Improved Differential Cryptanalysis on SPECK Using Plaintext Structures

Zhuohui Feng[1⋆], Ye Luo[1⋆], Chao Wang[1], Qianqian Yang[2,3], Zhiquan Liu[1], and Ling Song[1,4(✉)]

[1] College of Cyber Security, Jinan University, Guangzhou, China
hhfzhfzh@163.com, roylaw456@gmail.com, wangchao0edu@gmail.com, zqliu@vip.qq.com, songling.qs@gmail.com
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
yangqianqian@iie.ac.cn
[4] National Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University, Guangzhou, China

Abstract. Plaintext structures are a commonly-used technique for improving differential cryptanalysis. Generally, there are two types of plaintext structures: multiple-differential structures and truncated-differential structures. Both types have been widely used in cryptanalysis of S-box-based ciphers while for SPECK, an Addition-Rotation-XOR (ARX) cipher, the truncated-differential structure has not been used so far. In this paper, we investigate the properties of modular addition and propose a method to construct truncated-differential structures for SPECK. Moreover, we show that a combination of both types of structures is also possible for SPECK. For recovering the key of SPECK, we propose dedicated algorithms and apply them to various differential distinguishers, which helps to obtain a series of improved attacks on all variants of SPECK. Notably, on SPECK128, the time complexity of the attack can be reduced by a factor up to 2^{15}. The results show that the combination of both structures helps to improve the data and time complexity at the same time, as in the cryptanalysis of S-box-based ciphers.

Keywords: ARX ciphers · structures · differential cryptanalysis · SPECK

1 Introduction

Symmetric ciphers play a major role in providing confidentiality. According to the nonlinear operation used in the cipher, one popular category of symmetric ciphers is S-box-based ciphers and another is ARX ciphers that are built using only modular additions, bit rotations, and bitwise XORs. When a symmetric

⋆ These authors contributed equally to this work.

cipher is designed, the only way to build confidence in it is through a continuous effort to evaluate its security.

There are several families of attack against symmetric ciphers. Among them, differential cryptanalysis [BS90] and linear cryptanalysis [MY92] are the two major ones. Many attacks can usually be divided into two parts: a distinguisher and a key-recovery part. Specifically, a differential distinguisher constitutes a high-probability differential in a part of a cipher, say a differential $\rho \to \delta$ over E_d as shown in Fig. 1. The key-recovery part usually involves the rounds before and after the distinguisher, *i.e.*, E_b and E_f in Fig. 1. The idea is to guess the subkeys of E_b and E_f, and check if the differential $\rho \to \delta$ occurs with a high probability for E_d. If so, the key guess is likely correct. This paper focuses on the key-recovery part.

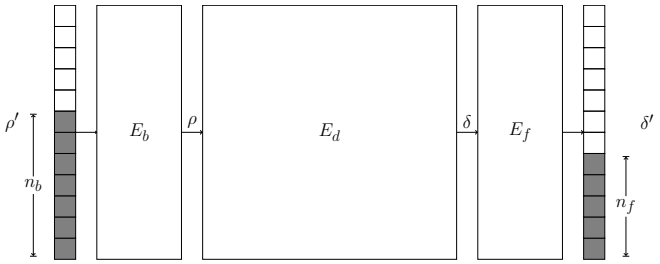

Fig. 1: Overview of differential attacks

Plaintext structures. Along with the invention of differential cryptanalysis, structures of plaintexts are used to improve the attack. Roughly, two types of structures were proposed and widely used in the literature. Before we recall the two types, we define a ratio $\mathcal{R}_{m/p}$ between the number of messages and the number of message pairs that satisfy the input difference of the distinguisher. Then the data complexity is the product of $\mathcal{R}_{m/p}$ and the number of required pairs satisfying the input difference.

Multiple-differential structures (Type-I) This type of plaintext structures is applied to the case where the target cipher $E = E_f \circ E_d$. Without using this type of structures, a pair costs two messages, *i.e.*, $\mathcal{R}_{m/p} = 2$. When this type of structures is used, the ratio $\mathcal{R}_{m/p}$ is expected to be lower than 2. A typical situation is that several differential trails over E_d are used simultaneously [BS91]. Suppose we use two trails with input differences Δ_1 and Δ_2. If we prepare N pairs of plaintexts for each input difference separately, it takes $4N$ plaintexts in total. However, when we pack them into structures, as shown in Fig. 2, we can get 2 pairs for each input difference from a structure of 4 plaintexts, resulting in $\mathcal{R}_{m/p} = 1$. That is to say, the use of Type-I structure helps to reduce the data complexity and potentially the

time complexity. Note, this type of structures was first used in the differential attack on 15-round DES [BS91].

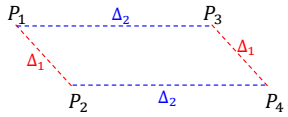

Fig. 2: Example of Type-I structures

Truncated-differential structures (Type-II) The second type of plaintext structures is applied to the case where the target cipher $E = E_f \circ E_d \circ E_b$ or $E = E_d \circ E_b$, *i.e.*, there are some rounds before the distinguisher. As illustrated in Fig. 1, the input difference ρ of the distinguisher propagates backward to ρ' through E_b, where n_b bits of ρ' are active. A structure of this type consists of 2^{n_b} plaintexts where inactive bits are constant while active bits are traversed. Among each structure of 2^{n_b} plaintexts, about 2^{n_b-1} out of 2^{2n_b-1} pairs satisfy the input difference ρ of the distinguisher. In order to have N pairs of plaintexts leading to ρ, still $2N$ plaintexts are needed. That is to say, both $\mathcal{R}_{m/p}$ and the data complexity remain the same. The possible gain is to attack more rounds or reduce the time complexity. Note, this type of structures was used to reach a full-round differential attack on DES [BS92].

As a common technique, both types of structures have been widely used in the cryptanalysis of S-box-based ciphers, *e.g.*, [BS91, BK09]. For ARX ciphers, Type-I structures have been used to reduce the data complexity of differential-like attacks on a series of ARX ciphers, like Chaskey [Leu16] and `SPECK` [Goh19], and Type-II structures have been used to mount impossible differential attacks and truncated differential attacks on ARX ciphers, such as XTEA, TEA and HIGHT [MHL+02, HHK+04, CWP12]. However, to the best of our knowledge, Type-II structures have not been applied to `SPECK`.

In differential cryptanalysis of ARX ciphers, extending some rounds before the distinguisher is possible, but it may bring no benefit in the general case. The problem is that these ciphers typically have 32-bit or 64-bit words. Even though the input difference ρ of the distinguisher may have a few active bits, full-word subkeys of E_b have to be guessed to ensure the ρ difference. In other words, adding rounds before the distinguisher brings no benefit (compared to adding rounds after the distinguisher). Therefore, this usually does not give efficient attacks. In this paper, we study in which case the Type-II structures can be applied to `SPECK` and what benefits they can bring.

`SPECK`. `SPECK` is a family of ARX ciphers which were designed in 2013 by researchers from the U.S. National Security Agency (NSA) [BSS+13]. Its structure is a generalized Feistel structure and it provides excellent performance in

both hardware and software. The SPECK family has 10 variants, denoted by SPECK$2n/mn$, where $2n$ is the block size and mn is the key size. Since the design of SPECK was published, it has attracted intensive cryptanalysis from the community [BRV14, ALLW14, Din14, FWG+16, SHY16, LLJW20, BdST+22, WW22]. Besides these classical cryptanalysis, there also exist some cryptanalysis using deep learning [Goh19, BGL+23] whose basis is differential characteristics. So far, differential-like cryptanalysis is the most powerful attack against SPECK. Note that none of the previous differential attacks employ the Type-II structures in the key-recovery attacks on SPECK.

Our contributions. We start with studying the differential properties of modular addition. Due to the fact of modular addition that the lower bits of the output are affected only by the lower bits of the inputs, the differential propagation can be confined to the higher bits when the inputs have zero difference in the lower bits. We then introduce two parameters n_{BIL} and n_{FIL} to denote the numbers of inactive lower bits for the output of modular subtraction and addition, respectively. Based on these properties of modular addition, we formalize the construction of Type-II structures for SPECK. Moreover, we show that Type-II structures and Type-I structures can be combined for ARX ciphers and applied to SPECK.

Further, we propose three algorithms for key recovery attacks on SPECK, *i.e.*, Algorithm A, B and C, which target the cases using Type-I structures, Type-II structures, and a combination of both, respectively. Algorithm A mainly achieves the goal of reducing the data complexity when compared with previous attacks. The aim of Algorithm B is to utilize Type-II structures so as to reduce the time complexity. Algorithm C combines the ideas of Algorithm A and B and helps to improve the data and time complexity at the same time. Note the improvement in the time complexity is proportional to n_{BIL} and n_{FIL}.

In order to prepare suitable distinguishers of SPECK for these three algorithms, we particularly search for differential trails with n_{BIL} and n_{FIL} as large as possible. We then mount attacks by applying the three algorithms to newly obtained differential distinguishers. The resulting attacks on SPECK and the previous works are presented in Table 1. More detailed results on SPECK in this work are displayed in Table 11. The results show that the use of Type-II structures helps to reduce the time complexity by a factor up to 2^{15} and in many cases, both the time and data complexity are reduced to a certain extent.

Our work shows that Type-II structures are possible for SPECK and help to improve the time complexity in certain cases. Their application should not be limited to standard differential cryptanalysis of SPECK. Other attacks to which Type-II structures can be potentially applied include boomerang attacks, differential-linear attacks, impossible differential attacks, etc.

Table 1: Comparison of our attack on SPECK with the previous works

Variants	Rounds	Probability	Data	Time	Memory	Reference
32/64	14/22	2^{-30}	2^{31}	2^{63}	2^{22}	[Din14]

Continued on next page

Table 1 – Continued from previous page

Variants	Rounds	Probability	Data	Time	Memory	Reference
	14/22	$2^{-29.47}$	$2^{30.47}$	$2^{62.47}$	2^{22}	[SHY16]
	14/22	$2^{-29.37}$	$2^{31.75}$	$2^{60.99}$	$2^{41.91}$	[BdST+22]
	14/22	$2^{-27.68}$	$\mathbf{2^{30.26}}$	$\mathbf{2^{60.58}}$	2^{36}	**This**
	15/22	$2^{-30.39}$	$2^{31.39}$	$2^{63.39}$	2^{22}	[LKK+18]
	15/22	$2^{-30.39}$	$2^{31.39}$	$2^{62.25}$	-	[BdST+22]
48/72	15/22	2^{-45}	2^{46}	2^{70}	2^{22}	[FWG+16]
	15/22	$2^{-44.31}$	$2^{45.31}$	$2^{69.31}$	2^{22}	[SHY16]
	15/22	$2^{-43.42}$	$\mathbf{2^{44.42}}$	2^{70}	2^{22}	**This**
	16/22	$2^{-46.80}$	$2^{47.80}$	$2^{71.80}$	2^{22}	[LKK+18]
	16/22	$2^{-45.78}$	$\mathbf{2^{46.78}}$	$\mathbf{2^{71.78}}$	2^{22}	**This**
48/96	16/23	2^{-45}	2^{46}	2^{94}	2^{22}	[FWG+16]
	16/23	$2^{-44.31}$	$2^{45.31}$	$2^{93.31}$	2^{22}	[SHY16]
	16/23	$2^{-43.42}$	$\mathbf{2^{44.42}}$	2^{94}	2^{22}	**This**
	17/23	$2^{-46.80}$	$2^{47.80}$	$2^{95.80}$	2^{22}	[LKK+18]
	17/22	$2^{-45.78}$	$\mathbf{2^{46.78}}$	$\mathbf{2^{95.78}}$	2^{22}	**This**
64/96	19/26	2^{-62}	2^{63}	2^{95}	2^{22}	[FWG+16]
	19/26	$2^{-59.30}$	$\mathbf{2^{61.88}}$	$\mathbf{2^{93.34}}$	2^{68}	**This**
	19/26	$2^{-60.56}$	$2^{61.56}$	$2^{93.56}$	2^{22}	[SHY16]
	19/26	$2^{-58.24}$	$\mathbf{2^{60.82}}$	$\mathbf{2^{92.28}}$	2^{68}	**This**
64/128	20/27	2^{-62}	2^{63}	2^{127}	2^{22}	[FWG+16]
	20/27	$2^{-59.30}$	$\mathbf{2^{61.88}}$	$\mathbf{2^{125.34}}$	2^{68}	**This**
	20/27	$2^{-60.56}$	$2^{61.56}$	$2^{125.56}$	2^{22}	[SHY16]
	20/27	$2^{-60.73}$	$2^{63.96}$	$2^{122.69}$	$2^{77.19}$	[BdST+22]
	20/27	$2^{-58.24}$	$\mathbf{2^{60.82}}$	$2^{124.28}$	2^{68}	**This**
96/96	19/28	2^{-87}	2^{88}	2^{88}	2^{22}	[FWG+16]
	19/28	2^{-86}	$\mathbf{2^{87}}$	2^{88}	2^{22}	**This**
	20/28	$2^{-94.94}$	$2^{95.94}$	$2^{95.94}$	2^{22}	[SHY16]
	20/28	$2^{-92.17}$	$\mathbf{2^{93.17}}$	$\mathbf{2^{95.75}}$	2^{22}	**This**
96/144	20/29	2^{-87}	2^{88}	2^{136}	2^{22}	[FWG+16]
	20/29	2^{-86}	$\mathbf{2^{87}}$	2^{136}	2^{22}	**This**
	21/29	$2^{-94.94}$	$2^{95.94}$	$2^{143.94}$	2^{22}	[SHY16]
	21/29	$2^{-92.17}$	$\mathbf{2^{93.17}}$	$\mathbf{2^{143.75}}$	2^{22}	**This**
	21/29	$2^{-91.03}$	$\mathbf{2^{93.61}}$	$\mathbf{2^{143.13}}$	2^{99}	**This**
128/128	22/32	2^{-119}	2^{120}	2^{120}	2^{22}	[FWG+16]
	22/32	2^{-117}	$\mathbf{2^{118}}$	$2^{120.81}$	2^{22}	**This**
	23/32	$2^{-124.35}$	$2^{125.35}$	$2^{125.35}$	2^{22}	[SHY16]
	23/32	$2^{-121.37}$	$\mathbf{2^{122.37}}$	$\mathbf{2^{124.95}}$	2^{22}	**This**
128/192	23/33	2^{-119}	2^{120}	2^{184}	2^{22}	[FWG+16]
	23/33	$2^{-117.19}$	$\mathbf{2^{119.77}}$	$\mathbf{2^{168.35}}$	2^{131}	**This**
	24/33	$2^{-124.35}$	$2^{125.35}$	$2^{189.35}$	2^{22}	[SHY16]
	24/33	$2^{-121.37}$	$\mathbf{2^{123.95}}$	$\mathbf{2^{174.53}}$	2^{129}	**This**
128/256	24/34	2^{-119}	2^{120}	2^{248}	2^{22}	[FWG+16]
	24/34	$2^{-117.19}$	$\mathbf{2^{119.77}}$	$\mathbf{2^{232.35}}$	2^{131}	**This**
	25/34	$2^{-124.35}$	$2^{125.35}$	$2^{253.35}$	2^{22}	[SHY16]
	25/34	$2^{-121.37}$	$\mathbf{2^{123.95}}$	$\mathbf{2^{238.53}}$	2^{129}	**This**

Organization. The paper is organized as follows. In Section 2, we introduce some preliminaries, including notations, a description of SPECK, and a previous key recovery attack on it. In Section 3, we study the properties of modular addition and propose some propositions, based on which Type-II structures and a combination of both types of structures can be constructed for ARX ciphers. In Section 4, we introduce three key recovery algorithms for SPECK using different types of structures. We apply them to all variants of SPECK and obtain a series of improved attacks in Section 5. Finally, we conclude this work in Section 6.

