A Deep Learning aided Key-Recovery Framework for Large-State Block Ciphers *

Application to Round-Reduced Large-State SPECK

Yi Chen[1], Zhenzhen Bao[2], Yantian Shen[1] and Hongbo Yu[1]

[1] Department of Computer Science and Technology, Tsinghua University, China,
[2] Institute for Network Sciences and Cyberspace, Tsinghua University, China

Abstract. In the seminal work published by Gohr in CRYPTO 2019, neural networks were successfully exploited to perform differential attacks on SPECK32/64, the smallest member in the block cipher family SPECK. The deep learning aided key-recovery attack by Gohr achieves considerable improvement in terms of time complexity upon the state-of-the-art result from the conventional cryptanalysis method. A further question is whether the advantage of deep learning aided attack can be kept on large-state members of SPECK and other primitives. Since there are several key points in Gohr's key-recovery frameworks that seem not fit for large-state ciphers, this question stays open for years.

This work provides an answer to this question by proposing a deep learning aided multi-stage key-recovery framework. To apply this key-recovery framework on large-state members of SPECK, multiple neural distinguishers (NDs) are trained and carefully combined into groups. Employing the groups of NDs under the multi-stage key-recovery framework, practical attacks are designed and trialed. Experimental results show the effectiveness of the framework. The practical attacks are then extended into theoretical attacks that cover more rounds. To do that, multi-round classical differentials (CDs) are used together with the NDs. To find the CDs' neutral bits to boost signals from the distinguishers, an efficient algorithm is proposed.

As a result, considerable improvement in terms of both time and data complexity of differential key-recovery attacks on round-reduced SPECK with the largest, *i.e.*, the 128-bit state, is obtained. Besides, efficient differential attacks are achieved on round-reduced SPECK with 96-bit and 64-bit states. Since most real-world block ciphers have a state size of no less than 64 bits, this work paves the way for performing cryptanalysis using deep learning on more block ciphers. The code is available at https://github.com/AI-Lab-Y/NAAF.

Keywords: Differential cryptanalysis · Key-recovery · Machine learning · Neural network · Large-state · SPECK

1 Introduction

In [Goh19], a simple neural network was trained to perform a real-or-random cryptographic distinguishing task. The target is SPECK32/64, the smallest member of a block cipher family SPECK [BSS+15]. Specifically, generic deep residual networks were trained to distinguish ciphertext pairs whose corresponding plaintext pairs hold particular differences and the random ones. The obtained neural distinguishers ($\mathcal{ND}$s) cover 5, 6, 7, and 8 rounds of SPECK32/64 and exhibit noticeable advantages over their classical counterparts.

*This is the English version. The initial version is published in SSI, which is available at https://doi.org/10.1360/SSI-2022-0298.

The $\mathcal{ND}$s are then prepended with a classical differential ($\mathcal{CD}$) and the formed hybrid distinguishers ($\mathcal{HD}$s) are used to do key-recovery attacks. In the key-recovery attack, the signal from the distinguisher is rather weak. To boost the signal, a combined score on a structure of ciphertext pairs is used, where the ciphertext pairs in a structure correspond to plaintext pairs that are expected to pass the $\mathcal{CD}$ and reach the input of the $\mathcal{ND}$ together. Such a structure of plaintext pairs can be created from a single plaintext pair using the $\mathcal{CD}$'s neutral bits (NBs) [BC04, Goh19], since flipping an NB of a conforming pair of $\mathcal{CD}$, the new data pair also conform to the $\mathcal{CD}$.

In the basic key-recovery framework in [Goh19], a sufficient number of plaintext structures are chosen, and the ciphertext structures are queried. For each ciphertext structure, ciphertext pairs are decrypted one round under all last-round subkeys. For each last-round subkey, the resulting intermediate state pairs are fed into an r-round $\mathcal{ND}$. The returned scores from the $\mathcal{ND}$ are combined, and the combined score is used to filter the subkey. If the combined score passes a threshold c_1, the one-round-decrypted state pairs within a structure are decrypted another round under all second-to-the-last round subkeys. Combined scores from an $(r-1)$-round $\mathcal{ND}$ are then used to filter the subkeys. If the score passes a threshold c_2, the current last-round and second-to-the-last round subkeys are output as a key guess. After peeling off the last two rounds, a similar (expected to be more efficient) procedure can be applied to recover other subkeys.

In an improved key-recovery framework upon the basic one in [Goh19], ciphertext structures are selectively used according to their best performance and visited frequency. Most importantly, for each ciphertext structure, instead of performing one round of trial decryption under all subkeys, only a few subkeys are trialed. These few subkeys are selected under a key-guessing policy based on a variant of Bayesian optimization in a procedure named as BayesianKeySearch. In BayesianKeySearch, the inconsistency of Speck with the wrong key randomization hypothesis for differential cryptanalysis is exploited. Specifically, the expected response of the $\mathcal{ND}$ upon wrong-key decryption will be not random but depend on the bitwise difference between the trial key and the real key. Accordingly, an effective BayesianKeySearch requires an important preliminary computation on the wrong key response profile (WKRP) using the $\mathcal{ND}$s. Such a WKRP typically requires heavy off-line computation and non-negligible memory, which has not been a problem for attacking small-state ciphers.

Using $\mathcal{HD}$s formed by a 2-round $\mathcal{CD}$ and 6-/7-round $\mathcal{ND}$s, applying the highly selective key-guessing policy, improved key-recovery attacks on 11-round Speck32/64 were presented. Concretely, compared to the state-of-the-art result on the same target [Din14], the time complexity is reduced from 2^{46} to 2^{38} with the data complexity slightly increasing from 2^{14} to $2^{14.5}$ chosen-plaintext pairs.

However, several key points in the above basic and improved key-recovery framework seem not fit for launching practical attacks on large-state versions of Speck. One problem lies in that the $\mathcal{ND}$s trained in [Goh19] take full states as input. A full intermediate state will depend on all bits of the last subkey. For large-state ciphers, the size of subkeys is typically too large to do last-round key guessing in one go and still kept considerable advantage when applying the basic key-recovery framework. Applying the improved framework, the critical WKRP required by the BayesianKeySearch should record the empirical mean and standard deviation of scores for every possible difference between trial subkey and right subkey, thus, is hard to be computed and stored.

Another more subtle problem lies in that, although the $\mathcal{ND}$ accepts all bits of the state pairs as input, it does not exploit the bits at every position equally. In other words, some bit information at certain positions has little effect on the $\mathcal{ND}$. As a consequence, those key bits that mainly relate to the non-informative bits are hard to be correctly guessed. This problem is more severe for large-state ciphers since most of the state bits are non-informative bits for an $\mathcal{ND}$ that covers a moderate number of rounds.

These two problems prohibit practical $\mathcal{ND}$-based key-recovery attacks on large-state ciphers, even for rather short round-reduced versions. In addition to the above two problems, another problem makes it difficult to extend practical $\mathcal{ND}$-based attacks on relatively short round-reduced versions to theoretical attacks on relatively long versions. Concretely, to attack as many rounds as possible, $\mathcal{CD}$s are typically prepended to the $\mathcal{ND}$s to form long $\mathcal{HD}$s. For a $\mathcal{CD}$ to be useful, apart from the number of rounds it covers and the differential probability it has, there is another critical factor. That is, it should have a sufficient number of neutral bits (NBs) for boosting the weak signal from the $\mathcal{HD}$. However, for large state ciphers, the $\mathcal{CD}$s covering a moderate number of rounds will have a low probability. In such cases, it is hard for a random sampling method to generate sufficiently large number of conforming pairs to filter out the NBs.

With these problems, the question that whether similar deep learning aided key recovery attacks are applicable to large-state versions stay open for years.

1.1 Our Contributions

This work presents ways to overcome these problems and proposes $\mathcal{ND}$-based key-recovery attacks to improve differential cryptanalysis of large-state Speck.

1. In this work, a multi-stage $\mathcal{ND}$-based key-recovery framework is proposed. This framework exploits multiple $\mathcal{ND}$s that accept different parts of the state depending on different subsets of key bits. The union of subsets of key bits is the full bit-set (or most part) of a subkey. Subsets of key bits are recovered stage-by-stage. Within each stage, both the basic and the improved key-recovery attacks employing a single $\mathcal{ND}$ can be applied.

2. For an $\mathcal{ND}$ to be useful in the multi-stage key-recovery framework, it should have a decent accuracy and rely on non-full states. Thus, one should selectively use certain state bits for individual $\mathcal{ND}$ and carefully organize multiple $\mathcal{ND}$s.

   In this work, multiple $\mathcal{ND}$s that are selected and retrained after identifying their informative bits are exhibited for various round-reduced large-state Speck. The selected $\mathcal{ND}$s form a group such that they together facilitate the recovery of the full subkey. In addition, such groups of $\mathcal{ND}$s for another three block ciphers, including Simon [BSS+15], DES [BS92], and Present [BKL+07], are exhibited.

3. Employing the groups of $\mathcal{ND}$s, practical attacks on short round-reduced Speck with 128-, 96-, and 64-bit states are designed, the experimental results are reported. To extend the practical attacks to cover more rounds, multiple-round $\mathcal{CD}$s that can be coupled with each $\mathcal{ND}$ are presented. To find their NBs to boost the signal of the distinguishers, an approach to efficiently found NBs of low-probability $\mathcal{CD}$s is proposed, which can be generally applied to Add-Rotate-Xor (ARX) ciphers.

4. As a result, considerable improvement in terms of both time and data complexity of differential key-recovery attacks on round-reduced Speck with the largest, *i.e.*, the 128-bit state, is obtained. Besides, efficient key-recovery attacks under the proposed framework on round-reduced Speck with 96-bit and 64-bit states are presented. The resulted attacks and comparisons with existing attacks are summarized in Table 1.

