More Insight on Deep Learning-aided Cryptanalysis

Zhenzhen Bao[1,6]✉, Jinyu Lu[2,3]✉, Yiran Yao[3]✉, and Liu Zhang[4,5,3]✉

[1] Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China zzbao@tsinghua.edu.cn

[2] College of Sciences, National University of Defense Technology, Hunan, Changsha 410073, China jinyu_smile@foxmail.com

[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore yiran005@e.ntu.edu.sg

[4] School of Cyber Engineering, Xidian University, Xi'an, China liuzhang@stu.xidian.edu.cn

[5] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China

[6] Zhongguancun Laboratory, Beijing, China

Abstract. In CRYPTO 2019, Gohr showed that well-trained neural networks could perform cryptanalytic distinguishing tasks superior to differential distribution table (DDT)-based distinguishers. This suggests that the differential-neural distinguisher ($\mathcal{ND}$) may use additional information besides pure ciphertext differences. However, the explicit knowledge beyond differential distribution is still unclear. In this work, we provide explicit rules that can be used alongside DDTs to enhance the effectiveness of distinguishers compared to pure DDT-based distinguishers. These rules are based on strong correlations between bit values in right pairs of XOR-differential propagation through addition modulo 2^n. Interestingly, they can be closely linked to the earlier study of the multi-bit constraints and the recent study of the fixed-key differential probability. In contrast, combining these rules does not improve the $\mathcal{ND}$s' performance. This suggests that these rules or their equivalent form have already been exploited by $\mathcal{ND}$s, highlighting the power of neural networks in cryptanalysis.

In addition, we find that to enhance the differential-neural distinguisher's accuracy and the number of rounds, regulating the differential propagation is imperative. Introducing differences into the keys is typically believed to help eliminate differences in encryption states, resulting in stronger differential propagations. However, differential-neural attacks differ from traditional ones as they don't specify output differences or follow a single differential trail. This questions the usefulness of introducing differences in a key in differential-neural attacks and the resistance of Speck against such attacks in the related-key setting. This work shows that the power of differential-neural cryptanalysis in the related-key setting can exceed that in the single-key setting by successfully conducting a 14-round key recovery attack on Speck32/64.

Keywords: Neural Network · Interpretability · Modular Addition · Related-key · Speck

1 Introduction

In 2019, Gohr [14] proposed differential-neural cryptanalysis, employing neural networks as superior distinguishers and exploiting them to perform efficient key recovery attacks. Impressively, the differential-neural distinguisher ($\mathcal{ND}$) outperformed the traditional pure differential distinguishers using full differential distribution tables (DDT). However, interpreting these neural network-based distinguishers remains challenging, hindering the comprehension of the additional knowledge learned by differential-neural distinguishers.

Despite the intricate nature of neural network interpretability, researchers have made primary progress in understanding the differential-neural distinguisher's inner workings. In EUROCRYPT 2021, Benamira *et al.* [6] proposed that Gohr's neural distinguisher effectively approximates the cipher's DDT during the learning phase. Moreover, the distinguisher relies on both the differential distribution of ciphertext pairs and that of the penultimate and antepenultimate rounds. Yet, the specific form of additional information remains undisclosed.

In AICrypt 2023, Gohr *et al.* [16] proved the differential-neural distinguisher for Simon32/64 can use only differential features and achieve accuracy same as pure differential ones. Applying the same neural network to both Speck and Simon yields different conclusions: neural networks learned or did not learn features beyond full DDT. These intriguing findings motivate us to delve deeper into the neural network's mechanisms, aiming to comprehend the specific features underpinning its conclusions for each cipher and to improve and exploit further the neural distinguishers should additional features be captured.

Our Contributions. In this work, we conclude that $\mathcal{ND}$s' advantage over pure DDT-based distinguishers is in exploiting the differential distribution under the partially known value input to the last non-linear operation. Specifically, $\mathcal{ND}$s exploit the correlation between the ciphertexts' partial value, ciphertext pair's differences, and intermediate states' differences. Furthermore, our work shows that differential-neural cryptanalysis in the related-key ($\mathcal{RK}$) setting can attack more rounds than in the single-key setting, which was not apparent before. The concrete contributions include the following.

- **Improving full DDT-based distinguisher.** We observe that, apart from the information of differences, one knows the patrial value of inputs, denoted by y, to the last modular addition of Speck, leveraging by which one can improve DDT-based distinguishers. We show that the differential probability conditioned on a fixed value of y can differ from the average differential probability over all possible y. This insight enables more accurate classification based on the ciphertext pair's differences and the ciphertexts' partial value. The high-level idea is to consider conditional probabilities and specific cases where the fulfillment of the differential constraints can be predicted based on the value of y. The results indicate that it is highly likely that $\mathcal{ND}$s rely on these specific cases to outperform pure DDT-based distinguishers.
- **Optimizing the performance and training process of $\mathcal{ND}$s.** Addressing the challenge of training high-round, especially 8-round, $\mathcal{ND}$ of Speck32/

Table 1: Summary of key recovery attacks on SPECK32/64

#R	Distinguisher	Configure	Time	Data	Succ. Rate	Key Space	Advantage	Ref.
13	$\mathcal{DD}$	1+8+4	2^{57}	2^{25}	-	2^{64}	2^{7}	[11]
	$\mathcal{DD}$	1+8+3	$2^{55.58}$	$2^{24.26}$	-	2^{64}	$2^{8.42}$	[13]
	$\mathcal{DD}$	1+8+2+2	$2^{50.16}$	$2^{31.13}$	63%	2^{64}	$2^{13.84}$	[8]
	$\mathcal{ND}$	1+3+8+1	$2^{50.17}$	2^{29}	82%	2^{63}	$2^{13.83}$	[3]
	$\mathcal{ND}$	1+3+8+1	$2^{44.36}$	2^{27}	21%	2^{63}	$2^{18.64}$	[28]
	$\mathcal{RK}$-$\mathcal{ND}$	1+2+9+1	$2^{34.57}$	2^{16}	54.29%	2^{50}	$2^{15.43}$	Sect. 5.2
	$\mathcal{RK}$-$\mathcal{ND}$	1+2+9+1	$2^{31.79}$	2^{10}	43.33%	2^{46}	$2^{14.21}$	Sect. 5.2
14	$\mathcal{DD}$	1+9+4	2^{63}	2^{31}	-	2^{64}	2^{1}	[11]
	$\mathcal{DD}$	1+9+4	$2^{62.47}$	$2^{30.47}$	-	2^{64}	$2^{1.53}$	[25]
	$\mathcal{DD}$	1+9+2+2	$2^{60.99}$	$2^{31.75}$	63%	2^{64}	$2^{3.01}$	[8]
	$\mathcal{DD}$	2+9+3	$2^{60.58}$	$2^{30.26}$	76.00%	2^{64}	$2^{3.42}$	[13]
	$\mathcal{ND}$	1+3+8+2	$2^{60.36}$	2^{27}	21%	2^{63}	$2^{2.64}$	[28]
	$\mathcal{RK}$-$\mathcal{ND}$	1+3+9+1	$\mathbf{2^{35.59}}$	$\mathbf{2^{16}}$	75.71%	2^{42}	$\mathbf{2^{6.41}}$	Sect. 5.2
	$\mathcal{RK}$-$\mathcal{ND}$	1+3+9+1	$\mathbf{2^{35.78}}$	$\mathbf{2^{15}}$	71.43%	2^{41}	$\mathbf{2^{5.22}}$	Sect. 5.2
15	$\mathcal{DD}$	1+10+4	$2^{63.39}$	$2^{30.39}$	-	2^{64}	$2^{0.61}$	[18]
	$\mathcal{DD}$	1+10+2+2	$2^{62.25}$	$2^{30.39}$	-	2^{64}	$2^{1.75}$	[8]

−: Not available; "Advantage" denotes the time complexity advantage over a brute force attack.

64, we introduce the Freezing Layer Method. By freezing all convolutional layers in a pre-trained 7-round $\mathcal{ND}$, we efficiently train an 8-round $\mathcal{ND}$ using simple basic training with unaltered hyperparameters. This method matches Gohr's accuracy but cuts training time and data.

- **Exploring differential-neural attacks in the related-key setting.** The conclusion that $\mathcal{ND}$s can efficiently capture features beyond full DDT encourages further exploration of $\mathcal{ND}$-based attacks. We observed that control over the differential propagation is vital for achieving effective high-round $\mathcal{ND}$s. Hence, we introduce related-key ($\mathcal{RK}$) differences to slow down the diffusion of differences, aiding in training $\mathcal{ND}$ for higher rounds. As a result, we achieve a 14-round key recovery attack on SPECK32/64 using related-key neural distinguishers ($\mathcal{RK}$-$\mathcal{ND}$s). Results are in Table 1. Furthermore, we constructed various distinguishers under various $\mathcal{RK}$ differential trails and conducted comprehensive comparisons, reinforcing $\mathcal{ND}$ explainability.

Organization. The paper's structure is as follows: Sect. 2 provides preliminaries. Sect. 3 provides insights on the $\mathcal{ND}$ explainability. Sect. 4 provides enhancements on the $\mathcal{ND}$ training. Sect. 5 details of related-key differential-neural cryptanalysis. The conclusion is presented in Sect. 6.

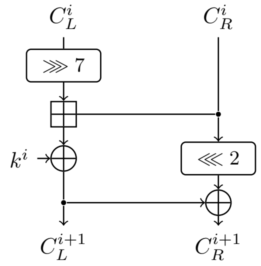
2 Preliminary

2.1 Notations

Denote by $C = (C_{n-1}, \ldots, C_0)$ the binary vector of n bits, , where C_i is the bit at position i and C_0 is the least significant. Define n as the word size in bits and $2n$ as the state size. Let (C_L^r, C_R^r) represent left and right state branches after r rounds, and k^r the r-round subkey. Bitwise XOR is denoted by $\oplus$, addition modulo 2^n by $\boxplus$, bitwise AND by $\odot$, and bitwise right/left rotation by $\ggg$ / $\lll$.

2.2 Brief Description of Speck32/64

In 2013, the National Security Agency (NSA) proposed Speck and Simon block ciphers, aiming to ensure security on resource-constrained devices [4]. By 2018, both ciphers were standardized by ISO/IEC for air interface communication. The Speck cipher uses a Feistel-like ARX design, emanating its non-linearity from modular addition and leveraging XOR and rotation for linear mixing. Speck32/64 is the smallest Speck variant [4]. Its round function, one of 22 rounds, takes a 16-bit subkey k^i and a state of two 16-bit words, (C_L^i, C_R^i). Its key schedule reuses the round function to generate round keys. With K as a master key and k^i the i-th round key, $K = (l^2, l^1, l^0, k^0)$. The round function's details are in Fig. 1.



$$C_L^{i+1} = ((C_L^i \ggg 7) \boxplus C_R^i) \oplus k^i$$
$$C_R^{i+1} = (C_R^i \lll 2) \oplus C_L^{i+1}$$

$$l^{i+3} = ((l^i \ggg 7) \boxplus k^i) \oplus i$$
$$k^{i+1} = (k^i \lll 2) \oplus l^{i+3}$$

Fig. 1: The round function and key schedule algorithm of Speck32/64

2.3 Overview of Differential-Neural Cryptanalysis

The differential-neural distinguisher operates as a supervised model, distinguishing whether ciphertext pairs originate from plaintext pairs with a defined input difference or from random pairs. Given m plaintext pairs $\{(P_j, P'_j), j \in [0, m-1]\}$, the corresponding ciphertext pairs $\{(C_j, C'_j), j \in [0, m-1]\}$ constitute a sample (In [14], $m = 1$). Each training sample is associated with a label Y defined as:

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

The $\mathcal{ND}$ architecture from [14] uses the prevalent ResNet. It comprises an initial input block, several residual blocks, and a prediction output layer.

In [14], three training schemes are proposed: a) Basic training for short-round distinguishers. b) An enhanced method using the KeyAveraging simulation and an $(r-1)$-round distinguisher, achieving the optimal 7-round $\mathcal{ND}$ for Speck. c) A staged training approach evolving a pre-trained $(r-1)$-round distinguisher to an r-round one in stages, yielding the most extended $\mathcal{ND}$ on Speck, covering 8 rounds. In [14], Gohr also showed how to combine a neural distinguisher with a classical differential and use a Bayesian-optimized key-guessing strategy for key recovery. Later, in [16], the authors provide general guidelines for optimizing Gohr's neural network and diverse optimization approaches across different ciphers, highlighting its efficacy and versatility. The authors also clarify which kind of ciphers the neural network can't learn beyond differential features.

3 Explicitly Explain Knowledge Beyond Full DDT

Studies show differential-based neural distinguishers often outperform DDT-based ones in certain ciphers [3, 14, 16]. However, what specific knowledge these neural distinguishers learn beyond DDT remains elusive. Prior research suggests that these distinguishers rely on differential distributions in the last two rounds and differential-linear (DL) properties [6, 10]. In [14], a "Real Differences Experiment" was conducted to observe how well neural networks could detect real differences beyond DDT. The experiment used randomized ciphertext pairs with a blinding value R introduced to obscure information beyond the difference. Results showed that neural networks could detect real differences without explicit training, and ciphertext pairs have non-uniform distributions within their difference equivalence classes. But, using blinding values in the form $R = aa$ (with a as any 16-bit word), the distinguishers failed (henceforth referred to as Gohr's $aaaa$-blinding experiment). This underlines that the neural distinguishers aren't exploiting the key schedule, and they can make finer distinctions than mere difference equivalence classes. These insights are crucial to explicitly explaining $\mathcal{ND}$'s superior classification mechanism. Based on these studies, this section takes a further step towards fully interpreting the knowledge that an $\mathcal{ND}$ has captured beyond full differential distribution.

We'll initiate by locating the root of the performance improvement, then deduce the specific pattern that causes the improvement, and finally use this pattern to improve the pure DDT-based distinguisher.

3.1 Locating Information used by $\mathcal{ND}$s of Speck Beyond DDT

In the following, we start with a generalized definition of information that the differential-neural distinguisher might use.

Generalized Definition of XOR Information. In Gohr's differential-neural distinguishers, given Speck's Feistel-like structure, samples are split into four

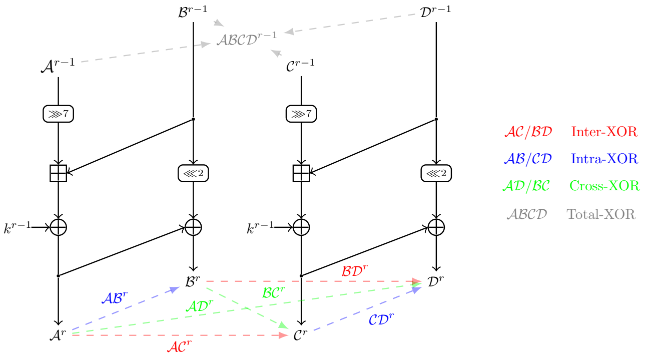


Fig. 2: Definition of XOR information

Table 2: Experimental results detailing the information harnessed by $\mathcal{ND}$s. Each set comprises both positive and negative samples. The notation $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ denotes ciphertext pairs derived from plaintext pairs with an input difference of `(0040,0000)`, while *Random* signifies pairs generated from random values. $\mathcal{R}_1$ refers to a random value.

Set.	Positive Samples	Negative Samples	Acc.
1-1	$(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$	*Random*	0.7906
1-2	$(\mathcal{AR}_1, \mathcal{BR}_1, \mathcal{CR}_1, \mathcal{DR}_1)$	*Random*	0.7911
1-3	$(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$	$(\mathcal{AR}_1, \mathcal{BR}_1, \mathcal{CR}_1, \mathcal{DR}_1)$	*Fail*

words: $\mathcal{A}, \mathcal{B}$ (forming the first ciphertext) and $\mathcal{C}, \mathcal{D}$ (forming the second), as depicted in Fig. 2. In subsequent discussions, a symbol's superscript denotes the number of encryption rounds. The absence of a superscript implies r rounds.

Traditional differential distinguishers focus solely on the difference of ciphertext pairs. Yet, as indicated in prior research [12, 23, 29], internal differentials can also be pivotal in cryptanalytic tasks.

We broaden the focus to include the XOR interactions among $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$. For brevity, XOR combinations like $\mathcal{A} \oplus \mathcal{B} \oplus \mathcal{C} \oplus \mathcal{D}$ are shortened to $\mathcal{ABCD}$. In other words, beyond the traditionally focused differences like $\mathcal{AC}$ and $\mathcal{BD}$, we explore under-emphasized XORs such as $\mathcal{AB}$, $\mathcal{CD}$, $\mathcal{AD}$, $\mathcal{BC}$, and $\mathcal{ABCD}$. For clarity, we classify these XORs as: **Inter-XOR** ($\mathcal{AC}$, $\mathcal{BD}$), **Intra-XOR** ($\mathcal{AB}$, $\mathcal{CD}$), **Cross-XOR** ($\mathcal{AD}$, $\mathcal{BC}$), and **Total-XOR** ($\mathcal{ABCD}$).

In Speck, Intra-XOR and Total-XOR relate to values and differences from the prior round. Specifically, Intra-XOR helps deduce the right-half values, and Total-XOR deduces the right-half differences of the preceding round.

Is XOR Information the Sole Basis for Differential-Neural Distinguisher's Decision Making? Using a mechanical method to determine relations between information sets, it became evident that focusing solely on specified XOR information is natural for finding the source of the information that $\mathcal{ND}$s exploit beyond the difference information.

Determine Relations Between Information Sets Mechanically. Consider a pair of ciphertexts from a round-reduced Speck, denoted as $C_0 = (C_{0L}, C_{0R})$ and $C_1 = (C_{1L}, C_{1R})$. Each ciphertext splits into two parts, with $C_{iJ} \in \mathbb{F}_2^b$ for $i \in \{0,1\}$ and $J \in \{L, R\}$. For Speck32/64, $b = 16$. Let K be the last round key, with $K \in \mathbb{F}_2^b$. For each C_i, let M_{iL} and M_{iR} represent the state value immediately preceding the XOR with key K and before the XOR between the left and right branches for $i \in \{0,1\}$. That is

$$C_{0L} = M_{0L} \oplus K,\ C_{0R} = M_{0L} \oplus M_{0R} \oplus K,\ C_{1L} = M_{1L} \oplus K,\ C_{1R} = M_{1L} \oplus M_{1R} \oplus K$$

The method to determine relations between information sets can be outlined in the following steps: Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two random values in $\mathbb{F}_2^b$.