2 Preliminaries

2.1 Notations

Given an n-bit word x, we denote its i^{th} bit for $i \in \{0, 1, ..., n-1\}$ by $x[i]$ and its i^{th} bit to j^{th} bit by $x[j:i]$, $i \leq j$. Given two n-bit words x and y, we denote by $x \boxplus y$ their addition modulo 2^n, by $x \boxminus y$ their modular subtraction, by $x \oplus y$ the bitwise XOR operation, by $x \vee y$ the bitwise OR operation and by $x \wedge y$ the bitwise AND operation between them. Given an n-bit word x and a positive integer i, we denote by $x \ggg i$ the n-bit word obtained by rotating x by i bits to the right, and by $x \lll i$ the word obtained by rotating x to the left.

In this paper, we denote the i^{th} round of SPECK by Round i where $i \in \{1, 2, \ldots, T\}$. The output of Round i is denoted by x_i and y_i and k_{i-1} is used to denote the round keys of Round i.

2.2 Specification of SPECK

SPECK is a family of lightweight block ciphers designed by researchers from the U.S. National Security Agency (NSA) [BSS+13]. There are 10 variants, each of which is characterized by its block size $2n$ and key size mn. Some parameters for all variants of SPECK are specified in Table 2.

Table 2: The parameters of SPECK

block size $2n$	key size mn	word size n	key word size m	rot a	rot b	Rnds T
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

The round function of SPECK is shown in Fig. 3, in which there are two n-bit words x_i and y_i as inputs and two n-bit words x_{i+1} and y_{i+1} are outputs after some operations of rotation, modular addition and XOR with the key word. The round function is defined as:

$$(x_{i+1}, y_{i+1}) = \mathcal{R}_{k_i}(x_i, y_i) = (((x_i \ggg a) \boxplus y_i) \oplus k_i, (y_i \lll b) \oplus ((x_i \ggg a) \boxplus y_i) \oplus k_i),$$

where k_i is the round key for $0 \leq i < T$.

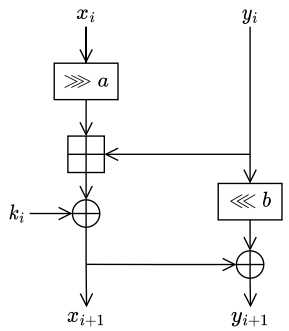

Fig. 3: The round function of SPECK

The SPECK key schedule takes an initial m-word master key $(l_{m-2}, \ldots, l_0, k_0)$ and from it generates a sequence of T round key words $k_0, \ldots, k_{T-1}$ as follows.

$$\begin{aligned} l_{i+m-1} &= (k_i \boxplus (l_i \ggg a)) \oplus i, \\ k_{i+1} &= (k_i \lll b) \oplus l_{i+m-1}, 0 \leq i < T. \end{aligned}$$

From the specification of SPECK, two simple properties can be obtained directly.

Property 1. If the output of Round $i+1, i \geq 0$ is known, *i.e.*, (x_{i+1}, y_{i+1}) is known, one of the input word y_i can be determined by $y_i = (x_{i+1} \oplus y_{i+1}) \ggg b$.

Property 2. If m consecutive round keys of SPECK are known, *i.e.*, $k_{i-m}, \cdots, k_{i-1}$ for $i > m$, we can efficiently invert the key schedule as follows to determine $k_{i-m-1}, \cdots$, and so on until we have the original m master key words.

$$\begin{aligned} l_{j+m-3} &= k_{j-1} \oplus (k_{j-2} \lll b), \\ l_{j-2} &= ((l_{j+m-3} \oplus (j-2)) \boxminus k_{j-2}) \lll a, \\ k_{j-m-1} &= k_{j-m} \oplus l_{j-2}) \ggg b. \end{aligned}$$

Consequently, the key recovery attacks of SPECK are equivalent to recovering m consecutive round keys.

2.3 Dinur's Key Recovery Attack

In [Din14], Dinur paid attention to the key-recovery part of differential attacks on SPECK and proposed an algorithm to recover the key of $1+r+m$ rounds using an r-round differential characteristic. This algorithm is very efficient and was adopted in later differential cryptanalysis of SPECK [FWG+16, SHY16].

In this algorithm, the core part was a sub-procedure called the 2-round attack. Due to Property 1 of SPECK, if we know the input difference $(\Delta x_i, \Delta y_i)$ and the output difference $(\Delta x_{i+2}, \Delta y_{i+2})$ of the 2-round transformation, we can derive the differences after Round $i+1$, *i.e.*, $\Delta y_{i+1} = (\Delta x_{i+2} \oplus \Delta y_{i+2}) \ggg b$ and $\Delta x_{i+1} = (\Delta y_i \lll b) \oplus \Delta y_{i+1}$ as shown in Fig. 4. In addition, we can derive the value of y_{i+1} from the known x_{i+2} and y_{i+2}.

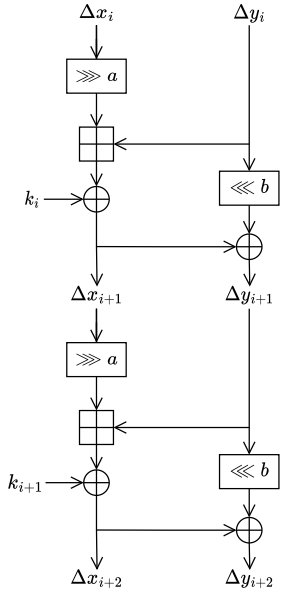

Fig. 4: The differences of the two rounds in Dinur's attack

In the whole key-recovery attack, these two rounds are located exactly after the differential distinguisher. Therefore, $(\Delta x_i, \Delta y_i)$ should match the output difference of the differential distinguisher. After these two rounds, there are additional $m-2$ rounds. Guessing the last $m-2$ round keys, one can compute the exact values of (x_{i+2}, y_{i+2}) and $(\Delta x_{i+2}, \Delta y_{i+2})$ as well.

In other words, all the input and output differences of the modular addition of Round $i+1$ and Round $i+2$ are known whereas x_i, y_i and x_{i+1} are unknown. The objective of the 2-round attack is to find all possible independent round keys k_i and k_{i+1} from x_i, y_i and x_{i+1} by solving differential equations of addition. On average, there are no more than two solutions for k_{r+1} and k_{r+2} [Din14]. The

time complexity of the 2-round attack is less than 2 encryptions of SPECK and the memory complexity is about 2^{22} bytes which is small. For more details of the 2-round attack, please refer to Appendix A.1.

Note that the first round of SPECK can be seen as a whitening layer and thus covered for free. Given an r-round differential $\rho \to \delta$ of probability 2^{-w}, the main steps of Dinur's algorithm to attack $1 + r + m$ rounds are listed as follows.

1. For $i = 1, \cdots, 2^w$, choose a random P and get P' such that $\bar{\mathcal{R}}(P') = \bar{\mathcal{R}}(P) \oplus \rho$, where $\bar{\mathcal{R}}$ is the round function without round key addition. Request the encryption of (P, P').
   (a) Guess the last $m - 2$ round keys, *i.e.*, none when $m = 2$, k_{r+3} when $m = 3$, and k_{r+3}, k_{r+4} when $m = 4$. Then do partial decryption.
      i. Run the 2-round attack using $(\Delta x_{r+1}, \Delta y_{r+1}) = \delta$, $(\Delta x_{r+3}, \Delta y_{r+3})$ and (x_{r+3}, y_{r+3}) and return solutions of k_{r+1} and k_{r+2}.
      ii. For each returned value of k_{r+1} and k_{r+2} together with the guessed $m - 2$ subkeys, recover the master key. Test the master key using trial encryptions, and return it if the trial encryptions succeed.

The above algorithm recovers the last m consecutive round keys, from which the master key can be derived according to Property 2. The complexities of the algorithm are summarized as follows.

- $D_{\text{Dinur}} = 2^{w+1}$ plaintexts;
- $T_{\text{Dinur}} = 2^{(m-2)n} \times 2^w \times 2 = 2 \times 2^{(m-2)n+w}$ encryptions, where $m = 2, 3, 4$;
- $M_{\text{Dinur}} = 2^{22}$ bytes.

Remark 1. Since this algorithm enumerates and tests all possible candidates of the last m round keys, the attack will succeed as long as the distinguisher succeeds. Consequently, it has a high success probability.

3 Properties of Modular Addition and Structures

In this section, we give some properties of modular addition, based on which one could construct plaintext structures for SPECK as in cryptanalysis of S-box-based ciphers.

3.1 Properties of Modular Addition

The addition modular 2^n in cryptographic primitives is a nonlinear transformation that is extremely fast in the calculation. Given two n-bit words x and y, we let $z = x \boxplus y$ where the carry word is denoted by c. Some operation rules about the carry word c are listed as follows.

$$\begin{aligned} c[0] &= 0, \\ c[i+1] &= x[i] \wedge y[i] \oplus x[i] \wedge c[i] \oplus y[i] \wedge c[i], i \in \{0, 1, \ldots, n-2\}, \\ z[i] &= x[i] \oplus y[i] \oplus c[i], i \in \{0, 1, \ldots, n-1\}. \end{aligned}$$

With these in mind, we look into the XOR differences of modular addition. Let $\tilde{x} = x \oplus \alpha$, $\tilde{y} = y \oplus \beta$, $\tilde{z} = \tilde{x} \boxplus \tilde{y}$, and $\tilde{z} \oplus z = \gamma$, where α, β and γ are differences, as shown in Fig. 5. Let c, $\tilde{c}$ respectively denote the carry words of $x \boxplus y$, $\tilde{x} \boxplus \tilde{y}$ and let $\Delta c = c \oplus \tilde{c}$. Then we have

$$\gamma[i] = z[i] \oplus \tilde{z}[i] = x[i] \oplus y[i] \oplus c[i] \oplus \tilde{x}[i] \oplus \tilde{y}[i] \oplus \tilde{c}[i] = \alpha[i] \oplus \beta[i] \oplus \Delta c[i].$$

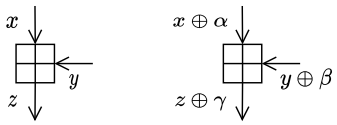

Fig. 5: The differences of the modular addition

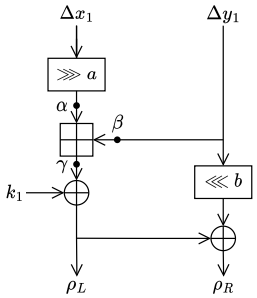

Fig. 6: The differences of Round 2

In the following, we give three propositions of the differences of modular addition.