2 Preliminary and Notations

2.1 A Brief Introduction of Speck

Speck is a family of lightweight block ciphers designed together with another family Simon by researchers from the U.S. National Security Agency (NSA) [BSS+15]. The Speck and

Table 1: Summary of key-recovery attacks on large-state SPECK

Target	#R	Time (#Enc)	Data (#CP)	Succ. Rate	Configure	Ref.
SPECK128/128	**17/32**	2^{113}	2^{113}	-	$1+14r_{\mathcal{CD}}+2$	[Din14]
		$\mathbf{2^{78.98}}$	$\mathbf{2^{61.28}}$	**0.52**	$1+6r_{\mathcal{CD}}+9r_{\mathcal{ND}}+1$	Sect. 6.2
	23/32	$2^{125.35}$	$2^{125.35}$	-	$1+20r_{\mathcal{CD}}+2$	[SHY16]
SPECK128/192	**18/33**	2^{177}	2^{113}	-	$1+14r_{\mathcal{CD}}+3$	[Din14]
		$\mathbf{2^{142.98}}$	$\mathbf{2^{61.28}}$	**0.52**	$1+6r_{\mathcal{CD}}+9r_{\mathcal{ND}}+2$	Sect. 6.2
	24/33	$2^{189.35}$	$2^{125.35}$	-	$1+20r_{\mathcal{CD}}+3$	[SHY16]
SPECK128/256	**19/34**	2^{241}	2^{113}	-	$1+14r_{\mathcal{CD}}+4$	[Din14]
		$\mathbf{2^{206.98}}$	$\mathbf{2^{61.28}}$	**0.52**	$1+6r_{\mathcal{CD}}+9r_{\mathcal{ND}}+3$	Sect. 6.2
	25/34	$2^{253.35}$	$2^{125.35}$	-	$1+20r_{\mathcal{CD}}+4$	[SHY16]
SPECK96/96	13/28	$\mathbf{2^{52.61}}$	$\mathbf{2^{36.60}}$	**0.81**	$1+4r_{\mathcal{CD}}+7r_{\mathcal{ND}}+1$	Sect. 6.2
	18/28	2^{82}	2^{82}	-	$1+15r_{\mathcal{CD}}+2$	[SHY16]
SPECK96/144	14/29	$\mathbf{2^{100.61}}$	$\mathbf{2^{36.60}}$	**0.81**	$1+4r_{\mathcal{CD}}+7r_{\mathcal{ND}}+2$	Sect. 6.2
	19/29	2^{130}	2^{82}	-	$1+15r_{\mathcal{CD}}+3$	[SHY16]
SPECK64/96	11/26	$\mathbf{2^{41.13}}$	$\mathbf{2^{26.47}}$	**0.90**	$1+3r_{\mathcal{CD}}+6r_{\mathcal{ND}}+1$	Sect. 6.2
	12/26	$\mathbf{2^{73.13}}$	$\mathbf{2^{26.47}}$	**0.90**	$1+3r_{\mathcal{CD}}+6r_{\mathcal{ND}}+2$	Sect. 6.2
	19/26	$2^{93.56}$	$2^{61.56}$	-	$1+15r_{\mathcal{CD}}+3$	[SHY16]
SPECK64/128	13/29	$\mathbf{2^{105.13}}$	$\mathbf{2^{26.47}}$	**0.90**	$1+3r_{\mathcal{CD}}+6r_{\mathcal{ND}}+3$	Sect. 6.2
	20/29	$2^{125.56}$	$2^{61.56}$	-	$1+15r_{\mathcal{CD}}+4$	[SHY16]

- Not available.

SIMON families have been standardized by ISO as a part of the RFID air interface standard (ISO/29167-21 for SIMON and ISO/29167-22 for SPECK).

The SPECK family contains ten members, each of which is characterized by its block size $2n$ and key size mn, thus is named as SPECK$2n/mn$. The concrete parameters of the members of the SPECK family can be found in Table 2. In the following presentation, when the key size does not affect the applicability, the members that have the same state size will be collectively named as SPECK$2n$.

The encryption of SPECK$2n$ maps a plaintext of two n-bit words (x_0, y_0) into a ciphertext (x_r, y_r), using a sequence of r round functions. The round function is defined as

$$\begin{aligned} x_{i+1} &= ((x_i^{\ggg\alpha}) \boxplus y_i) \oplus rk_{i+1}, \\ y_{i+1} &= (y_i^{\lll\beta}) \oplus x_{i+1}. \end{aligned}$$

where rk_{i+1} is the $(i+1)$-th round key for $i \in [0, r-1]$. The round keys $rk_i, i \in [1, r]$ are generated from an m-word master key by reusing the round function and adding constants.

2.2 Notations

For SPECK, denote by n the word size in bits, $2n$ the state size in bits.
For SPECK, denote by (x_r, y_r) the left and right branches of a state after the encryption of r rounds.
For SPECK, denote by rk_r the r-round subkey added before obtain (x_r, y_r).
Denote by $x[i]$ (resp. $y[i]$ and $rk[i]$) the i-th bit of x (resp. y and rk) counted starting from 0.
Denote by $rk[i_2 \sim i_1]$ the $rk[i_2] \| rk[i_2-1] \| \ldots \| rk[i_1+1] \| rk[i_1]$.
For SPECK, denote by $[j]$ the index of the j-th bit of the state, *i.e.*, the concatenation of x and y, where $y[0]$ is the 0-th bit, and $x[0]$ is the n-th bit.
Denote by $\{i_2 \sim i_1\}$ the set of bit index $\{i_2, i_2-1, \ldots, i_1+1, i_1\}$.

Denote by $\{[i_2] \sim [i_1]\}$ the set of neutral bits $\{[i_2], [i_2-1], \ldots, [i_1+1], [i_1]\}$.
Denote by $\oplus$ the bit-wise XOR, $\boxplus$ the addition modulo 2^n.
Denote by $x^{\lll s}$ (resp. $x^{\ggg s}$) the bit-wise left (resp. right) rotation by s positions.
Denote by $x^{\ll s}$ (resp. $x^{\gg s}$) the bit-wise left (resp. right) shift x by s positions.
For any bit-strings x, y, and z, define $\texttt{eq}(x,y,z) := (\neg x \oplus y) \wedge (\neg x \oplus z)$ (*i.e.*, $\texttt{eq}(x,y,z)[i] = 1$ if and only if $x[i] = y[i] = z[i]$).
For any bit-strings x, y, and z, define $\texttt{xor}(x,y,z) := x \oplus y \oplus z$.
For any integer n, let $\texttt{mask}(n) := 2^n - 1$.
The notation $i \in [1,n]$ means i is an integer in range $[1,n]$.
$\Delta_{[\mathrm{i}]}$ represents a single-bit difference whose i-th bit is the only active bit.
$hw(x,y)$ represents the Hamming distance between x and y.

2.3 Neural Distinguishers

Gohr in [Goh19] shows that a deep residual network could be trained to capture the non-randomness of the distribution of values of output pairs when the input pairs to round reduced Speck32/64 are of specific difference, and thus play the role of distinguisher in cryptanalysis.

Many researchers further verify that neural distinguishers against other ciphers could also be obtained using the method proposed by Gohr [BBCD21, CY21]. In the following, the way of training neural distinguishers introduced in [Goh19] is briefly recalled.

The Training Data and Input Representation. For a target cipher, the deep residual network is to be trained to distinguish between ciphertext pairs whose corresponding plaintext pairs hold a given difference, denoted by Δ, and those whose corresponding plaintext pairs are randomly selected (for the resulted neural distinguisher, we say Δ is its input difference.) Thus, each sample of the training data is a ciphertext pair together with a label taking a value 0 or 1, where 0 means the difference of the corresponding plaintext pair is random, and 1 means the difference is the given value Δ. Among the training dataset and validation dataset, half are positive samples labeled by 1, and the other half are negative samples labeled by 0.

Assume that the block size of the target cipher is mn. A ciphertext pair is written as a sequence of n-bit words $(w_0, \cdots, w_{m-1})$. Then the w_i are directly interpreted as the row-vectors of an $m \times n$-matrix. The input layer of the deep residual network consists of mn units likewise arranged in an $m \times n$ array. When we train neural distinguishers against different ciphers, only the number of units in the input layer is changed. More details of the deep residual network refer to [Goh19].

Training Scheme. Three training schemes are introduced in [Goh19]. We introduce and use the basic training scheme in this paper. Training is run for 50 epochs on the training dataset of size 10^7. The training dataset is processed in batches of size 5000. The deep residual network is validated on the validation dataset. Optimization is performed against mean square error loss plus a small penalty based on L2 weights regularization (with regularization parameter 10^{-5}) using the Adam algorithm [KB15] with default parameters in Keras [C+15]. A cyclic learning rate schedule is used, setting the learning rate l_i for epoch i to $l_i = a + \frac{(n-i) \mod (n+1)}{n} \times (b-a)$, with $a = 10^{-4}, b = 2 \times 10^{-3}$ and $n = 9$.

Once the training is finished and the deep residual network achieves a distinguishing accuracy higher than 0.5 on the validation dataset, it is a valid neural distinguisher. Feed a ciphertext pair (C_0, C_1) into this network, it will output a score Z where $0 \leqslant Z \leqslant 1$. If $Z > 0.5$, the prediction label of the input is 1. Otherwise, the prediction label is 0.

Table 2: The SPECK parameters.

Block Size ($2n$)	Key Size (mn)	Rounds	α	β
32	64	22	7	2
48	72	22	8	3
	96	23	8	3
64	96	26	8	3
	128	27	8	3
96	96	28	8	3
	144	29	8	3
128	128	32	8	3
	192	33	8	3
	256	34	8	3

3 Deep Learning aided Key-Recovery Framework for Large-State Ciphers

This section introduces an $\mathcal{ND}$-based key-recovery framework. The proposal of the following framework is to overcome problems when applying existing $\mathcal{ND}$-based attacks to large-state block ciphers.

In the sequel, the core idea of this key-recovery framework is firstly introduced. Then, the explicit attack procedure and the complexity analysis are presented, which will be directly applied in the concrete key-recovery attacks on various versions of SPECK. At last, additional factors for applying the framework are discussed.

3.1 Core Idea

The high-level idea is to recover a full subkey with a multi-stage procedure. Each stage employs an individual $\mathcal{ND}$ (typically prepended with a $\mathcal{CD}$) to recover partial key bits. For that, multiple $\mathcal{ND}$s that accept different parts of the state are selected so that their feeding data depend on different subsets of key bits whose union covers the full subkey. Subsets of key bits are recovered either in parallel stages or in sequential stages, which depends on the targeted cipher. Within each stage, the existing basic or the improved key-recovery attacks can be applied. Figure 1 is the schematic diagram of this multi-stage key-recovery framework.

Selection Criteria. For an x-stage key-recovery attack, the x neural distinguishers are selected with the following criteria.

1. Each neural distinguisher $\mathcal{ND}_i$ accepts a partial state and undertakes the task of recovering a partial key. Denote the set of the partial state bits by $\mathcal{C}_i$ and the set of partial key bits by $\mathcal{B}_i$ for $i \in [1, x]$.

2. The union $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_x$ should cover as many key bits as possible. In an ideal scenario, the union contains all bits of the key to be recovered.