1. **Setup**:
   (a) Set up a vector space $\mathcal{V}$ over the field $\mathbb{F}_2$ with dimension 7.
   (b) Define various basis vectors for $\mathcal{V}$, acting as linear masks whose non-zero bits indicate the variable selection from the following vector $[M_{0L}, M_{0R}, M_{1L}, M_{1R}, K, \mathcal{R}_1, \mathcal{R}_2]$. Concretely,

$$\begin{aligned} \Gamma_{M_{0L}} &= \texttt{[1,0,0,0,0,0,0]} & \Gamma_{M_{1L}} &= \texttt{[0,0,1,0,0,0,0]} \\ \Gamma_{M_{0R}} &= \texttt{[0,1,0,0,0,0,0]} & \Gamma_{M_{1R}} &= \texttt{[0,0,0,1,0,0,0]} \\ \Gamma_{K} &= \texttt{[0,0,0,0,1,0,0]} & \Gamma_{\mathcal{R}_1} &= \texttt{[0,0,0,0,0,1,0]} \\ & & \Gamma_{\mathcal{R}_2} &= \texttt{[0,0,0,0,0,0,1]} \end{aligned}$$

   Accordingly, $[C_{0L}, C_{0R}, C_{1L}, C_{1R}]$ can be obtained using the following masks:

$$\begin{aligned} \Gamma_{C_{0L}} &:= \Gamma_{\mathcal{A}} = \Gamma_{M_{0L}} \oplus \Gamma_K &&= \texttt{[1,0,0,0,1,0,0]}, \\ \Gamma_{C_{0R}} &:= \Gamma_{\mathcal{B}} = \Gamma_{M_{0L}} \oplus \Gamma_{M_{0R}} \oplus \Gamma_K &&= \texttt{[1,1,0,0,1,0,0]}, \\ \Gamma_{C_{1L}} &:= \Gamma_{\mathcal{C}} = \Gamma_{M_{1L}} \oplus \Gamma_K &&= \texttt{[0,0,1,0,1,0,0]}, \\ \Gamma_{C_{1R}} &:= \Gamma_{\mathcal{D}} = \Gamma_{M_{1L}} \oplus \Gamma_{M_{1R}} \oplus \Gamma_K &&= \texttt{[0,0,1,1,1,0,0]}. \end{aligned}$$

   Besides, we have $\Gamma_{\mathcal{XY}} = \Gamma_{\mathcal{X}} \oplus \Gamma_{\mathcal{Y}}$ for $\mathcal{X}, \mathcal{Y} \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{AC}, \mathcal{BD}, \mathcal{AB}, \mathcal{R}_1, \mathcal{R}_2\}$.
2. **Subspace Generation**: Create the subspaces from given vectors and combinations:
   - `Set-1-1`: span of $\{\Gamma_{\mathcal{A}}, \Gamma_{\mathcal{B}}, \Gamma_{\mathcal{C}}, \Gamma_{\mathcal{D}}\}$.
   - `Set-1-2`: span of $\{\Gamma_{\mathcal{AR}_1}, \Gamma_{\mathcal{BR}_1}, \Gamma_{\mathcal{CR}_1}, \Gamma_{\mathcal{DR}_1}\}$.
   - `Set-1-X`: span of $\{\Gamma_{\mathcal{AC}}, \Gamma_{\mathcal{BD}}, \Gamma_{\mathcal{AB}}\}$.
   - `Set-2-1`: span of $\{\Gamma_{\mathcal{AR}_1}, \Gamma_{\mathcal{BR}_2}, \Gamma_{\mathcal{CR}_1}, \Gamma_{\mathcal{DR}_2}\}$.
   - `Set-2-2`: span of $\{\Gamma_{\mathcal{AR}_1}, \Gamma_{\mathcal{BR}_2}, \Gamma_{\mathcal{CR}_2}, \Gamma_{\mathcal{DR}_1}\}$.
   - `Set-2-3`: span of $\{\Gamma_{\mathcal{ABCD}}\}$.

   Note that `Set-1-2` is the setting of Gohr's *aaaa*-blinding experiment.
3. **Remove randomness**: In light of the observations from [14], where it's determined that $\mathcal{ND}$s in the single-key attack setting don't leverage the key schedule, we can adapt the Speck32/64 key schedule to employ independent subkeys. This means we treat K along with $\mathcal{R}_1$ and $\mathcal{R}_2$ as random variables.

Table 3: Experimental results $\mathcal{ND}$ leveraging select XOR information.

Set.	Positive Samples	Negative Samples	Acc.
2-1	$(\mathcal{AR}_1, \mathcal{BR}_2, \mathcal{CR}_1, \mathcal{DR}_2)$	*Random*	0.7558
2-2	$(\mathcal{AR}_1, \mathcal{BR}_2, \mathcal{CR}_2, \mathcal{DR}_1)$	*Random*	0.6722
2-3	$(\mathcal{ABCD}, \mathcal{ABCD}, \mathcal{ABCD}, \mathcal{ABCD})$	*Random*	0.6721

Consequently, any vector that has a component of Γ_K, or $\Gamma_{\mathcal{R}_1}$, or $\Gamma_{\mathcal{R}_2}$ is deemed random, and hence, devoid of information. For example, $\Gamma_{C_{iJ}}$ has a linear component Γ_K, thus, a standalone C_{iJ} lacks information, where $i \in \{0, 1\}$ and $J \in \{L, R\}$. Accordingly, we do as follows.

(a) After creating each subspace, randomness is removed from each subspace according to whether a vector has a component from Γ_K, or $\Gamma_{\mathcal{R}_1}$, or $\Gamma_{\mathcal{R}_2}$. Without ambiguity, the sanitized sets are also denoted by `Set-i-j` for $i \in \{$`1,2`$\}$ and $j \in \{$`1,2,3,X`$\}$.

4. **Comparison**: The sanitized sets are then compared against each other to determine if one set equals or is a subset of the other.

The result shows that `Set-1-1` equals `Set-1-2` and `Set-1-X`, meaning that the combination of Inter-XOR and Intra-XOR is exactly what an information-theoretically optimal distinguisher accepting ciphertext pairs can use under the assumption that it does not use key-schedule.

As we proceed, we delve deeper to ascertain the specific XOR information that holds significance.

Which of the XOR Information is significant for Differential-Neural Distinguisher? To isolate the pivotal XOR information, we conducted experiments where a differential-neural distinguisher was given access to only selected XOR data.

All our subsequent experiments were conducted on a 6-round Speck32/64 with an input difference of `(0040,0000)`, adhering to the configurations presented in Table 17. The differential-neural distinguishers, trained as per Table 2 **Set.**1-1 to **Set.**1-3, serve as baselines (**Set.**1-2 and **Set.**1-3 correspond to Gohr's *aaaa*-blinding experiment)). In the sequel, we use **Set.**`i-j` to refer to the experimental setup, while `Set-i-j` represents the associated information set for the positive samples, where $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$.

Defining $\mathcal{R}_1$ and $\mathcal{R}_2$ as two distinct random values, **Set.**2-1 in Table 3 retains only Inter-XOR and Total-XOR, while **Set.**2-2 keeps only Cross-XOR and Total-XOR. **Set.**2-3, on the other hand, exclusively considers Total-XOR. Firstly, our mechanical analysis on sanitized subspaces reveals the following relations:

- `Set-2-1` $\subset$ `Set-1-X`,
- `Set-2-2` $\subset$ `Set-1-X`,
- `Set-2-1` $\not\subseteq$ `Set-2-2`,
- `Set-2-2` $\not\subseteq$ `Set-2-1`,
- `Set-2-3` $\subset$ `Set-2-1`,
- `Set-2-3` $\subset$ `Set-2-2`.

In Table 3 **Set.**2-1, the differential-neural distinguisher's access is limited to Inter-XOR and Total-XOR – equivalent to what the DDT distinguisher utilizes.

Its accuracy aligns closely with the 5-round DDT's accuracy of 0.758, without any noticeable enhancement. This underscores the differential-neural distinguisher's advantage over the DDT arising from its access to extra information. From this observation, we reinforce the subsequent conclusion.

Conclusion 1 *The differential-neural distinguisher $\mathcal{ND}^{\textsc{Speck}_{rR}}$'s superiority over $\mathcal{DD}^{\textsc{Speck}_{rR}}$ is mainly due to its exploit of Intra-XOR and Cross-XOR.*

This conclusion naturally prompts a more intricate query: How does the differential-neural distinguisher effectively exploit Intra-XOR and Cross-XOR? Upon closer inspection, we can further dismiss the significance of Cross-XOR. Given that `Set-2-3` $\subset$ `Set-2-2`, it's evident that `Set-2-3` provides inherently less data than `Set-2-2`. While in **Set.**2-2, combining Total-XOR with either Intra-XOR or Cross-XOR results in a valid distinguisher, solely using Total-XOR in **Set.**2-3 yields an accuracy identical to the distinguisher in **Set.**2-2. From this, we conclude that Cross-XOR on its own lacks significance. The differential-neural distinguisher likely uncovers new patterns by melding Inter-XOR with either Intra-XOR or Cross-XOR. This line of reasoning culminates in the following conclusion.

Conclusion 2 *Unlike Inter-XOR, neither Intra-XOR nor Cross-XOR independently offers useful information. The differential-neural distinguisher relies on combinations of Inter-XOR with either Intra-XOR or Cross-XOR.*

Remark 1 (On $\mathcal{ND}$ exploiting the key schedule). Gohr's study in [14] indicates that $\mathcal{ND}$s, in a single-key attack on Speck, do not exploit the key schedule. It naturally raises the question: Do $\mathcal{ND}$s behave similarly in related-key scenarios? Motivated by this, we conduct comparison experiments similar to Gohr's *aaaa*-blinding experiment (comparing $\mathcal{RK}$-$\mathcal{ND}$s in **Set.**1-1 and **Set.**2-1), investigating whether $\mathcal{RK}$-$\mathcal{ND}$s use the same ciphertext equivalence classes as the single-key $\mathcal{ND}$s by [14]. In Sect. 5.1, we delve deep into our $\mathcal{RK}$-$\mathcal{ND}$s and present an interesting observation reinforcing our following $\mathcal{ND}$ explainability in Sect. 3.2.

3.2 Explicitly Rules to Exploit the Information Beyond Full DDT: From a Cryptanalytic Perspective

In this section, we delve into the exact patterns harnessed by the differential-neural distinguisher. Our exploration commences with an intriguing observation from **Experiment** A, as described in [6]. The experiment unfolds as follows:

1. For each 5-round ciphertext pair difference, δ, which results in extreme scores surpassing 0.9 (indicative of a good score) and exhibiting a high frequency of occurrence:
   (a) Generate a set of 10^4 random 32-bit numbers.
   (b) Utilize the difference δ to construct a dataset encompassing 10^4 data pairs, each bearing the difference δ.

(c) Feed the dataset to the differential-neural distinguisher and count the predicted labels.

While DDT-based distinguishers would predict **Experiment** A's entire data as positive, the differential $\mathcal{ND}$ does not. For $\mathcal{ND}$, the proportion of each difference is consistently at 0.75 (refer to Table 19), suggesting that the $\mathcal{ND}$ employs criteria beyond simple differential probability in its classifications. The consistent proportion of 0.75 also implies a discernible pattern linked to two specific bits. If a ciphertext pair aligns with this bi-bit pattern, it's classified as negative, regardless of high output difference probabilities. This observation prompts an investigation into the potential two-bit pattern, motivating us to look into properties of the addition modular 2^n ($\boxplus$) from a cryptanalytic perspective.

Enhancing DDT-based Distinguishers via Conditional Probabilities. In the r-round Speck32/64, denote the input and output differences of the last $\boxplus$ by (α, β, γ), and their respective values by $(x\ y\ z)$ and $(x'\ y'\ z')$. For each output pairs $((C_L, C_R), (C'_L, C'_R))$, one knows the following information: $\gamma = C_L \oplus C'_L$, $\beta = (C_L \oplus C_R \oplus C'_L \oplus C'_R)^{\ggg 2}$, and $y = (C_L \oplus C_R)^{\ggg 2}$. Namely, apart from knowing two differences (*i.e.*, β and γ), one knows a value (*i.e.*, y) around the last $\boxplus$. Besides, the input difference α is unknown but might be biased among positive samples and thus is predictable. Concretely, attributes of the information around the last $\boxplus$ are as follows:

α	unknown but biased	x	unknown and balanced
β	known	y	known
γ	known	z	unknown and balanced

The knowledge of y, which is one of two inputs of the last $\boxplus$, provides additional information apart from the differences. The concrete analysis is as follows.

When conditioned on a fixed y, the differential probability can differ from the average probability over all possible y. For a valid differential propagation $(\alpha, \beta \mapsto \gamma)$ through $\boxplus$, consider each bit position i where $0 \le i < n-1$: If $\texttt{eq}(\alpha,\beta,\gamma)_i = 1$, the difference propagation at the $(i+1)$-th position is deterministic, as elucidated in [21]; Conversely, for $\texttt{eq}(\alpha,\beta,\gamma)_i = 0$, the $(i+1)$-th bit's difference propagation is probabilistic; for a given $(i+1)$-th bit differences to be fulfilled, the input values at the i-th position (namely, x_i, y_i, c_i – the carry's i-th bit) must satisfy a certain linear constraint, detailed in Observation 1.

Observation 1 ([9]) *Let $\delta = (\alpha, \beta \mapsto \gamma)$ be a possible XOR-differential through addition modulo 2^n ($\boxplus$). Let (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 $\texttt{eq}(\alpha,\beta,\gamma)_i = 0$*

$$\left.\begin{array}{ll} x_i \oplus y_i = \texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \alpha_i, & \textit{if } \alpha_i \oplus \beta_i = 0, \\ \left.\begin{array}{ll} x_i \oplus c_i = \texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \alpha_i, & \textit{if } \alpha_i \oplus \texttt{xor}(\alpha,\beta,\gamma)_i = 0, \\ y_i \oplus c_i = \texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i, & \textit{if } \alpha_i \oplus \texttt{xor}(\alpha,\beta,\gamma)_i = 1, \end{array}\right\} & \textit{if } \alpha_i \oplus \beta_i = 1, \end{array}\right\}$$

where c_i is the i-th carry bit, $x \boxplus y = z$, $\texttt{eq}(a,b,d) = (\neg a \oplus b) \wedge (\neg a \oplus d)$ (i.e., $\texttt{eq}(a,b,d) = 1$ if and only if $a = b = d$), and $\texttt{xor}(a,b,d) = a \oplus b \oplus d$.

Table 4: Necessary and sufficient conditions for a one-bit difference from Observation 1

Case No.	Difference	Constraint on values	Known
$\mathrm{Cxy}_{(i+1,i)}$	$\begin{cases}\texttt{eq}(\alpha,\beta,\gamma)_i = 0,\\ \alpha_i \oplus \beta_i = 0.\end{cases}$	$\texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \alpha_i = x_i \oplus y_i$	None
$\mathrm{Cxc}_{(i+1,i)}$	$\begin{cases}\texttt{eq}(\alpha,\beta,\gamma)_i = 0,\\ \alpha_i \oplus \beta_i = 1,\\ \alpha_i \oplus \texttt{xor}(\alpha,\beta,\gamma)_i = 0.\end{cases}$	$\texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \alpha_i = x_i \oplus c_i$	None
$\mathrm{Cyc}_{(i+1,i)}$	$\begin{cases}\texttt{eq}(\alpha,\beta,\gamma)_i = 0,\\ \alpha_i \oplus \beta_i = 1,\\ \alpha_i \oplus \texttt{xor}(\alpha,\beta,\gamma)_i = 1.\end{cases}$	$\texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i = y_i \oplus c_i$	$y_i \oplus c_i$

The column titled "Known" indicates whether the fulfilment of the condition might be known in Speck's last $\boxplus$.

In other words, at bit positions i and $i+1$, a valid difference tuple $(\alpha_{i+1,i}, \beta_{i+1,i}, \gamma_{i+1,i})$ that satisfies $\texttt{eq}(\alpha_i, \beta_i, \gamma_i) = 0$ imposes a 1-bit linear constraint on the tuple (x_i, y_i, c_i). As c_i is determined by lower bits, the freedom for conforming to the constraint comes exclusively from the i-th bits of x and y, independent of constraints at other bit positions. Accordingly, the constraints on $(x_i,\ y_i)$, $(x_i,\ c_i)$, or $(y_i,\ c_i)$ as listed in Observation 1 are necessary and sufficient. Therefore, when the constraint at a bit position is fulfilled, the conditional probability $\tilde{p}$ of a differential whose unconditional probability is p should be calculated as $2 \cdot p$; when unfulfilled, it is 0. In comparison, the conditional probability for random pairs is still at most 2^{-n}. Hence, leveraging conditional probability for classification amplifies the advantage.

To clarify when the fulfilment of the constraints at the last $\boxplus$ can be effectively predicted, we catalog cases from Observation 1 in Table 4, naming them $\mathrm{Cxy}_{(i+1,i)}$, $\mathrm{Cxc}_{(i+1,i)}$, and $\mathrm{Cyc}_{(i+1,i)}$. As above analyzed, in Speck32/64's last $\boxplus$, among the tuple (x, y, c) (with $c = z \oplus x \oplus y$ and unknown z), only y is known. Hence, exploiting knowledge of y requires examining bit positions with differential constraints fulfilling $\mathrm{Cyc}_{(i+1,i)}$ in Table 4.