Proposition 1. *Let $z = x \boxplus y$ and $\tilde{z} = \tilde{x} \boxplus \tilde{y}$, where $\alpha = x \oplus \tilde{x}, \beta = y \oplus \tilde{y}$ and $\gamma = z \oplus \tilde{z}$. Given β, γ where $\beta[j:0] = \gamma[j:0] = 0$, $0 \le j < n$, then $\alpha[j:0] = 0$.*

Proof. When $\beta[0] = 0$ and $\gamma[0] = 0$, we have $\alpha[0] = \gamma[0] \oplus \beta[0] \oplus \Delta c[0] = 0$. Therefore, $\Delta c[1] = 0$. Then when $\beta[1] = 0, \gamma[1] = 0$, as $\alpha[1] = \gamma[1] \oplus \beta[1] \oplus \Delta c[1] = 0$, we have $\Delta c[2] = 0$, and so on. □

Definition 1 (Backward inactive lower bits (BIL)). *Under the setting in Proposition 1, $\alpha[j:0] = 0$, $0 \le j < n$. We call $\alpha[j:0]$ the backward inactive lower bits for the addition and BIL for short. Denote the length of $\alpha[j:0]$ in bits by n_{BIL}.*

Proposition 2. *Let $z = x \boxplus y$ and $\tilde{z} = \tilde{x} \boxplus \tilde{y}$, where $\alpha = x \oplus \tilde{x}, \beta = y \oplus \tilde{y}$ and $\gamma = z \oplus \tilde{z}$. Given β, γ where $\beta[j:0] = \gamma[j:0] = 0$ and $\beta[j+1] \vee \gamma[j+1] = 1$, $0 \le j < n-1$, then $\alpha[j+1] = \beta[j+1] \oplus \gamma[j+1]$ and $\Pr[\alpha[u] = 1]$ for any $u \in (j+1, n)$ is non-zero if z and y are taken randomly (or equivalently x, y are taken randomly).*

Proof. According to Proposition 1, we have $\alpha[j:0] = 0$ and $\Delta c[j+1] = 0$. When $\beta[j+1] \vee \gamma[j+1] = 1$, we have $\alpha[j+1] = \beta[j+1] \oplus \gamma[j+1] \oplus \Delta c[j+1] = \beta[j+1] \oplus \gamma[j+1]$. We prove the subsequent cases by exhaustive calculation which is shown in Table 3(a) and 3(b).

i) For $u = j + 2$, as $\beta[u-1] \vee \gamma[u-1] = 1$ and $\Delta c[u-1] = 0$, we have $\Pr[\Delta c[u] = 0] = \Pr[\Delta c[u] = 1] = \frac{1}{2}$, according to Table 3(b);
ii) For $u \geq j + 3$, let $\Pr[\Delta c[u-1] = 1] = p$. According to Table 3(b),
  a) when $(\beta[u-1], \gamma[u-1])$=(0,0), $\Pr[\Delta c[u] = 0] = 1 - p + \frac{1}{2}p = 1 - \frac{1}{2}p$, $\Pr[\Delta c[u] = 1] = \frac{1}{2}p > 0$;
  b) when $(\beta[u-1], \gamma[u-1])$=(0,1), $\Pr[\Delta c[u] = 0] = \frac{1}{2}(1-p) + \frac{1}{2}p = \frac{1}{2}$, $\Pr[\Delta c[u] = 1] = \frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}$;
  c) when $(\beta[u-1], \gamma[u-1])$=(1,0), $\Pr[\Delta c[u] = 0] = \frac{1}{2}$, $\Pr[\Delta c[u] = 1] = \frac{1}{2}$;
  d) when $(\beta[u-1], \gamma[u-1])$=(1,1), $\Pr[\Delta c[u] = 0] = \frac{1}{2}(1-p)$, $\Pr[\Delta c[u] = 1] = \frac{1}{2}(1-p) + p = \frac{1}{2} + \frac{1}{2}p$.

In summary, for any $u \in (j+1, n)$, $0 < \Pr[\Delta c[u] = 1] < 1$. Therefore, due to $\alpha[u] = \beta[u] \oplus \gamma[u] \oplus \Delta c[u]$, we will get $\alpha[u] = 0$ or $\alpha[u] = 1$ with certain non-zero probability. □

Similarly, on the other side we have the following proposition.

Proposition 3. *Let $z = x \boxplus y$ and $\tilde{z} = \tilde{x} \boxplus \tilde{y}$, where $\alpha = x \oplus \tilde{x}, \beta = y \oplus \tilde{y}$ and $\gamma = z \oplus \tilde{z}$. Given α, β where $\alpha[j:0] = \beta[j:0] = 0$, $0 \leq j < n$, then $\gamma[j:0] = 0$.*

Definition 2 (Forward inactive lower bits (FIL)). *Under the setting of Proposition 3, $\gamma[j:0] = 0$, $0 \leq j < n$. We call $\gamma[j:0]$ the forward inactive lower bits of the addition and FIL for short. Denote the length of $\gamma[j:0]$ in bits by n_{FIL}.*

3.2 Type-II Structures for SPECK

Let us come to the key recovery part of differential attacks. Given a differential $\rho \to \delta$ of an ARX cipher, suppose we prepend one round to it. For the modular addition of this extra round, we still denote its input differences and the output difference by $\alpha, \beta \to \gamma$, where the differences β and γ should match the input difference of the differential.

If β and γ are some random differences, α will be regarded as random as well. In terms of notations in Fig. 1, it leads to a large n_b, *i.e.*, a large number of plaintext bits will be active. In contrast, if the lower bits or the least significant bit of β and γ are 0, we can know for sure the same number of lower bits of α are zero as well, while its higher bits or the most significant bit can be either 1 or 0. That is to say, for certain proper differentials, the number of active bits in the plaintext can be small.

A common experience in differential cryptanalysis of S-box-based ciphers is that a small n_b is desirable, so it would be interesting to see the effect of building Type-II structures on the active higher bits of some input words of SPECK.

In the following, we will show how Type-II structures of SPECK can be built by exploiting Proposition 1 and 2 and discuss the possibility of applying both types of structures to ARX ciphers simultaneously.

Type-II structures for SPECK. Note that the first round of SPECK acts as a whitening layer, so it can be covered for free. Now we consider adding two rounds

Table 3: All cases of the differences of the carry bits $\Delta c[u]$ in the modular addition. (a)All cases of $\beta[u-1]$ and $\gamma[u-1]$. (b)The exhaustive calculation of $\Delta c[u]$ for all possible $y[u-1]$, $\tilde{y}[u-1]$, $z[u-1]$, $\tilde{z}[u-1]$, $c[u-1]$, $\tilde{c}[u-1]$ where $j+1<u$.

(a)

ID	$\beta[u-1]$	$\gamma[u-1]$	$y[u-1]$,$\tilde{y}[u-1]$ $z[u-1]$,$\tilde{z}[u-1]$	$\alpha[u-1]$	
				$\Delta c[u-1]=0$	$\Delta c[u-1]=1$
1	0	0	0, 0, 0, 0	0	1
2			1, 1, 0, 0		
3			0, 0, 1, 1		
4			1, 1, 1, 1		
5	0	1	0, 0, 0, 1	1	0
6			0, 0, 1, 0		
7			1, 1, 0, 1		
8			1, 1, 1, 0		
9	1	0	0, 1, 0, 0	1	0
10			0, 1, 1, 1		
11			1, 0, 0, 0		
12			1, 0, 1, 1		
13	1	1	0, 1, 0, 1	0	1
14			0, 1, 1, 0		
15			1, 0, 0, 1		
16			1, 0, 1, 0		

(b)

ID	$x[u-1]$, $\tilde{x}[u-1]$, $c[u]$, $\tilde{c}[u]$, $(\Delta c[u])$			
	$(c[u-1], \tilde{c}[u-1])$ $=(0, 0)$	$(c[u-1], \tilde{c}[u-1])$ $=(1, 1)$	$(c[u-1], \tilde{c}[u-1])$ $=(0, 1)$	$(c[u-1], \tilde{c}[u-1])$ $=(1, 0)$
1	0, 0, 0, 0 (0)	1, 1, 1, 1 (0)	0, 1, 0, 1 (1)	1, 0, 1, 0 (1)
2	1, 1, 1, 1 (0)	0, 0, 1, 1 (0)	1, 0, 1, 1 (0)	0, 1, 1, 1 (0)
3	1, 1, 0, 0 (0)	0, 0, 0, 0 (0)	1, 0, 0, 0 (0)	0, 1, 0, 0 (0)
4	0, 0, 0, 0 (0)	1, 1, 1, 1 (0)	0, 1, 0, 1 (1)	1, 0, 1, 0 (1)
5	0, 1, 0, 0 (0)	1, 0, 1, 0 (1)	0, 0, 0, 0 (0)	1, 1, 1, 0 (1)
6	1, 0, 0, 0 (0)	0, 1, 0, 1 (1)	1, 1, 0, 1 (1)	0, 0, 0, 0 (0)
7	1, 0, 1, 0 (1)	0, 1, 1, 1 (0)	1, 1, 1, 1 (0)	0, 0, 1, 0 (1)
8	0, 1, 0, 1 (1)	1, 0, 1, 1 (0)	0, 0, 0, 1 (1)	1, 1, 1, 1 (0)
9	0, 1, 0, 1 (1)	1, 0, 1, 1 (0)	0, 0, 0, 1 (1)	1, 1, 1, 1 (0)
10	1, 0, 0, 0 (0)	0, 1, 0, 1 (1)	1, 1, 0, 1 (1)	0, 0, 0, 0 (0)
11	1, 0, 1, 0 (1)	0, 1, 1, 1 (0)	1, 1, 1, 1 (0)	0, 0, 1, 0 (1)
12	0, 1, 0, 0 (0)	1, 0, 1, 0 (1)	0, 0, 0, 0 (0)	1, 1, 1, 0 (1)
13	0, 0, 0, 0 (0)	1, 1, 1, 1 (0)	0, 1, 0, 1 (1)	1, 0, 1, 0 (1)
14	1, 1, 0, 1 (1)	0, 0, 0, 1 (1)	1, 0, 0, 1 (1)	0, 1, 0, 1 (1)
15	1, 1, 1, 0 (1)	0, 0, 1, 0 (1)	1, 0, 1, 0 (1)	0, 1, 1, 0 (1)
16	0, 0, 0, 0 (0)	1, 1, 1, 1 (0)	0, 1, 0, 1 (1)	1, 0, 1, 0 (1)

before a differential of SPECK. Suppose the input difference of the differential is $\rho = (\rho_L, \rho_R)$ and its probability is 2^{-w}. Further, suppose the input difference ρ of the differential propagates backward to (α, β) marked in Fig. 6 such that, according to Proposition 1 and 2, α has the form

$$\alpha = 0b\underbrace{*\cdots*}_{s}c\underbrace{0\cdots0}_{n-s-1},$$

where the most significant s bits of α can take any possible value and c is a known constant. As for β, it can be computed from ρ and thus is fixed.

We then construct structures at the beginning of the second round. In this situation, pairs formed from the structures will satisfy the input difference ρ of the differential probabilistically rather than deterministically. We will show that the required data remains the same as that in Dinur's attacks [Din14].

The goal of the data collection phase is to generate pairs (x_1, y_1) and (x'_1, y'_1) whose difference is (α, β). From such pairs, the difference between $(x_1 \oplus k_0 \ggg a)\boxplus(y_1 \oplus k_0)$ and $(x'_1 \oplus k_0 \ggg a)\boxplus(y'_1 \oplus k_0)$ is expected to equal γ with probability 2^{-s} under some unknown round keys k_0. To achieve this, we could generate such pairs using twin structures:

$$\begin{aligned} &S = \{(x_1, y_1) | x_1[n-s-1:0], y_1 = c_1, x_1[n-1:n-s] \in \{0,1\}^s\}, \\ &S' = S \oplus \Delta = \{(x'_1, y'_1) | (x_1, y_1) \in S, x'_1[n-s-1:0] = x_1[n-s-1:0] \oplus c0\cdots0, \\ &\quad y'_1 = y_1 \oplus \beta, x'_1[n-1:n-s] \in \{0,1\}^s\}, \text{where } \Delta = 0b0\cdots0c0\cdots0||\beta. \quad (1) \end{aligned}$$

From such a pair of twin structures, we can generate 2^{2s} pairs of (x_1, y_1) and (x'_1, y'_1), and 2^s pairs are expected to lead to γ, meeting the input difference of the distinguisher.

Suppose 2^t pairs of twin structures are used. We need 2^w pairs of (x_1, y_1) and (x'_1, y'_1) that satisfy the input difference of the distinguisher to have at least one right pair for the differential. Therefore, $2^{s+t} = 2^w$ and $s + t = w$. The total data complexity for 2^t pairs of twin structures is

$$D = 2^{t+s+1} = 2^{w+1},$$

which is the same as that in Dinur's attacks. This confirms again that Type-II structures do not help to reduce the data complexity. Therefore, in this paper, the purpose of using Type-II structures is not to reduce the data complexity, but to reduce the time complexity. This will be shown in Section 4.2.

3.3 Combining Both Types of Structures

Recall that the use of Type-I structures helps to reduce the data complexity. What if we use both types of structures simultaneously? Can we reduce the time and data complexity at the same time?

Suppose we have two differentials $(\rho_1 \to \delta)$ and $(\rho_2 \to \delta)$ of the same probability for SPECK and ρ_1 and ρ_2 propagate back over one round to (α_1, β_1)

and (α_2, β_2), respectively, where α_1 and α_2 share the same number of zero bits in the lower part. Then we choose four structures as follows.

$$S, S \oplus \Delta_1, S \oplus \Delta_2, S \oplus \Delta_1 \oplus \Delta_2, \tag{2}$$

where Δ_1 and Δ_2 are derived from (α_1, β_1) and (α_2, β_2) in a way presented in Equation (1). Actually, these four sets form a Type-I structure of Type-II structures.

If the size of each Type-II structure is 2^s, then we can get about 2×2^s pairs satisfying ρ_1 and about 2×2^s pairs satisfying ρ_2. Now the ratio between the number of messages and the number of message pairs meeting one of the input differences becomes 1, *i.e.*, $\mathcal{R}_{m/p} = 1$. Generally, if there are h differentials, $\mathcal{R}_{m/p} = \frac{2}{h}$, which is much lower than 2. This shows that the combination of both types of structures may bring benefits of each together. This will be shown in Section 4.3.

4 New Key Recovery Algorithm for SPECK

In this section, we propose three generic key recovery algorithms for SPECK which respectively use Type-I structures, Type-II structures and a combination of both types of structures. In particular, we will highlight the benefit each algorithm may bring.

4.1 Algorithm A Using Type-I Structures

Suppose we have a set of h differentials $\{\rho^j \to \delta\}$ for $j = 1, \cdots, h$ over r rounds and the corresponding probabilities are p^j where

$$\sum_{j=1}^{h} p^j = \hat{p} = 2^{-\hat{w}},$$

as given in the left part of Fig. 7. We build Type-I structures using the h input differences and adapt Dinur's attack to the multiple-differential case. The goal is to attack $1 + r + m$ rounds. The steps of Algorithm A are as follows.