3. The size of the partial state accepted by each $\mathcal{ND}$ is appropriate, such that their accuracy is sufficient, at the same time the allocated workload for recovering partial key-bits is feasible and balanced. That is, $|\mathcal{C}_i|$ and $|\mathcal{B}_i|$ are well allocated for $i \in [1, x]$.

Based on the x neural distinguishers, the whole key recovery attack is divided into x stages. In Stage i for $i \in [1, x]$, the $|\mathcal{B}_i|$ key bits are recovered using a single $\mathcal{ND}_i$, which can be reviewed as attacking a small-state cipher.

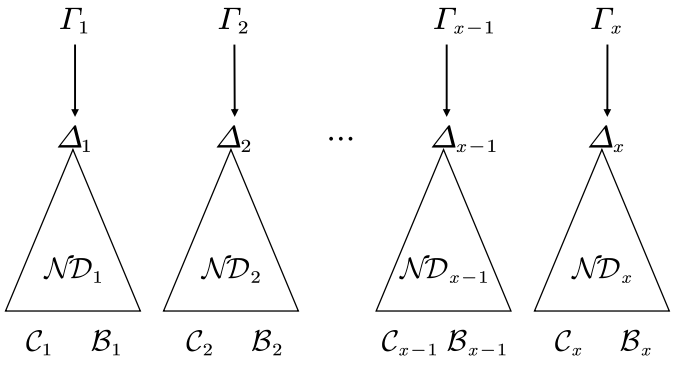

Figure 1: The schematic diagram of the multi-stage key-recovery framework for large-state block ciphers. A total of x neural distinguishers $\mathcal{ND}_i$ are used, whose input differences are Δ_i; each $\mathcal{ND}$ is prepended with a $\mathcal{CD}$, and $\mathcal{CD}_i$ is defined as $\Gamma_i \to \Delta_i$ for $i \in [1, x]$. Each $\mathcal{ND}_i$ is trained on partial state bits $\mathcal{C}_i$, and is used to recover partial key bits $\mathcal{B}_i, i \in [1, x]$.

3.2 The Attack Procedure

Suppose one has x neural distinguishers $\mathcal{ND}_i$ and x prepended differentials $\mathcal{CD}_i := \Gamma_i \xrightarrow{p_i} \Delta_i$, where p_i is the differential probability of $\mathcal{CD}_i$ for $i \in [1, x]$ (see Figure 1). The target is to recover the last-round subkey rk.

The whole attack contains x stages performed one by one. In Stage i, $|\mathcal{B}_i|$ bits of the last subkey is to be recovered with the knowledge of the previously recover key-bits $\bigcup_{j\in[1,i-1]} \mathcal{B}_j$. The concrete method to recover $\mathcal{B}_i$ can be either the basic one or the improved one (when the targeted cipher has inconsistency with the wrong key randomization hypothesis such that BayesianKeySearch can be applied) in [Goh19]. For generality, let us describe the procedure using the basic attack within each stage. The whole procedure goes as follows. Let ϵ be a small constant.

1. For $i \in [1, x]$, do as follows.
   (a) Launch Stage i by choosing $\frac{\epsilon}{p_i}$ plaintext pairs with difference Γ_i. Expand the plaintext pairs into plaintext structures using the $\log_2(N_i)$ neutral bits of $\mathcal{CD}_i$.
   (b) For each of the $\frac{\epsilon}{p_i}$ plaintext structures, do as follows.
      i. Query the encryption oracle and obtain the corresponding ciphertext structure. Note that each ciphertext structure contains N_i ciphertext pairs.
      ii. Initialize a list $L_i \leftarrow \emptyset$
      iii. For each of the β_i top-ranked partial key guesses for bits in $\bigcup_{j\in[1,i-1]} \mathcal{B}_j$ that were recommended from the previous stages, denoted it by $\vec{kg}_{i-1} := kg_{i-1} \| \cdots \| kg_1$ (for Stage 1, $\beta_1 = 1$ and $\vec{kg}_{i-1} = \varnothing$)
         A. For each of the $2^{|\mathcal{B}_i|}$ possible value kg_i of the key bits in $\mathcal{B}_i$, denote the concatenation $kg_i \| kg_{i-1} \| \cdots \| kg_1$ by $\vec{kg}_i$.
            - Partially decrypt the N_i ciphertext pairs by one round using $\vec{kg}_i$ to obtain pairs of values for state bits in $\mathcal{C}_i$.
            - Feed N_i partial state pairs into $\mathcal{ND}_i$, obtain N_i scores Z_j for $j \in [1, N_i]$.
            - Combine the scores using the following formula

$$v_{\vec{kg}_i} := \sum_{j=1}^{N_i} \log_2 \left(\frac{Z_j}{1 - Z_j} \right). \tag{1}$$

B. If $v_{\vec{kg}_i} > c_i$, store $(\vec{kg}_i, v_{\vec{kg}_i})$ in L_i.

iv. If $L_i \neq \emptyset$, sort L_i according to the scores of the guessed key bits, and take the β_{i+1} top-ranked values as the guessed value for the key-bits in $\bigcup_{j\in[1,i]} \mathcal{B}_j$. Go to Step 1a.

(c) If all $\frac{\epsilon}{p_i}$ ciphertext structures have been used and no values of $\vec{kg}_i$ obtain a score passing c_i, terminate the attack with output $\perp$.

2. Return the concatenated key bits $\vec{kg}_x$ with the highest score in the last stage as the guessed value for rk.

Complexity Analysis. In the worst case, all the $\frac{\epsilon}{p_i}$ ciphertext structures are queried at each Stage i for $i \in [1, x]$. Thus, the data complexity of the above procedure is upper bounded by

$$\sum_{i=1}^{x} \frac{\epsilon}{p_i} \times N_i. \tag{2}$$

In the above attack procedure, there are three basic operations (within the inner most loop in Step 1(b)iiiA):

1. perform one round decryption on a ciphertext pair using a key guess.
2. feed a decrypted ciphertext pair into a neural distinguisher and obtain the score Z.
3. compute the value of $\log_2\left(\frac{Z}{1-Z}\right)$.

Taking the combination of the three operations as the atomic operation, the computational complexity in terms of the number of the atomic operations is upper bounded by

$$\sum_{i=1}^{x} \beta_i \times 2^{|\mathcal{B}_i|} \times \frac{\epsilon}{p_i} \times N_i, \tag{3}$$

where N_i is the number of ciphertext pairs in a ciphertext structure in Stage i, and $\beta_1 = 1$.

Suppose the atomic operation in Stage i is equivalent to η_i encryptions of the targeted cipher, for $i \in [1, x]$. The computational complexity in terms of the number of equivalent encryptions is then is upper bounded by

$$\sum_{i=1}^{x} \beta_i \times 2^{|\mathcal{B}_i|} \times \frac{\epsilon}{p_i} \times N_i \times \eta_i. \tag{4}$$

3.3 Applicable Scenario

Apart from the careful combination of x neural distinguishers for applying the x-stage key-recovery framework, there might be other factors one should consider.

- The first factor that needs to be considered is the selection of the input difference of the multiple $\mathcal{ND}$s. Assume that the x neural distinguishers are built using input differences Δ_i for $i \in [1, x]$. The x input differences $\Delta_i, i \in [1, x]$ can be the same one for reusing data among different stages. However, typically, it is hard to find such an input difference suitable for multiple $\mathcal{ND}$s for a large-state cipher. Nevertheless, even if the input differences of multiple $\mathcal{ND}$s are different, it is possible that the influence on the data requirement is not significant. For various targeted ciphers in this work, it is much easier to find a good combination of x neural distinguishers using different input differences, at the same time, the influence on the data complexity is kept to be a constant factor.

- The second factor is about the $\mathcal{CD}$s to be prepended to each $\mathcal{ND}$ (as shown in Figure 1, a $\mathcal{CD}_i := \Gamma_i \to \Delta_i$ is prepended to an $\mathcal{ND}_i$ for $i \in [1, x]$). The numbers of rounds covered by the $\mathcal{CD}$s, their differential probabilities, and their number of neutral bits should be matched with their coupled $\mathcal{ND}$s and the other $\mathcal{CD}$ and $\mathcal{ND}$ couples. For example, the number of neutral bits of a $\mathcal{CD}$ should be sufficient with respect to the accuracy of the corresponding $\mathcal{ND}$.
- The third factor is the execution order of x stages. The x stages can be launched in parallel if the recovery of one subset of key bits does not rely on the recovery of other subsets. Otherwise, the x stages must be executed sequentially. For example, for SPECK, the attacks are triggered from the recovery of the least significant bits (with respect to $\boxplus$) to the most significant bits stage-by-stage. In sequential execution, in case that Stage i wrongly guessed the partial key-bits in $\mathcal{B}_i$, that might influence the guessing of key bits in $\bigcup_{j\in[i+1,x]} \mathcal{B}_i$. The degree of influence depends on the property of the targeted cipher. For SPECK (ARX cipher), the influential factors include both the number and the position of the wrongly guessed key bits. In our experiments on SPECK, when the previous stages get outputs, even if there are a few wrongly guessed key bits, it is still possible that the later stages correctly guess the remaining key bits.

Apart from the above considerations, there is no strict limitations for the application of the framework.

4 Neural Distinguishers on Large-State SPECK

This section presents a simple way to build multiple $\mathcal{ND}$s that are ready to be employed in the multi-stage key-recovery framework. Groups of $\mathcal{ND}$s are exhibits for large-state members of SPECK.

Since the size of the key has no influence on the distinguishers, for convenience, the members of SPECK with different key sizes but with the same state size ($2n$-bit) will be collectively named as SPECK$2n$.

4.1 A Procedure to Build a Group of $\mathcal{ND}$s

To found a proper combination of $\mathcal{ND}$s to launch a multi-stage key-recovery attack, one takes the following steps.

1. Train b neural distinguishers with input difference $\Delta_{[\mathtt{i}]}$ for $i \in [0, b-1]$. Where $\Delta_{[\mathtt{i}]}$ means a single-bit difference whose i-th bit is the only active bit [1]. The fed data are full-state ciphertext pairs. For SPECK$2n$, $b = 2n$, and the fed data are pairs of $2n$-bit data.
2. Identify *informative bits*, denote the set by $\mathcal{C}_i$, *i.e.*, ciphertext bits that have a significant influence on the accuracy of $\mathcal{ND}$s. This can be done using the *Bit Sensitivity Test* introduced in [CY20].
3. Select from b neural distinguishers a proper combination of x neural distinguishers according to the $\mathcal{C}_i$'s, the related subkey bits, and the criteria listed in Sect 3.1.
4. Train each of the x selected $\mathcal{ND}$s from the scratch and with input state pairs being truncated such that only the corresponding informative bits are left. For SPECK$2n$, the fed data are pairs of $|\mathcal{C}_i|$-bit data.

