSNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of Simon32/64 [BGL+22]

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	[18,4,3,10]	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 Key Recovery Attack on 17-round Simon32/64

Combining a 4-round classical differential with an 11-round teacher distinguisher, we examine how far a practical attack can go on 17-round Simon32/64 in this subsection.

Experiment 4: The components of the key recovery attack $\mathcal{A}^{\textsc{Simon}17R}$ of the 17-round Simon32/64 are shown below.

1. 4-round classical differential (0x1000, 0x4440)→ (0x0000, 0x0040);
2. generalized neutral bits of generate multiple-ciphertext pairs: $\{[2], [6], [12, 26]\}$ (refer to Table 14); generalized neutral bits of combined response of differential-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 Table 15).
3. 11(10)-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Simon}_{11(10)r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\textsc{Simon}_{11(10)r}} \cdot \mu$ and $\mathcal{ND}^{\textsc{Simon}_{11(10)r}} \cdot \delta$.

In the beginning, we guess two key bits of k_0, that is, $k_0[3]$ and $k_0[5]$, because of 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 Table 15). Thus, n_{kg} is 2^8. However, to make the experimental verification economic, we tested the core of the attack with the eight conditions being fulfilled only. The concrete parameters used in our 17-round key recovery attack $\mathcal{A}^{\textsc{Simon}17R}$ are listed below.

$n_{kg} = 2^8$	$m = 8$	$n_b = 2^6$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 15, c_2 = 65$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

For the 17-round key recovery attack $\mathcal{A}^{\textsc{Simon}17R}$, the GPU memory utilization was 9187MB, and the GPU utilization was around 83%. We conducted predictions on 10^6 instances using a GPU and a single-core CPU, calculating the required time. The time required on the GPU was 37.589 seconds, while on the single-core CPU, it was 1402.609 seconds. Consequently, the prediction speed ratio P_s between the single-core CPU and GPU is $2^{5.22}$. The data complexity D_c is $2^8 \times 2^6 \times 2^{11} \times 8 \times 2 = 2^{29}$ plaintexts. In total, 100 trials are running, and 9 successful trials, that is $sr = 9\%$. The average running time of the experiment is 2435.63 seconds. Thus, the time complexity $T_c = n_{kg} \times rt \times \frac{E_s}{R} \times P_s = 2^8 \times 2435.63 \times \frac{2^{25.48}}{17} \times 2^{5.22} = 2^{40.64+5.22}$.

C Procedure of $(1 + s + r + 1)$-round key recovery attack

The attack procedure is as follows.

1. Initialize variables $Gbest_{key} \leftarrow$ (None, None), $Gbest_{score} \leftarrow -\infty$.

2. For each of the n_{kg} guessed key bits, on which the conditions depend,

   (a) Generate n_{cts} random data with difference ΔP, and satisfying the conditions being conforming pairs (refer to Appendix A).

   (b) Using n_{cts} random data and $\log_2 m$ neutral bit with probability one to generate n_{cts} data pairs. Every data pairs have m data.

   (c) From the n_{cts} random data pairs, generate n_{cts} structures using the n_b generalized neutral bit.

   (d) Decrypt one round using zero as the subkey for all data in the structures and obtain the n_{cts} plaintext structure.

   (e) Query for the ciphertexts under $(1 + s + r + 1)$-round Speck or Simon of the $n_{cts} \times n_b \times 2$ plaintext structures, thus obtaining n_{cts} ciphertext structures, denoted by $\{\mathcal{C}_1, \ldots, \mathcal{C}_{n_{cts}}\}$.

   (f) Initialize an array $\omega_{\max}$ and an array n_{visit} to record the highest distinguisher score obtained so far and the number of visits have received in the last subkey search for the ciphertext structures.

   (g) Initialize variables $best_{score} \leftarrow -\infty$, $best_{key} \leftarrow$ (None, None), $best_{pos} \leftarrow$ None to record the best score, the corresponding best recommended values for the two subkeys obtained among all ciphertext structures and the index of these ciphertext structures.

   (h) For j from 1 to n_{it}:

      i. Compute the priority of each of the ciphertext structures as follows: $s_i = \omega_{\max i} + \alpha \cdot \sqrt{\log_2 j / n_{\text{visit}i}}$, for $i \in \{1, \ldots, n_{cts}\}$, and $\alpha = \sqrt{n_{cts}}$; The formula of priority is designed according to a general method in reinforcement learning to achieve automatic exploitation versus exploration trade-off based on Upper Confidence Bounds. It is motivated to focus the key search on the most promising ciphertext structures [Goh19].

      ii. Pick the ciphertext structure with the highest priority score for further processing in this j-th iteration, denote it by $\mathcal{C}$, and its index by idx, $n_{\text{visit}idx} \leftarrow n_{\text{visit}idx} + 1$.

      iii. Run the BayesianKeySearch Algorithm [Goh19] with $\mathcal{C}$, the r-round differential-neural distinguisher $\mathcal{ND}^r$ and its wrong key response profile $\mathcal{ND}^r \cdot \mu$ and $\mathcal{ND}^r \cdot \sigma$, n_{cand1}, and n_{byit1} as input parameters; obtain the output, that is, a list L_1 of $n_{byit1} \times n_{cand1}$ candidate values for the last subkey and their scores, i.e., $L_1 = \{(g_{1i}, v_{1i}) : i \in \{1, \ldots, n_{byit1} \times n_{cand1}\}\}$.

      iv. Find the maximum $v_{1\max}$ among v_{1i} in L_1, if $v_{1\max} > \omega_{\max idx}$, $\omega_{\max idx} \leftarrow v_{1\max}$.

      v. For each of recommended last subkey $g_{1i} \in L_1$, if the score $v_{1i} > c_1$,

         A. Descrypt the ciphertext in $\mathcal{C}$ using the g_{1i} by one round and obtain the ciphertext structures $\mathcal{C}'$ of $(1 + s + r)$-round Speck or Simon.

B. Run BayesianKeySearch Algorithm with $\mathcal{C}'$, the differential-neural distinguisher $\mathcal{ND}^{r-1}$ and its wrong key response profile $\mathcal{ND}^{r-1} \cdot \mu$ and $\mathcal{ND}^{r-1} \cdot \sigma$, n_{cand2}, and n_{byit2} as input parameters; obtain the output, that is a list L_2 of $n_{byit2} \times n_{cand2}$ candidate values for the last subkey and their scores, i.e., $L_2 = \{(g_{2i}, v_{2i}) : i \in \{1, \ldots, n_{byit2} \times n_{cand2}\}\}$.

C. Find the maximum v_{2i} and the corresponding g_{2i} in L_2, and denote them by $v_{2\max}$ and $g_{2\max}$.

D. If $v_{2\max} > best_{score}$, update $best_{score} \leftarrow v_{2\max}$, $best_{key} \leftarrow (g_{1i}, g_{2\max})$, $best_{pos} \leftarrow idx$.

vi. If $best_{score} > c_2$, go to Step 2i.

(i) Make a final improvement using VerifierSearch [Goh19] on the value of $best_{key}$ by examining whether the scores of a set of keys obtained by changing at most 2 bits on top of the incrementally updated $best_{key}$ could be improved recursively until no improvement is obtained, update $best_{score}$ to the best score in the final improvement; If $best_{score} > Gbest_{score}$, update $Gbest_{score} \leftarrow best_{score}$, $Gbest_{key} \leftarrow best_{key}$.

3. Return $Gbest_{key}, Gbest_{score}$.