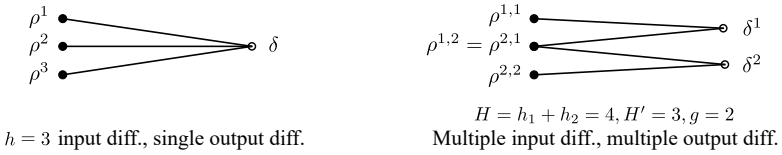

Fig. 7: Two examples where Type-I structures can be applied

For $i = 1, \cdots, 2^t$:

1. Construct a structure S of 2^h data by taking the h input differences ρ^j as a basis.
2. Guess the last $m-2$ round keys. Do partial decryptions to have intermediate values $(x_{r+3},\, y_{r+3})$.
   (a) For each input difference ρ^j, $j = 1, \cdots, h$, generate 2^{h-1} pairs of (P, P') meeting the ρ^j difference. For each pair,
      i. Run the 2-round attack using $(\Delta x_{r+1}, \Delta y_{r+1}) = \delta$, $(\Delta x_{r+3}, \Delta y_{r+3})$ and $(x_{r+3},\, y_{r+3})$ and return solutions of k_{r+1} and k_{r+2};
      ii. For each solution together with the guessed $m-2$ subkeys, recover the master key and test it using trial encryptions, and return it if the trial encryptions succeed.

It takes $2^t \times 2^h$ data and let $2^t \times 2^{h-1} \times \sum_j p^j = 1$, meaning the expected number of right pair is one. Therefore $h + t = \hat{w} + 1$. Then we can summarize the complexities of Attack A as follows.

- $D_{\mathrm{A}} = 2^{\hat{w}+1}$ plaintexts;
- $T_{\mathrm{A}} = 2^t \times 2^{(m-2)n} \times h \times 2^{h-1} \times 2 = c_{\mathrm{A}} \cdot 2^{(m-2)n+\hat{w}} \times h = c_{\mathrm{A}} \cdot 2^{(m-2)n} \frac{1}{(\sum p^j)/h}$ encryptions, where $c_{\mathrm{A}} = 2$, $m = 2, 3, 4$;
- $M_{\mathrm{A}} = \max\{2^{22}, 2^h\}$.

Extended to multiple output differences. The above algorithm can also be extended to the case with multiple output differences (see the right part of Fig. 7). Suppose we have g output differences δ^i, $i = 1, \cdots, g$. Each output difference δ^i corresponds to h_i input differences, *i.e.*, $\{\rho^{i,j} \to \delta^i\}$ of probability $p^{i,j}$ for $j = 1, \cdots, h_i$. Let

$$\sum_{i,j} p^{i,j} = \hat{p} = 2^{-\hat{w}}, H = h_1 + \cdots + h_g.$$

Suppose among the H input differences $\rho^{i,j}$s, H' of them are linearly independent when we treat them as binary vectors. We can attack $1 + r + m$ rounds following the procedure below.

For $i = 1, \cdots, 2^t$:

1. Construct a structure S of $2^{H'}$ data by taking the H' independent input differences $\rho^{i,j}$ as a basis.
2. Guess the last $m-2$ round keys. Do partial decryptions to have intermediate values $(x_{r+3},\, y_{r+3})$.
   (a) For each input difference $\rho^{i,j}$ and each output difference δ^i, $j = 1, \cdots, h_i$, $i = 1, \cdots, g$, generate $2^{h_i-1} \times 2^{H'-h_i}$ pairs of (P, P') meeting the $\rho^{i,j}$ difference. For each pair:
      i. Run the 2-round attack using $(\Delta x_{r+1}, \Delta y_{r+1}) = \delta$, $(\Delta x_{r+3}, \Delta y_{r+3})$ and $(x_{r+3},\, y_{r+3})$ and return solutions of k_{r+1} and k_{r+2};
      ii. For each returned value of k_{r+1} and k_{r+2} together with the guessed $m-2$ subkeys, recover the master key and test it using trial encryptions.

It takes $2^t \times 2^{H'}$ data and let $2^t \times 2^{H'-1} \times \sum_{i,j} p^{i,j} = 1$. Therefore $H' + t = \hat{w} + 1$. Then we can summarize the complexities of Attack A as follows.

- $D_{\mathrm{A}} = 2^{\hat{w}+1}$ plaintexts;
- $T_{\mathrm{A}} = 2^t \times 2^{(m-2)n} \times H \times 2^{H'-1} \times 2 = c_{\mathrm{A}} \cdot 2^{(m-2)n+\hat{w}} \times H = c_{\mathrm{A}} \cdot 2^{(m-2)n} \frac{1}{\left(\sum_{i,j} p^{i,j}\right)/H} = c_{\mathrm{A}} \cdot 2^{(m-2)n} \cdot \frac{H}{\hat{p}}$ encryptions, where $c_{\mathrm{A}} = 2$, $m = 2, 3, 4$;
- $M_{\mathrm{A}} = \max\{2^{22}, 2^{H'}\}$.

Example. Table 4 lists some examples which demonstrate the improvement brought by Algorithm A. In these two attacks on 14-round SPECK32/64, the probability of all differential trails we use is 2^{-30}. The result shows the data complexity is reduced by using Type-I structures.

Table 4: Example of attacks using Algorithm A

Variants	Split	#Trails	h	g	Data	Time	Mem.	Ref
32/64	1+9+4	1	1	1	2^{31}	2^{63}	2^{22}	[Din14]
	1+9+4	5	5	1	$2^{28.68}$	2^{63}	2^{22}	Alg. A
	1+9+4	15	15	6	$2^{27.09}$	2^{63}	2^{22}	Alg. A

Experiment. We mainly test whether we can get right pairs as expected. We construct Type-I structures and use a 5-round distinguisher for the 10-round SPECK32/64 attack. The experiment results demonstrate that no matter how many right pairs we set, we can always get the expected number of right pairs.

4.2 Algorithm B Using Type-II Structures

Suppose we have a differential $\rho = (\rho_L, \rho_R) \to \delta = (\delta_L, \delta_R)$ with probability $p = 2^{-w}$. Also, suppose the output difference δ leads to n_{FIL} forward inactive lower bits for the addition in the next round and the input difference results in n_{BIL} backward inactive lower bits for the addition in the earlier round. The goal of Algorithm B is to attack $2 + r + (m-1)$ rounds using Type-II structures for $m = 3, 4$.

Switch to the counting method. When we use Type-II structures, there are two rounds before the distinguisher. If we guess the last $m-2$ round keys, the time complexity will exceed 2^{mn}, because we need to process more pairs than that in Dinur's attack. Thus, in our attack we only guess the last $m-3$ round keys. For Round $r+3$ and Round $r+4$, we can also mount a 2-round attack to recover round keys k_{r+2} and k_{r+3}. However, due to Property 2, testing the recovered round keys using trial encryptions is possible only when we have information of m consecutive round keys. In this case, we only have the last $m-1$ consecutive

rounds keys (or with the first round key). This makes the enumerating method infeasible.

A way to get around this issue is to switch to the common counting method. The number of message pairs that meet the input difference ρ and the output difference δ under each possible key value is recorded. The key values with the highest counters are the likeliest right key. Thus, a shortlist of round key candidates can be obtained according to the counters, which helps to give an efficient key recovery of the master key.

The algorithm. In this algorithm, we guess the involved round key bits of k_0 which are needed for verifying the input difference ρ. As the higher bits of both k_0 and $k_0 \ggg a$ affect the verification, more than $s = n - n_{\mathsf{BIL}} - 1$ bits of k_0 are needed. In the following procedures, we guess these bits of k_0 together with k_{r+m} if $m = 4$ and make counters for other involved round keys k_{r+2}, k_{r+3}. The detailed procedures of Algorithm B are as follows.

1. Construct 2^t pairs of twin structures (S, S') in a way described in Equation 1. Each structure has 2^s plaintext-ciphertext pairs, where $s = n - n_{\mathsf{BIL}} - 1$.
2. Guess $\min\{n - n_{\mathsf{BIL}} + a, n\}$ bits of k_0 and the full k_{r+m} if $m = 4$. Do partial encryptions and decryptions.
   (a) Initialize counters for k_{r+2} and k_{r+3}.
   (b) For each pair of twin structures:
      i. Store the data of one structure into a hash table according to the state before the key addition of the second round and $(x_{r+4} \oplus y_{r+4})[n_{\mathsf{FIL}} + b : b]$ and look up the table with data from the other structure. There will be $2^{s - n_{\mathsf{FIL}} - 1}$ pairs of data meeting the input difference and the $(n_{\mathsf{FIL}} + 1)$-bit fixed difference.
      ii. For each pair, mount the 2-round attack to recover k_{r+2}, k_{r+3} and update the counters.
   (c) Select 2^ℓ (*e.g.*, $l < n$) candidates for k_{r+2}, k_{r+3} with top counters. Guess k_{r+1} and test the correctness by trial encryptions.

If we set the number of right pairs to one, then $2^t \times 2^s \times p = 1$, $t + s = w$, and the data complexity is 2^{w+1}, which is the same as that in Dinur's attack. If we want to have a higher success probability, we can increase the data complexity. For the computation of success probability, we follow the formula in Selçuk's work [Sel08] (see Equation 7). However, for a convenient comparison of time complexities between Algorithm B and Dinur's attack, we simply let the data complexity be the same (while we may trade the data for a higher success probability in concrete applications). The memory complexity for storing data and counters is $D + 2^{2n} \approx 2^{2n}$, and the time complexity is dominated by step (b) and (c). Specifically,

- $D_{\mathrm{B}} = 2^{w+1}$ plaintexts;
- $T_{\mathrm{B}} = 2^{(m-3)n + \min\{n - n_{\mathsf{BIL}} + a, n\} + t}(2^{s+1} + 2^{s - n_{\mathsf{FIL}} - 1}) + 2^{(m-2)n + \min\{n - n_{\mathsf{BIL}} + a, n\} + \ell}$ encryptions, $m = 3, 4$;
- $M_{\mathrm{B}} = 2^{2n}$ for counters.

As the number of candidate keys 2^ℓ is flexible, it is reasonable to assume the second term of T_{B} is not dominant. Let us focus on the first term. We can see that if n_{BIL} is greater than the rotation number a, this attack is already more efficient than Dinur's attack in terms of time complexity.

Improvement. Note that the time complexities of step (i) and (ii) are 2^{s+1} and $2^{s-n_{\mathsf{FIL}}-1}$ under each guess, respectively, which are not balanced. A strategy to balance them is to guess fewer key bits of k_0. There are three types of bits in k_0: bits used once (bits in the red lines in Fig. 8), twice (bits in the blue line in Fig. 8) and none for verifying ρ difference. Our strategy is to guess all the bits that are used twice and partial bits that are used only once. Let us analyze case by case.

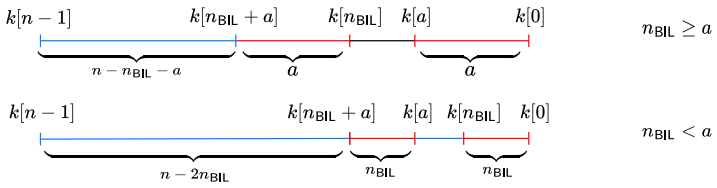

Fig. 8: Guess bits of k_0, where bits in the blue line are all guessed and y bits in the red lines are guessed.

- When $n_{\mathsf{BIL}} \geq a$, $n - n_{\mathsf{BIL}} + a$ bits of k_0 are needed. If we guess $n - n_{\mathsf{BIL}} - a + y$ bits of k_0 with $0 \leq y \leq 2a$, we will get a $(n - n_{\mathsf{BIL}} - 2a + y - 1)$-bit filter, *i.e.*, an $(s - 2a + y)$-bit filter, instead of an s-bit filter. Now the time complexity of step (ii) becomes $2^{s-n_{\mathsf{FIL}}-1+2a-y}$. Meanwhile, the number of guessed key bits of k_0 is reduced by $2a - y$ bits and the memory complexity is increased by 2^{2a-y}.
  - When $2a \geq n_{\mathsf{FIL}} + 1$, choose $y = 2a - n_{\mathsf{FIL}} - 1$, the advantage of time complexity over Dinur's attack is $ad = n_{\mathsf{BIL}} - a + n_{\mathsf{FIL}} + 1$.
  - When $2a < n_{\mathsf{FIL}} + 1$, choose $y = 0$, the advantage of time complexity over Dinur's attack is $ad = n_{\mathsf{BIL}} + a$.
- When $n_{\mathsf{BIL}} < a$, a full k_0 is needed. If we guess $n - 2n_{\mathsf{BIL}} + y$ bits of k_0 with $0 \leq y \leq 2n_{\mathsf{BIL}}$, we will get an $(s - 2n_{\mathsf{BIL}} + y)$-bit filter, instead of an s-bit filter. Now the time complexity of step (ii) becomes $2^{s-n_{\mathsf{FIL}}-1+2n_{\mathsf{BIL}}-y}$. Meanwhile, the number of guessed key bits of k_0 is reduced by $2n_{\mathsf{BIL}} - y$ bits and the memory complexity is increased by $2^{2n_{\mathsf{BIL}}-y}$.
  - When $2n_{\mathsf{BIL}} \geq n_{\mathsf{FIL}} + 1$, choose $y = 2n_{\mathsf{BIL}} - n_{\mathsf{FIL}} - 1$, the advantage of time complexity over Dinur's attack is $ad = n_{\mathsf{FIL}} + 1$.
  - When $2n_{\mathsf{BIL}} < n_{\mathsf{FIL}} + 1$, choose $y = 0$, the advantage of time complexity over Dinur's attack is $ad = 2n_{\mathsf{BIL}}$.

The advantage of Algorithm B over Dinur's attack in terms of the time complexity is summarized in Table 5.