[1] More generally, these input differences can be not limited to single-bit ones as long as the obtained multiple $\mathcal{ND}$s have different sets of informative bits.

4.2 Neural Distinguishers against large-state SPECK

Using the procedure in Sect 4.1, a combination of five 9-round $\mathcal{ND}$s with the following input differences is obtained for SPECK128:

$$\Delta_1 = \Delta_{[64]},\ \ \Delta_2 = \Delta_{[76]},\ \ \Delta_3 = \Delta_{[90]},\ \ \Delta_4 = \Delta_{[105]},\ \ \Delta_5 = \Delta_{[117]}. \tag{5}$$

A combination of four 7-round $\mathcal{ND}$s with the following input differences is obtained for SPECK96:

$$\Delta_1 = \Delta_{[53]},\ \ \Delta_2 = \Delta_{[65]},\ \ \Delta_3 = \Delta_{[77]},\ \ \Delta_4 = \Delta_{[89]}. \tag{6}$$

A combination of three 6-round $\mathcal{ND}$s with the following input differences is obtained for SPECK64:

$$\Delta_1 = \Delta_{[42]},\ \ \Delta_2 = \Delta_{[47]},\ \ \Delta_3 = \Delta_{[33]}. \tag{7}$$

Table 4 summarizes the details of the combination of $\mathcal{ND}$s for the three large-state subfamilies of SPECK. These combinations of $\mathcal{ND}$s will be employed in the key-recovery attacks in the next sections.

Table 4: Neural distinguishers against round-reduced large-state SPECK.

	9-round SPECK128			7-round SPECK96			6-round SPECK64		
$\mathcal{ND}_i$	Δ_i	$\mathcal{C}_i$	Acc.	Δ_i	$\mathcal{C}_i$	Acc.	Δ_i	$\mathcal{C}_i$	Acc.
$\mathcal{ND}_1$	$\Delta_{[64]}$	$\{22\sim18, 14\sim9\}$	0.559	$\Delta_{[53]}$	$\{19\sim8\}$	0.633	$\Delta_{[42]}$	$\{17\sim8\}$	0.613
$\mathcal{ND}_2$	$\Delta_{[76]}$	$\{34\sim30, 26\sim21\}$	0.586	$\Delta_{[65]}$	$\{31\sim20\}$	0.621	$\Delta_{[47]}$	$\{29\sim18\}$	0.677
$\mathcal{ND}_3$	$\Delta_{[90]}$	$\{48\sim44, 40\sim34\}$	0.609	$\Delta_{[77]}$	$\{43\sim32\}$	0.628	$\Delta_{[33]}$	$\{31, 30, 7\sim0\}$	0.653
$\mathcal{ND}_4$	$\Delta_{[105]}$	$\{63\sim59, 55\sim49\}$	0.616	$\Delta_{[89]}$	$\{47\sim44, 7\sim0\}$	0.634			
$\mathcal{ND}_5$	$\Delta_{[117]}$	$\{11, 7, 4, 3, 0\}$	0.559						

Δ_i: the input difference Acc.: the accuracy of the $\mathcal{ND}_i$

$\mathcal{C}_i$: the index of bits of x_r (at the same time, take the same index of bits of y_r) that are fed into $\mathcal{ND}_i$, where x_r (resp. y_r) is the left (resp. right) n-bit word of a full r-round output state.

5 Practical Key Recovery Attacks on Large State SPECK

This section presents practical key recovery attacks on large stage SPECK employing the groups of neural distinguishers presented in Section 4.

5.1 Practical Attack on 12-Round SPECK128

Applying the multi-stage key-recovery framework proposed in Sect 3, and employing the combinations of $\mathcal{ND}$s found in Sect 4, a practical attack on 12-round SPECK128 can be launched to recover the full last subkey.

Concretely, each of the five 9-round neural distinguishers $\mathcal{ND}_i$ (shown in Table 4) is prepended with a 1-round classical differential $\Gamma_i \to \Delta_i$ (shown in Table 5) and form a 10-round hybrid distinguisher $\mathcal{HD}_i$, which can be freely extended another round at the top since there is no whitening key added before the first nonlinear operation in SPECK. Hence, at the top of the last round, there are 11 rounds in total, and the goal is to recover the 12-th subkey rk_{12}. For launching the attack procedure in Sect. 3.2, concrete parameters are as follows.

Attack Setting. The 64 bits of the round key rk_{12} is recovered through five stages:

1. Stage 1 guesses $|\mathcal{B}_1| = 15$ bits, *i.e.*, $rk_{12}[14 \sim 0]$.

2. Stage 2 guesses $|\mathcal{B}_2| = 12$ bits, *i.e.*, $rk_{12}[26 \sim 15]$.
3. Stage 3 guesses $|\mathcal{B}_3| = 14$ bits, *i.e.*, $rk_{12}[40 \sim 27]$.
4. Stage 4 guesses $|\mathcal{B}_4| = 15$ bits, *i.e.*, $rk_{12}[55 \sim 41]$.
5. Stage 5 guesses $|\mathcal{B}_5| = 8$ bits, *i.e.*, $rk_{12}[63 \sim 56]$.

Table 5 summarizes the to be used five 1-round $\mathcal{CD}$s together with their differential probability p_i and their 10 neutral bits (thus, $N_i = 2^{10}$ for $i \in [1, 5]$). The small constant ϵ is set to be 4. At Stage i, at most $\frac{4}{p_i}$ ciphertext structures are generated, for $i \in [1, 5]$. In the five stages, the thresholds on the scores for filtering wrong key guesses are set to be $c_1 = c_2 = c_3 = c_4 = c_5 = 10$. For Stage i, the upper bound of the number of kept surviving key guesses is set to be $\beta_{i+1} = 3$ for $i \in [1, 4]$ (note that β_1 is always 1).

Table 5: One-round classical differentials to be prepended to the $\mathcal{ND}$s in Table 4.

	1-round Speck128			1-round Speck96			1-round Speck64		
$\mathcal{CD}_i$	$\Gamma_i \to \Delta_i$	NB's	p_i	$\Gamma_i \to \Delta_i$	NB's	p_i	$\Gamma_i \to \Delta_i$	NB's	p_i
$\mathcal{CD}_1$	$\Delta_{[72,69,61]} \to \Delta_{[64]}$	$\{[20] \sim [11]\}$	2^{-1}	$\Delta_{[61,58,2]} \to \Delta_{[53]}$	$\{[25] \sim [16]\}$	2^{-2}	$\Delta_{[50,47,7]} \to \Delta_{[42]}$	$\{[39] \sim [30]\}$	2^{-2}
$\mathcal{CD}_2$	$\Delta_{[84,81,9]} \to \Delta_{[76]}$	$\{[32] \sim [23]\}$	2^{-2}	$\Delta_{[73,70,14]} \to \Delta_{[65]}$	$\{[37] \sim [28]\}$	2^{-2}	$\Delta_{[55,52,12]} \to \Delta_{[47]}$	$\{[39] \sim [30]\}$	2^{-2}
$\mathcal{CD}_3$	$\Delta_{[98,95,23]} \to \Delta_{[90]}$	$\{[46] \sim [37]\}$	2^{-2}	$\Delta_{[85,82,26]} \to \Delta_{[77]}$	$\{[49] \sim [40]\}$	2^{-2}	$\Delta_{[41,38,30]} \to \Delta_{[33]}$	$\{[29] \sim [20]\}$	2^{-2}
$\mathcal{CD}_4$	$\Delta_{[113,110,38]} \to \Delta_{[105]}$	$\{[61] \sim [52]\}$	2^{-2}	$\Delta_{[94,49,38]} \to \Delta_{[89]}$	$\{[61] \sim [52]\}$	2^{-2}			
$\mathcal{CD}_5$	$\Delta_{[125,122,50]} \to \Delta_{[117]}$	$\{[73] \sim [64]\}$	2^{-2}						

Experimental Results. Under the above attack setting, we performed the attack in 100 trials. The trials were performed on a PC with a modern graphic card (GeForce GTX 1080 Ti GPU). The average time consumption of the attack using a single CPU core and a GPU is about 5060 seconds.

Out of the 100 trials, there are 80 trails that return a key candidate. Denote the Hamming distance between the returned key candidate kg and the real round key rk by $hw(kg, rk)$. The mean value of $hw(kg, rk)$ over the 80 trials is 3.2. The concrete statistic on $hw(kg, rk)$ is reported in Table 6. From Table 6, when we count an attack as successful as long as $hw(kg, rk) \leqslant 3$, the success rate is 0.52.

Table 6: Statistic on $hw(kg, rk)$ over 100 trials of the 12-round attack on Speck128

$hw(kg, rk)$	0	1	2	3	4	5	6	7	8	9
# trials	4	12	17	19	8	9	7	3	0	1

We argue that an attack can be considered as successful as long as $hw(kg, rk) \leqslant 3$. The reasons are as follows. Firstly, note that correctly guessing at least $(64 - 3)$ bits is equivalent to reducing the round key space from 2^{64} to $C(64, 3) \times 2^3 = 2^{18.3}$. Actually, the remaining key space could be smaller since we observe that some round key bits are more likely to be wrongly guessed than other key bits. [2]

Secondly, to fully correct the few bit errors in the returned key guessing and recover other round keys, the complexity is expected to be small. To do that, one can employ i-round neural distinguishers where $i \leqslant 8$, and reuse the ciphertext structures that were used to recommend the returned last-round key. The ciphertext structures that were used to recommend the returned key can be taken as the correct ones (their corresponding plaintext structures conform to the prepended $\mathcal{CD}$s and reach the input difference of the

[2] This phenomenon also occurs in our attacks on other variants of reduced Speck. However, the positions of wrongly guessed key bits do not have common characteristic among different versions.

$\mathcal{ND}$s). The reason is that the thresholds c_i are set such that using wrong ciphertext structures, there will be no key candidates able to be recommended (note that the numbers of neutral bits of the $\mathcal{CD}$s that we used are relatively high, such that the signals from the distinguishers are very strong. Thus, the thresholds can be easily set to block almost all wrong ciphertext structures.)

5.2 Practical Attacks on Reduced Speck96, Speck64

Similar to the practical attack on round-reduced Speck128, practical attacks on round-reduced Speck96 and Speck64 can be launched to recover the full last subkey. The attacks are direct applications of the multi-stage key-recovery framework proposed in Sect 3 and the combinations of $\mathcal{ND}$s found in Sect 4.