In the $\mathrm{Cyc}_{(i+1,i)}$ case, the constraint is on $y_i \oplus c_i$. While c_i may seem unknown, it is determined by lower bits: $c_i = x_{i-1}y_{i-1} \oplus (x_{i-1} \oplus y_{i-1})c_{i-1}$. The knowledge on c_i might be inferred if the $(i-1)$-th bit differences meet the condition $\texttt{eq}(\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) = 0$, as per Observation 1. For example, when $\begin{cases}(\alpha_i, \beta_i, \gamma_i) = (0, 1, 0),\\ (\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}) = (1, 1, 0)\end{cases}$, one knows that $\begin{cases}\texttt{eq}(\alpha,\beta,\gamma)_{i-1} = 0,\\ \alpha_{i-1} \oplus \beta_{i-1} = 0,\\ \texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 0.\end{cases}$ From Table 4, one has $x_{i-1} \oplus y_{i-1} = 0$. Thus, $c_i = x_{i-1}y_{i-1} \oplus (x_{i-1} \oplus y_{i-1})c_{i-1} = y_{i-1}$. Therefore, $y_i \oplus c_i = y_i \oplus y_{i-1}$. As a consequence, one can predict the

Table 5: Cases for deducing the i-th carry bit c_i

Case No.	Difference	Value	Known
$\text{Cy0c0}_{(i,i-1)}$		$y_{i-1}=0, c_{i-1}=0$	$c_i = 0$
$\text{Cy1c1}_{(i,i-1)}$		$y_{i-1}=1, c_{i-1}=1$	$c_i = 1$
$\text{Cxy0}_{(i,i-1)}$	$\text{Cxy}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 0$	$x_{i-1} \oplus y_{i-1} = 0$	$c_i = y_{i-1}$
$\text{Cxy1}_{(i,i-1)}$	$\text{Cxy}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 1$	$x_{i-1} \oplus y_{i-1} = 1$	$c_i = c_{i-1}$
$\text{Cxc0}_{(i,i-1)}$	$\text{Cxc}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 0$	$x_{i-1} \oplus c_{i-1} = 0$	$c_i = c_{i-1}$
$\text{Cxc1}_{(i,i-1)}$	$\text{Cxc}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 1$	$x_{i-1} \oplus c_{i-1} = 1$	$c_i = y_{i-1}$
$\text{Cyc0}_{(i,i-1)}$	$\text{Cyc}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \beta_{i-1} = 0$	$y_{i-1} \oplus c_{i-1} = 0$	$c_i = y_{i-1}$
$\text{Cyc1}_{(i,i-1)}$	$\text{Cyc}_{(i,i-1)}$ and $\texttt{xor}(\alpha,\beta,\gamma)_i \oplus \alpha_{i-1} = 1$	$y_{i-1} \oplus c_{i-1} = 1$	$c_i = x_{i-1}$

Table 6: Cases where the knowledge on y can be used to check the fulfilment of the differential constraints

Case No.	Difference	Known
C1	$\text{Cyc}_{(0,-1)}$	$\texttt{xor}(\alpha,\beta,\gamma)_1 \oplus \beta_0 = y_0$
C2	$\text{Cyc}_{(2,1)}$ and $\text{Cy0}_{(1,0)}$	$\texttt{xor}(\alpha,\beta,\gamma)_2 \oplus \beta_1 = y_1$
C3	$\text{Cyc}_{(i+1,i)}$ and ($\text{Cxy0}_{(i,i-1)}$ or $\text{Cxc1}_{(i,i-1)}$ or $\text{Cyc0}_{(i,i-1)}$)	$\texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i = y_i \oplus y_{i-1}$
C4	$\text{Cyc}_{(i+1,i)}$ and ($\text{Cxy1}_{(i,i-1)}$ or $\text{Cxc0}_{(i,i-1)}$) and ($\text{Cxy0}_{(i-1,i-2)}$ or $\text{Cxc1}_{(i-1,i-2)}$ or $\text{Cyc0}_{(i-1,i-2)}$)	$\texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i = y_i \oplus y_{i-2}$

fulfilment of constraint in case $\text{Cyc}_{(i+1,i)}$ by observing whether $y_i \oplus y_{i-1} = \texttt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i$. Table 5 lists more cases where c_i might be known.

Incorporating observations from Table 4 and Table 5, one gets Table 6, which lists various cases where the knowledge of y can be used to determine the satisfaction of differential constraints.

Note that apart from the general cases (C3 and C4) at the i-th bit, special cases (C1 and C2) emerge at the two least significant bits due to the carry bit c_0 being 0. For example,

1. at the 0th bit position, observing $\beta_0 = 0$ and $\gamma_0 = 1$ determines $\alpha_0 = 1$ based on Alg. 3. From case $\text{Cyc}_{(i+1,i)}$ in Table 4 and given $c_0 = 0$, one knows that $\texttt{xor}(\alpha,\beta,\gamma)_1 \oplus \beta_0 = y_0 \oplus c_0 = y_0$;
2. at the 1st bit position, $c_1 = x_0 y_0 \oplus (x_0 \oplus y_0) c_0 = x_0 y_0$. Given an observed $y_0 = 0$, one knows $c_1 = 0$. Consequently, in case $\text{Cyc}_{(2,1)}$ and $y_0 = 0$, one knows $\texttt{xor}(\alpha,\beta,\gamma)_2 \oplus \beta_1 = y_1 \oplus c_1 = y_1$;
3. in general case C3, based on Table 5, c_i is determined as y_{i-1}, leading to the use of $y_i \oplus y_{i-1}$;
4. in general case C4, applying Table 5 to the $(i-1, i-2)$-th bit position, it is inferred that $c_i = c_{i-1} = y_{i-2}$, leading to the use of $y_i \oplus y_{i-2}$;

5. for cases where $c_{i-1} = c_{i-2}$, one can further observe differences at the $(i-2)$-th bit position and continues deducting c_{i-2} by observing bit differences at the $i-3$ position.

Table 7 lists some concrete examples of differential patterns where the observation of y enables prediction of whether differential constraints are met.

Remark 2. These constraints on values for valid differential propagation resonate with established concepts. Specifically, insights derived from Table 6 align with findings on multi-bit constraints from [19, 20], quasi-differential trails in [7], and extended differential-linear approximations in [10]. Table 7 exhibits the correspondence between examples of cases in Table 6 and these established concepts. For instance, given a differential propagation $(\alpha_{i+1,i,i-1}, \beta_{i+1,i,i-1} \mapsto \gamma_{i+1,i,i-1}) = (\texttt{*01}, \texttt{*11} \mapsto \texttt{*00})$ (for $0 < i < n-1$),

1. using the 1.5-bit constraints concept and the finite state machines representing the differential properties of modular addition from [19, 20], one can get a new constraint and refine the propagation $(\texttt{--x}, \texttt{-xx} \mapsto \texttt{---})$ to $(\texttt{--x}, \texttt{->x} \mapsto \texttt{---})$ (where the notations {`-`, `x`, `>`, `<`, `=`, `!`} are explained below Table 7); more generally, C3 cases correspond to the 1.5-bit constraints {`>`, `<`, `=`, `!`} in [19, 20];
2. using the quasi-differential trail concept from [7], the differential trail $(\texttt{001}, \texttt{011} \mapsto \texttt{000})$ comprises a non-trivial quasi-differential trail with a mask of $(\texttt{000}, \texttt{011} \mapsto \texttt{000})$. The non-trivial quasi-differential trail has correlation -2^{-1} (*i.e.*, additional weight of 0). Consequently, the "fixed-y" probability of this differential trail is $(1-(-1)^{y_i \oplus y_{i-1}}) \cdot 2^{-1}$, *i.e.*, the probability equals 1 when $y_i \oplus y_{i-1} = 1$ and 0 in the opposite case;
3. using the extended differential-linear connectivity table (EDLCT) concept from [10], assessing the constraint $y_i \oplus y_{i-1} = \alpha_{i+1} \oplus \beta_{i+1} \oplus \gamma_{i+1} \oplus \beta_i$ aligns with gauging the bias of the linear approximation $(x_{i+1} \oplus x'_{i+1}) \oplus (y_{i+1} \oplus y'_{i+1}) \oplus (z_{i+1} \oplus z'_{i+1}) \oplus (y_i \oplus y'_i) \oplus (y_i \oplus y_{i-1})$ that corresponds to selecting bits $[x_{i+1}, y_{i+1}, z_{i+1}, y_{i-1}]$ and $[x'_{i+1}, y'_{i+1}, z'_{i+1}, y'_i]$.
   As noted in [6], $\mathcal{ND}$s rely on differential-linear (DL) properties. We note that pure DL properties do not provide additional information beyond full DDT; the differential-linear distribution can be directly derived from the full differential distribution. It is the *extended* differential-linear distribution [10] (which includes the selection of ciphertext values apart from differences) that contains additional information.

Table 7: Concrete examples of differential patterns where one can predict the fulfilment of the differential constraints by observing the value of y

Case	Observation 1			Multi-bit Constraints in [19, 20]		Quasi-differential in [7]		Extended DLCT in [10]
No.	Difference local map	Value	Observations	org	new	diff	mask (add. w)	selected bits
C1	$\alpha_{1,0}$ *1 $\beta_{1,0}$ *0 $\gamma_{1,0}$ *1	$x_{1,0}$ ** $y_{1,0}$ ** $z_{1,0}$ **	$y_0 = \alpha_1 \oplus \beta_1 \oplus \gamma_1 \oplus 0$	-x -- -x	-x -0 -x	01 00 01	00 01 $(+2^0\ )$ 00	$[x_1,\ y_1,\ z_1]$, $[x'_1,\ y'_1,\ z'_1,\ y'_0]$
C2	$\alpha_{2,1,0}$ *1* $\beta_{2,1,0}$ *0* $\gamma_{2,1,0}$ *1*	$x_{2,1,0}$ *** $y_{2,1,0}$ **0 $z_{2,1,0}$ ***	$y_1 = \alpha_2 \oplus \beta_2 \oplus \gamma_2 \oplus 0$	-x? --0 -x?	-x? -00 -x?	010 000 010	000 011 $(+2^{-1})$ 000	$[x_2,\ y_2,\ z_2,\ y_0]$, $[x'_2,\ y'_2,\ z'_2,\ y'_1]$
C3	$\alpha_{i+1,i,i-1}$ *01 $\beta_{i+1,i,i-1}$ *11 $\gamma_{i+1,i,i-1}$ *00	$x_{i+1,i,i-1}$ *** $y_{i+1,i,i-1}$ *** $z_{i+1,i,i-1}$ ***	$y_i \oplus y_{i-1} =$ $\alpha_{i+1} \oplus \beta_{i+1} \oplus \gamma_{i+1} \oplus 1$	--x -xx ---	--x ->x ---	001 011 000	000 011 $(-2^0\ )$ 000	$[x_{i+1},\ y_{i+1},\ z_{i+1},\ y_{i-1}]$, $[x'_{i+1},\ y'_{i+1},\ z'_{i+1},\ y'_i]$
C3	$\alpha_{i+1,i,i-1}$ *11 $\beta_{i+1,i,i-1}$ *00 $\gamma_{i+1,i,i-1}$ *11	$x_{i+1,i,i-1}$ *** $y_{i+1,i,i-1}$ *** $z_{i+1,i,i-1}$ ***	$y_i \oplus y_{i-1} =$ $\alpha_{i+1} \oplus \beta_{i+1} \oplus \gamma_{i+1} \oplus 0$	-xx --- -xx	-xx -=- -xx	011 000 011	000 011 $(+2^0\ )$ 000	$[x_{i+1},\ y_{i+1},\ z_{i+1},\ y_{i-1}]$, $[x'_{i+1},\ y'_{i+1},\ z'_{i+1},\ y'_i]$
C4	$\alpha_{i+1,i,i-1,i-2}$ *111 $\beta_{i+1,i,i-1,i-2}$ *010 $\gamma_{i+1,i,i-1,i-2}$ *101	$x_{i+1,i,i-1,i-2}$ **** $y_{i+1,i,i-1,i-2}$ **** $z_{i+1,i,i-1,i-2}$ ****	$y_i \oplus y_{i-2} =$ $\alpha_{i+1} \oplus \beta_{i+1} \oplus \gamma_{i+1} \oplus 0$	-xxx --x- -x-x	-xxx -$^{2}_{0}$x- -x-x	0111 0010 0101	0000 0101 $(+2^0\ )$ 0000	$[x_{i+1},\ y_{i+1},\ z_{i+1},\ y_{i-2}]$, $[x'_{i+1},\ y'_{i+1},\ z'_{i+1},\ y'_i]$

0: $y_i = y'_i = 0$ 1: $y_i = y'_i = 1$ -: $y_i = y'_i$ x: $y_i \neq y'_i$ $^{2}_{0}$: uncommon 2.5-bit constraint “28000014”
=: $y'_i = y_i = y_{i-1}$!: $y'_i = y_i \neq y_{i-1}$ <: $y'_i \neq y_i = y_{i-1}$ >: $y'_i \neq y_i \neq y_{i-1}$

To directly exploit these observations for an r-round Speck32/64, a preliminary is to effectively predict the input difference α at the last $\boxplus$, which equals $((\delta_R^{r-2})^{\lll 2} \oplus \delta_R^{r-1})^{\ggg 7}$. Given the known δ_R^{r-1} from r-round outputs, the focus shifts to predicting $(\delta_R^{r-2})^{\lll 2}$. Notably, for $r \le 7$ and input difference `(0040, 0000)`, some bits of $(\delta_R^{r-2})^{\lll 2}$ exhibit bias, as detailed in Table 8, enabling predictions of α for positive samples.

Table 8: Bit bias towards '0' of $(\delta_R^{r-2})^{\lll 2}$ for $4 \le r \le 7$, where the input difference of the plaintext is `(0040,0000)`. A positive (resp. negative) value indicates a bias towards '0' (resp. '1').

Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$(\delta_R^2)^{\lll 2}$	0.4689	0.4377	0.3752	0.2498	-0.0002	-0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	-0.5000	0.5000	-0.4922	0.4844
$(\delta_R^3)^{\lll 2}$	0.3749	0.2809	0.1247	-0.1241	0.0100	0.4687	0.4451	0.4062	0.3435	0.2498	-0.1251	-0.0000	-0.4922	0.4844	-0.4608	0.4297
$(\delta_R^4)^{\lll 2}$	0.0926	-0.0347	0.0004	-0.0617	0.0028	-0.3709	0.3012	0.2046	-0.0578	-0.0004	0.0312	-0.0009	0.4451	0.4059	-0.2931	0.2035
$(\delta_R^5)^{\lll 2}$	-0.0002	-0.0016	-0.0002	-0.0238	0.0002	0.1531	-0.0243	-0.0001	-0.0034	-0.0011	-0.0076	-0.0001	0.2700	0.1772	-0.0597	-0.0061
$(\delta_R^6)^{\lll 2}$	0.0001	-0.0006	0.0005	0.0103	0.0002	-0.0002	-0.0009	-0.0003	-0.0004	-0.0002	-0.0033	0.0007	-0.0438	-0.0048	-0.0007	-0.0010

A simple procedure to improve the DDT-based distinguisher. To improve a DDT-based distinguisher for an r-round Speck32/64 using its $\mathrm{DDT}_{\texttt{(0040, 0000)}}$, we proceed as follows, resulting in distinguishers named $\mathcal{YD}^{\textsc{Speck}_{rR}}$:

1. Compute the bias (towards 0) of each bit of $(\delta_R^{r-2})^{\lll 2}$,
2. Predict bit values for $(\delta_R^{r-2})^{\lll 2}$ based on their biases: assign a value of 0 if bias ≥ 0 and 1 otherwise,
3. Define the absolute bias of the i-bit of $((\delta_R^{r-2})^{\lll 2})^{\ggg 7}$ as $\epsilon_\alpha(i)$,
4. For each output pair of r-round Speck32/64, use Alg. 1 to predict its classification.

Results of improving the DDT-based distinguisher. Table 9 presents the performance of $\mathcal{YD}^{\textsc{Speck}_{rR}}$ distinguishers, derived from the described enhancement of $\mathcal{DD}^{\textsc{Speck}_{rR}}$. For rounds $4 \le r \le 7$, $\mathcal{YD}^{\textsc{Speck}_{rR}}$ typically shows improvement. In contrast, when applying a similar method to adjust the $\mathcal{ND}^{\textsc{Speck}_{rR}}$ score Z (converting score Z to probability p using $p = Z/(1-Z) \cdot 2^{-n}$), the accuracy does not get improved. It is unchanged for $\mathcal{ND}^{\textsc{Speck}_{4R}}$ and marginally degrades for rounds $5 \le r \le 7$ since the threshold τ is set less than 0.5. This suggests that the additional information useful in improving DDT-based distinguishers does not help improve $\mathcal{ND}$'s; thus, the $\mathcal{ND}$'s might have maximally utilized this information already. Thus, we conclude as follows.

Conclusion 3 *By utilizing conditional differential distributions when the input and/or output values of the last nonlinear operation are observable, a distinguisher can surpass pure DDT-based counterparts. Accordingly, if these conditional distributions differ greatly from the averaged differential distribution, and*

Algorithm 1: A simple procedure to improve the DDT-based distinguisher: $\mathcal{YD}^{\textsc{Speck}_{rR}}$

```
1. Get the differential probability $p$ of $(\texttt{0040, 0000}) \mapsto (C_L \oplus C'_L, C_R \oplus C'_R)$ by
   looking up the table $\mathrm{DDT}_{(\texttt{0040, 0000})}[(C_L \oplus C'_L, C_R \oplus C'_R)]$
2. Compute the following information around the last $\boxplus$ from $((C_L, C_R), (C'_L, C'_R))$:
   (a) $\gamma \leftarrow C_L \oplus C'_L, \quad \beta \leftarrow (C_L \oplus C_R \oplus C'_L \oplus C'_R)^{\ggg 2}$,
   (b) $\alpha \leftarrow ((\delta_R^{r-2})^{\lll 2} \oplus \beta)^{\ggg 7}, \quad y \leftarrow (C_L \oplus C_R)^{\ggg 2}$.
3. For bit position 0, if $\epsilon_\alpha(1) > \tau$ and $\epsilon_\alpha(0) > \tau$, do:
   (a) If $\mathrm{Cyc}_{(0,-1)}$, do: $p \leftarrow (1 + (-1)^{\mathtt{xor}(\alpha,\beta,\gamma)_1 \oplus \beta_0 \oplus y_0}) \cdot p$.
4. For bit position 1, if $\epsilon_\alpha(2) > \tau$ and $\epsilon_\alpha(1) > \tau$, do:
   (a) If $\mathrm{Cyc}_{(2,1)}$ and $y_0 = 0$, do: $p \leftarrow (1 + (-1)^{\mathtt{xor}(\alpha,\beta,\gamma)_2 \oplus \beta_1 \oplus y_1}) \cdot p$.
5. For each bit position $i$ $(1 < i < n - 1)$, if $\epsilon_\alpha(i + 1) > \tau$ and $\epsilon_\alpha(i) > \tau$ and
   $\epsilon_\alpha(i - 1) > \tau$, do:
   (a) If $\mathrm{Cyc}_{(i+1,i)}$ and $(\mathrm{Cxy0}_{(i,i-1)}$ or $\mathrm{Cxc1}_{(i,i-1)}$ or $\mathrm{Cyc0}_{(i,i-1)})$, do:
       $p \leftarrow (1 + (-1)^{\mathtt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i \oplus y_i \oplus y_{i-1}}) \cdot p$.
   (b) If $\mathrm{Cyc}_{(i+1,i)}$ and $(\mathrm{Cxy1}_{(i,i-1)}$ or $\mathrm{Cxc0}_{(i,i-1)})$ and $(\mathrm{Cxy0}_{(i-1,i-2)}$ or
       $\mathrm{Cxc1}_{(i-1,i-2)}$ or $\mathrm{Cyc0}_{(i-1,i-2)})$ and $\epsilon_\alpha(i - 2) > \tau$, do:
       $p \leftarrow (1 + (-1)^{\mathtt{xor}(\alpha,\beta,\gamma)_{i+1} \oplus \beta_i \oplus y_i \oplus y_{i-2}}) \cdot p$.
6. If $p > 2^{-n}$, predict $Z \leftarrow 1$; else predict $Z \leftarrow 0$.
```

the satisfaction of the conditions is either observable or effectively predictable, then r-round $\mathcal{ND}$s can outperform r-round DDT-based distinguishers.