Table 5: Advantage of Algorithm B

$n_{\mathsf{BIL}} \geq a$	$2a \geq n_{\mathsf{FIL}} + 1$	$y = 2a - n_{\mathsf{FIL}} - 1$	$ad = n_{\mathsf{BIL}} - a + n_{\mathsf{FIL}} + 1$
	$2a < n_{\mathsf{FIL}} + 1$	$y = 0$	$ad = n_{\mathsf{BIL}} + a$
$n_{\mathsf{BIL}} < a$	$2n_{\mathsf{BIL}} \geq n_{\mathsf{FIL}} + 1$	$y = 2n_{\mathsf{BIL}} - n_{\mathsf{FIL}} - 1$	$ad = n_{\mathsf{FIL}} + 1$
	$2n_{\mathsf{BIL}} < n_{\mathsf{FIL}} + 1$	$y = 0$	$ad = 2n_{\mathsf{BIL}}$

Example. We take the attack on SPECK128/256 as an example, as shown in Table 6. In the attack a 19-round differential trail with probability 2^{-119} is used where $n_{\mathsf{BIL}} = 23, n_{\mathsf{FIL}} = 2$. We mount attacks on 24 rounds using Algorithm B and set the number of right pairs $\mu = 3$ and let $\ell = 55$ so that the success probability is high (about 90% in the calculation). As can be seen the time complexity is reduced by using Type-II structures and by taking $n_{\mathsf{BIL}}, n_{\mathsf{FIL}}$ into account.

Table 6: Example of attacks using Algorithm B

Variants	Split	Prob.	n_{BIL}	n_{FIL}	Data	Time	Mem.	Ref
128/256	1+19+4	2^{-119}	-	-	2^{120}	2^{248}	2^{22}	[FWG+16]
	2+19+3	2^{-119}	23	2	$2^{121.58}$	$2^{231.58}$	2^{131}	Alg. B

Experiment. Using Algorithm B, we try to mount a 13-round attack on SPECK32/64 with a 8-round distinguisher. The result shows that the correct key ranks high. We set $\mu = 2$ and the highest counter is 6. Specifically, among all possible keys, 16 values take this count, including the correct key.

4.3 Algorithm C Using Type-I and Type-II Structures

Suppose we have g output differences δ^i, $i = 1, \cdots, g$, all of which lead to at least n_{FIL} forward inactive lower bits for the addition in the next round. Each output difference δ^i corresponds to h_i input differences, *i.e.*, $\rho^{i,j} \to \delta^i$ of probability $p^{i,j}$ for $j = 1, \cdots, h_i$. Suppose all these input differences result in at least n_{BIL} backward inactive lower bits for the addition in the earlier round. Let

$$\sum_{i,j} p^{i,j} = \hat{p} = 2^{-\hat{w}}, H = h_1 + \cdots + h_g.$$

Suppose among the H input differences $\rho^{i,j}$s, H' of them are linearly independent. The goal is to attack $2 + r + (m-1)$ rounds for $m = 3, 4$.

The algorithm. This algorithm is a combination of Algorithm A and B.

1. Construct 2^t Type-I structures of $2^{H'}$ Type-II structures and get the corresponding ciphertexts. Each type-II structure has 2^s plaintexts, where $s = n - n_{\mathsf{BIL}} - 1$.
2. If $n_{\mathsf{BIL}} \geq a$, guess $n - n_{\mathsf{BIL}} - a + y$ bits of k_0 with $0 \leq y \leq 2a$; otherwise, guess $n - 2n_{\mathsf{BIL}} + y$ bits of k_0 with $0 \leq y \leq 2n_{\mathsf{BIL}}$. Guess the full k_{r+m} if $m = 4$. Do partial encryptions and decryptions.
   (a) Initialize counters for k_{r+2}, k_{r+3}, and $2 \times \min\{a, n_{\mathsf{BIL}}\} - y$ bits of k_0.
   (b) For each Type-I structure, each input difference $\rho^{i,j}$ and each output difference δ^i, $i = 1, \cdots, g$, $j = 1, \cdots, h_i$, there are $2^{h_i - 1} \times 2^{H' - h_i}$ pairs of twin structures.
      i. For each pair of twin structures, store the data of one structure into a hash table according to the state before the key addition of the second round and $(x_{r+m} \oplus y_{r+m})[n_{\mathsf{FIL}} + b : b]$ and look up the table with data from the other structure. Then, when $n_{\mathsf{BIL}} \geq a$ (resp. $n_{\mathsf{BIL}} < a$) there will be $2^{s - n_{\mathsf{FIL}} - 1 + 2a - y}$ (resp. $2^{s - n_{\mathsf{FIL}} - 1 + 2n_{\mathsf{BIL}} - y}$) pairs of data partly meeting the input difference and the $(n_{\mathsf{FIL}} + 1)$-bit fixed difference.
      ii. For each pair of data, mount the 2-round attack to recover k_{r+2}, k_{r+3}; recover other involved bits of k_0 by looking up a precomputed table[5]. Update the counters accordingly.
   (c) Select 2^ℓ (*e.g.*, $\ell < n$) candidates with top counters. Guess k_{r+1} and test the correctness by trial encryptions.

It takes $2^t \times 2^{H'} \times 2^s$ data. For a convenient comparison of time complexities between Algorithm C and Dinur's attack, we simply set the number of right pairs to one. Then we have $2^t \times 2^{H'-1} \times 2^s \times \sum_{i,j} p^{i,j} = 1$ and thus $H' + t + s = \hat{w} + 1$. However, we may trade the data for a higher success probability in attacks on concrete applications. Assume n_{FIL} is much smaller than s, which holds in general. Then we can summarize the complexities of Algorithm C as follows.

- $D_{\mathrm{C}} = 2^{\hat{w}+1}$ plaintexts;
- When $n_{\mathsf{BIL}} \geq a$, $T_{\mathrm{C}} = 2^{(m-3)n+n-n_{\mathsf{BIL}}-a+y} \times (2^t \times H \times 2^{H'-1} \times (2^{s+1} + 2^{s-n_{\mathsf{FIL}}-1+2a-y}) + 2^{n+\ell}))$ encryptions for $m = 3, 4$.
  - If $2a \geq n_{\mathsf{FIL}} + 1$, $T_{\mathrm{C}} = 2^{(m-2)n-(n_{\mathsf{BIL}}-a+n_{\mathsf{FIL}}+1)} \times (\frac{H}{\hat{p}} \times 2 + 2^{n+\ell})$. $M_{\mathrm{C}} = 2^{2n+n_{\mathsf{FIL}}+1}$ for counters.
  - If $2a < n_{\mathsf{FIL}} + 1$, $T_{\mathrm{C}} = 2^{(m-2)n-(n_{\mathsf{BIL}}+a)} \times (\frac{H}{\hat{p}} \times 2 + 2^{n+\ell})$. $M_{\mathrm{C}} = 2^{2n+2a}$ for counters.
- When $n_{\mathsf{BIL}} < a$, $T_{\mathrm{C}} = 2^{(m-3)n+n-2n_{\mathsf{BIL}}+y} \times (2^t \times H \times 2^{H'-1} \times (2^{s+1} + 2^{s-n_{\mathsf{FIL}}-1+2n_{\mathsf{BIL}}-y}) + 2^{n+\ell}))$ encryptions for$m = 3, 4$.
  - If $2n_{\mathsf{BIL}} \geq n_{\mathsf{FIL}} + 1$, $T_{\mathrm{C}} = 2^{(m-2)n-(n_{\mathsf{FIL}}+1)} \times (\frac{H}{\hat{p}} \times 2 + 2^{n+\ell})$. $M_{\mathrm{C}} = 2^{2n+n_{\mathsf{FIL}}+1}$ for counters.
  - If $2n_{\mathsf{BIL}} < n_{\mathsf{FIL}} + 1$, $T_{\mathrm{C}} = 2^{(m-2)n-2n_{\mathsf{BIL}}} \times (\frac{H}{\hat{p}} \times 2 + 2^{n+\ell})$. $M_{\mathrm{C}} = 2^{2n+2n_{\mathsf{BIL}}}$ for counters.

[5] This precomputed table takes a small memory of $2^{3 \times (2 \times \min\{a, n_{\mathsf{BIL}}\} - y)}$.

As Algorithm A, the data complexity is reduced. The total advantage of this attack in time complexity over Dinur's attack is the same as shown in Algorithm B of Section 4.2.

Data and the probability of success. We can calculate the success probability using Equation 7 as well. Notably, to have a competitive probability of success, having one right pair in the counting based algorithm is usually not enough. Therefore, in concrete attack we increase the data complexity by a factor μ to $\mu \times 2^{\hat{w}+1}$. We typically set $\mu = 3$ to have a reasonable success probability. In turn, this increases the data complexity, probably leading to a data complexity higher than that of the Dinur's attack.

Example. We take SPECK32/64 as an example. The best 9-round differential trails of SPECK32 have a probability 2^{-30}. Among them, a few have $n_{\mathsf{BIL}} = 3$, $n_{\mathsf{FIL}} = 3$. Using Algorithm C, we set $\mu = 3$ and let $\ell = 14$. When 3 trails with total probability $2^{-28.42}$ are used, the success probability of attack is about 66%. When two more trails are added, the total probability of all 5 differential trails is $2^{-27.68}$, and the success probability is about 76%. The results of these attacks are listed in Table 7.

Table 7: Example of attacks using Algorithm C

Variants	Split	#Trail(s)	n_{BIL}	n_{FIL}	Data	Time	Mem.	Ref
32/64	1+9+4	1	-	-	2^{31}	2^{63}	2^{22}	[Din14]
	2+9+3	3	3	3	2^{31}	$2^{60.58}$	2^{36}	Alg. C
	2+9+3	5	2	3	$2^{30.26}$	$2^{60.58}$	2^{36}	Alg. C

Experiment. We mount a 10-round attack on SPECK32/64 using Algorithm C. In this experiment, we use four trails, the probability of which are all 2^{-13}. When we provide 2^{14} message pairs, *i.e*, $\mu = 2$, whose differences match the input difference of the distinguisher, the results show that the highest counter is 5 and the correct key does take this count. Among all possible keys, there are roughly 2^{12} candidates whose counters are more than or equal to 2.

4.4 Discussions and Extensions

In this paper, we find a way to construct Type-II structures for SPECK. It allows to add one more round before the differential distinguisher and leads to better attacks in certain situations. From the application to SPECK, we can see that the benefit of Type-II structures is weakened by the nonlinearity of the key schedule and the rotation $\ggg a$ in the round function. Thus, it is expected that the benefit of Type-II structures may vary from cipher to cipher.

Limitation of Algorithm B and C. The algorithms using Type-II structures can be applied to SPECK variants where $m = 3, 4$. When $m = 2$, the number of pairs to be processed 2^{t+2s} may exceed 2^{mn} before guessing any round key. Note this is usually the case since $t + s$ is close to $2n$ in most attacks. Therefore, our new attack is difficult to apply to the variants of SPECK with $m = 2$, namely, when the block size equals the key size.

Extensions to variants of differential attacks. Besides the standard differential cryptanalysis, the new technique for constructing Type-II structures is potentially useful in various variants of differential attacks on ARX ciphers. Such variants include boomerang/rectangle attacks, differential-linear attacks, impossible differential attacks, etc.

Comparison with [BdST$^+$22]. Different from plaintext structures, in [BdST$^+$22] the authors propose a meet-in-the-middle technique for improving key recovery attacks on ciphers with slow diffusion and apply it to SPECK. They focus on the bottom part of the cipher. Instead of checking the output difference δ of the distinguisher, they propagate δ forward for some rounds and build a table for storing (almost) all possible differences. Then they guess the round keys of the last rounds, decrypt to the meeting point and check if the obtained intermediate difference falls in the table. This technique helps to reduce the time complexity of attacks on SPECK32 and SPECK64.

5 Applications to SPECK

In this section, we apply the three new algorithms proposed in the previous section to key recovery attacks of SPECK. Before we mount key recovery attacks on SPECK, we need to prepare suitable differentials or differential trails for these algorithms so that better attacks can be realized. Roughly, high-probability differential trails or differentials that have large $n_{\mathsf{BIL}}, n_{\mathsf{FIL}}$ are desirable. With this in mind, we reuse the SAT-based method [SWW21], which is briefly described in Appendix B, for searching good differential trails for SPECK. In the following, we provide information on the trails we find and present new attacks on all variants of SPECK.

5.1 New Differential Trails and Differentials

In this subsection, we find suitable differential trails and differentials for each algorithm separately. In a concrete attack, the underlying distinguisher may be composed of a single trail/differential or multiple trails/differentials. When multiple trails/differentials are used, we may select different sets of trails or differentials to mount better attacks. To make it clear which distinguisher is used in the attack, we label each distinguisher with an ID.

For Algorithm A, we aim to find all differential trails and differentials with the best probability. However, for $2n > 64$, the search becomes time-consuming and we can only find some good ones. The results are displayed in Table 8.

Table 8: Differential trails and differentials of SPECK for Algorithm A, where 'Number' means the number of trails/differentials

Versions	Rnds	Trails or Differentials	Number	$\sum p_i$	ID
32	9	Trails	15	$2^{-26.09}$	1
48	11	Trails	3	$2^{-43.42}$	2
	12	Differentials	2	$2^{-45.78}$	3
64	15	Trails	4	2^{-60}	4
	15	Differentials	4	$2^{-58.91}$	5
96	16	Trails	2	2^{-86}	6
	17	Differentials	6	$2^{-92.17}$	7
128	19	Trails	7	2^{-117}	8
	20	Differentials	6	$2^{-121.37}$	9

For Algorithm B, we search for single trails and differentials of SPECK, whose probability and n_{BIL}, n_{FIL} are as large as possible. We summarize the trails and differentials we obtain in Table 9.