Practical Attack on 10-round Speck96. Four 7-round $\mathcal{ND}$s are to be employed together with the 1-round $\mathcal{CD}$s (see Tables 4 and 5). After freely extending another round, there are 9 rounds at the top, and the goal is to recover the 10-th subkey rk_{10} in four stages.

In the four stages, each stage guesses 12 bits of the round key rk_{10}, namely $rk_{10}[11 \sim 0]$, $rk_{10}[23 \sim 12]$, $rk_{10}[35 \sim 24]$, and $rk_{10}[47 \sim 36]$, respectively. Table 5 summarizes the to be used four 1-round $\mathcal{CD}$s together with their differential probability p_i and their 10 neutral bits (thus, $N_i = 2^{10}$ for $i \in [1, 4]$). The other attack parameters are set to be the same as that in the 12-round attack on Speck128 in Sect. 5.1. That is, $c_i = 10$ for $i \in [1, 4]$ and $\beta_i = 3$ for $i \in [2, 4]$.

Under the above attack setting, the attack was performed in 100 trials. The average time consumption of the attack on the same PC (with a GeForce GTX 1080 Ti GPU card) is about 825 seconds. Out of the 100 trials, there are 87 trails that return a key candidate. The concrete statistic on the Hamming distance between the returned key candidate kg and the real round key rk, *i.e.*, $hw(kg, rk)$, is reported in Table 7. From Table 7, if an attack is considered as successful as long as $hw(kg, rk) \leqslant 3$, the success rate is 0.81.

Table 7: Statistic on $hw(kg, rk)$ over 100 trials of the 10-round attack on Speck96

$hw(kg, rk)$	0	1	2	3	4	5
# trials	23	30	18	10	3	3

Practical Attack on 9-round Speck64. Three 6-round $\mathcal{ND}$s are to be employed together with the 1-round $\mathcal{CD}$s (see Tables 4 and 5). After freely extending another round, there are 8 rounds at the top, and the goal is to recover the 9-th subkey rk_9 in three stages.

At the three stages, there are 10, 12, and 10 bits of rk_9 to be guessed, namely $rk_9[9 \sim 0]$, $rk_9[21 \sim 10]$, and $rk_9[31 \sim 22]$, respectively. Table 5 summarizes the to be used four 1-round $\mathcal{CD}$s together with their differential probability p_i and their 10 neutral bits (thus, $N_i = 2^{10}$ for $i \in [1, 3]$). The other attack parameters are set to be the same as that in the 12-round attack on Speck128 in Sect. 5.1. That is, $c_i = 10$ for $i \in [1, 3]$ and $\beta_i = 3$ for $i \in [2, 3]$.

Under the above attack setting, the attack was performed in 100 trials. The average time consumption of the attack on the same PC (with a GeForce GTX 1080 Ti GPU card) is about 90 seconds. Out of the 100 trials, there are 98 trails that return a key candidate. The concrete statistic on the Hamming distance between the returned key candidate kg and the real round key rk, *i.e.*, $hw(kg, rk)$, is reported in Table 8. From Table 8, if an attack is considered as successful as long as $hw(kg, rk) \leqslant 3$, the success rate is 0.90.

Table 8: Statistic on $hw(kg, rk)$ over 100 trials of the 9-round attack on SPECK64

$hw(kg, rk)$	0	1	2	3	4	5	6	7
# trials	22	30	29	9	4	1	2	1

6 Theoretical Cryptanalysis of Large-State SPECK

6.1 Prepending Multiple-Round $\mathcal{CD}$s to the $\mathcal{ND}$s

The practical attacks can be extended for several rounds after prepending a longer classical differential ($\mathcal{CD}$) to each of the $\mathcal{ND}$s. The number of rounds of a $\mathcal{CD}$ that can be prepended to an $\mathcal{ND}$ is limited by the number of neutral bits of the $\mathcal{CD}$. The latter should satisfy the requirement for boosting weak signals from the distinguisher, while it decreases sharply with the increase of the former.

6.1.1 Find Neutral Bits of Low-Probability Differential Trails

For large-state SPECK, finding neutral bits of multiple-round $\mathcal{CD}$s is not as straightforward as for small-state SPECK, since the probability of $\mathcal{CD}$s of a moderate number of rounds can be very low. For low-probability differentials, it is hard for a random sampling method to generate a sufficiently large number of conforming pairs to filter out NBs. Hence, this part of the work produces a method to find neutral bits of low-probability differential trails.

For the key-recovery attacks covering as many rounds as possible, both (probabilistic) neutral bits (NBs) and simultaneous neutral bit-sets (SNBSs) [BGL+21] are considered. For convenience, we collectively refer to them as NBs. To find NBs of a given differential, one way is to obtain many conforming pairs, flip the candidate bit/bit-set and examine the influence on the conformity. Thus, an efficient way to generate many conforming pairs for a differential is needed. For this, we provide the following method.

Generating Conforming Pairs and Finding NBs for Low-Probability Differentials. When a bit is neutral for a differential trail in the setting where round-keys are independently randomly selected (free round-key setting), it is also neutral for the differential trail in the setting where round-keys are generated using a key schedule (real round-key setting). Thus, the following approach reduces the real round-key setting into the free round-key setting and generates conforming pairs for a given differential trail by adjusting randomly sampled round-keys. The obtained conforming pairs can then be used to filter out candidate NBs.

Firstly, we deduced conditions on conforming pairs for an XOR-difference propagating through addition modular 2^n, denoted by $\delta = (\alpha, \beta \mapsto \gamma)$.

Define the differential probability of XOR-difference through addition modular 2^n by $\mathrm{DP}^+(\alpha, \beta \mapsto \gamma) := \Pr_{(x,y)\in\mathbb{F}_2^n\times\mathbb{F}_2^n}[(x \boxplus y) \oplus ((x \oplus \alpha) \boxplus (y \oplus \beta)) = \gamma]$.

Observation 1. *Let $\delta = (\alpha, \beta \mapsto \gamma)$ be a possible XOR-differential through addition modulo 2^n ($\boxplus$). For (x, y) and $(x \oplus \alpha, y \oplus \beta)$ be a conforming pair of δ, x and y should satisfy the follows. For $0 \le i < n-1$, if $\mathtt{eq}(\alpha, \beta, \gamma)[i] = 0$*

$$\left.\begin{array}{ll} x[i] \oplus y[i] = \mathtt{xor}(\alpha, \beta, \gamma)[i+1] \oplus \alpha[i], & \text{if } \alpha[i] \oplus \beta[i] = 0, \\ \left.\begin{array}{ll} x[i] \oplus c[i] = \mathtt{xor}(\alpha, \beta, \gamma)[i+1] \oplus \alpha[i], & \text{if } \alpha[i] \oplus \mathtt{xor}(\alpha, \beta, \gamma)[i] = 0, \\ y[i] \oplus c[i] = \mathtt{xor}(\alpha, \beta, \gamma)[i+1] \oplus \beta[i], & \text{if } \alpha[i] \oplus \mathtt{xor}(\alpha, \beta, \gamma)[i] = 1, \end{array}\right\} & \text{if } \alpha[i] \oplus \beta[i] = 1, \end{array}\right\}$$

where $c[i]$ is the i-th carry bit, and the definitions of $\mathtt{eq}(\cdot,\cdot,\cdot)$ and $\mathtt{xor}(\cdot,\cdot,\cdot)$ can be found in Sect. 2.2.

Proof. See Appendix A.1. □

With Observation 1, given a differential trail of SPECK, we use Algorithm 1 to generate conforming pairs and filter candidate NBs. Since the round-keys are adjusted such that most conditions on conforming pairs are satisfied, the number of repeats from Step 1a to 1e is far less than $N/\mathrm{DP}^{+}(\delta)$. Therefore, Algorithm 1 is much more efficient than a purely random sampling method. As for the choice of N, preliminary experiments were performed using $N \in \{2^{10} \sim 2^{13}\}$. The obtained empirical neutrality by taking $N = 2^{10}$ do not differ obviously from that obtained by taking larger N. Thus, all presented results by applying this algorithm (in Table 11) are obtained by taking $N = 2^{10}$.

Algorithm 1 Generate conforming pairs and filter candidate neutral bit/bit-set for differential trails of SPECK

INPUT:

1. an r-round differential trail $\delta = (\delta x_0, \delta y_0) \mapsto (\delta x_1, \delta y_1) \mapsto \cdots \mapsto (\delta x_r, \delta y_r)$
2. a candidate neutral bit-set $I_s = [\mathtt{i}_1, \mathtt{i}_2, \cdots, \mathtt{i}_\mathtt{j}]$

OUTPUT: empirical neutrality p of I_s.

1. Generate N conforming pairs for δ as follows.
   (a) Randomly generate an input data (x_0, y_0).
   (b) On this input data, compute the i-th round function by selecting the $(i-1)$-th round-key. Concretely, during the i-th round-computation, generate the $(i-1)$-th round-key by firstly determining those bits that made the inputs to the i-th $\boxplus$ satisfy conditions listed in Observation 1, then randomly generate other bits. Some conditions that depend on carry bit cannot be fulfilled by simply adjusting one key-bit without affecting conditions on other bits. For such conditions, simply bypass them by randomly selecting the key-bits. Compute in this way through r-round and obtain output (x_r, y_r).
   (c) Use the adjusted round-keys to compute the r-round output on the input $(x_0 \oplus \delta x_0, y_0 \oplus \delta y_0)$ (denoted by (x'_0, y'_0)).
   (d) If the output (x'_r, y'_r) satisfy $x_r \oplus x'_r = \delta x_r$ and $y_r \oplus y'_r = \delta y_r$, store $((x_0, y_0), (x'_0, y'_0))$ and all round-keys as a conforming pair.
   (e) Repeat Step 1a until N conforming pairs are obtained.
2. For each conforming pair, flip bits in I_s, test whether the resulted pair is a conforming pair (encrypted under the round-keys associated with the original conforming pair). Count the number of cases that the conformity is kept, denote the number by cnt.
3. Return $p \leftarrow cnt/N$ as the empirical neutrality of I_s.

6.1.2 $\mathcal{CD}$s and NBs of the $\mathcal{CD}$s for Large-State SPECK

Applying Algorithm 1, we searched all NBs (simultaneously flipping at most 3 bits) for $\mathcal{CD}$s that can be prepended to the obtained $\mathcal{NCD}$s.