For Speck, one of the two inputs of the last non-linear operation ($\boxplus$) is observable. If conditioned on this input, the conditional differential distribution can diverge significantly from the averaged one. Therefore, an optimal distinguisher can obviously outperform a pure DDT-based counterpart. A similar analysis applies to Simon. In Simon, the values that go through the last nonlinear operation are fully observable. Consequently, it is interpretable that in the case of Simon, an r-round $\mathcal{ND}$ can achieve an accuracy close to the $(r-1)$-round DDT [3].

This conclusion can be further supported by the following experimental result: In a modified r-round Speck32/64 where the last key XORing is omitted, revealing both z and y (equating to full awareness of the satisfaction of the last round's differential constraints given a predictable input difference α), a well-trained r-round $\mathcal{ND}$ achieves an accuracy close to the $(r-1)$-round $\mathcal{DD}$. Interestingly, subsequent observations on $\mathcal{RK}$-$\mathcal{ND}$s reinforce our conclusion, while the conclusion itself aids in interpreting those observations.

3.3 Distinguishers using Systematic Computation of Conditional Differential Probability under Known y

The simple process in Alg. 1 is fast, but it requires evaluating the bias of each bit of the difference on the right branch of round $r-2$ to estimate the input difference α for the last modular addition. The differential probability can only be adjusted if the estimated bias of the corresponding bit of α exceeds a certain threshold. As a result, it does not make the most of the information in y. Therefore, we further designed a process, described in Alg. 2, to systematically calculate the differential probability conditioned under the known value of y and predict based on the $(r-1)$-round DDT [7].

In essence, the systematic process involves using β, γ, and y to determine all possible αs and the conditional differential probabilities of the last round. It combines this information with the probabilities of the previous $(r-1)$ rounds to calculate the conditional differential probability for r rounds under the known value of y. Finally, it uses the systematically computed conditional probability for prediction.

More concretely, in the process, we have the following procedures:

1. **Precomputation**: We generate three b-bit conditional DDTs, denoted as $\mathbf{A}_0$, $\mathbf{A}_{\text{next}}$, and $\mathbf{A}^c_{\text{next}}$, of the single modular addition operation $\boxplus$. These resemble Dinur's b-bit filter in [11]:
   (a) $\mathbf{A}_0$ tells all valid b-bit values of α with their associated probability pr for given b-bit inputs β, γ, and y at the first b least significant bits (LSB) where the first carry bits are zeros.
   (b) $\mathbf{A}_{\text{next}}$ tells all valid 1-bit values of α_{next} with their associated probability pr for given b-bit inputs β, γ, y, and $(b-1)$-bit α at intermediate consecutive b bits where the LSB of carry is undetermined.
   (c) $\mathbf{A}^c_{\text{next}}$ is similar to $\mathbf{A}_{\text{next}}$ but serves scenarios with known carry LSBs.
2. **Initialization**: From a received ciphertext pair, we derive the output difference γ, input difference β, and input value y; initialize the to-be-calculated probability p and the last round's probability factor q with 0 and 1.
3. **Generate candidate LSB b-bit of α**:
   (a) Using table $\mathbf{A}_0$, we obtain candidates for the LSB b-bit of α based on the LSB b-bit of β, γ, and y, update q with the associated pr.
   (b) For each valid LSB b-bit of α, we invoke 'ComputeCarryNextBit' to determine the carry bits wherever possible according to Table 5.
4. **Iterative Calculation**: For each valid LSB b-bit of α,
   (a) Starting from the $(b-1)$-th bit, we invoke 'ComputeAlphaPrNextBit' to sequentially determine α's later bits and the respective augmentation of the probability factor to q; alongside, we use 'ComputeCarryNextBit' to determine the carry bits wherever possible, preparing to be used to derive later bits of α in case of Cyc or be used to look up $\mathbf{A}^c_{\text{next}}$.

   Within procedure 'ComputeAlphaPrNextBit':
   (a) Once α is fully assigned, we calculate the output difference of the penultimate round and use it to look up the $(r-1)$-round DDT. The resultant value, upon multiplied by the last round's probability factor q, yields a contribution term to the final probability p.
   (b) At an intermediate bit position i, equal three input/output bits differences facilitate the direct determination of the subsequent α bit.

[7] Please refer to [1] for the implementation codes and experimental results.

(c) When input/output bits differences at position $(i+1, i)$ conforms to the $\mathrm{Cyc}_{(i+1,i)}$ condition with an determined value for c_i, the subsequent α bit is deduced using $y_i \oplus c_i$. After determining α_{i+1}, we invoke 'ComputeCarryNextBit' to determine the carry bit c_{i+1} wherever possible.

(d) Otherwise (in the absence of conformity or a determined c_i value), α_{i+1} is enumerated using either $\mathbf{A}_{\mathrm{next}}$ or $\mathbf{A}^{c}_{\mathrm{next}}$, depending on whether the carry bit before b bits of the $(i+1)$-th bit is determined.

(e) After obtaining α_{i+1} and its probability pr, we continue to determine the next α bit, updating the probability factor by multiplying pr to q.

The resulting procedure is slower than the simple one; however, the resulting distinguishers, named "$\mathcal{AD}_{\mathbf{YD}}$", have accuracy exceeds not only that of the distinguishers $\mathcal{DD}$s but also the neural distinguishers $\mathcal{ND}$s, comparable to the $r-1$-round DDT-based key-averaging distinguishers $\mathcal{AD}_{\mathbf{KD}}$s [2] (refer to Tables 9 and 20), indicating an exemplary accuracy for $\mathcal{ND}$s.

3.4 Discussion on $\mathcal{ND}$'s Advantages

Based on the above observations and experiments, we can conclude that $\mathcal{ND}$'s advantage over pure differential-based distinguishers comes from exploiting the conditional differential distribution under the partially known value from ciphertexts input to the last non-linear operation. More specifically, $\mathcal{ND}$s exploited the correlation between the ciphertexts' partial value, the ciphertext pair's differences, and the intermediate states' differences. Specifically, when some of the last-round nonlinear operations' inputs and outputs are known (*i.e.*, not XORed with independently randomized key bits), a distinguisher can achieve higher distinguishing accuracy than an r-round pure differential-based distinguisher.

These findings apply not only to the Speck but also to other block ciphers, such as Simon and Gift (refer to Appendix D.1), and demonstrate the ability of neural networks to capture and utilize complex relationships between ciphertext values and intermediate state differences. Note that the neural distinguishers are not aware of the specific details of the ciphers, including their non-linear components and structure. Therefore, these neural distinguishers can be used for ciphers that have unknown components.

On the performance of various distinguishers. Experiments showed that $\mathcal{ND}$s can be more efficient while achieving comparable accuracy to sophisticated manual methods (Alg. 2). Please refer to Table 9 for detailed benchmarks. Note that in benchmarks listed in Table 9, all DDT-based distinguishers are implemented in `C++`, whereas $\mathcal{ND}$-based distinguishers are implemented in `Python Tensorflow`. Although implementations in `C++` might be inherently faster than its `Python` counterpart, $\mathcal{ND}^{\textsc{Speck}_{*R}}$s in `Python` are still more efficient than $\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{*R}}$ and $\mathcal{AD}_{\mathbf{KD}}^{\textsc{Speck}_{*R}}$ in `C++` (all restricted to run in a single `CPU` thread). Therefore, we can conclude that the neural network-based distinguishers provide a good trade-off between efficiency and accuracy.

Algorithm 2: Known-y differential distinguishers: $\mathcal{AD}^{\text{SPECK}_{rR}}_{\mathbf{YD}}$

```
1. $b \leftarrow 6$ // for practical reason, we consider 6-bit conditional DDT of $\boxplus$
2. $\mathbf{A}_0, \mathbf{A}_{\text{next}}, \mathbf{A}^c_{\text{next}} \leftarrow$ GenMultiBitsConditionalDDTs($b$)
3. $p \leftarrow 0.0$, $q \leftarrow 1.0$
4. Compute the following around the last $\boxplus$ from $((C_L, C_R), (C'_L, C'_R))$:
   (a) $\gamma \leftarrow C_L \oplus C'_L$, $\beta \leftarrow (C_L \oplus C_R \oplus C'_L \oplus C'_R)^{\ggg 2}$, $y \leftarrow (C_L \oplus C_R)^{\ggg 2}$.
5. $\alpha \leftarrow \mathbf{0}, c \leftarrow \mathbf{0}$
6. $\beta_b \leftarrow$ LSB $b$ bits of $\beta$, $\gamma_b \leftarrow$ LSB $b$ bits of $\gamma$, $y_b \leftarrow$ LSB $b$ bits of $y$
7. For $(\alpha_b, pr) \in \mathbf{A}_0[\beta_b, \gamma_b, y_b]$
   (a) $\alpha \leftarrow \alpha_b$
   (b) For $i$ in $\{0, 1, \ldots, b-2\}$: ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$)
   (c) ComputeAlphaPrNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $b-1$, $q \times pr$, $p$)
8. If $p > 2^{-n}$, predict $Z \leftarrow 1$; else predict $Z \leftarrow 0$.

ComputeAlphaPrNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$, $q$, $p$) // update $c_{i+1}$, $\alpha_{i+1}$, $p$ in-place
1. If $i =$ WordSize $- 1$: $p \leftarrow p + q \times \mathcal{DD}^{\text{SPECK}_{r-1R}}(\alpha^{\lll 7} \| \beta)$; return
2. If $\mathtt{eq}(\alpha_i, \beta_i, \gamma_i)$:
   (a) $\alpha_{i+1} \leftarrow \beta_{i+1} \oplus \gamma_{i+1} \oplus \beta_i$; ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$);
   (b) ComputeAlphaPrNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i+1$, $q \cdot 1$, $p$); return
3. Else if $\text{Cyc}_{(i+1,i)}$ and $c_i \neq \perp$:
   (a) $\alpha_{i+1} \leftarrow \beta_{i+1} \oplus \gamma_{i+1} \oplus \beta_i \oplus y_i \oplus c_i$; ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$)
   (b) ComputeAlphaPrNextBit($\alpha$, $\beta$, $\gamma$, $y$, $i+1$, $q \cdot 1$, $p$); return
4. Else:
   (a) $\beta_b \leftarrow \beta_{\{i+1,\ldots,i+2-b\}}$, $\gamma_b \leftarrow \gamma_{\{i+1,\ldots,i+2-b\}}$, $y_b \leftarrow y_{\{i+1,\ldots,i+2-b\}}$,
       $\alpha_b \leftarrow \alpha_{\{i,\ldots,i+2-b\}}$
   (b) If $c_{i+2-b} \neq \perp$: For $(\alpha_{i+1}, pr) \leftarrow \mathbf{A}^c_{\text{next}}[\beta_b, \gamma_b, y_b, \alpha_b, c_{i+2-b}]$
       – ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$)
       – ComputeAlphaPrNextBit($\alpha$, $\beta$, $\gamma$, $y$, $i+1$, $q \cdot pr$, $p$)
   (c) If $c_{i+2-b} = \perp$: For $(\alpha_{i+1}, pr) \leftarrow \mathbf{A}_{\text{next}}[\beta_b, \gamma_b, y_b, \alpha_b]$
       – ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$)
       – ComputeAlphaPrNextBit($\alpha$, $\beta$, $\gamma$, $y$, $i+1$, $q \cdot pr$, $p$)

ComputeCarryNextBit($c$, $\alpha$, $\beta$, $\gamma$, $y$, $i$) // update $c_{i+1}$ in-place
1. If $y_i = 0$ and $c_i = 0$: $c_{i+1} \leftarrow 0$
2. Else if $y_i = 1$ and $c_i = 1$: $c_{i+1} \leftarrow 1$
3. Else if $\text{Cxy0}_{(i+1,i)}$ or $\text{Cxc1}_{(i+1,i)}$ or $\text{Cyc0}_{(i+1,i)}$: $c_{i+1} \leftarrow y_i$
4. Else if $\text{Cxy1}_{(i+1,i)}$ or $\text{Cxc0}_{(i+1,i)}$: $c_{i+1} \leftarrow c_i$
5. Else: $c_{i+1} \leftarrow \perp$. // $\perp$ means unknown

GenMultiBitsConditionalDDTs($b$)
1. $\mathbf{A}_0 \leftarrow$ Generate $b$-bit conditional DDT of $\boxplus$, each entry is indexed by ($b$-bit $\beta$,
   $b$-bit $\gamma$, $b$-bit $y$), the values are ($b$-bit $\alpha$, non-zero $pr$). // $\mathbf{A}_0$ will be used for the
   first $b$ bits since one knows that both LSB carry bits are 0.
2. $\mathbf{A}_{\text{next}} \leftarrow$ Generate $b$-bit conditional DDT of $\boxplus$, each entry is indexed by ($b$-bit $\beta$,
   $b$-bit $\gamma$, $b$-bit $y$, $(b-1)$-bit $\alpha$), the values are (1-bit $\alpha_{\text{next}}$, non-zero $pr$). // $\mathbf{A}_{\text{next}}$
   will be used for the intermediate bits when LSB carry $c$ is unknown.
3. $\mathbf{A}^c_{\text{next}} \leftarrow$ Generate $b$-bit conditional DDT of $\boxplus$, each entry is indexed by ($b$-bit $\beta$,
   $b$-bit $\gamma$, $b$-bit $y$, $(b-1)$-bit $\alpha$, 1-bit carry $c$), the values are (1-bit $\alpha_{\text{next}}$, non-zero
   $pr$). // $\mathbf{A}^c_{\text{next}}$ will be used for the intermediate bits when LSB carry $c$ is known.
4. Output $\mathbf{A}_0, \mathbf{A}_{\text{next}}, \mathbf{A}^c_{\text{next}}$
```

Table 9: Performance of the improved DDT-based distinguishers ($\mathcal{YD}$s and $\mathcal{AD}_{\mathbf{YD}}$s) on Speck32/64 and comparisons with pure DDT-based distinguishers ($\mathcal{DD}$s), neural distinguishers ($\mathcal{ND}$s), and DDT-based key-averaging distinguishers ($\mathcal{AD}_{\mathbf{KD}}$s)

#R	Name	ACC	TPR	TNR	Mem (GBytes)	Time (Secs per 2^{20})
4	$\mathcal{DD}^{\text{Speck}4R}$	0.9869	0.9869	0.9870	32.5	$2^{-4.98}$
4	$\mathcal{YD}^{\text{Speck}4R}$	0.9907	0.9887	0.9928	32.5	$2^{-2.37}$
5	$\mathcal{DD}^{\text{Speck}5R}$	0.9107	0.8775	0.9440	32.5	$2^{-4.94}$
5	$\mathcal{YD}^{\text{Speck}5R}$	0.9215	0.8947	0.9484	32.5	$2^{-1.87}$
5	$\mathcal{ND}^{\text{Speck}5R}$	0.9273	0.9011	0.9536	0.0277	$2^{+3.56}$
5	$\mathcal{AD}_{\mathbf{YD}}^{\text{Speck}5R}$	0.9362	0.9173	0.9552	32.5	$2^{+5.46}$
5	$\mathcal{AD}_{\mathbf{KD}}^{\text{Speck}5R}$	0.9364	0.9171	0.9557	32.5	$2^{+7.03}$
6	$\mathcal{DD}^{\text{Speck}6R}$	0.7584	0.6795	0.8371	32.5	$2^{-4.53}$
6	$\mathcal{YD}^{\text{Speck}6R}$	0.7663	0.7118	0.8207	32.5	$2^{-2.05}$
6	$\mathcal{ND}^{\text{Speck}6R}$	0.7876	0.7197	0.8554	0.0277	$2^{+3.54}$
6	$\mathcal{AD}_{\mathbf{YD}}^{\text{Speck}6R}$	0.7949	0.7309	0.8587	32.5	$2^{+5.12}$
6	$\mathcal{AD}_{\mathbf{KD}}^{\text{Speck}6R}$	0.7946	0.7309	0.8583	32.5	$2^{+7.03}$
7	$\mathcal{DD}^{\text{Speck}7R}$	0.5913	0.5430	0.6397	32.5	$2^{-4.49}$
7	$\mathcal{YD}^{\text{Speck}7R}$	0.5962	0.5582	0.6343	32.5	$2^{-2.18}$
7	$\mathcal{ND}^{\text{Speck}7R}$	0.6155	0.5325	0.6985	0.0277	$2^{+3.57}$
7	$\mathcal{AD}_{\mathbf{YD}}^{\text{Speck}7R}$	0.6237	0.5428	0.7048	32.5	$2^{+5.33}$
7	$\mathcal{AD}_{\mathbf{KD}}^{\text{Speck}7R}$	0.6240	0.5435	0.7046	32.5	$2^{+7.04}$
8	$\mathcal{DD}^{\text{Speck}8R}$	0.5116	0.4963	0.5268	32.5	$2^{-4.64}$
8	$\mathcal{YD}^{\text{Speck}8R}$	0.5117	0.4967	0.5268	32.5	$2^{-2.99}$
8	$\mathcal{ND}^{\text{Speck}8R}$	0.5135	0.5184	0.5085	0.0277	$2^{+3.55}$
8	$\mathcal{AD}_{\mathbf{YD}}^{\text{Speck}8R}$	0.5187	0.4914	0.5460	32.5	$2^{+5.51}$
8	$\mathcal{AD}_{\mathbf{KD}}^{\text{Speck}8R}$	0.5194	0.4919	0.5469	32.5	$2^{+7.04}$

– ACC: Accuracy, TPR: True Positive Rate, TNR: True Negative Rate

– For $\mathcal{YD}$s, the thresholds τ's for $\sigma_\alpha(i)$'s in building $\mathcal{YD}^{\text{Speck}4R}$, $\mathcal{YD}^{\text{Speck}5R}$, $\mathcal{YD}^{\text{Speck}6R}$, $\mathcal{YD}^{\text{Speck}7R}$ are 0.50, 0.30, 0.20, and 0.02, respectively. The number of samples for the accuracy testing is 2^{24}.

– The number of samples for benchmark is 2^{20}. Thus, the times are seconds taken by making predictions on 2^{20} samples.

– All DDT-based distinguishers ($\mathcal{DD}$s, $\mathcal{YD}$s, $\mathcal{AD}_{\mathbf{YD}}$s, and $\mathcal{AD}_{\mathbf{KD}}$s) are implemented in `C++` (compiled using g++ 9.4.0 with optimization option '`-O3`'), whereas $\mathcal{ND}$-based distinguishers ($\mathcal{ND}$s) are implemented in `Python`.

– The benchmark environment is as follows: OS: Ubuntu 20.04; Processor: Intel(R) Xeon(R) Gold 6330 `CPU` @ `2.00GHz`; Memory: 256 `GB` `DDR4` memory; all timings were restricted to run using a single `CPU` thread.

– We profiled the memory requirements of $\mathcal{ND}$s using the `Tracemalloc` module in `Python`. We specifically measured the peak allocated memory, excluding the memory allocated for storing the testing dataset. This was calculated by determining the memory usage when loading the $\mathcal{ND}$ and making predictions, and then subtracting the memory usage when these operations were excluded (for example, 0.246898 GB − 0.219219 GB). For other distinguishers, we assessed memory requirements by referencing the '`RES`' column associated with the process in the '`htop`' command.