Table 9: Differential trails and differentials of SPECK for Algorithm B

Versions	Rnds	Trail or Differential	Prob.	n_{BIL}	n_{FIL}	ID
32	9	Trail	2^{-30}	3	3	10
	10	Differential	$2^{-30.39}$	2	2	11
48	11	Trail	2^{-46}	11	7	12
	12	Differential	$2^{-46.78}$	0	7	13
64	15	Trail	2^{-62}	5	2	14
	15	Differential	$2^{-60.53}$	5	2	15
96	16	Trail	2^{-87}	7	0	16
	16	Trail	2^{-92}	15	0	17
	17	Differential	$2^{-93.98}$	7	2	18
128	19	Trail	2^{-119}	23	2	19
	20	Differential	$2^{-123.17}$	23	0	20

For Algorithm C, we form different sets of trails/differentials from the single trails/differentials we obtained for Algorithm B, so that the average probability and n_{BIL}, n_{FIL} are as large as possible. The results are shown in Table 10.

Table 10: New characteristics and differentials for Algorithm C, where 'Number' means the number of trails/differentials

Versions	Rnds	Trails or Differentials	Number	$\sum p_i$	n_{BIL}	n_{FIL}	ID
32	8	Trails	10	$2^{-21.68}$	3	3	21
	9	Trails	5	$2^{-27.68}$	2	3	22
48	11	Trails	2	$2^{-45.42}$	11	7	23
	12	Differentials	6	$2^{-45.89}$	3	2	24
64	15	Trails	11	$2^{-59.30}$	2	3	25
	15	Differentials	11	$2^{-58.24}$	2	3	26
96	16	Trails	6	$2^{-85.19}$	7	0	27
	17	Differentials	23	$2^{-91.03}$	7	2	28
128	19	Trails	6	$2^{-117.19}$	23	2	29
	20	Differentials	6	$2^{-121.37}$	23	0	30

5.2 Summary of Results

In this subsection, we mount key recovery attacks on SPECK by applying the new algorithms in Section 4 using differential distinguishers in Table 8, 9 and 10.

The results are summarized in Table 11 and the comparison to the related works is presented in Table 1.

Table 11: All improved attacks using our new Algorithm

Variants	Split	ID	Prob.	n_{BIL},n_{FIL}	Data	Time	Mem.	Method
32/64	2+8+3	21	$2^{-21.68}$	3,3	$2^{24.26}$	$2^{55.58}$	2^{36}	Alg. C
	1+9+4	1	$2^{-26.09}$	-	$2^{27.09}$	2^{63}	2^{22}	Alg. A
	2+9+3	22	$2^{-27.68}$	2,3	$2^{30.26}$	$2^{60.58}$	2^{36}	Alg. C
48/72	1+11+3	2	$2^{-43.42}$	-	$2^{44.42}$	2^{70}	2^{22}	Alg. A
	2+11+2	23	$2^{-45.42}$	11,7	2^{48}	2^{62}	2^{56}	Alg. C
	1+12+3	3	$2^{-45.78}$	-	$2^{46.78}$	$2^{71.78}$	2^{22}	Alg. A
48/96	1+11+4	2	$2^{-43.42}$	-	$2^{44.42}$	2^{94}	2^{22}	Alg. A
	2+11+3	23	$2^{-45.42}$	11,7	2^{48}	2^{86}	2^{56}	Alg. C
	1+12+4	3	$2^{-45.78}$	-	$2^{46.78}$	$2^{95.78}$	2^{22}	Alg. A
64/96	1+15+3	4	2^{-60}	-	2^{61}	2^{95}	2^{22}	Alg. A
	1+15+3	5	$2^{-58.91}$	-	$2^{59.91}$	$2^{93.91}$	2^{22}	Alg. A
	2+15+2	15	$2^{-60.53}$	5,2	$2^{63.11}$	$2^{92.11}$	2^{67}	Alg. B
	2+15+2	25	$2^{-59.30}$	2,3	$2^{61.88}$	$2^{93.34}$	2^{68}	Alg. C
	2+15+2	26	$2^{-58.24}$	2,3	$2^{60.82}$	$2^{92.28}$	2^{68}	Alg. C
64/128	1+15+4	4	2^{-60}	-	2^{61}	2^{127}	2^{22}	Alg. A
	1+15+4	5	$2^{-58.91}$	-	$2^{59.91}$	$2^{125.91}$	2^{22}	Alg. A
	2+15+3	15	$2^{-60.53}$	5,2	$2^{63.11}$	$2^{124.11}$	2^{67}	Alg. B
	2+15+3	25	$2^{-59.30}$	2,3	$2^{61.88}$	$2^{125.34}$	2^{68}	Alg. C
	2+15+3	26	$2^{-58.24}$	2,3	$2^{60.82}$	$2^{124.28}$	2^{68}	Alg. C
96/96	1+16+2	6	2^{-86}	-	2^{87}	2^{88}	2^{22}	Alg. A
	1+17+2	7	$2^{-92.17}$	-	$2^{93.17}$	$2^{95.75}$	2^{22}	Alg. A

Continued on next page

Table 11 – continued from previous page

Variants	Split	ID	Prob.	n_{BIL},n_{FIL}	Data	Time	Mem.	Method
96/144	1+16+3	6	2^{-86}	-	2^{87}	2^{136}	2^{22}	Alg. A
	2+16+2	16	2^{-87}	7,0	$2^{89.58}$	$2^{136.58}$	2^{97}	Alg. B
	2+16+2	17	2^{-92}	15,0	$2^{94.58}$	$2^{134.58}$	2^{97}	Alg. B
	2+16+2	27	$2^{-85.19}$	7,0	$2^{87.77}$	$2^{137.35}$	2^{97}	Alg. C
	1+17+3	7	$2^{-92.17}$	-	$2^{93.17}$	$2^{143.75}$	2^{22}	Alg. A
	2+17+2	28	$2^{-91.03}$	7,2	$2^{93.61}$	$2^{143.13}$	2^{99}	Alg. C
128/128	1+19+2	8	2^{-117}	-	2^{118}	$2^{120.81}$	2^{22}	Alg. A
	1+20+2	9	$2^{-121.37}$	-	$2^{122.37}$	$2^{124.95}$	2^{22}	Alg. A
128/192	1+19+3	8	2^{-117}	-	2^{118}	$2^{184.81}$	2^{22}	Alg. A
	2+19+2	19	2^{-119}	23,2	$2^{121.58}$	$2^{167.58}$	2^{131}	Alg. B
	2+19+2	29	$2^{-117.19}$	23,2	$2^{119.77}$	$2^{168.35}$	2^{131}	Alg. C
	1+20+3	9	$2^{-121.37}$	-	$2^{122.37}$	$2^{188.95}$	2^{22}	Alg. A
	2+20+2	20	$2^{-123.17}$	23,0	$2^{125.75}$	$2^{173.75}$	2^{129}	Alg. B
	2+20+2	30	$2^{-121.37}$	23,0	$2^{123.95}$	$2^{174.53}$	2^{129}	Alg. C
128/256	1+19+4	8	2^{-117}	-	2^{118}	$2^{248.81}$	2^{22}	Alg. A
	2+19+3	19	2^{-119}	23,2	$2^{121.58}$	$2^{231.58}$	2^{131}	Alg. B
	2+19+3	29	$2^{-117.19}$	23,2	$2^{119.77}$	$2^{232.35}$	2^{131}	Alg. C
	1+20+4	9	$2^{-121.37}$	-	$2^{122.37}$	$2^{252.95}$	2^{22}	Alg. A
	2+20+3	20	$2^{-123.17}$	23,0	$2^{125.75}$	$2^{237.75}$	2^{129}	Alg. B
	2+20+3	30	$2^{-121.37}$	23,0	$2^{123.95}$	$2^{238.53}$	2^{129}	Alg. C

Now we highlight some features of the results shown in Table 11. Algorithm A is used to attack all variants while Algorithm B and C are applied to variants with $m = 3, 4$. We use proper differentials for reduced `SPECK` with the largest number of rounds that can be attacked and for `SPECK` reduced to fewer rounds, we use differential trails. When we use Algorithm B and C which are based on the counting method, in order to have a reasonable success probability, we set the number of right pairs to a larger value than the one used in the enumerating method of Dinur's attack and Algorithm A. This increases the data complexity and the time complexity by a factor. For distinguishers with small signal-to-noise ratio, *e.g.*, the ones for `SPECK32` and `SPECK48`, there is no advantage if we use Algorithm B. As a result, Algorithm A and C, which uses multiple trails or differentials, give better results in most cases.

Note that the results of Algorithm A achieve the goal of reducing the data complexity when compared with previous attacks. The aims of using Algorithm B are to make use of Type-II structure and reduce the time complexity. Algorithm C combines the ideas of Algorithm A and B and helps to improve the data and time complexity at the same time. However, the memory complexity of the attacks using Algorithm B and C might be higher than the previous attacks which use Dinur's algorithm.

6 Conclusion

In this paper, we study the properties of modular addition and find a method to construct Type-II structures for `SPECK`. We also show that a combination of both

types of structures is possible for ARX ciphers. To demonstrate the effect of these structures, we apply them to SPECK and obtain a series of improved attacks on all variants of SPECK. Our results confirm that Type-II structures help to reduce the time complexity and the combination of both types of structures helps to improve the data and time complexity at the same time, as in the cryptanalysis of S-box-based ciphers. Besides the standard differential cryptanalysis, we believe Type-II structures can be potentially applied to other differential-like cryptanalysis for ARX ciphers, such boomerang attacks, differential-linear attacks, impossible attacks, etc.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported by the National Natural Science Foundation of China (Grants 62022036, 62132008, 62202460).

References

ALLW14. Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential cryptanalysis of round-reduced SIMON and SPECK. In *International Workshop on Fast Software Encryption*, pages 525–545. Springer, 2014.

BdST+22. Alex Biryukov, Luan Cardoso dos Santos, Je Sen Teh, Aleksei Udovenko, and Vesselin Velichkov. Meet-in-the-filter and dynamic counting with applications to SPECK. *Cryptology ePrint Archive*, 2022.

BGL+23. Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Enhancing differential-neural cryptanalysis. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I*, pages 318–347. Springer, 2023.

BK09. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *International conference on the theory and application of cryptology and information security*, pages 1–18. Springer, 2009.

BRV14. Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential analysis of block ciphers SIMON and SPECK. In *International Workshop on Fast Software Encryption*, pages 546–570. Springer, 2014.

BS90. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537, pages 2–21. Springer, 1990.

BS91. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

BS92. Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In *Annual international cryptology conference*, pages 487–496. Springer, 1992.

BSS+13. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *cryptology eprint archive*, 2013.

CWP12. Jiazhe Chen, Meiqin Wang, and Bart Preneel. Impossible differential cryptanalysis of the lightweight block ciphers TEA, XTEA and HIGHT. In *Progress in Cryptology-AFRICACRYPT 2012: 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings 5*, pages 117–137. Springer, 2012.

Din14. Itai Dinur. Improved differential cryptanalysis of round-reduced SPECK. In *International Conference on Selected Areas in Cryptography*, pages 147–164. Springer, 2014.

FWG+16. Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-based automatic search algorithms for differential and linear trails for SPECK. In *International Conference on Fast Software Encryption*, pages 268–288. Springer, 2016.

Goh19. Aron Gohr. Improving attacks on round-reduced SPECK32/64 using deep learning. In *Annual International Cryptology Conference*, pages 150–179. Springer, 2019.

HHK+04. Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Sangjin Lee. Differential cryptanalysis of TEA and XTEA. In *Information Security and Cryptology-ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers 6*, pages 402–417. Springer, 2004.

KLT15. Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. *IACR Cryptol. ePrint Arch.*, page 145, 2015.

Leu16. Gaëtan Leurent. Improved differential-linear cryptanalysis of 7-round Chaskey with partitioning. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 344–371. Springer, 2016.

LKK+18. HoChang Lee, Seojin Kim, HyungChul Kang, Deukjo Hong, Jaechul Sung, and Seokhie Hong. 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.

LLJW20. Zhengbin Liu, Yongqiang Li, Lin Jiao, and Mingsheng Wang. A new method for searching optimal differential and linear trails in ARX ciphers. *IEEE Transactions on Information Theory*, 67(2):1054–1068, 2020.

LM01. Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In *International Workshop on Fast Software Encryption*, pages 336–350. Springer, 2001.

LWR16. Yunwen Liu, Qingju Wang, and Vincent Rijmen. Automatic search of linear trails in ARX with applications to SPECK and Chaskey. In *International Conference on Applied Cryptography and Network Security*, pages 485–499. Springer, 2016.

MHL+02. Dukjae Moon, Kyungdeok Hwang, Wonil Lee, Sangjin Lee, and Jongin Lim. Impossible differential cryptanalysis of reduced round XTEA and TEA. In *Fast Software Encryption: 9th International Workshop, FSE 2002 Leuven, Belgium, February 4–6, 2002 Revised Papers 9*, pages 49–60. Springer, 2002.

MP13. Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Paper 2013/328, 2013. `https://eprint.iacr.org/2013/328`.

MY92. Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of FEAL cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 81–91. Springer, 1992.

QUE19. SEPARATE DECISION QUEUE. CaDiCaL at the SAT race 2019. *SAT RACE 2019*, page 8, 2019.

Sel08. Ali Aydın Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008.

SHY16. Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In *Australasian Conference on Information Security and Privacy*, pages 379–394. Springer, 2016.

Sin05. Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

SNC09. Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 244–257. Springer, 2009.

SWW21. Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Transactions on Symmetric Cryptology*, pages 269–315, 2021.

WFH+22. Senpeng Wang, Dengguo Feng, Bin Hu, Jie Guan, Tairong Shi, and Kai Zhang. The simplest SAT model of combining matsui's bounding conditions with sequential encoding method. *Cryptology ePrint Archive*, 2022.

WW22. Feifan Wang and Gaoli Wang. Improved differential-linear attack with application to round-reduced SPECK32/64. In *International Conference on Applied Cryptography and Network Security*, pages 792–808. Springer, 2022.