$\mathcal{CD}$s for Large State SPECK. For each version of SPECK, the $\mathcal{CD}$s to be prepended to the $\mathcal{NCD}$s can be seen as dominated by a single differential trail. Thus, we will use the dominant differential trails as the $\mathcal{CD}$s. Different differential trails prepended to different $\mathcal{NCD}$s can be obtained from the same base differential trail by rotating intermediate differences to the left (circular shift by word). The obtained trails are valid as long as the condition on the least

significant bit, *i.e.*, $\texttt{xor}(\alpha,\beta,\gamma)[0]=0$ is fulfilled by the input/output differences (α,β,γ) to all involved $\boxplus$'s on the trails (refer to Algorithm 2) [3]. The differential weight of each trail obtained by cyclically shifting the base differs from that of the base trail by at most t, where $t=\max_{i,0\le i<n}(\sum_{j=1}^{r}\neg\texttt{eq}(\alpha_j,\beta_j,\gamma_j)[i])$ and $(\alpha_j,\beta_j,\gamma_j)$ are the input/output differences through the j-th $\boxplus$ along the r-round differential trail. In our following cases, t is at most 3. The base trails exploited in the following theoretical attacks are listed in Table 9.

NBs of the $\mathcal{CD}$s. When restricting that the neutrality be no less than 0.7 for $\mathcal{CD}$s of Speck128 (resp. 0.8 for $\mathcal{CD}$s of Speck96 and Speck64), the number of independent NBs together with the differential weight of each rotated differential trail are listed in Table 10. The concrete exploitable NBs for the $\mathcal{CD}$s that will be prepended to the used $\mathcal{NCD}$s are listed in Table 11.

Similar to the differential trails, we found that most NBs of rotated trails can be obtained from the NBs of the base trail by a circular shift to the left (by word). For example, the bit-set [26, 37, 98] is an NB of the base differential trail listed in column 'Speck128' of Table 10 and denoted by $\mathcal{CD}_{[64]}$ in Table 11. For the differential trail obtained by rotating 53 bits to the left (denoted by $\mathcal{CD}_{[117]}$), the bit-set $[(26+53) \mod 64, (37+53) \mod 64, 64+((98+53) \mod 64)]$, *i.e.*, [15, 26, 87] is its NB.

In summary, there are more than 10 exploitable NBs for 6-round, 4-round, and 3-round differential trails of Speck128, Speck96, and Speck64, respectively. For longer differential trails of each version, the number of high neutrality NBs is no more than 3.

Table 9: Differential Trails used for Speck. The listed are base trails. Other trails can be obtained by rotating (by word) the base trails to the left. The notation (w $w_1 \sim w_2$) in the first row represents the range of the differential weights of the trails that can be obtained by rotating the base trails.

R	Speck128 (w 43 $\sim$ 46)			Speck96 (w 20 $\sim$ 22)			Speck64 (w 10 $\sim$ 12)		
	δx_i	δy_i	w_i	δx_i	δy_i	w_i	δx_i	δy_i	w_i
-6	4041041440401000	024040240640d010							
-5	0200012012009000	1002000020061080	14						
-4	1000000100141010	9010000000249410	10	100100141010	901000249410				
-3	8000000001248000	0080000000002084	9	800001248000	008000002084	9	81248000	00802084	
-2	0000000000010404	0400000000000024	6	000000010404	040000000024	6	00010404	04000024	6
-1	0000000000000120	2000000000000000	4	000000000120	200000000000	4	00000120	20000000	4
0	0000000000000001	0000000000000000	2	000000000001	000000000000	2	00000001	00000000	2

6.2 Theoretical Cryptanalysis of Extended Round-Reduced Speck128, Speck96, and Speck64

For Speck128/128, prepending the five 6-round $\mathcal{CD}$s (in Table 11) to the five 9-round $\mathcal{NCD}$s (in Table 4), one can obtain five $\mathcal{HD}$s covering 15 rounds, which can be denoted by $6r_{\mathcal{CD}}-\Delta_{[64]}-9r_{\mathcal{NCD}}$, $6r_{\mathcal{CD}}-\Delta_{[76]}-9r_{\mathcal{NCD}}$, $6r_{\mathcal{CD}}-\Delta_{[90]}-9r_{\mathcal{NCD}}$, $6r_{\mathcal{CD}}-\Delta_{[105]}-9r_{\mathcal{NCD}}$, $6r_{\mathcal{CD}}-\Delta_{[117]}-9r_{\mathcal{NCD}}$. Exploiting the 15-round $\mathcal{HD}$s, one can launch a multi-stage attack to recover the full last subkey of 17-round Speck128/128. Each stage of the 17-round attack composes of a freely invertible round at the top, a 15-round $\mathcal{HD}$ at the middle, and 1-round partial subkey guessing at the end (denoted by $1r+6r_{\mathcal{CD}}-\Delta_{[\cdot]}-9r_{\mathcal{NCD}}+1r$).

[3] There are differential trails with the same output and have a high probability (by a factor at most 2^2) than the used differential trails. However, those trails are impossible for some keys (weak-key differential trails). Hence, we employ the ones that have the highest differential probability among non-weakkey trails. Exploiting weak-key trails, the attack complexity can be slightly improved but only applies to a partial key space.

Table 10: The differential weights (denoted by w) and the numbers of independent NBs (denoted by #NB) of differential trails obtained by rotating (to the left by rol bits) the base trail in Table 9.

SPECK128 (#NB with $\Pr_{nb} > 0.7$)

rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB
0	**45**	**12**	8	45	12	16	46	13	24	46	13	32	46	13	40	46	12	48	44	12	56	44	12
1	46	12	9	45	12	17	45	13	25	46	13	33	46	13	**41**	**45**	**12**	49	45	12	57	46	12
2	45	12	10	46	12	18	46	12	**26**	**45**	**14**	34	45	12	42	45	12	50	45	12	58	45	12
3	44	12	11	45	12	19	46	12	27	46	13	35	45	12	43	45	12	51	43	12	59	43	12
4	46	12	**12**	**46**	**12**	20	46	12	28	46	13	36	46	12	44	46	12	52	46	12	60	46	12
5	45	12	13	46	12	21	46	12	29	45	12	37	45	12	45	44	12	**53**	**44**	**12**	61	44	12
6	45	12	14	45	13	22	46	12	30	46	13	38	45	12	46	45	12	54	46	12	62	46	12
7	46	12	15	46	15	23	45	14	31	45	13	39	45	12	47	45	12	55	45	12	63	45	12

SPECK96 (#NB with $\Pr_{nb} > 0.8$)

rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB
0	21	25	6	22	23	12	22	23	18	22	21	24	22	22	30	22	23	36	22	21	42	21	22
1	22	23	7	22	23	13	22	23	19	22	22	25	22	23	31	21	24	37	20	22	43	21	21
2	21	24	8	21	24	14	22	22	20	22	22	26	21	23	32	21	22	38	22	21	44	22	21
3	21	23	9	22	26	15	22	22	21	22	21	27	22	22	33	22	24	39	21	22	45	20	24
4	22	23	10	22	24	16	22	21	22	22	21	28	22	23	34	21	21	40	21	21	46	22	22
5	**21**	**23**	11	21	24	**17**	**22**	**22**	23	21	23	**29**	**21**	**22**	35	21	21	**41**	**22**	**21**	47	21	25

SPECK64 (#NB with $\Pr_{nb} > 0.8$)

rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB	rol	w	#NB
0	12	18	4	12	16	8	11	16	12	12	14	16	12	15	20	12	15	24	11	14	28	12	14
1	**12**	**16**	5	11	17	9	12	14	13	12	14	17	12	15	21	11	15	25	12	15	29	10	16
2	11	18	6	12	17	**10**	**12**	**14**	14	12	14	18	11	14	22	12	17	26	11	15	30	12	15
3	12	15	7	12	15	11	12	14	**15**	**11**	**16**	19	12	15	23	11	15	27	12	14	31	11	26

Table 11: The concrete NBs and their empirical probabilities for the $\mathcal{CD}$s to be prepended to the $\mathcal{ND}$s. The $\mathcal{CD}$s can be obtained by rotating by word (to the left by rol bits) the base trail in Table 9 (corresponding to the ones in bold in Table 10). For each $\mathcal{CD}$, its NBs can be obtained by rotating by word (to the left by rol bits) the NBs listed in the first row for each version of SPECK. Concretely, for SPECK2n, suppose $i < n$ and $j >= n$, then $[\mathtt{i},\mathtt{j}]^{\lll k}$ means $[(\mathtt{i}+\mathtt{k})\%\mathtt{n},\ \mathtt{n}+(\mathtt{j}+\mathtt{k})\%\mathtt{n}]$. The column titled $\Pr_{nb}$ shows the empirical probability that *all* 2^{10} pairs in a structure created from a conforming pair using the 10 NBs are conforming pairs. Each empirical probability in this table is obtained by performing 1024 independent trials.

SPECK128 (10 NBs for each of the five $\mathcal{CD}$s)

$\mathcal{CD}$	rol	$[26,37,98]^{\lll rol}$	$[34,106]^{\lll rol}$	$[41]^{\lll rol}$	$[42]^{\lll rol}$	$[43]^{\lll rol}$	$[112]^{\lll rol}$	$[113]^{\lll rol}$	$[114]^{\lll rol}$	$[115]^{\lll rol}$	$[116]^{\lll rol}$	$\Pr_{nb}$
$\mathcal{CD}_{[64]}$	0	1.000	0.998	0.947	0.901	0.818	1.000	0.981	0.954	0.924	0.852	0.567
$\mathcal{CD}_{[76]}$	12	0.999	1.000	0.970	0.947	0.840	0.999	0.979	0.964	0.929	0.873	0.649
$\mathcal{CD}_{[90]}$	26	0.994	1.000	0.935	0.873	0.778	0.995	0.977	0.932	0.883	0.833	0.502
$\mathcal{CD}_{[105]}$	41	0.995	0.997	0.951	0.872	0.785	0.994	0.979	0.951	0.917	0.836	0.516
$\mathcal{CD}_{[117]}$	53	0.996	0.998	0.936	0.896	0.768	0.996	0.975	0.944	0.896	0.820	0.512

SPECK96 (10 NBs for each of the four $\mathcal{CD}$s)