Table 10: The accuracy of differential-neural distinguishers using distinct differences obtained by (0040, 0000) after i rounds of propagation. Prob. represents the probability of the highest probability differential `(0040,0000)` → "Diff.".

i	Diff.	Prob.	Acc.	i	Diff.	Prob.	Acc.
0	`(0040,0000)`	1	0.6137	3	`(8000,840a)`	2^{-3}	0.7394
1	`(8000,8000)`	1	0.6137	4	`(850a,9520)`	2^{-7}	0.9166
2	`(8100,8102)`	2^{-1}	0.6705				

4 Insights and Improvements on Training Differential-Neural Distinguisher

4.1 Relations between Distinguisher Accuracy and Differential Distribution

Traditional differential cryptanalysis predominantly utilizes high-probability differentials as distinguishers. However, differential-neural cryptanalysis exploits all output differences for distinguishing while fixing input differences for plaintext pairs. In EUROCRYPT 2021, Benamira *et al.* [6] argued that differential-neural distinguisher is inherently building a very good approximation of the DDT during the learning phase.

Our study delves into the relation between the accuracy of the differential-neural distinguisher and the differential distribution of ciphertext pairs. We modify the input difference of plaintext pairs, inspired by Gohr's staged training method [14]. In [14], while the basic training method can produce a valid 7-round distinguisher, an 8-round distinguisher must be trained using the staged training approach. The core of the staged training method is training a pre-trained 7-round distinguisher to learn 5-round Speck32/64's output pairs with the input difference `(8000,804a)` (the most likely difference to appear three rounds after the input difference `(0040,0000)`). Employing such plaintext pairs aims to concentrate the difference distribution of ciphertext pairs, escalating the output difference's likelihood and simplifying the distinguisher's learning task.

In our work, we first introduce a 4-round highest probability differential trail starting from `(0040,0000)`.

`(0040,0000)` → `(8000,8000)` → `(8100,8102)` → `(8000,840a)` → `(850a,9520)`

Our experiments (see Table 10) initially employ a 4-round high-probability differential trail starting from `(0040,0000)`, leading to `(850a,9520)`.

By default, we use `(0040,0000)` as the input difference of the plaintext pair to generate the ciphertext pair. Here, in Table 10, we use the difference of the highest probability of `(0040,0000)` after i ($1 \leq i \leq 4$) rounds of propagation as the input difference of the plaintext pair, respectively.

From Table 8, we can observe that the larger i is, the higher the accuracy of the differential-neural distinguisher. As i increases, the difference distribution in

Table 11: The accuracy of the differential-neural distinguisher using distinct differences obtained by `(0040, 0000)` after 2 rounds of propagation. Prob. represents the probability of differential `(0040,0000)` → "Diff.". Round 2+i represents the positive sample of the training set is the ciphertext pair obtained by encrypting the plaintext pair that satisfies this difference for i rounds

Diff.	Prob.	Acc. (2+4)	Acc. (2+5)	Acc. (2+6)	Diff.	Prob.	Acc. (2+4)	Acc. (2+5)	Acc. (2+6)
(8100,8102)	2^{-1}	0.8720	0.6811	0.5270	(8f00,8f02)	2^{-4}	0.6746	Fail	Fail
(8300,8302)	2^{-2}	0.8191	0.6218	Fail	(9f00,9f02)	2^{-5}	Fail	Fail	Fail
(8700,8702)	2^{-3}	0.7492	Fail	Fail	(bf00,bf02)	2^{-6}	Fail	Fail	Fail

the ciphertext becomes more concentrated, and the probability of each difference increases. Therefore, the more significant the difference between the ciphertext and the random number, the accuracy of the differential-neural distinguisher is continuously improved.

To more comprehensively demonstrate the relation between the accuracy of the differential-neural distinguisher and the differential distribution of the ciphertext pairs, we conducted some experiments from another perspective. We fixed the number of rounds of differential but chose multiple 2-round differences with gradually decreasing probabilities. In Table 11, we notice that the higher the fixed probability of the differential, the higher the accuracy of the differential-neural distinguisher obtained. In other words, a lower probability means that after i rounds of encryption, the differential distribution of the ciphertext is more dispersed, and the neural network is more difficult to learn, resulting in a continuous decrease in the number of rounds and accuracy of the differential-neural distinguisher.

In conclusion, controlling differential propagation is imperative to enhance the differential-neural distinguisher's accuracy and the number of rounds. We thus propose a method to control the differential propagation and reduce the diffusion of features, thereby increasing the number of rounds of the differential-neural distinguisher. However, before the formal introduction, we introduce one method that can simplify the training process of high round distinguisher.

4.2 Freezing Layer Method

In existing experiments on Speck32/64, especially with an input difference of `(0040,0000)`, there has been a notable limitation. Researchers have been able to directly train a differential-neural distinguisher for up to only 7 rounds. Direct training for higher rounds from scratch has been challenging. A potential avenue that has garnered attention is the utilization of various network fine-tuning strategies. Specifically, continuing the training phase from pre-trained models has been proposed to potentially overcome these limitations and expand the distinguisher's round capability. Examples include the staged training method in [14] and the staged pipeline method in [5].

The inability to directly train the 8-round distinguisher likely stems from feature diffusion associated with the input difference (0040,0000) over increasing rounds. This makes the 8-round features considerably challenging for the distinguisher to learn directly from limited data, as compared to lower rounds. One approach is to either mitigate feature diffusion or narrow the distinguisher's solution space. While a technique to constrain feature diffusion is discussed in the subsequent chapter, in this context, we employ the classic network fine-tuning strategy, the freezing layer method, to limit the solution space.

Our distinguishers consist of two parts: the convolutional layers and fully connected layers. In the field of artificial intelligence, all convolutional layers are viewed as feature extractors, while all fully connected layers are viewed as a classifier. We argue that the feature extractor can be reused, and the classifiers are relatively similar in adjacent rounds. Therefore, to train an 8-round distinguisher for Speck32/64, we can simply load a well-trained 7-round model and freeze all its convolutional layers, meaning that only parameters in fully connected layers can be updated. Then, we can obtain an 8-round distinguisher with accuracy identical to the ones in [5, 14], remaining all hyperparameters in the training process unchanged.

Relative to the staged training method [14], our approach maintains the same hyperparameters and does not require more samples in the final stage. In comparison with the method in [5], we only need two training rounds instead of multiple rounds in a row as required by the simple training pipeline in [5]. Besides, the simple training pipeline [5] did not produce $\mathcal{ND}$s with the same accuracy as Gohr's on 8-round Speck32/64; it needs a further polishing step to achieve similar accuracy, demanding more time and data. Our freezing layer method also speeds up the training process due to the reduction of trainable parameters. Therefore, we recommend trying the freezing layer method once the number of the distinguisher is too high to train directly.

5 Related-Key Differential-Neural Cryptanalysis

The $\mathcal{ND}$ explainability concept serves as a fundamental theoretical underpinning when aiming to enhance and leverage its capabilities. With the outcome being that $\mathcal{ND}$s can effectively capture additional features and provide a better trade-off between efficiency and accuracy, there is substantial motivation for us to continue refining and exploiting their potential.

In this section, we introduce the related-key into differential-neural cryptanalysis, enabling control over differential propagation and facilitating the training of high-round $\mathcal{ND}$s. Furthermore, we enhance the DDT-based distinguisher under the $\mathcal{RK}$ setting by employing the analytical methods and conclusions outlined in Sect. 3. As a result of these advancements, we successfully implement a 14-round key recovery attack for Speck32/64 using the proposed $\mathcal{RK}$-$\mathcal{ND}$s.

5.1 Related-key Differential-Neural Distinguisher for Speck32/64

Here we present the related-key differential-neural distinguishers on Speck32/64 obtained in this work.

The choice of the input difference. The input difference is a crucial and central component of differential-neural cryptanalysis, and numerous papers delve into the study of the input difference, such as [3,5,14,16,22]. To maximize the number of rounds for both $\mathcal{ND}$ and $\mathcal{CD}$, as well as the weak key space as large as possible to perform the longest key recovery attack, we use the SMT-based method to search for appropriate $\mathcal{RK}$ differential or differential trails. It is important to note that the largest weak key space does not necessarily equate to the largest $\mathcal{ND}$ or $\mathcal{CD}$, thus requiring a compromise between the three factors. In this paper, the choice of the best input difference is given under different compromises. Table 12 lists the $\mathcal{RK}$ differential trails used to constrain the key space in Speck32/64, where we label each distinguisher with an ID. Specifically, ID_1 is used to restrict the weak key space for the 13-round, ID_2 and ID_3 are used to restrict the 14-round. Note that part of the $\mathrm{ID}_2/\mathrm{ID}_3$ (2-round to 11-round) $\mathcal{RK}$ difference are same as the 10-round optimal $\mathcal{RK}$ differential trail for speck32/64 given in Table 9 of [24]. In addition, the round-reduced of the trails are used to restrict the weak key space for shorter rounds, *e.g.*, ID_2 and ID_3 are used to restrict the weak key space for 13-round starting from the second round.

Network architecture. Given the success of the neural network consisting of the Inception block and residual network in Speck, Simon and Simeck [27,28], as well as its superior performance in differential-neural distinguisher, we use this neural network proposed in [28] to train $\mathcal{RK}$ differential-neural distinguisher. However, we also made some modifications to the network architecture. In deep learning, odd numbers such as 3, 5, and 7 are often used as the size of the convolution kernel. However, according to the cyclic shift of the round function of Speck32/64, we choose 2 and 7 as the size of the convolution kernel. Furthermore, using 2 as the convolution kernel size can make the model's accuracy converge faster than 3. In [28], the size of the convolution kernel continues to increase as the depth of the residual network increases. We think it is reasonable to increase the convolution kernel's size to improve the network's receptive field, but it cannot always be increased. Therefore, we will limit the size of the convolution kernel to less than or equal to 7.

The training of related-key differential-neural distinguisher. This work still uses the basic training method to train short-round distinguishers. When the basic training method fails, we train the r-round distinguisher with the $(r-1)$-round distinguisher by using the freezing layer method. Please refer to Appendix F for the detailed training method.

Performance evaluation of the distinguisher. In artificial intelligence, the model's accuracy is the most critical evaluation indicator. In differential-neural cryptanalysis, it is judged whether the guessed key is correct based on the score of the

Table 12: Related-key differential trails used to constrain the key space in Speck32/64 where we label each distinguisher with an ID. For example, ID$_1$ represents the 13-round $\mathcal{RK}$ differential trail for the key schedule algorithm with $(\Delta l^2, \Delta l^1, \Delta l^0, \Delta k^0) =$ `(0044,0011,4000,0080)`

	ID$_1$		ID$_2$/ID$_3$	
r	Differential in Key	$\log_2$ Pr	Differential in Key	$\log_2$ Pr
0	`(0044,0011,4000,0080)`		`(0200,0080,0011,4a00)`	
1	`(0000,0044,0011,0200)`	-1	`(2800,0200,0080,0001)`	-4
2	`(2000,0000,0044,2800)`	-2	`(0000,2800,0200,0004)`	-1
3	`(a000,2000,0000,0000)`	-2	`(0000,0000,2800,0010)`	-1
4	`(0000,a000,2000,0000)`	-0	`(0040,0000,0000,0000)`	-2
5	`(0040,0000,a000,0040)`	-1	`(0000,0040,0000,0000)`	0
6	`(0100,0040,0000,0000)`	-2	`(0000,0000,0040,0000)`	0
7	`(0000,0100,0040,0000)`	0	`(8000,0000,0000,8000)`	0
8	`(8000,0000,0100,8000)`	0	`(8000,8000,0000,8002)`	0
9	`(8002,8000,0000,8000)`	-1	`(8002,8000,8000,8008)`	-1
10	`(8000,8002,8000,8002)`	0	`(8108,8002,8000,812a)`	-2
11	`(8102,8000,8002,8108)`	-2	`(802a,8108,8002,8480)`	-4
12	`(8408,8102,8000,802a)`	-3	`(8180,802a,8108,9382)`/ `(8280,802a,8108,9082)`	-3/-4
13			`(8180,8180,802a,cf8a)`/ `(8080,8280,802a,c28a)`	-4/-4
	$\log_2 (P_r (Q_K))$: -14		$\log_2 (P_r (Q_K))$: -22/-23	

distinguisher. Therefore, we evaluate the performance of the differential-neural distinguisher regarding both the accuracy and the score.

- *Test accuracy.* We summarize the accuracy of the differential-neural distinguisher in Table 13. The 8, and 9-round distinguishers were trained using the basic training method, while the 10-round distinguishers were trained using the freezing layer method. For more insight on related-key differential-neural distinguishers, please refer to Appendix F.2.
- *Wrong key response profile (WKRP).* In [14], the key search policy depends on the observation that a distinguisher's response to wrong-key decryption varies with the bitwise difference between the guessed and real key. Instead of exhaustive trial decryption, it suggests specific subkeys and scores them. Fig. 3 shows the mean response for varying Hamming distances between guessed and actual keys in ID$_1$. Notably, high scores emerge when differences in keys are small, especially if the difference relates to {16384, 32768, 49152}. This indicates that errors in the 14th and 15th bits of the subkey minimally impact scores, allowing for a reduced key guessing space. This accelerated key recovery in [14]. For WKRPs of ID$_2$ and ID$_3$, see Appendix B.2.

Table 13: The summary of related-key differential-neural distinguishers on Speck32/64, where the plaintext difference is `(0000,0000)`.

Diff.	#R	Name	Accuracy	True Positive Rate	True Negative Rate
ID_1	8	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}8R}$	0.7584	0.6836	0.8332
ID_1	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}$	0.5620	0.5212	0.6028
ID_2/ID_3	8	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}8R}$	0.9259	0.9063	0.9455
ID_2	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}$	0.7535	0.7035	0.8036
ID_2	10	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}10R}$	0.5643	0.5382	0.5893
ID_3	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}$	0.7726	0.7247	0.8206
ID_3	10	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}10R}$	0.5562	0.5361	0.5765

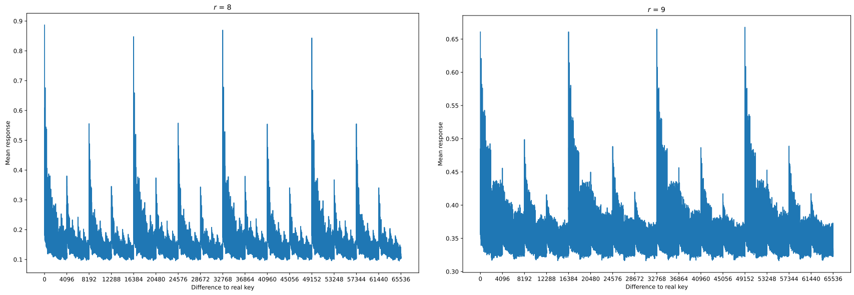


Fig. 3: Wrong key response profile of ID_1

On $\mathcal{RK}$-$\mathcal{ND}$'s Explainability. Beyond constructing and comparing various $\mathcal{RK}$ distinguishers (see Appendix F.2), we further undertook experiments analogous to Gohr's *aaaa*-blinding experiment. Some $\mathcal{RK}$-$\mathcal{ND}$s behaved similarly to single-key setting $\mathcal{ND}$s, while others varied. Refer to $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(2,9182)}}$ and $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(2,9382)}}$ in Table 14 for example for the former and latter case, where the differential trail $\text{ID}_{(2,9182)}$ differs from $\text{ID}_{(2,9382)}$ only at the last round key, and $\text{ID}_{(2,9382)}$ is ID_2 from round 4 to 12. Notably, the behavior of $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(3,9082)}}$ presented intriguing phenomena ($\text{ID}_{(3,9082)}$ is ID_3 from round 4 to 12):

1. $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(3,9082)}}$ performed differently on `Set-1-1` $:= \{\Gamma_{\mathcal{A}}, \Gamma_{\mathcal{B}}, \Gamma_{\mathcal{C}}, \Gamma_{\mathcal{D}}\}$ and `Set-1-2` $:= \{\Gamma_{\mathcal{AR}_1}, \Gamma_{\mathcal{BR}_1}, \Gamma_{\mathcal{CR}_1}, \Gamma_{\mathcal{DR}_1}\}$, which, under the assumption of a random last-round key K, defines the same information set per Sect. 3.1 (please refer to Table 14).
2. $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(3,9082)}}$ showed superior performance over $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(2,9382)}}$ (0.7726 vs. 0.7535, refer to Table 22), while theoretically, if there is no information on the key being revealed beyond the key difference, $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(3,9082)}}$ should perform exactly the same as $\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}9R}_{\text{ID}_{(2,9382)}}$, since the two differential trails

Table 14: Experiments detailing the information harnessed by $\mathcal{RK}$-$\mathcal{ND}$s using 9 round $\text{ID}_{(2,9182)}$, $\text{ID}_{(2,9382)}$, and $\text{ID}_{(3,9082)}$, with similar settings in Table 2.

ID	Set.	Positive Samples	Negative Samples	Acc.
$\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}{}_{\text{ID}_{(2,9182)}}$	1-1	$(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D})$	*Random*	0.7531
	1-2	$(\mathcal{AR}_1,\mathcal{BR}_1,\mathcal{CR}_1,\mathcal{DR}_1)$	*Random*	0.7534
$\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}{}_{\text{ID}_{(2,9382)}}$	1-1	$(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D})$	*Random*	0.7574
	1-2	$(\mathcal{AR}_1,\mathcal{BR}_1,\mathcal{CR}_1,\mathcal{DR}_1)$	*Random*	0.7529
$\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}{}_{\text{ID}_{(3,9082)}}$	1-1	$(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D})$	*Random*	0.7746
	1-2	$(\mathcal{AR}_1,\mathcal{BR}_1,\mathcal{CR}_1,\mathcal{DR}_1)$	*Random*	0.7539

differ only at the last round key difference thus the two output difference distributions are affine-equivalent.

3. Surprisingly, $\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}{}_{\text{ID}_{(3,9082)}}$ even outperformed our manually enhanced distinguisher $\mathcal{RK}$-$\mathcal{AD}^{\text{SPECK}9R}_{\mathbf{YD}}$ (0.7726 vs. 0.7574, refer to Table 22).

Upon closer examination of the differential trail of $\text{ID}_{(3,9082)}$, we identified the causative factor. Let's denote input/output differences and values around the last $\boxplus$ in the key schedule producing the 8-round (counting start from 0) key k^8 as $\alpha, \beta, \gamma, x, y, z$. Then from the differential trail $\text{ID}_{(3,9082)}$, specifically focus on the 7- and 8-round, we have
$$\begin{cases} \alpha = \texttt{0x8002}^{\lll 7} & = \texttt{0b 0000 0101 0000 0000}, \\ \beta = \texttt{0x8480} & = \texttt{0b 1000 0100 1000 0000}, \\ \gamma = \texttt{0x8280} & = \texttt{0b 1000 0010 1000 0000}, \end{cases}$$
According to Tables 4 and 5, we have follows.