A More Details of the Key Recovery Phase

A.1 The 2-Round Attack of Dinur's Algorithm

As described in Section 2.3, the 2-round attack is to enumerate all the possible round keys k_i and k_{i+1} under which after the 2-round decryption the difference of the pairs is equal to $(\Delta x_i, \Delta y_i)$, if we know an input difference $(\Delta x_i, \Delta y_i)$, an output difference $(\Delta x_{i+2}, \Delta y_{i+2})$ and the output pairs of the 2-round transformation. Since $k_{i+1} = (y_{i+1} \boxplus (x_{i+1} \ggg a)) \oplus x_{i+2}$ and $k_i = (y_i \boxplus (x_i \ggg a)) \oplus x_{i+1}$, the key point is to derive the values of x_i and x_{i+1}. To obtain x_i and x_{i+1}, one needs to solve the following two differential equations of addition (DEA) in sequel.

$$((x_{i+1} \oplus \Delta x_{i+1}) \ggg a) \boxplus (y_{i+1} \oplus \Delta y_{i+1}) = ((x_{i+1} \ggg a) \boxplus y_{i+1}) \oplus \Delta x_{i+2}, \quad (3)$$

$$((x_i \oplus \Delta x_i) \ggg a) \boxplus (y_i \oplus \Delta y_i) = ((x_i \ggg a) \boxplus y_i) \oplus \Delta x_{i+1}. \quad (4)$$

This type of DEA has an average of two solutions. However, for almost any value of $(\Delta x_i, \Delta y_i)$, a large part of ciphertext pairs lead to no solutions. Therefore, to save time, a filtering process is needed before solving these two equations. As proved in [LM01], a valid differential $(\alpha, \beta \to \gamma)$ for addition should satisfy the following equation.

$$\mathrm{eq}(\alpha \ll 1, \beta \ll 1, \gamma \ll 1)(\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0, \quad (5)$$

$$\text{where} \quad \mathrm{eq}(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z). \quad (6)$$

This means checking whether the equation holds for the two addition given all differences of the 2-round scheme. Using the filter, the complexity of the 2-round attack can be optimized to less than 2 encryptions of SPECK, which was verified by a lot of experiments on SPECK in [Din14] and we also confirmed this through experiments. The memory complexity of this attack is calculated in terms of bytes which is 2^{22}.

A.2 Success Probability

For differential attacks on SPECK, we use the formula in Selçuk's work to evaluate the success probability [Sel08], which is

$$P_s = \Phi\left(\frac{\sqrt{\mu S_N} - \Phi^{-1}(1 - 2^{-(n_k-\ell)})}{\sqrt{S_N + 1}}\right), \tag{7}$$

where μ is the number of right pairs, S_N is the signal-to-noise ratio, n_k is the number of key bits involved in the key recovery phase, and 2^ℓ is the size of the short list of the key candidates.

B Search for Differential Trails and Differentials of SPECK

In this section, we present the SAT model for differential cryptanalysis of ARX ciphers [SWW21] and give detailed information on differential trails.

In literature, Mouha and Preneel are the first to apply the SAT method to the search of differential trails [MP13], where they only take ARX ciphers into account. Before long, lots of works in search of characteristics based on SAT spring up, including searching of differential and linear characteristics for the SIMON-like round function [KLT15] and linear trails for ARX ciphers [LWR16]. Recently, Wang et.al [WFH+22] proposed a new method based on SAT to accelerate the trail search .

In this work, we also use the SAT method to search for trails of SPECK. In reality, there are several operations modeled in SAT. For convenience, we only give a brief description of how to model the nonlinear operation of SPECK, *i.e.*, Modular Addition, and how to model the constraint of weight.

A detailed description of other operations can be found in [SWW21]. Finally, we apply the above methods to our search of differential trails and differentials.

Modular addition [SWW21]. For the n-bit modular addition, let α and β be the input variables, and γ be the output variable. Then, the differential (α, β, γ)

is valid if and only if the following equations hold:

$$\left.\begin{array}{l}
\alpha_i \vee \beta_i \vee \overline{\gamma_i} \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} = 1 \\
\alpha_i \vee \overline{\beta_i} \vee \gamma_i \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} = 1 \\
\overline{\alpha_i} \vee \beta_i \vee \gamma_i \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} = 1 \\
\overline{\alpha_i} \vee \overline{\beta_i} \vee \overline{\gamma_i} \vee \alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} = 1 \\
\alpha_i \vee \beta_i \vee \gamma_i \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}} = 1 \\
\alpha_i \vee \overline{\beta_i} \vee \overline{\gamma_i} \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}} = 1 \\
\overline{\alpha_i} \vee \beta_i \vee \overline{\gamma_i} \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}} = 1 \\
\overline{\alpha_i} \vee \overline{\beta_i} \vee \gamma_i \vee \overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}} = 1
\end{array}\right\} 0 \le i \le n-2$$
$$\alpha_{n-1} \oplus \beta_{n-1} \oplus \gamma_{n-1} = 0$$

Since the modular addition is a non-linear operation, we also need to model the probability of the operation. Generally, the probability will be represented by the sum of weight ω_i. The differential probability $\sum_{i=0}^{n-2} \omega_i$ is valid if and only if the following equations hold:

$$\left.\begin{array}{l}
\overline{\alpha_{i+1}} \vee \gamma_{i+1} \vee \omega_i = 1 \\
\beta_{i+1} \vee \overline{\gamma_{i+1}} \vee \omega_i = 1 \\
\alpha_{i+1} \vee \overline{\beta_{i+1}} \vee \omega_i = 1 \\
\alpha_{i+1} \vee \beta_{i+1} \vee \gamma_{i+1} \vee \overline{\omega_i} = 1 \\
\overline{\alpha_{i+1}} \vee \overline{\beta_{i+1}} \vee \overline{\gamma_{i+1}} \vee \overline{\omega_i} = 1
\end{array}\right\} 0 \le i \le n-2$$

Sequential encoding method. In the search, we want to confine the weight of the trail, *i.e.*, the probability of the trail. This can be abstracted as the Boolean cardinality constraint $\sum_{j=0}^{n-1} x_j \le k$, where x_j is the Boolean variable, and k is a non-negative integer. However, this constraint can not easily be modeled into CNF formulas. Fortunately, we can take the sequential encoding method [Sin05] to eliminate this difficulty as [SWW21], where the constraint $\sum_{j=0}^{n-1} x_j \le k$ is translated into the following CNF formulas:

$$\begin{array}{l}
\overline{x_0} \vee x_{0,0} = 1 \\
\overline{x_{0,j}} = 1, 1 \le j \le k-1 \\
\overline{x_i} \vee x_{i,0} = 1 \\
\overline{x_{i-1,0}} \vee x_{i,0} = 1 \\
\left.\begin{array}{l}
\overline{x_i} \vee \overline{x_{i-1,j-1}} \vee x_{i,j} = 1 \\
\overline{x_{i-1,j}} \vee x_{i,j} = 1
\end{array}\right\} 1 \le j \le k-1 \\
\overline{x_i} \vee \overline{x_{i-1,k-1}} = 1 \\
\overline{x_{n-1}} \vee \overline{x_{n-2,k-1}} = 1
\end{array}\quad \left.\begin{array}{l}\\\\\\\\\\\\\end{array}\right\} 1 \le i \le n-2 \tag{8}$$

In general, all bounding conditions can be replaced with an inequality constraint of the following form

$$\sum_{i=e_1}^{e_2} x_i \leq m, \tag{9}$$

where $e_1 \geq 0, e_2 \leq n-1$, and $m \leq k$. Based on the above, we can apply Equation (8) to Equation (9) to model the bounding conditions.

In [SWW21], Sun et al. build the SAT model of bounding conditions without claiming any new variable. They split the encoding problem into three different cases as follows:

Case 1. $\sum_{j=e_1}^{e_2} x_j \leq m$ with $e_1 = 0$ and $e_2 < n-1$, this bounding condition can be converted into the following CNF formulas:

$$\overline{x_i} \vee \overline{x_{i-1,m-1}} = 1, 1 \leq i \leq e_2.$$

Case 2. $\sum_{j=e_1}^{e_2} x_j \leq m$ with $e_1 > 0$ and $e_2 < n-1$, this bounding condition can be converted into the following CNF formulas:

$$x_{e_1-1,j} \vee \overline{x_{e_2,j+m}} = 1, 0 \leq j \leq k-m-1.$$

Case 3. $\sum_{j=e_1}^{e_2} x_j \leq m$ with $e_1 > 0$ and $e_2 = n-1$, this bounding condition can be converted into the following CNF formulas:

$$\begin{aligned} &x_{e_1-1,j} \vee \overline{x_{n-2,j+m}} = 1, 0 \leq j \leq k-m-1, \\ &x_{e_1-1,j} \vee \overline{x_{n-1}} \vee \overline{x_{n-2,j+m-1}} = 1, 0 \leq j \leq k-m. \end{aligned}$$

Search for trails and differentials. For SPECK32/48/64, We use CaDiCaL [QUE19] to search for the differential trails. As for the search of differentials, we fix the input difference of the first round and the output difference of the last round to search for all trails with probability larger than a threshold and sum up all probabilities as the differential probability. In order to obtain all solutions to the problem, we switch to another tool CryptoMiniSat [SNC09], while for the other versions, other than the search strategy, the others are the same. We exploit the heuristic methods as [SHY16] to improve the search efficiency. More specifically, we split one long trail into two short trails, i.e, combining two trails to form a long one. By these heuristic strategies can we obtain better trails and differentials. For those trails and differentials that are applied to our attacks which are shown in Table 1, we list them as below.

Table 12: Characteristics of ID:2

SPECK48									
Number	1			2			3		
Round	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight
0	001202	020002	0	080048	080800	0	0800c8	080800	0
1	000010	100000	3	400000	004000	3	400000	004000	3
2	000000	800000	1	000000	020000	1	000000	020000	1
3	800000	800004	0	020000	120000	1	020000	120000	1
4	808004	808020	2	120200	820200	3	120200	820200	3
5	8400a0	8001a4	4	821006	920002	5	821002	920006	4
6	608da4	608080	9	918216	018202	9	918236	018202	9
7	042003	002400	11	0c1080	000090	11	0c1080	000090	12
8	012020	000020	5	800480	800000	4	800480	800000	4
9	200100	200000	3	008004	008000	2	008004	008000	2
10	202001	202000	3	048080	008080	3	048080	008080	3
11	210020	200021	4	808400	848000	3	808400	848000	3
$\sum_r$ weight			45			45			45

Table 13: Characteristics of ID:6

SPECK96						
Number	1			2		
Round	ΔL	ΔR	weight	ΔL	ΔR	weight
0	010420040000	000024000400	0	240004000009	010420040000	0
1	000120200000	000000202000	5	082020000000	000120200000	6
2	000001000000	000000010000	3	000900000000	000001000000	4
3	000000000000	000000080000	1	000008000000	000000000000	2
4	000000380000	000000780000	3	000000080000	000000080000	1
5	000000080800	000003c80800	7	000000080800	000000480800	2
6	000000480008	00001e084008	7	000000480008	000002084008	4
7	080006080808	0800f64a0848	9	0800fe080808	0800ee4a0848	12
8	0007b2400040	400000104200	18	000772400040	400000104200	21
9	000000820200	000000001202	10	000000820200	000000001202	11
10	000000009000	000000000010	4	000000009000	000000000010	4
11	000000000080	000000000000	2	000000000080	000000000000	2
12	800000000000	800000000000	0	800000000000	800000000000	0
13	808000000000	808000000004	1	808000000000	808000000004	1
14	800080000004	840080000020	3	800080000004	840080000020	3
15	808080800020	a08480800124	5	808080800020	a08480800124	5
16	800400008124	842004008801	9	800400008124	842004008801	9
$\sum_r$ weight			87			87

Table 14: Characteristics of ID:8

SPECK128				
Number	Round	ΔL	ΔR	weight
1	0	0124000400000010	0801042004000000	0
	1	0800202000000000	4808012020000000	7
	2	4800010000000000	0840080100000002	6
	3	0808080000000006	4a08480800000016	7
	4	4000400000000032	1042004000000080	12
	5	0202000000000080	8012020000000480	7
	6	0010000000000480	0080100000002084	5
	7	8080000000006080	84808000000164a0	6
	8	0400000000032400	2004000000080104	11
	9	2000000000080020	2020000000480801	7
	10	0000000000480001	0100000002084008	6
	11	000000000e080808	080000001e4a0848	8
	12	00000000f2400040	4000000000104200	15
	13	0000000000820200	0000000000001202	8
	14	0000000000009000	0000000000000010	4
	15	0000000000000080	0000000000000000	2
	16	8000000000000000	8000000000000000	0
	17	8080000000000000	8080000000000004	1
	18	8000800000000004	8400800000000020	3
	19	8080808000000020	a084808000000124	5
	$\sum$ weight			120

Note: the remaing 6 trails are the same as Characteristics of ID: 35

Table 15: Characteristics of ID:22

SPECK32												
Number	1			2			3			4		
Round	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight
0	7458	b0f8	0	7c58	b0f8	0	1488	1008	0	7448	b0f8	0
1	01e0	c202	5	01e0	c202	5	0021	4001	4	01e0	c202	5
2	020f	0a04	5	020f	0a04	5	0601	0604	4	020f	0a04	5
3	2800	0010	5	2800	0010	5	1800	0010	6	2800	0010	5
4	0040	0000	2	0040	0000	2	0040	0000	3	0040	0000	2
5	8000	8000	0	8000	8000	0	8000	8000	0	8000	8000	0
6	8100	8102	1	8100	8102	1	8100	8102	1	8100	8102	1
7	8000	840a	2	8000	840a	2	8000	840a	2	8000	840a	2
8	850a	9520	4	850a	9520	4	850a	9520	4	850a	9520	4
9	802a	d4a8	6	802a	d4a8	6	802a	d4a8	6	802a	d4a8	6
$\sum_r$ weight			30			30			30			30