$\mathcal{CD}$	rol	$[10,21,66]^{\lll rol}$	$[25]^{\lll rol}$	$[26]^{\lll rol}$	$[27]^{\lll rol}$	$[28]^{\lll rol}$	$[80]^{\lll rol}$	$[81]^{\lll rol}$	$[82]^{\lll rol}$	$[83]^{\lll rol}$	$[84]^{\lll rol}$	$\Pr_{nb}$
$\mathcal{CD}_{[53]}$	5	1.000	1.000	0.999	0.996	0.998	1.000	1.000	0.999	0.999	0.999	0.993
$\mathcal{CD}_{[65]}$	17	1.000	1.000	0.999	0.999	0.993	1.000	0.999	1.000	1.000	1.000	0.990
$\mathcal{CD}_{[77]}$	29	1.000	1.000	0.999	0.997	0.994	1.000	1.000	0.999	0.997	0.999	0.987
$\mathcal{CD}_{[89]}$	41	1.000	0.998	0.997	0.998	0.994	1.000	1.000	1.000	0.997	0.995	0.987

SPECK64 (10 NBs for each of the three $\mathcal{CD}$s)

$\mathcal{CD}$	rol	$[7,47]^{\lll rol}$	$[13,53]^{\lll rol}$	$[17]^{\lll rol}$	$[18]^{\lll rol}$	$[19]^{\lll rol}$	$[56]^{\lll rol}$	$[57]^{\lll rol}$	$[58]^{\lll rol}$	$[59]^{\lll rol}$	$[60]^{\lll rol}$	$\Pr_{nb}$
$\mathcal{CD}_{[42]}$	10	1.000	1.000	0.998	0.992	0.967	1.000	0.999	0.998	0.996	0.998	0.951
$\mathcal{CD}_{[47]}$	15	1.000	1.000	0.983	0.971	0.938	1.000	0.990	0.982	0.971	0.948	0.854
$\mathcal{CD}_{[33]}$	1	1.000	1.000	0.983	0.981	0.969	1.000	0.991	0.982	0.975	0.955	0.887

The attack can be launched by directly applying the procedure described in Sect. 3.2, and adopting almost the same parameters as that in the five-stage procedure for practically attacking 12-round SPECK128 in Sect 5.1. Compared to the practical attack on 12-round that employs 1-round $\mathcal{CD}$s in Sect. 5.1, the attack employing 6-round $\mathcal{CD}$s requires more ciphertext structures in each stage, since the differential probabilities are much lower;

another difference is that the used NBs are probabilistic instead of complete ones. Thus, to obtain a ciphertext structure in which all pairs correspond to conforming pairs, slightly more data is required (by a factor of the reciprocal of the neutrality). Along the recovery of the last subkey, correct ciphertext structures are identified. After recovering the last subkey and peeling off the last round, a procedure employing the identified correct ciphertext structures and stronger 8-round $\mathcal{ND}$s can be launched to recover the second-to-the-last subkey.

Similarly, basing on the practical attacks on 10-round Speck96 and 9-round Speck64 in Sect. 5.2, one can directly devise theoretical attacks on 13-round Speck96 and 11-round Speck64 by simply replacing the 1-round $\mathcal{CD}$'s and their NBs in Table 5 with the longer ones in Table 11, at the same time increasing the number of ciphertext structures according to the differential probability and the neutrality of the NBs.

According to the complexity analysis in Sect. 3, the worst-case data and time complexity of the theoretical attacks can be directly computed with formulas 2 and 4. The results on the complexity analysis of the above theoretical attacks are listed in Table 12, where $\epsilon, N_i, \beta_i, |\mathcal{B}_i|$ are the same as the practical attacks in Sect. 5; The $\mathrm{Pr}_{\mathsf{nb}}$'s are additional factors compared to the complexity of practical attacks since the NBs are probabilistic. These $\mathrm{Pr}_{\mathsf{nb}}$'s can be found in the last column of Table 11; The factor η_i for computing the time complexity is the ratio between the time for an atomic operation in the attacks (refer to Sect. 3) and the time for encrypting one plaintext. These ratios are obtained by experiments with data processed in a batch of size 2^{10} and the timings are averaged over 2^{12} trials. The success rates of the attacks are expected to be the same as the corresponding practical attacks, where 17-round corresponds to 12-round attack on Speck128/128, 13-round corresponds to 10-round attack on Speck96/96, and 11-round corresponds to 9-round attack on Speck64/96, respectively.

The attack on 17-round Speck128/128 can be directly converted to attacks on 18-round Speck128/192 (resp. 19-round Speck128/256). To do that, one simply guesses the 18-round subkey (resp. 19- and 18-round subkeys) to peel off one round (resp. two rounds), and repeat the 17-round attack at most 2^{64} (resp. 2^{128}) times. It is expected that only for the correctly guessed last-round subkey (resp. last-round and second-to-the-last subkeys), the inner procedure has a return. A similar analysis applies to attacks on 13-round Speck96/96 (resp. 14-round Speck96/144) and 11-round Speck64/96 (resp. 12-round Speck64/96 and 13-round Speck64/128) (refer to Table 12 and Table 1).

6.3 Experimental Verification of 11-round Attack on Speck64

The theoretical analysis of the 11-round attack on reduced Speck64 was further experimentally verified. The recovery of the last subkey was performed in 55 trials on a server with a professional graphic card (Tesla V100-SXM2-32GB GPU). The average time consumption of the attack using a single CPU core and a GPU is about 19395 seconds. The average number of cipertext structures queried in the attack is 7891, which is $2^{23.95}$ chosen plaintexts.

Out of the 55 trials, there are 53 trails return a key candidate. The mean value of $hw(kg, rk)$ over the 53 trials is 2.0. The concrete statistic on $hw(kg, rk)$ is reported in Table 13. From Table 13, when we count an attack as successful as long as $hw(kg, rk) \leqslant 3$, the success rate is 0.87.

From the above, for Speck64, the experimental result on 11-round attack is largely consistent with its theoretical analysis in Table 12.

Table 12: Complexity of the theoretical attacks using multiple-round $\mathcal{CD}$s with the $\mathcal{ND}$s

Attack Stage$_i$	DC $\epsilon/(p_i \times \Pr_{\mathsf{nb}}) \times N_i$ pairs	#CP	TC $\beta_i \times 2^{\lvert\mathcal{B}_i\rvert} \times$ DC pairs $\times\eta_i$	#Enc	Ps.
17-round Speck128/128					
$1r + 6r_{\mathcal{CD}} - \Delta_{[64]} - 9r_{\mathcal{ND}} + 1r$	$4/(2^{-45} \times 0.567) \times 2^{10}$	$2^{58.82}$	$1 \times 2^{15} \times 2^{57.82} \times 10.69$	$2^{76.24}$	
$1r + 6r_{\mathcal{CD}} - \Delta_{[76]} - 9r_{\mathcal{ND}} + 1r$	$4/(2^{-46} \times 0.649) \times 2^{10}$	$2^{59.62}$	$3 \times 2^{12} \times 2^{58.62} \times 10.41$	$2^{75.58}$	
$1r + 6r_{\mathcal{CD}} - \Delta_{[90]} - 9r_{\mathcal{ND}} + 1r$	$4/(2^{-45} \times 0.502) \times 2^{10}$	$2^{58.99}$	$3 \times 2^{14} \times 2^{57.99} \times 10.77$	$2^{77.00}$	
$1r + 6r_{\mathcal{CD}} - \Delta_{[105]} - 9r_{\mathcal{ND}} + 1r$	$4/(2^{-45} \times 0.516) \times 2^{10}$	$2^{58.95}$	$3 \times 2^{15} \times 2^{57.95} \times 10.90$	$2^{77.98}$	
$1r + 6r_{\mathcal{CD}} - \Delta_{[117]} - 9r_{\mathcal{ND}} + 1r$	$4/(2^{-44} \times 0.512) \times 2^{10}$	$2^{57.97}$	$3 \times 2^{8} \times 2^{56.97} \times 9.74$	$2^{69.84}$	
$\sum_i$ #CP	-	$\mathbf{2^{61.28}}$	$\sum_i$ #Enc	$\mathbf{2^{78.98}}$	**0.52**
13-round Speck96/96					
Attack Stage$_i$	DC $\epsilon/(p_i \times \Pr_{\mathsf{nb}}) \times N_i$ pairs	#CP	TC $\beta_i \times 2^{\lvert\mathcal{B}_i\rvert} \times$ DC pairs $\times\eta_i$	#Enc	Ps.
$1r + 4r_{\mathcal{CD}} - \Delta_{[53]} - 7r_{\mathcal{ND}} + 1r$	$4/(2^{-21} \times 0.993) \times 2^{10}$	$2^{34.01}$	$1 \times 2^{12} \times 2^{33.01} \times 12.00$	$2^{48.59}$	
$1r + 4r_{\mathcal{CD}} - \Delta_{[65]} - 7r_{\mathcal{ND}} + 1r$	$4/(2^{-22} \times 0.990) \times 2^{10}$	$2^{35.01}$	$3 \times 2^{12} \times 2^{34.01} \times 11.89$	$2^{51.17}$	
$1r + 4r_{\mathcal{CD}} - \Delta_{[77]} - 7r_{\mathcal{ND}} + 1r$	$4/(2^{-21} \times 0.987) \times 2^{10}$	$2^{34.02}$	$3 \times 2^{12} \times 2^{33.02} \times 12.12$	$2^{50.20}$	
$1r + 4r_{\mathcal{CD}} - \Delta_{[89]} - 7r_{\mathcal{ND}} + 1r$	$4/(2^{-22} \times 0.987) \times 2^{10}$	$2^{35.02}$	$3 \times 2^{12} \times 2^{34.02} \times 12.30$	$2^{51.23}$	
$\sum_i$ #CP	-	$\mathbf{2^{36.60}}$	$\sum_i$ #Enc	$\mathbf{2^{52.61}}$	**0.81**
11-round Speck64/96					
Attack Stage$_i$	DC $\epsilon/(p_i \times \Pr_{\mathsf{nb}}) \times N_i$ pairs	#CP	TC $\beta_i \times 2^{\lvert\mathcal{B}_i\rvert} \times$ DC pairs $\times\eta_i$	#Enc	Ps.
$1r + 3r_{\mathcal{CD}} - \Delta_{[42]} - 6r_{\mathcal{ND}} + 1r$	$4/(2^{-12} \times 0.951) \times 2^{10}$	$2^{25.07}$	$1 \times 2^{10} \times 2^{24.07} \times 11.93$	$2^{37.65}$	
$1r + 3r_{\mathcal{CD}} - \Delta_{[47]} - 6r_{\mathcal{ND}} + 1r$	$4/(2^{-11} \times 0.854) \times 2^{10}$	$2^{24.23}$	$3 \times 2^{12} \times 2^{23.23} \times 12.40$	$2^{40.45}$	
$1r + 3r_{\mathcal{CD}} - \Delta_{[33]} - 6r_{\mathcal{ND}} + 1r$	$4/(2^{-12} \times 0.887) \times 2^{10}$	$2^{25.17}$	$3 \times 2^{10} \times 2^{24.17} \times 12.00$	$2^{39.34}$	
$\sum_i$ #CP	-	$\mathbf{2^{26.47}}$	$\sum_i$ #Enc	$\mathbf{2^{41.13}}$	**0.90**

DC: data complexity TC: time complexity Ps: success rate
#CP: number of chosen plaintexts #Enc: number of equivalent encryptions under the targeted cipher.