1. The $(8,7)$-th bit position is in case $\text{Cxc1}_{(8,7)}$, we have $c_8 = y_7$.
2. The $(9,8)$-th bit position is in case $\text{Cxc0}_{(9,8)}$, we have $c_9 = c_8$.
3. The $(10,9)$-th bit position is in case $\text{Cxy1}_{(10,9)}$, we have $x_9 \oplus y_9 = z_9 \oplus c_9 = 1$.

Consequently, we have $z_9 \oplus y_7 = 1$. Note that $z_9 \oplus y_7 = 1$ implies that the 9th bit of the last round key is constantly 1. This does not obscure 1-bit information of the output of the last $\boxplus$ in the encryption path, allowing for better accuracy of the resulting distinguisher. This explains all the odds on $\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}_{\text{ID}_{(3,9082)}}$.

Additionally, for $\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}_{\text{ID}_{(2,9382)}}$, the 10th bit of the last-round key conforming to the round difference has a bias towards 0 (equals 0 with a probability of 3/4), which could explain its slightly differed accuracy between **Set.**1-1 and **Set.**1-2 (refer to Table 14). After fixing the 10th bit to be 0 and re-training the distinguisher, it achieves almost the same accuracy as $\mathcal{RK}$-$\mathcal{ND}^{\text{SPECK}9R}_{\text{ID}_{(3,9082)}}$. When analyzing the probability of related-key pairs under these conditions, we deduced that restricting the 10th bit for $\text{ID}_{(2,9382)}$ still results in a larger weak-key space compared with $\text{ID}_{(3,9082)}$ while achieving the same high $\mathcal{RK}$-$\mathcal{ND}$ accuracy.

5.2 Key Recovery Attack on Round-Reduced SPECK32/64

This subsection describes the implementation of $\mathcal{RK}$ differential-neural cryptanalysis using the trained distinguisher. The key recovery framework is similar to [3,14,28]. Since the whole attack is in the $\mathcal{RK}$ setting, we need to specify the difference between each round of subkeys. Specifically, it is unclear how to perform a key recovery attack if only applying a difference to the master key without specifying the difference in the round-key state. In such cases, the guessed one last-round key cannot directly infer the other last-round key in the related pair, as the difference in the last-round key is not specified.

We first introduce some preparatory work before officially implementing the key recovery attack.

Generalized neutral bits. We incorporate $\mathcal{CD}$ before $\mathcal{ND}$ to increase the number of rounds for the key recovery attack. Furthermore, to enhance predictive performance, we employ the distinguisher to estimate the scores of multiple ciphertexts with the same distribution (ciphertext structure) and combine them to obtain the scores for the guessed subkey. However, the $\mathcal{CD}$ is probabilistic, and the randomly generated plaintext structure does not retain the same distribution after encryption. Hence, we require neutral bits to generate the plaintext structure, which we encrypt to obtain the ciphertext structure, achieving a successful key recovery attack. Therefore, the $\mathcal{CD}$ should have a high probability and a sufficient number of neutral bits. Appendix B.3 lists the NBs/SNBSs we used to perform the key recovery attack.

The parameters for key recovery attack. The attacks follow the framework of the improved key recovery attacks in [14]. An r-round main and an $(r-1)$-round helper $\mathcal{ND}s$ are employed, and an s-round $\mathcal{CD}$ is prepended. The key guessing procedure applies a simple reinforcement learning procedure. The last subkey and the second to last subkey are to be recovered without exhaustively using all candidate values to perform one-round decryption. Moreover, a Bayesian key search employing the *wrong key response profile* will be used. We count a key guess as successful if the last round key was guessed correctly and if the second round key is at the hamming distance at most two of the real keys. The parameters to recover the last two subkeys are indicated below.

Parameter	Definition
n_{cts}	The number of ciphertext structures.
n_b	The number of ciphertext pairs in each ciphertext structure, that is, $2^{\lvert \text{NB} \rvert}$.
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.

Complexity evaluation of key recovery attack. The experiment is conducted by Python 3.7.15 and Tensorflow 2.5.0 in Ubuntu 20.04. The device information is

Table 15: Summary of key recovery attacks on Speck32/64

Diff.	#R	Configure	wks	n_{cts}	n_{it}	n_b	c_1	c_2	sr	Time	Data	Advantage
ID$_1$	13	1+2+9+1	2^{50}	2^7	2^8	2^8	8	5	54.28%	$2^{34.57}$	2^{16}	$2^{15.43}$
ID$_2$	13	1+2+9+1	2^{46}	2^6	2^7	2^5	5	5	93.33%	$2^{33.95}$	2^{12}	$2^{12.05}$
				2^5	2^6	2^5	5	5	72.86%	$2^{33.01}$	2^{11}	$2^{12.99}$
				2^4	2^5	2^5	10	1	44.28%	$2^{31.79}$	2^{10}	$2^{14.21}$
	14	1+3+9+1	2^{42}	2^9	2^{10}	2^6	8	10	75.71%	$2^{35.59}$	2^{16}	$2^{6.41}$
				2^9	2^{10}	2^5	8	5	55.71%	$2^{35.32}$	2^{15}	$2^{6.68}$
ID$_3$	13	1+2+9+1	2^{45}	2^6	2^7	2^5	5	5	95.24%	$2^{34.26}$	2^{12}	$2^{10.75}$
				2^5	2^6	2^5	5	5	77.62%	$2^{33.55}$	2^{11}	$2^{11.45}$
				2^4	2^5	2^5	10	1	46.67%	$2^{32.20}$	2^{10}	$2^{12.80}$
	14	1+3+9+1	2^{41}	2^{10}	2^{11}	2^7	10	25	90%	$2^{36.39}$	2^{18}	$2^{4.61}$
				2^9	2^{10}	2^7	10	15	71.43%	$2^{35.78}$	2^{17}	$2^{5.22}$
				2^9	2^{10}	2^5	5	5	68.57%	$2^{35.40}$	2^{15}	$2^{5.6}$

Intel Xeon E5-2680V4*2 with 2.40GHz, 256GB RAM, and NVIDIA RTX3080Ti 12GB*7. To reduce the experimental error, we perform 210 key recovery attacks for each parameter setting, take the average running time rt as the running time of an experiment, and divide the number of successful experiments by the total experimental number as the success rate sr of the key recovery attack.

1. *Data complexity.* The data complexity of the experiment is calculated using the formula $n_b \times n_{ct} \times 2$, which is a theoretical value. In the actual experiment, when the accuracy of the differential-neural distinguisher is high, the key can be recovered quickly and successfully. Not all data are used, so the actual data complexity is lower than theoretical.

2. *Time complexity.* We use 2^{32} data to test the speed of encryption and decryption on our device, and each core can perform $2^{26.814}$ rounds of decryption operations per second for Speck32/64. The formula for calculating the time complexity in our experiments: $2^{26.814} \times rt$.

The result of key recovery attacks. We list the results of key recovery attacks in multiple differential modes in Table 15. We calculate the corresponding weak key space wks according to the probabilities of ID$_1$, ID$_2$, and ID$_{(3,9082)}$. Adv. represents the advantage compared to the time complexity of brute forcing. The time and data complexity can be reduced by reducing n_{cts} and n_{it}, but the success rate sr also decreases accordingly. The first metric for our experiment is to reduce the time complexity.

Remark 3 (The profiling information of the key-recovery attack). To pinpoint the attack's bottleneck, we profiled a 14-round key-recovery attack using ID3. The main result is detailed in Table 16. From the profiling result, the performance of our implementation of the attack is mostly limited by the speed of neural

Table 16: Profiling information of the key-recovery attack

Function	Time (Percentage)
test_bayes	242 s (100 %)
| – bayesian_key_recovery	| – 229.39 s (94.79 %)
| – | – (GPU) net.predict	| – | – 191.62 s (79.18 %)
| – | – (CPU) bayesian_rank_kr	| – | – 28.99 s (11.98 %)
| – verifier_search	| – 12.63 s (5.22 %)
| – | – (GPU) net.predict	| – | – 12.51 s (5.17 %)

– test_bayes: a full run of the attack excluding the generation of the related-key, load models, and generation of ciphertext structures.

– bayesian_key_recovery: the run of the BayesianKeySearch algorithm.

– verifier_search: the run of the final improvement [3].

– net.predict: using $\mathcal{ND}$s to score the ciphertext structures decrypted by one round.

– bayesian_rank_kr: computing the weighted Euclidean distance with WKRPs.

network evaluation (the proportion taken by $\mathcal{ND}$ making the prediction is 79.18 % + 5.17 % = 84.35 %). The next limiting factor is the speed of computing the weighted Euclidean distance with the wrong key response profile.

Remark 4 (Efficiency measures in symmetric-key cryptanalysis attacks). Assessing the efficiency of distinguishers and key recovery attacks in symmetric-key cryptanalysis poses intricate challenges, particularly when pinpointing computational complexities based on real-time attack timings and then extrapolating these to equivalent primitive evaluations, as done in both $\mathcal{ND}$-based and traditional attacks in [11, 13, 25] (listed in Table 1).

Factors influencing these complexities include architecture compatibility and algorithmic suitability, varied computation intensity and various operation costs across platforms, memory constraints and flexible trade-offs, and implementation factors. Given these complexities, it is a good idea to have secondary metrics for comparison, for instance, power consumption and cost efficiency (please refer to Appendix E for detailed discussions). While there's a pressing need for universal metrics, formulating such benchmarks is challenging, warranting caution when interpreting the comparison results and warranting further exploration.

6 Conclusion

This paper provides explicit rules that a distinguisher can use beyond the full differential distribution table to achieve better distinguishing performance. These rules are based on high correlations between values of bits in right pairs of differential propagation through addition modular 2^n. By leveraging the value-dependent differential probability, which is not typically applied in traditional differential distinguishers, we can equip additional knowledge to DDT-based distinguishers, enhancing their accuracy. These rules or their equivalent form are likely the additional features beyond full DDT that the neural distinguishers exploit. While these rules are not difficult to derive with careful analysis, they rely on non-trivial relations that traditional distinguishers often overlook. This indicates that neural networks help break the limitations of traditional cryptanalysis. Studying this unorthodox model can provide new opportunities to understand cryptographic primitives better.

Another investigation in this paper revealed that controlling differential propagation is crucial to enhance the accuracy of differential-neural distinguisher. It is typically believed that introducing differences into the keys provides chances to cancel differences in the encryption states, thus resulting in stronger differential propagations. However, unlike traditional differential attacks, differential-neural attacks do not specify the output difference and, thus, are not limited to a single differential trail. Therefore, it is unclear whether the difference in a key is helpful in differential-neural attacks. It is also unclear how resistant Speck is against differential-neural attacks in the $\mathcal{RK}$ setting. This work confirmed that differential-neural cryptanalysis in the $\mathcal{RK}$ setting could be more powerful than in the single-key setting by conducting a 14-round key recovery attack on Speck32/64.

Acknowledgments

The authors would like to thank anonymous reviewers for their insightful and helpful comments. Special thanks go to the anonymous shepherd, who provided invaluable constructive feedback and patient guidance, which helped us improve the manuscript significantly. This research is partially supported by Nanyang Technological University in Singapore under Start-up Grant 04INS000397C230, and Ministry of Education in Singapore under Grants RG91/20. Zhenzhen Bao was partially supported by the National Key R&D Program of China (Grant No. 2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008), the High Performance Computing Center of Tsinghua University. Jinyu Lu was supported by the National Natural Science Foundation of China (Grant No. 62172427). Liu Zhang was supported by the National Natural Science Foundation of China (Grant No. 62172319), and the Fundamental Research Funds for the Central Universities (Grant No. QTZX23090).

References

1. Source codes in this work (2023), https://www.dropbox.com/sh/yleufeiu0wqwcjv/AADUpM15q86Uk1lM8z99fU2ia?dl=0
2. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Enhancing differential-neural cryptanalysis. Cryptology ePrint Archive, Report 2021/719 (2021), https://eprint.iacr.org/2021/719
3. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Enhancing differential-neural cryptanalysis. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol.

13791, pp. 318–347. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22963-3_11

4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), https://eprint.iacr.org/2013/404
5. Bellini, E., Gerault, D., Hambitzer, A., Rossi, M.: A cipher-agnostic neural training pipeline with automated finding of good input differences. Cryptology ePrint Archive, Report 2022/1467 (2022), https://eprint.iacr.org/2022/1467
6. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 805–835. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_28
7. Beyne, T., Rijmen, V.: Differential cryptanalysis in the fixed-key model. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 687–716. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_23
8. Biryukov, A., dos Santos, L.C., Teh, J.S., Udovenko, A., Velichkov, V.: Meet-in-the-filter and dynamic counting with applications to speck. IACR Cryptol. ePrint Arch. p. 673 (2022)
9. Chen, Y., Bao, Z., Shen, Y., Yu, H.: A deep learning aided key recovery framework for large-state block ciphers. Cryptology ePrint Archive, Report 2022/1659 (2022), https://eprint.iacr.org/2022/1659
10. Chen, Y., Yu, H.: Bridging machine learning and cryptanalysis via EDLCT. Cryptology ePrint Archive, Report 2021/705 (2021), https://eprint.iacr.org/2021/705
11. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 147–164. Springer (2014)
12. Dinur, I., Dunkelman, O., Shamir, A.: Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 219–240. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-662-43933-3_12
13. Feng, Z., Luo, Y., Wang, C., Yang, Q., Liu, Z., Song, L.: Improved differential cryptanalysis on speck using plaintext structures. In: Australasian Conference on Information Security and Privacy. pp. 3–24. Springer (2023)
14. Gohr, A.: Improving attacks on round-reduced Speck32/64 using deep learning. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 150–179. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_6
15. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39. pp. 150–179. Springer (2019)
16. Gohr, A., Leander, G., Neumann, P.: An assessment of differential-neural distinguishers. Cryptology ePrint Archive, Report 2022/1521 (2022), https://eprint.iacr.org/2022/1521
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (Poster) (2015)
18. Lee, H., Kim, S., Kang, H., Hong, D., Sung, J., Hong, S.: Calculating the approximate probability of differentials for arx-based cipher using sat solver. Journal of the Korea Institute of Information Security & Cryptology **28**(1), 15–24 (2018)

19. Leurent, G.: Analysis of differential attacks in ARX constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 226–243. Springer, Heidelberg (Dec 2012). `https://doi.org/10.1007/978-3-642-34961-4_15`
20. Leurent, G.: Construction of differential characteristics in ARX designs application to Skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 241–258. Springer, Heidelberg (Aug 2013). `https://doi.org/10.1007/978-3-642-40041-4_14`
21. Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (Apr 2002). `https://doi.org/10.1007/3-540-45473-X_28`
22. Lu, J., Liu, G., Liu, Y., Sun, B., Li, C., Liu, L.: Improved neural distinguishers with (related-key) differentials: Applications in SIMON and SIMECK. Cryptology ePrint Archive, Report 2022/030 (2022), `https://eprint.iacr.org/2022/030`
23. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 370–392. Springer, Heidelberg (Aug 2010). `https://doi.org/10.1007/978-3-642-14623-7_20`
24. Sadeghi, S., Rijmen, V., Bagheri, N.: Proposing an milp-based method for the experimental verification of difference-based trails: application to speck, simeck. Designs, Codes and Cryptography **89**, 2113–2155 (2021)
25. Song, L., Huang, Z., Yang, Q.: Automatic differential analysis of arx block ciphers with application to speck and lea. In: Australasian Conference on Information Security and Privacy. pp. 379–394. Springer (2016)
26. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. IACR Trans. Symm. Cryptol. **2021**(1), 269–315 (2021). `https://doi.org/10.46586/tosc.v2021.i1.269-315`
27. Zhang, L., Lu, J., Wang, Z., Li, C.: Improved differential-neural cryptanalysis for round-reduced simeck32/64. arXiv preprint arXiv:2301.11601 (2023)
28. Zhang, L., Wang, Z., Wang, B.: Improving differential-neural cryptanalysis with inception blocks. Cryptology ePrint Archive, Report 2022/183 (2022), `https://eprint.iacr.org/2022/183`
29. Zhang, Z., Hou, C., Liu, M.: Collision attacks on round-reduced SHA-3 using conditional internal differentials. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 220–251. Springer, Heidelberg (Apr 2023). `https://doi.org/10.1007/978-3-031-30634-1_8`

A Network Architecture

The general architecture of our neural network to train the differential-neural distinguisher is shown in Fig. 4. The network architecture consists of four parts: an input layer consisting of multiple-ciphertext pairs, an initial convolutional layer consisting of four parallel convolutional layers, a residual tower with multiple two-layer convolutional neural networks, and a prediction head consisting of multiple fully connected layers.

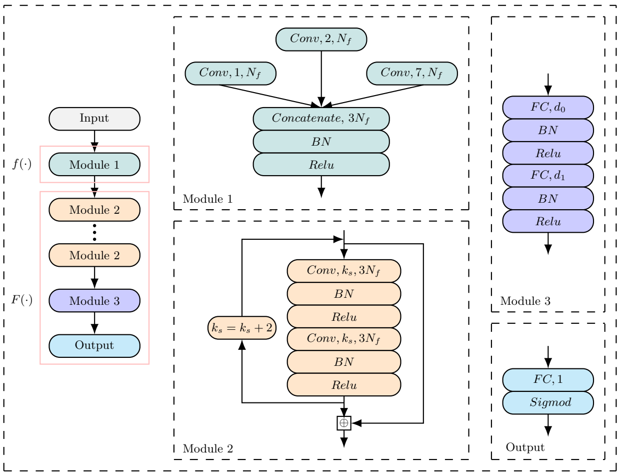


Fig. 4: The network architecture for Speck32/64

- *Input representation.* If the output of the r-th round $(C, C') = (C_L^r || C_R^r, {C'}_L^r || {C'}_R^r)$ is known, one can directly compute $(C_R^{r-1}, {C'}_R^{r-1})$ without knowing the $(r-1)$-th subkey according to the round function of Speck. Thus, the neural network accepts data of the form $(C_L^r, {C'}_L^r, C_R^r, {C'}_R^r, C_R^{r-1}, {C'}_R^{r-1})$. The input layer has $6n$ units likewise arranged in a $[n, 6]$ array, where $n = 16$ for Speck32/64.

- *Initial convolution (module 1).* The input layer is connected to the initial convolutional layer, which comprises three convolutional layers with $N_f = 16$ channels of different kernel sizes. The three convolution layers are concatenated at the channel dimension. Batch normalization is applied to the output of the concatenate layers. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[n, 3N_f]$ matrix is passed to the convolutional blocks layer.