Table 13: Statistic on $hw(kg, rk)$ over 55 trials of the 11-round attack on Speck64

$hw(kg, rk)$	0	1	2	3	4
# trials	4	16	12	16	5

7 Application to More Ciphers

To demonstrate the generality of the multi-stage key-recovery framework, we present combinations of neural distinguishers that are facilitated to be employed to perform key-recovery attacks on three other block ciphers, including Simon [BSS+15], Present [BKL+07], and DES [BS92]. Table 14 shows one combination of six $\mathcal{ND}$s against 16-round reduced Simon128, one combination of four $\mathcal{ND}$s against 5-round reduced Present, and one combination of three $\mathcal{ND}$s against 5-round reduced DES. Note that multi-round $\mathcal{CD}$s are to be prepended to these $\mathcal{ND}$s for the attacks to cover more rounds.

8 Summary and Conclusions

This paper presents a multi-stage key-recovery framework for attacking large-stage block ciphers basing on differential neural distinguishers. To apply this framework on cryptanalysis of large-stage members of the block cipher family Speck, multiple neural distinguishers were trained and carefully selected. Under the proposed multi-stage key-recovery framework and with the combinations of neural distinguishers, various practical attacks were designed and trialed on round-reduced Speck with 128-bit, 96-bit, and 64-bit states. Basing on the practical attacks, theoretical attacks covering more rounds were devised, for which classical differentials and their neutral bits are searched to cooperate with the neural distinguishers. As a result, considerable improvement in terms of both time and

Table 14: Neural distinguishers against round-reduced SIMON, PRESENT, and DES.

	16-round SIMON128			5-round PRESENT			5-round DES		
$\mathcal{ND}_i$	Δ_i	$\mathcal{C}_i$	Acc.	Δ_i	$\mathcal{S}_i$	Acc.	Δ_i	$\mathcal{S}_i$	Acc.
$\mathcal{ND}_1$	$\Delta_{[8]}$	$\{9\sim0\}$	0.669	$\Delta_{[9]}$	$\{2,4,7,10,12,15\}$	0.832	$\Delta_{[41]}$	$\{1\}$	0.669
$\mathcal{ND}_2$	$\Delta_{[21]}$	$\{22\sim10\}$	0.717	$\Delta_{[22]}$	$\{11\}$	0.838	$\Delta_{[45]}$	$\{2,4,7\}$	0.717
$\mathcal{ND}_3$	$\Delta_{[34]}$	$\{35\sim23\}$	0.717	$\Delta_{[25]}$	$\{0,1,3,8,9\}$	0.839	$\Delta_{[62]}$	$\{3,5,6,8\}$	0.717
$\mathcal{ND}_4$	$\Delta_{[47]}$	$\{48\sim36\}$	0.717	$\Delta_{[34]}$	$\{5,6,13,14\}$	0.800			
$\mathcal{ND}_5$	$\Delta_{[54]}$	$\{55\sim49\}$	0.718						
$\mathcal{ND}_5$	$\Delta_{[62]}$	$\{63\sim56\}$	0.711						

Δ_i: the input difference Acc.: the accuracy of the $\mathcal{ND}_i$

$\mathcal{C}_i$: the index of bits of x_r (at the same time, take the same index of bits of y_r) that are fed into $\mathcal{ND}_i$, where x_r (resp. y_r) is the left (resp. right) n-bit word of a full r-round output state.

$\mathcal{S}_i$: the index of S-boxes that are fed into $\mathcal{ND}_i$.

data complexity of differential key-recovery attacks on 17-round (resp. 18- and 19-round) reduced SPECK with the largest, *i.e.*, 128-bit state and 128-bit key (resp. 192- and 256-bit key) is obtained. Besides, efficient attacks are achieved on round-reduced SPECK with 96-bit and 64-bit states.

A Appendix

A.1 Proof of Observation 1

Property 1 (Basic property of addition modulo 2^n [LM01])**.** If $(x, y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, then $x \boxplus y = x \oplus y \oplus \mathtt{carry}(x, y)$, where the carry, $\mathtt{carry}(x, y) := c \in \mathbb{F}_2^n$, of addition $x \boxplus y$ is defined recursively as follows. Firstly, $c[0] := 0$. Secondly, $c[i+1] := (x[i] \wedge y[i]) \oplus (x[i] \wedge c[i]) \oplus (y[i] \wedge c[i])$, for every $i \geq 0$.

Lemma 1 ([LM01])**.** $\mathrm{DP}^+(\alpha, \beta \mapsto \gamma) = \Pr_{(x,y)\in\mathbb{F}_2^n\times\mathbb{F}_2^n}[\mathtt{carry}(x, y)\oplus\mathtt{carry}(x\oplus\alpha, y\oplus\beta) = \mathtt{xor}(\alpha, \beta, \gamma)]$*, where* $\mathtt{xor}(x, y, z) := x \oplus y \oplus z$*.*

Theorem 1 ([LM01])**.** *Let* $\delta = (\alpha, \beta \mapsto \gamma)$ *be an arbitrary XOR-differential through addition modulo* 2^n*. Algorithm 2 returns* $\mathrm{DP}^+(\delta)$ *in time* $\Theta(\log n)$*. More precisely, it works in time* $\Theta(1) + t$*, where* t *is the time it takes to compute* $\mathtt{w_h}$*.*

Algorithm 2 Compute $\mathrm{DP}^+(\delta)$ [LM01]

INPUT: $\delta = (\alpha, \beta \mapsto \gamma)$
OUTPUT: $\mathrm{DP}^+(\delta)$

1. If $\mathtt{eq}(\alpha^{\ll 1}, \beta^{\ll 1}, \gamma^{\ll 1}) \wedge (\mathtt{xor}(\alpha, \beta, \gamma) \oplus (\beta^{\ll 1})) \neq 0$ then return 0;
2. Return $2^{-\mathtt{w_h}(\neg\mathtt{eq}(\alpha,\beta,\gamma)\wedge\mathtt{mask}(n-1))}$;

Proof of Observation 1. Denote $\mathtt{carry}(x, y) \oplus \mathtt{carry}(x \oplus \alpha, y \oplus \beta)$ by Δc. From Lemma 1, for an arbitrary conforming pair, (x, y) and $(x \oplus \alpha, y \oplus \beta)$, of a possible differential

$\delta := (\alpha, \beta \mapsto \lambda)$, one have

$$\begin{array}{l}
\mathtt{xor}(\alpha,\beta,\gamma)[i+1] \\
= \Delta c[i+1] \\
= (x[i]y[i] \oplus x[i]c[i] \oplus y[i]c[i]) \ \oplus \\
\quad ((x[i] \oplus \alpha[i])(y[i] \oplus \beta[i]) \oplus (x[i] \oplus \alpha[i])(c[i] \oplus \Delta c[i]) \oplus (y[i] \oplus \beta[i])(c[i] \oplus \Delta c[i])) \\
= x[i]\beta[i] \oplus y[i]\alpha[i] \oplus \alpha[i]\beta[i] \oplus (x[i] \oplus y[i] \oplus \alpha[i] \oplus \beta[i])\Delta c[i] \oplus (\alpha[i] \oplus \beta[i])c[i]
\end{array} .$$

Since δ is a possible differential, from Theorem 1, one only need to consider those bits i, $0 \le i < n-1$, such that $\mathtt{eq}(\alpha,\beta,\gamma)[i] = 0$. For those bits i, one has the follows.
If $\alpha[i] \oplus \beta[i] = 0$, then $\alpha[i] = \beta[i] \neq \lambda[i] = \Delta c[i]$ (note that $\mathtt{xor}(\alpha[i], \beta[i], \lambda[i]) = \Delta c[i]$). Thus, $\alpha[i]\beta[i] = 0$ and $\alpha[i] \oplus \Delta c[i] = 1$. One has

$$\begin{array}{ll}
\mathtt{xor}(\alpha,\beta,\gamma)[i+1] & = (x[i] \oplus y[i])(\alpha[i] \oplus \Delta c[i]) \oplus \alpha[i], \text{ thus} \\
x[i] \oplus y[i] & = \mathtt{xor}(\alpha,\beta,\gamma)[i+1] \oplus \alpha[i];
\end{array}$$

if $\alpha[i] \oplus \beta[i] = 1$, one has

$$\begin{array}{l}
\mathtt{xor}(\alpha,\beta,\gamma)[i+1] = (x[i] \oplus y[i])(\alpha[i] \oplus \Delta c[i]) \oplus x[i] \oplus \Delta c[i] \oplus c[i], \text{ thus} \\
\left.\begin{array}{ll}
x[i] \oplus c[i] = \mathtt{xor}(\alpha,\beta,\gamma)[i+1] \oplus \alpha[i], & \text{if } \alpha[i] \oplus \mathtt{xor}(\alpha,\beta,\gamma)[i] = 0, \\
y[i] \oplus c[i] = \mathtt{xor}(\alpha,\beta,\gamma)[i+1] \oplus \beta[i], & \text{if } \alpha[i] \oplus \mathtt{xor}(\alpha,\beta,\gamma)[i] = 1.
\end{array}\right\}
\end{array}$$

□

References

[BBCD21] Anubhab Baksi, Jakub Breier, Yi Chen, and Xiaoyang Dong. Machine learning assisted differential distinguishers for lightweight ciphers. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*, pages 176–181. IEEE, 2021.

[BC04] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[BGL+21] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Conditional differential-neural cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 719, 2021.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BS92] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 175:1–175:6. ACM, 2015.

[C+15] Francois Chollet et al. Keras, 2015. https://github.com/fchollet/keras.

[CY20] Yi Chen and Hongbo Yu. Neural aided statistical attack for cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 1620, 2020.

[CY21] Yi Chen and Hongbo Yu. A new neural distinguisher model considering features derived from multiple ciphertext pairs. *IACR Cryptol. ePrint Arch.*, page 310, 2021.

[Din14] Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

[Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[LM01] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. *IACR Cryptol. ePrint Arch.*, page 209, 2016.