- *Convolutional blocks (module 2).* Each convolutional block consists of two layers of $3N_f$ filters. Each block applies first the convolution with kernel

Table 17: Hyperparameters of network architecture and training process

Parameter	Value	Parameter	Value
Epoch	20	Filters	32
Batch Size	5000	Regularization Factor	0
#Residual Blocks (depth)	10	#Densely Connected Neurons	512, 256, 64
Circular Learning Rate		$3.5e-3$ to $2e-4$ every 10 epochs	

size k_s, then a batch normalization, and finally a rectifier layer. At the end of the convolutional block, a skip connection is added to the output of the final rectifier layer of the block to the input of the convolutional block. It transfers the result to the next block. After each convolutional block, the kernel size increases by 2 if $k_s < 7$. The number of convolutional blocks is 5 in our model (determined by experiment).

- *Prediction head (module 3 and output).* The prediction head consists of two hidden layers and one output unit. Before the first hidden layer, we add a dropout layer to prevent model overfitting. The two fully connected layers comprise 64, and 64 units, followed by the batch normalization and rectifier layers. The final layer consists of a single output unit using the activation function *Sigmoid.*

Rationale. First, we take the ciphertext of the last round and the right half of the penultimate round as input, hoping to provide more information to the neural network. Second, the purpose of using multiple convolutional layers with different kernel sizes is to capture information in multiple dimensions. The convolution layer with kernel size 1 is to capture the differential features in the ciphertext pairs. In the field of deep learning, odd numbers such as 3, 5, and 7 are often used as the size of the convolution kernel. But according to the cyclic shift of the speck round function, we choose 2 and 7 as the size of the convolution kernel. In addition, using 2 as the convolution kernel size can make the accuracy of the model converge faster than 3. Third, to increase the convolution's receptive field, the convolution kernel's size increases by 2 with the increase of the depth of the Residual Network.

A.1 Hyperparameters for Training $\mathcal{ND}$s used in Sect. 3.1

B Differential-Neural Cryptanalysis

B.1 Framework of Key Recovery Attack

Gohr [14] proposed a framework for differential neural cryptanalysis dedicated to recovering the last two rounds of subkeys for Speck32/64. We decrypt the ciphertext using the guessed subkey and use the differential-neural distinguisher to estimate the distance between the guessed subkey and the real key.

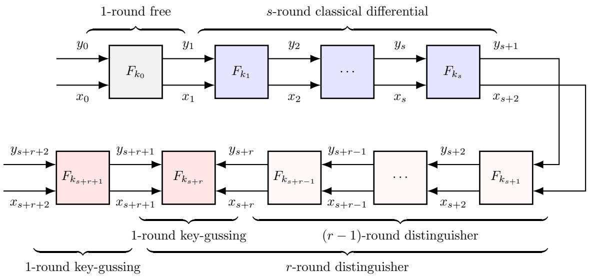


Fig. 5: $(1+s+r+1)$-round key recovery attack [3]

The overall processing of a key recovery attack based on a differential-neural distinguisher is shown in Fig. 5, 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 ΔP. 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 ΔP after s-round encryption, about $c \cdot 2^p$ (denoted by n_{cts}) data pairs with the difference ΔS are required according to the probability of difference propagation, where c is a small constant.

Neutral bits (NB) of the s-round classical differential is 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 Speck32/64. 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* (UCB). 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 response $v_{i,k}$ 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 Alg. [14].

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 $s_k = \sum_{i=0}^{n_b-1} \log_2\left(\frac{v_{i,k}}{1-v_{i,k}}\right)$ as the score of the recommended subkey by using large amounts of instances with the same distribution, which can be satisfied by NB [14]. 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 [14]. 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 [3].

B.2 Wrong Key Response Profile

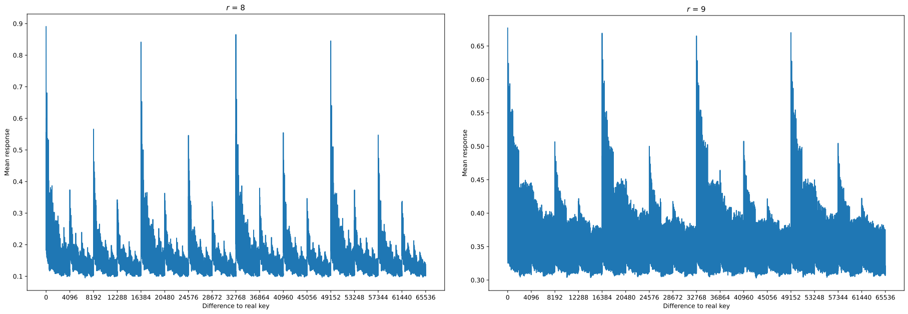


Fig. 6: Wrong key response profile of ID_2

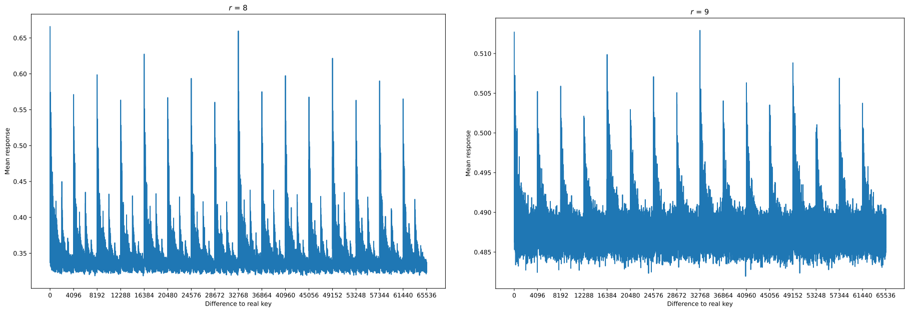


Fig. 7: Wrong key response profile of ID_3

B.3 The (probabilistic) NBs/SNBSs of Speck32/64

Table 18: The (probabilistic) NBs/SNBSs of Speck32/64. The statistics were performed on 1000 correct pairs, each with a different related-key

$\texttt{(0881,0005)} \xrightarrow[\texttt{ID}_1]{\text{2-round } (P_r(Q_D)=2^{-6})} \texttt{(0000,0000)}$									
NBs	[13,20]	[2]	[27]	[0,23]	[15,22]	[14]	[21]	[28]	
Prob.	1.0	0.99	0.99	0.99	0.95	0.91	0.91	0.91	
$\texttt{(0205,0200)} \xrightarrow[\texttt{ID}_2/\texttt{ID}_3]{\text{2-round } (P_r(Q_D)=2^{-4})} \texttt{(0000,0000)}$									
NBs	[12]	[13]	[18]	[19]	[20]	[21]	[22]	[9,16]	[4,27]
Prob.	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.97
$\texttt{(02a1,4001)} \xrightarrow[\texttt{ID}_2/\texttt{ID}_3]{\text{3-round } (P_r(Q_D)\approx 2^{-8})} \texttt{(0000,0000)}$									
NBs	[0,23,25]	[11,18]	[4,27]	[19]	[12]	[28]			
Prob.	0.99	0.89	0.80	0.79	0.77	0.77			

C Algorithm for Computing XOR-Differential Probability of Addition

Theorem 1 ([21]). *Let $\delta = (\alpha, \beta \mapsto \gamma)$ be an arbitrary XOR-differential through addition modulo 2^n. Alg. 3 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 $\texttt{w}_\texttt{h}$.*

Algorithm 3: Compute $\mathrm{DP}^+(\delta)$ [21]

```
INPUT: $\delta = (\alpha, \beta \mapsto \gamma)$
OUTPUT: $\mathrm{DP}^+(\delta)$
 1. If $\texttt{eq}(\alpha^{\lll 1}, \beta^{\lll 1}, \gamma^{\lll 1}) \wedge (\texttt{xor}(\alpha, \beta, \gamma) \oplus (\beta^{\lll 1})) \neq 0$ then return 0;
 2. Return $2^{-\texttt{w}_\texttt{h}(\neg \texttt{eq}(\alpha, \beta, \gamma) \wedge \texttt{mask}(n-1))}$;
```

D More Details on $\mathcal{ND}$ Explainability

In Table 19, we list some output differences whose probabilities are higher than 2^{-32}.

Table 19: Experimental results of **Experiment** A. Prop. refers to the proportion of predicted positive labels. The probabilities of these output differences are all higher than 2^{-32}.

Difference	Prop.	Difference	Prop.	Difference	Prop.	Difference	Prop.
`(802a,d4a8)`	0.75	`(803a,d4b8)`	0.75	`(822a,d6a8)`	0.75	`(802e,d4ac)`	0.75
`(8e2a,daa8)`	0.75	`(b82a,eca8)`	0.75	`(882a,dca8)`	0.75	`(a02a,f4a8)`	0.75
`(806a,d4e8)`	0.75	`(801a,d498)`	0.75	`(be2a,eaa8)`	0.75	`(8e26,daa4)`	0.75
`(8026,d4a4)`	0.75	`(a026,f4a4)`	0.75	`(be26,eaa4)`	0.75	`(83ea,d768)`	0.75
`(883a,dcb8)`	0.75	`(801e,d49c)`	0.75	`(be1a,ea98)`	0.75	`(821a,d698)`	0.75

D.1 Applying to Gift64/128

Although the above observations are based on specific analysis of the differential properties of Speck's modulo addition component, the analysis methods and conclusions are actually applicable in a general sense.

Specifically, this analysis also applies to differential neural distinguishers on SBox-based substitution-permutation network (SPN) block ciphers.

For example, let's consider the block cipher Gift with a block size of 64 bits and a key size of 128 bits. Fig. 8 is the schematic diagram of its two round functions. As can be seen, in each round, only half of the state is XORed with the round key. Therefore, given a pair of ciphertexts, not only the output difference of the final round's nonlinear layer (*i.e.*, S-box layer) is known, but also half of the state values are known.

Based on this property, one can construct r-round distinguishers for Gift by using conditional differential distribution tables for the inverse S-box of Gift under known partial values (refer to Fig. 9) and based on $(r-1)$ round distinguishers, similar to Alg. 2 used to construct $\mathcal{AD}_{\mathbf{YD}}$.

Due to the 64-bit block size of Gift64/128, we cannot calculate an exact $(r-1)$ round DDT as we did for Speck32/64. However, we can use the $(r-1)$ round neural distinguisher for construction. Furthermore, we can go beyond the boundary of the round function and construct r-round distinguishers based on $(r-1)$.75- and $(r-1)$.5-round neural distinguishers, as well as vDDTs with partial values for 4 S-boxes or 8 S-boxes (refer to Figures 10 and 11).

The resulting r-round distinguishers demonstrate significantly higher accuracy compared to distinguishers based on ciphertext differences and are comparable to, or even superior to, pure neural distinguishers based on ciphertext values (refer to Table 20) [8]. This indicates that neural distinguishers with ciphertext values as inputs not only utilize differential distributions but also exploit value-based conditional differential distributions.

D.2 Applying to Classical Cryptanalysis

The conclusion is not restricted to neural distinguishers and can be used to enhance classical differential cryptanalysis. Concretely, one has the following.

[8] Please refer to [1] for the implementation codes and experimental results.

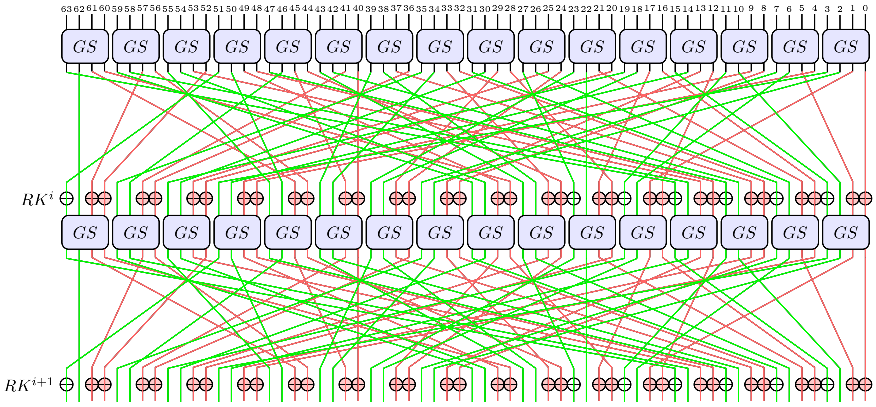


Fig. 8: Tow rounds of GIFT64/128

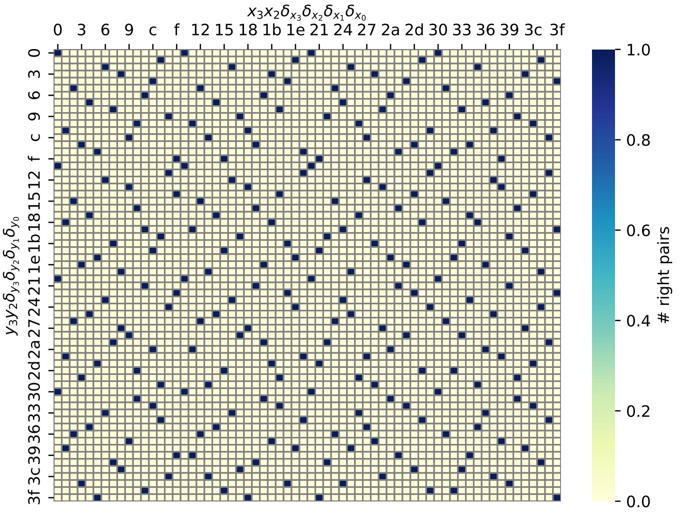
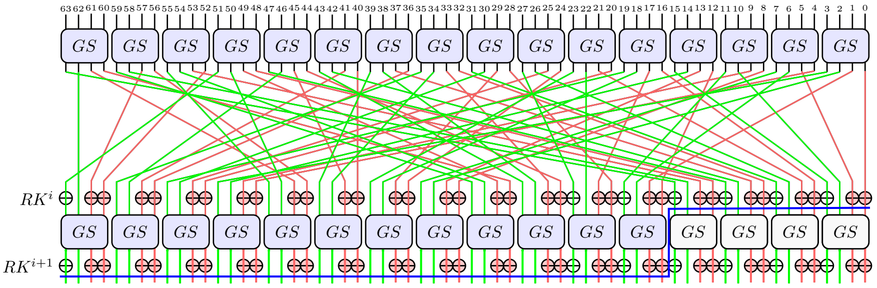
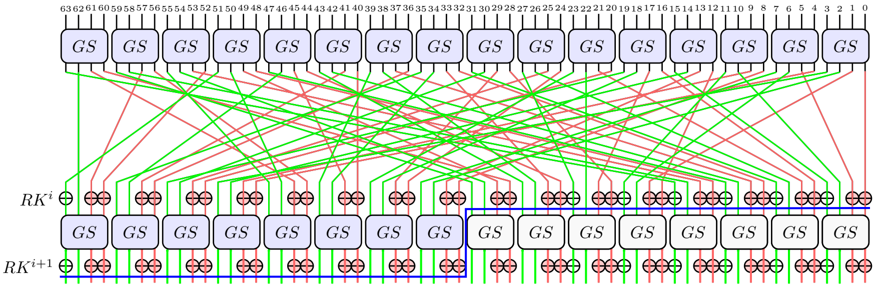


Fig. 9: The vDDT of GIFT64/128's inverse SBox: vDDT$_{y_3y_2\delta_{y_3}\delta_{y_2}\delta_{y_1}\delta_{y_0}\to x_3x_2\delta_{x_3}\delta_{x_2}\delta_{x_1}\delta_{x_0}}$. Based on the computation, there is only one right pair for every valid conditional differential propagation and there are only four possible output $x_3x_2\delta_{x_3}\delta_{x_2}\delta_{x_1}\delta_{x_0}$ for each input $y_3y_2\delta_{y_3}\delta_{y_2}\delta_{y_1}\delta_{y_0}$.



Taking average over $\prod_{i=0}^{3} \# v\delta_i = 4^4 = 2^8$ possibilities

Fig. 10: $\mathcal{ND}$s on $(r-1)$.75-round Gift64/128 combined with 4 vDDTs of the Gift64/128's inverse SBox



Taking average over $\prod_{i=0}^{7} \# v\delta_i = 4^8 = 2^{16}$ possibilities

Fig. 11: $\mathcal{ND}$s on $(r-1)$.5-round Gift64/128 combined with 8 vDDTs of the Gift64/128's inverse SBox

Table 20: Comparison between $\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_R}$s, $\mathcal{ND}_{\mathbf{DD}}^{\textsc{Gift}_R}$s, and $\mathcal{AD}^{\textsc{Gift}_R}$s (*i.e.*, $\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{(r-1).(1-n/16)R}}$ + n vDDTs) on Gift64/128

#R	Name	Acc.	TPR	TNR
4.5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{4.5\ R}}$	0.9694	0.9663	0.9725
4.75	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{4.75R}}$	0.9372	0.9163	0.9579
5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{4.5\ R}}$ + 8 vDDTs	0.9009	0.8615	0.9398
5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{4.75R}}$ + 4 vDDTs	0.9008	0.8620	0.9396
5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{5R}}$	0.9001	0.8623	0.9378
5	$\mathcal{ND}_{\mathbf{DD}}^{\textsc{Gift}_{5R}}$	0.8428	0.7693	0.9160
5.5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{5.5\ R}}$	0.7681	0.6639	0.8726
5.75	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{5.75R}}$	0.7203	0.5918	0.8484
6	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{5.5\ R}}$ + 8 vDDTs	0.6885	0.5692	0.8066
6	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{5.75R}}$ + 4 vDDTs	0.6826	0.5356	0.8287
6	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{6R}}$	0.6802	0.5571	0.8029
6	$\mathcal{ND}_{\mathbf{DD}}^{\textsc{Gift}_{6R}}$	0.6305	0.4988	0.7623
6.5	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{6.5\ R}}$	0.5741	0.4741	0.6743
6.75	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{6.75R}}$	0.5523	0.4317	0.6730
7	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{6.5\ R}}$ + 8 vDDTs	0.5361	0.5116	0.5633
7	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{6.75R}}$ + 4 vDDTs	0.5398	0.4195	0.6599
7	$\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{7R}}$	0.5348	0.5266	0.5431
7	$\mathcal{ND}_{\mathbf{DD}}^{\textsc{Gift}_{7R}}$	0.5019	0.4525	0.5513

For accuracy testing, 2^{19} samples were used except for $\mathcal{ND}_{\mathbf{VV}}^{\textsc{Gift}_{r.5\ R}}$ + 8 vDDTs, where only 2^{11} samples were used due to slow processing.

1. A classical (non-machine learning) distinguisher can also use this conditional differential probability to achieve superior accuracy than a pure differential-based distinguisher.
2. The observation can slightly improve the multi-bit filter used in the attacks by Dinur on Speck in [11]. Specifically, Table 8 can be combined with the 6-bit filter in [11], improving the filtering power by $2^{1.1}$, $2^{1.7}$, and $2^{2.7}$ on Speck32/*, Speck48/*, and Speck64/*, respectively (please refer to "ImprovingDDT/ImprovingClassical/DinurMultiBitFilter" in [1] for the codes and results).
3. In addition, the observations can be directly used to improve the Meet-in-the-Filter (MiF) attack on Speck [8]. Consider the last modular addition, without conditioned on the value of y, given the output difference and one input difference (γ, β), there are $2^{12.10}$ possible α in average for Speck32/64 [8, page 23]. However, since the value of y is known, according to our observation and experimental results, using the knowledge of y, the number of possible α is $2^{9.99}$. Accordingly, for those MiF attacks in [8] that use Step 2 [8, page 23], the computational complexity T_{mif} can be optimized by a factor of $2^{2.01}$ for Speck32/64. Similarly, for MiF attacks on Speck64/128, according to our observation and experimental results, the T_{mif} can be improved by a factor of $2^{4.24}$ (from $2^{25.03}$ to $2^{20.79}$) (please refer to "ImprovingDDT/ImprovingClassical/MiFnumAlpha" in [1] for the codes and results).
4. Furthermore, this kind of conditional differential probability can be used to improve the key-recovery attack on ciphers whose inputs to its non-linear operation are not directly masked by round keys. For example, in Gift, two out of the four bits input to each SBox are not XORed by key bits. In the key-recovery attack on Gift in [26, Table 11], before guessing any key bits, using the partially known value and the input differences and looking up the vDDT of the SBox, one can already filter out many candidate pairs.

E On Efficiency Measure Across Attacks

When discussing distinguishers and key recovery attacks in symmetric-key cryptanalysis, producing precise point estimates for attack efficiency can be challenging, especially when estimating the computational complexity of various attacks using the way of first timing part of the attacks in real-time, then conversing the time to the equivalent number of primitive evaluations using an assumption derived from experiments (like both the $\mathcal{ND}$-based attacks and the traditional attacks in [11,13,25] listed in Table 1):

1. **Architecture Differences** and **Algorithmic Suitability**: Algorithms optimized for CPUs might not parallelize efficiently on GPUs and vice versa. Some attacks, like traditional guess-and-determine procedures in [11], are inherently sequential. These might not see benefits from GPU parallelism. Conversely, neural network-based attacks naturally embrace GPU's parallelism.

2. **Varied Computational Intensity** and **Operation Costs Across Platforms**: The key guessing phases of $\mathcal{ND}$-based attacks, using BeyesianKeySearch on Speck32/64 deviate from the traditional counting scheme. Operations within these attacks aren't uniform in computational demand, making the conversion of the cost of various operations to the number of atomic operations produce unrealistic theoretical estimates. Besides, the cost of operations varies depending on the platform; such variances make it hard to have a standardized efficiency measure across attacks.
3. **Memory Access and Overheads** and **Various Trade-offs**: Data transfer between main memory and GPU memory can be a limiting factor. Additionally, algorithms, like the advanced key-recovery phase in [11], might allow for a direct time and memory trade-off, complicating how we measure the efficiency of an attack that prioritizes one over the other.
4. **Implementation Influence**: Efficiency can vary based on the programming language, optimization degree, and platform. An attack coded in C++ might be inherently faster than its Python counterpart due to the low-level optimizations available in C++. For example, we first implemented Alg. 2 in Python, requiring 16499 seconds to score 2^{19} pairs of Speck32/64 ciphertexts. We then implemented the same algorithm in C++ and got a significant speed up: 2^{14} seconds in Python vs. $2^{4.23}$ seconds in C++.

Given these complexities, it is a good idea to have secondary metrics for comparison. It's worth considering:

1. **Power Consumption**: While GPUs might accelerate a given attack, they might also consume more power than CPUs. For large-scale cryptographic attacks that run for extended periods, power consumption becomes a crucial metric. An attack that's slightly slower but consumes significantly less power might be considered more efficient in real-world scenarios.
2. **Cost Efficiency**: This metric evaluates the cost of the required hardware against the speedup achieved.

In conclusion, producing precise point estimates for attack efficiency in symmetric-key cryptanalysis is a complex task, affected by multiple factors ranging from algorithmic nature to the specific hardware and software implementations used. It emphasizes the need for standard metrics that can be universally applied, but developing such a metric is a challenge in itself. We will leave this for future research.

F on Related-Key Differential-Neural Distinguisher

F.1 The Training of Related-key Differential-Neural Distinguisher

- *Data generation.* Training and test sets were generated using the Linux random number generator to obtain uniformly distributed key pairs $(K_{i,0}, K_{i,1})$ with the key differential trail (list in Table 12) and plaintext pairs $(P_{i,0}, P_{i,1})$

with input difference $\Delta = $ (0000,0000) and a vector of binary-valued labels Y_i. Note that we randomly generate a large number of $(K_{i,0}, K_{i,1})$ and apply the key schedule algorithm to derive the corresponding subkeys. We then save the key pairs $(K_{i,0}, K_{i,1})$ that satisfy the differential trail of the key. During the production of training or test sets for r-round Speck32/64, the plaintext pairs were then encrypted for r rounds if $Y_i = 1$, while otherwise the second plaintext of the pairs was replaced with a freshly generated random plaintext and then encrypted for r rounds.

- *Training distinguisher using a basic training method.* We conducted training for 20 epochs in the dataset for the train set $N = 2^{24}$, and test set $M = 2^{21}$. In particular, we generate a new train set every epoch. 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 using the Adam algorithm [17]. 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 $\alpha = 2 \times 10^{-4}$, $\beta = 3.5 \times 10^{-3}$ 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 distinguisher using the freezing layer method.* We load the $r-1$-round distinguisher and then freeze the network parameters of all layers except the fully connected layer so that these parameters are not updated during training. Other settings are the same as the basic training method, such as the size and generation method of the training set and test set, batch size, optimization algorithm, learning rate, etc.

F.2 More Insight on Related-Key Differential-Neural Distinguisher

Comparing the $\mathcal{ND}$s in the single-key setting (Table 9) with those in the $\mathcal{RK}$ setting (Table 13), it can be observed that the $\mathcal{ND}$ in the $\mathcal{RK}$ setting offers more significant advantages in distinguishing the number of rounds, *i.e.*, they permit to reach more rounds.

Naturally, we wonder about the underlying reasons that contribute to the improved results in the $\mathcal{RK}$ setting. Below, we provide more explanations of the insight gained from $\mathcal{RK}$-$\mathcal{ND}$s on Speck.

Computing the pure DDT-based distinguishers in the $\mathcal{RK}$ setting. We compute the pure DDT-based distinguishers in the $\mathcal{RK}$ setting for Speck32/64, providing baselines for $\mathcal{RK}$-$\mathcal{ND}$s. To accomplish this, we make essential modifications to Gohr's Speck32/64 implementation framework of $\mathcal{DD}$s [15]. Specifically, we add an XOR operation between the r-round output difference and the related-key difference, to obtain the probability of the final output difference in the $\mathcal{RK}$ setting.

- *Under the ID_1 setting.* The input difference (0000,0000) of the plaintext pairs transitions deterministically to the low-weight difference (0040,0040) with $\mathcal{RK}$ difference $(\varDelta k^0 = \texttt{0000}, \varDelta k^1 = \texttt{0000}, \varDelta k^2 = \texttt{0040})$ after 3 rounds. Thus, starting from the input difference `(0040,0040)`, we calculate the full predicted induced output distribution of Speck32/64 for up to 6 rounds with $\mathcal{RK}$ difference $(\varDelta k^3 = \texttt{0000}, \varDelta k^4 = \texttt{0000}, \varDelta k^5 = \texttt{8000}, \varDelta k^6 = \texttt{8000}, \varDelta k^7 = \texttt{8002}, \varDelta k^8 = \texttt{8108})$ $(6 + 3 = 9$ rounds $\mathcal{RK}$-$\mathcal{DD}$s in all).

- *Under the ID$_2$/ID$_3$ setting.* The input difference `(0000,0000)` transitions deterministically to the low-weight difference `(8000,8000)` with $\mathcal{RK}$ difference $(\varDelta k^0 = \texttt{0000}, \varDelta k^1 = \texttt{0000}, \varDelta k^2 = \texttt{0000}, \varDelta k^3 = \texttt{8000})$ after 4 rounds. Thus, starting from the input difference `(8000,8000)`, we calculate the full predicted induced output distribution of Speck32/64 for up to 6 rounds with $\mathcal{RK}$ difference $(\varDelta k^4 = \texttt{8002}, \varDelta k^5 = \texttt{8008}, \varDelta k^6 = \texttt{812a}, \varDelta k^7 = \texttt{8480}, \varDelta k^8 = \texttt{9382/9082}, \varDelta k^9 = \texttt{cf8a/c28a})$ $(6 + 4 = 10$ rounds $\mathcal{RK}$-$\mathcal{DD}$s in all).

$\mathcal{RK}$-$\mathcal{ND}$s can efficiently capture additional features. The accuracy of the distinguishers $\mathcal{RK}$-$\mathcal{DD}$s obtained by using the above procedure is summarized in Table 22. It can be seen that the accuracy of $\mathcal{RK}$-$\mathcal{ND}$s is higher than that of $\mathcal{RK}$-$\mathcal{DD}$s. For example, under the ID$_1$ setting, DDT distinguishers of 8-, and 9-round have an accuracy of 0.7226 and 0.5475, respectively; in contrast, $\mathcal{ND}$s for ID$_1$ have an accuracy of 0.7584 and 0.5620, respectively. It is shown that $\mathcal{ND}$s can also efficiently capture additional features under the $\mathcal{RK}$ setting.

Improving the DDT-based distinguisher under the $\mathcal{RK}$ setting. The analysis methods and conclusions in Sect. 3 are applicable under the $\mathcal{RK}$ setting. Actually, given a pair of r-round ciphertexts $((C_L, C_R), (C'_L, C'_R))$ under the $\mathcal{RK}$ setting ($\mathcal{RK}$ difference $\varDelta k^0, \varDelta k^1, \ldots, \varDelta k^{r-1}$), not only the output difference of the final round's nonlinear layer is known, but also half of the state values are known. Specifically, one can compute the following information around the last $\boxplus$ from $((C_L, C_R), (C'_L, C'_R))$ and $\varDelta k^{r-1}$:

1. $\gamma \leftarrow C_L \oplus C'_L \oplus \varDelta k^{r-1}$,
2. $\beta \leftarrow (C_L \oplus C_R \oplus C'_L \oplus C'_R)^{\ggg 2}$,
3. $\alpha \leftarrow ((\delta_R^{r-2})^{\lll 2} \oplus \beta)^{\ggg 7}$,
4. $y \leftarrow (C_L \oplus C_R)^{\ggg 2}$.

Thus, building upon r-round distinguishers $\mathcal{RK}$-$\mathcal{DD}$s, similar to the methodology illustrated in Alg. 1 for constructing $\mathcal{YD}^{\text{Speck}_{rR}}$, we can effectively construct r-round $\mathcal{YD}^{\text{Speck}_{rR}}$ under the $\mathcal{RK}$ setting (referred to as $\mathcal{RK}$-$\mathcal{YD}^{\text{Speck}_{rR}}$). Specifically, we first compute the bias (towards 0) of each bit of $(\delta_R^{r-2})^{\lll 2}$ (Table 21 lists the values in the $\mathcal{RK}$ setting of ID$_1$, ID$_2$, and ID$_3$). Subsequently, we predict the value of each bit of $(\delta_R^{r-2})^{\lll 2}$ based on its bias (assuming it to be 0 if its bias ≥ 0 and 1 if its bias < 0). We further denote the absolute bias of the i-bit of $((\delta_R^{r-2})^{\lll 2})^{\ggg 7}$ by $\epsilon_\alpha(i)$. For each output pair $((C_L, C_R), (C'_L, C'_R))$ of r-round Speck32/64 under the $\mathcal{RK}$ setting, we utilize Alg. 1 to predict its classification, where $\gamma = C_L \oplus C'_L \oplus \varDelta k^{r-1}$ (distinct from $\gamma = C_L \oplus C'_L$ in the single-key setting).

Table 21: Bit bias towards '0' of $(\delta_R^{r-2})^{\lll 2}$ for ID_1, ID_2, and ID_3, where $8 \leq r \leq 9$. A positive (resp. negative) value indicates a bias towards '0' (resp. '1').

Diff.	Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_1	$(\delta_R^6)^{\lll 2}$	0.2344	-0.0560	-0.0002	0.0467	0.0227	0.0612	0.0000	-0.3378	0.2555	0.1538	-0.0470	0.0154	-0.1232	0.0021	0.3951	0.3313
ID_1	$(\delta_R^7)^{\lll 2}$	0.0004	-0.0074	0.0009	-0.0152	-0.0000	-0.0231	-0.0001	-0.1037	-0.0075	0.0000	-0.0068	0.0001	0.0562	0.0000	0.1750	-0.0170
$\mathrm{ID}_2/\mathrm{ID}_3$	$(\delta_R^6)^{\lll 2}$	0.3048	0.1721	-0.0003	0.1249	0.0011	-0.4688	0.4454	0.4062	0.3442	0.2502	-0.1250	0.0007	-0.4923	0.4844	-0.4434	0.3946
$\mathrm{ID}_2/\mathrm{ID}_3$	$(\delta_R^7)^{\lll 2}$	-0.0006	-0.0180	0.0003	0.0572	0.0020	-0.3363	0.2532	0.1466	0.0241	-0.0005	0.0327	0.0007	0.4287	0.3734	0.2296	0.1131

Moreover, based on $(r-1)$-round distinguishers $\mathcal{RK}$-$\mathcal{DD}$s, similar to Alg. 2 used to construct $\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{rR}}$, we can construct r-round $\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{rR}}$ under the $\mathcal{RK}$ setting (referred to as $\mathcal{RK}$-$\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{rR}}$).

Results of improving the DDT-based distinguisher under the $\mathcal{RK}$ setting. The accuracy of the distinguishers $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{rR}}$ and $\mathcal{RK}$-$\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{rR}}$ obtained by using the previous procedure is summarized in Table 22. For ID_1, the performance of $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{8R}}$ demonstrates an improvement. But the limited progress in enhancing $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{9R}}$ is due to the fact that $(\delta_R^7)^{\lll 2}$ no longer exhibits obvious bias with multiple bits. For ID_2 and ID_3, the performance of $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{8R}}$ and $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{9R}}$ is generally improved. Similarly, the constrained progress in enhancing $\mathcal{RK}$-$\mathcal{YD}^{\textsc{Speck}_{10R}}$ can be attributed to the absence of notable multi-digit bias in $(\delta_R^8)^{\lll 2}$. Furthermore, the obtained r-round distinguishers $\mathcal{RK}$-$\mathcal{AD}_{\mathbf{YD}}^{\textsc{Speck}_{rR}}$ exhibit comparable, or even superior, performance compared to pure neural distinguishers that rely solely on ciphertext values as inputs. This indicates that neural distinguishers under the $\mathcal{RK}$ setting with ciphertext values as inputs not only utilize differential distributions but also exploit value-based conditional differential distributions.

Remark 5. It can be seen that the number of rounds of $\mathcal{RK}$-$\mathcal{DD}$ and $\mathcal{RK}$-$\mathcal{ND}$ is longer than that of the current best $\mathcal{DD}$ and $\mathcal{ND}$, respectively. We believe that the main reason is that the utilization of the $\mathcal{RK}$ difference can control the number of active bits and make them be injected into the nonlinear operation as slowly as possible, thus extending the number of rounds of the differential. For example, under the $\mathrm{ID}_2/\mathrm{ID}_3$ setting, when the input difference of the plaintext pairs is chosen as `(0000,0000)`, the introduction of active bits does not begin until the end of round 4.

Table 22: Accuracy of the $\mathcal{RK}\text{-}\mathcal{DD}$s on $\textsc{Speck}32/64$ and comparisons with $\mathcal{RK}\text{-}\mathcal{ND}$s, $\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{rR}}$, and $\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{rR}}_{\mathbf{YD}}$

Diff.	#R	Name	Accuracy	True Positive Rate	True Negative Rate
ID_1	8	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{8R}}$	0.7226	0.6447	0.8005
ID_1	8	$\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{8R}}$	0.7333	0.6838	0.7828
ID_1	8	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{8R}}$	0.7584	0.6836	0.8332
ID_1	8	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{8R}}_{\mathbf{YD}}$	0.7642	0.6946	0.8340
ID_1	9	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{9R}}$	0.5475	0.5221	0.5728
ID_1	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{9R}}$	0.5620	0.5212	0.6028
ID_1	9	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{9R}}_{\mathbf{YD}}$	0.5745	0.5306	0.6187
ID_2/ID_3	8	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{8R}}$	0.8989	0.8714	0.9264
ID_2/ID_3	8	$\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{8R}}$	0.9118	0.8886	0.9350
ID_2/ID_3	8	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{8R}}$	0.9259	0.9063	0.9455
ID_2/ID_3	8	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{8R}}_{\mathbf{YD}}$	0.9315	0.9159	0.9470
ID_2	9	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{9R}}$	0.7128	0.6644	0.7612
ID_2	9	$\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{9R}}$	0.7190	0.6845	0.7534
ID_2	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{9R}}$	0.7535	0.7035	0.8036
ID_2	9	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{9R}}_{\mathbf{YD}}$	0.7574	0.7114	0.8035
ID_2	10	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{10R}}$	0.5484	0.5387	0.5581
ID_2	10	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{10R}}$	0.5643	0.5382	0.5893
ID_2	10	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{10R}}_{\mathbf{YD}}$	0.5706	0.5359	0.6053
ID_3	9	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{9R}}$	0.7128	0.6644	0.7612
ID_3	9	$\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{9R}}$	0.7185	0.6843	0.7527
ID_3	9	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{9R}}$	0.7726	0.7247	0.8206
ID_3	9	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{9R}}_{\mathbf{YD}}$	0.7574	0.7113	0.8035
ID_3	10	$\mathcal{RK}\text{-}\mathcal{DD}^{\textsc{Speck}_{10R}}$	0.5484	0.5343	0.5624
ID_3	10	$\mathcal{RK}\text{-}\mathcal{ND}^{\textsc{Speck}_{10R}}$	0.5562	0.5361	0.5765
ID_3	10	$\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{10R}}_{\mathbf{YD}}$	0.5713	0.5357	0.6069

The thresholds τ for $\sigma_\alpha(i)$ in constructing $\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{8R}}$ for ID_1, ID_2, and ID3 are 0.1, 0.3, and 0.3, respectively. Similarly, the threshold τ for $\sigma\alpha(i)$ in constructing $\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{9R}}$ for either ID_2 or ID_3 is set to 0.1.

The number of samples for the accuracy testing of $\mathcal{RK}\text{-}\mathcal{YD}^{\textsc{Speck}_{rR}}$ and $\mathcal{RK}\text{-}\mathcal{AD}^{\textsc{Speck}_{rR}}_{\mathbf{YD}}$ are 2^{19}.