(continued)

Table 15: (continued)

SPECK32			
Number	5		
Round	ΔL	ΔR	weight
0	7c48	b0f8	0
1	01e0	c202	5
2	020f	0a04	5
3	2800	0010	5
4	0040	0000	2
5	8000	8000	0
6	8100	8102	1
7	8000	840a	2
8	850a	9520	4
9	802a	d4a8	6
$\sum_r$ weight			30

Table 16: Characteristics of ID:25

SPECK64									
Number	1			2			3		
Round	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight
0	92400040	40104200	0	924000c0	40104200	0	96400040	40104200	0
1	00820200	00001202	6	00820200	00001202	6	00820200	00001202	7
2	00009000	00000010	4	00009000	00000010	4	00009000	00000010	4
3	00000080	00000000	2	00000080	00000000	2	00000080	00000000	2
4	80000000	80000000	0	80000000	80000000	0	80000000	80000000	0
5	80800000	80800004	1	80800000	80800004	1	80800000	80800004	1
6	80008004	84008020	3	80008004	84008020	3	80008004	84008020	3
7	80808060	a0848164	6	80808060	a0848164	6	80808060	a0848164	6
8	00040f24	04200401	13	00040f24	04200401	13	00040f24	04200401	13
9	20200008	01202000	8	20200008	01202000	8	20200008	01202000	8
10	09000000	00010000	4	09000000	00010000	4	09000000	00010000	4
11	00080000	00000000	2	00080000	00000000	2	00080000	00000000	2
12	00000800	00000800	1	00000800	00000800	1	00000800	00000800	1
13	00000808	00004808	2	00000808	00004808	2	00000808	00004808	2
14	08004800	08020840	4	08004800	08020840	4	08004800	08020840	4
15	080a0808	481a4a08	6	080a0808	481a4a08	6	080a0808	481a4a08	6
$\sum_r$ weight			62			62			63

Table 16: Characteristics of ID:25

SPECK64									
Number	4			5			6		
Round	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight
0	b2400040	40104200	0	b24000c0	40104200	0	92440040	40104200	0
1	00820200	00001202	7	00820200	00001202	7	00820200	00001202	7
2	00009000	00000010	4	00009000	00000010	4	00009000	00000010	4
3	00000080	00000000	2	00000080	00000000	2	00000080	00000000	2
4	80000000	80000000	0	80000000	80000000	0	80000000	80000000	0
5	80800000	80800004	1	80800000	80800004	1	80800000	80800004	1
6	80008004	84008020	3	80008004	84008020	3	80008004	84008020	3
7	80808060	a0848164	6	80808060	a0848164	6	80808060	a0848164	6
8	00040f24	04200401	13	00040f24	04200401	13	00040f24	04200401	13
9	20200008	01202000	8	20200008	01202000	8	20200008	01202000	8
10	09000000	00010000	4	09000000	00010000	4	09000000	00010000	4
11	00080000	00000000	2	00080000	00000000	2	00080000	00000000	2
12	00000800	00000800	1	00000800	00000800	1	00000800	00000800	1
13	00000808	00004808	2	00000808	00004808	2	00000808	00004808	2
14	08004800	08020840	4	08004800	08020840	4	08004800	08020840	4
15	080a0808	481a4a08	6	080a0808	481a4a08	6	080a0808	481a4a08	6
$\sum_r$ weight			63			63			63

Table 16: Characteristics of ID:25

SPECK64									
Number / Round	7			8			9		
	ΔL	ΔR	weight	ΔL	ΔR	weight	ΔL	ΔR	weight
0	924400c0	40104200	0	92c000c0	40104200	0	964000c0	40104200	0
1	00820200	00001202	7	00820200	00001202	7	00820200	00001202	7
2	00009000	00000010	4	00009000	00000010	4	00009000	00000010	4
3	00000080	00000000	2	00000080	00000000	2	00000080	00000000	2
4	80000000	80000000	0	80000000	80000000	0	80000000	80000000	0
5	80800000	80800004	1	80800000	80800004	1	80800000	80800004	1
6	80008004	84008020	3	80008004	84008020	3	80008004	84008020	3
7	80808060	a0848164	6	80808060	a0848164	6	80808060	a0848164	6
8	00040f24	04200401	13	00040f24	04200401	13	00040f24	04200401	13
9	20200008	01202000	8	20200008	01202000	8	20200008	01202000	8
10	09000000	00010000	4	09000000	00010000	4	09000000	00010000	4
11	00080000	00000000	2	00080000	00000000	2	00080000	00000000	2
12	00000800	00000800	1	00000800	00000800	1	00000800	00000800	1
13	00000808	00004808	2	00000808	00004808	2	00000808	00004808	2
14	08004800	08020840	4	08004800	08020840	4	08004800	08020840	4
15	080a0808	481a4a08	6	080a0808	481a4a08	6	080a0808	481a4a08	6
$\sum_r$ weight			63			63			63

Table 16: Characteristics of ID:25

SPECK64						
Number / Round	10			11		
	ΔL	ΔR	weight	ΔL	ΔR	weight
0	92c00040	40104200	0	09240004	04010420	0
1	00820200	00001202	7	00082020	20000120	6
2	00009000	00000010	4	00000900	00000001	4
3	00000080	00000000	2	00000008	00000000	2
4	80000000	80000000	0	08000000	08000000	1
5	80800000	80800004	1	08080000	48080000	2
6	80008004	84008020	3	48000800	08400802	4
7	80808060	a0848164	6	08080806	4a084816	7
8	00040f24	04200401	13	400040f2	10420040	14
9	20200008	01202000	8	82020000	00120200	7
10	09000000	00010000	4	00900000	00001000	4
11	00080000	00000000	2	00008000	00000000	2
12	00000800	00000800	1	00000080	00000080	1
13	00000808	00004808	2	80000080	80000480	1
14	08004800	08020840	4	00800480	00802084	3
15	080a0808	481a4a08	6	8080a080	8481a4a0	5
$\sum_r$ weight			63			63

Table 17: Characteristics of ID:29

SPECK128				
Number / Round	1		2	
	$\Delta L/\Delta R$	weight	$\Delta L/\Delta R$	weight
0	0124000400000000/ 0801042004000000	0	0324000400000000/ 0801042004000000	0
1	0800202000000000/ 4808012020000000	6	0800202000000000/ 4808012020000000	7
2	4800010000000000/ 0840080100000002	6	4800010000000000/ 0840080100000002	6
3	0808080000000006/ 4a08480800000016	6	0808080000000006/ 4a08480800000016	7
4	4000400000000032/ 1042004000000080	10	4000400000000032/ 1042004000000080	12
5	0202000000000080/ 8012020000000480	9	0202000000000080/ 8012020000000480	7
6	0010000000000480/ 0080100000002084	6	0010000000000480/ 0080100000002084	5
17	8080000000002080/ 84808000000124a0	5	8080000000002080/ 84808000000124a0	5
8	0400000000012440/ 2004000000080144	9	0400000000012440/ 2004000000080144	9
9	2000000000080220/ 2020000000480801	9	2000000000080220/ 2020000000480801	9
10	0000000000480001/ 0100000002084008	7	0000000000480001/ 0100000002084008	7
11	000000000e080808/ 080000001e4a0848	8	000000000e080808/ 080000001e4a0848	8
12	00000000f2400040/ 4000000000104200	15	00000000f2400040/ 4000000000104200	15
13	0000000000820200/ 0000000000001202	8	0000000000820200/ 0000000000001202	8
14	0000000000009000/ 0000000000000010	4	0000000000009000/ 0000000000000010	4
15	0000000000000080/ 0000000000000000	2	0000000000000080/ 0000000000000000	2
16	8000000000000000/ 8000000000000000	0	8000000000000000/ 8000000000000000	0
17	8080000000000000/ 8080000000000004	1	8080000000000000/ 8080000000000004	1
18	8000800000000004/ 8400800000000020	3	8000800000000004/ 8400800000000020	3
19	8080808000000020/ a084808000000124	5	8080808000000020/ a084808000000124	5
$\sum_r$ weight		119		120

Table 17: Characteristics of ID:29

SPECK128				
Number / Round	3		4	
	$\Delta L/\Delta R$	weight	$\Delta L/\Delta R$	weight
0	0124000c00000000/ 0801042004000000	0	0124400400000000/ 0801042004000000	0
1	0800202000000000/ 4808012020000000	7	0800202000000000/ 4808012020000000	7
2	4800010000000000/ 0840080100000002	6	4800010000000000/ 0840080100000002	6
3	0808080000000006/ 4a08480800000016	7	0808080000000006/ 4a08480800000016	7
4	4000400000000032/ 1042004000000080	12	4000400000000032/ 1042004000000080	12
5	0202000000000080/ 8012020000000480	7	0202000000000080/ 8012020000000480	7
6	0010000000000480/ 0080100000002084	5	0010000000000480/ 0080100000002084	5
17	8080000000006080/ 84808000000164a0	6	8080000000002080/ 84808000000124a0	5
8	0400000000032400/ 2004000000080104	11	0400000000012440/ 2004000000080144	9
9	2000000000080020/ 2020000000480801	7	2000000000080220/ 2020000000480801	9
10	0000000000480001/ 0100000002084008	6	0000000000480001/ 0100000002084008	7
11	000000000e080808/ 080000001e4a0848	8	000000000e080808/ 080000001e4a0848	8
12	00000000f2400040/ 4000000000104200	15	00000000f2400040/ 4000000000104200	15
13	0000000000820200/ 0000000000001202	8	0000000000820200/ 0000000000001202	8
14	0000000000009000/ 0000000000000010	4	0000000000009000/ 0000000000000010	4
15	0000000000000080/ 0000000000000000	2	0000000000000080/ 0000000000000000	2
16	8000000000000000/ 8000000000000000	0	8000000000000000/ 8000000000000000	0
17	8080000000000000/ 8080000000000004	1	8080000000000000/ 8080000000000004	1
18	8000800000000004/ 8400800000000020	3	8000800000000004/ 8400800000000020	3
19	8080808000000020/ a084808000000124	5	8080808000000020/ a084808000000124	5
$\sum_r$ weight		120		120

Table 17: Characteristics of ID:29

SPECK128				
Number / Round	5		6	
	$\Delta L/\Delta R$	weight	$\Delta L/\Delta R$	weight
0	012c000400000000/ 0801042004000000	0	0164000400000000/ 0801042004000000	0
1	0800202000000000/ 4808012020000000	7	0800202000000000/ 4808012020000000	7
2	4800010000000000/ 0840080100000002	6	4800010000000000/ 0840080100000002	6
3	0808080000000006/ 4a08480800000016	7	0808080000000006/ 4a08480800000016	7
4	4000400000000032/ 1042004000000080	12	4000400000000032/ 1042004000000080	12
5	0202000000000080/ 8012020000000480	7	0202000000000080/ 8012020000000480	7
6	0010000000000480/ 0080100000002084	5	0010000000000480/ 0080100000002084	5
7	8080000000006080/ 84808000000164a0	6	8080000000006080/ 84808000000164a0	6
8	0400000000032400/ 2004000000080104	11	0400000000032400/ 2004000000080104	11
9	2000000000080020/ 2020000000480801	7	2000000000080020/ 2020000000480801	7
10	0000000000480001/ 0100000002084008	6	0000000000480001/ 0100000002084008	6
11	000000000e080808/ 080000001e4a0848	8	000000000e080808/ 080000001e4a0848	8
12	00000000f2400040/ 4000000000104200	15	00000000f2400040/ 4000000000104200	15
13	0000000000820200/ 0000000000001202	8	0000000000820200/ 0000000000001202	8
14	0000000000009000/ 0000000000000010	4	0000000000009000/ 0000000000000010	4
15	0000000000000080/ 0000000000000000	2	0000000000000080/ 0000000000000000	2
16	8000000000000000/ 8000000000000000	0	8000000000000000/ 8000000000000000	0
17	8080000000000000/ 8080000000000004	1	8080000000000000/ 8080000000000004	1
18	8000800000000004/ 8400800000000020	3	8000800000000004/ 8400800000000020	3
19	8080808000000020/ a084808000000124	5	8080808000000020/ a084808000000124	5
$\sum_r$ weight		120		120

Table 18: The detailed differentials

ID	Variant 2n	Rounds	Input difference	output difference	weight	Number
3	48	12	0800c8,080800	840084,a00080	46.78	1
			080048,080800	840084,a00080	46.78	2
26	64	15	924000c0,40104200	080a0808,481a4a08	61.04	1
			92400040,40104200	080a0808,481a4a08	61.04	2
			96400040,40104200	080a0808,481a4a08	62.05	3
			b2400040,40104200	080a0808,481a4a08	62.05	4
			b24000c0,40104200	080a0808,481a4a08	62.05	5
			92440040,40104200	080a0808,481a4a08	62.00	6
			924400c0,40104200	080a0808,481a4a08	62.00	7
			92c000c0,40104200	080a0808,481a4a08	62.05	8
			964000c0,40104200	080a0808,481a4a08	62.05	9
			92c00040,40104200	080a0808,481a4a08	62.05	10
			09240004,04010420	8080a080,8481a4a0	62.05	11
7	96	17	010c20040000, 000024000400	a0a000008880, 81a02004c88c	94.99	1
			0104200c0000, 000024000400	a0a000008880, 81a02004c88c	94.99	2
			010460040000, 000024000400	a0a000008880, 81a02004c88c	94.94	3
			014420040000, 000024000400	a0a000008880, 81a02004c88c	94.99	4
			030420040000, 000024000400	a0a000008880, 81a02004c88c	94.99	5
			010420040000, 000024000400	a0a000008880, 81a02004c88c	93.98	6
28	96	17	00c420040000, 000024000400	a0a000008880, 81a02004c88c	96.00	1
			01c420040000, 000024000400	a0a000008880, 81a02004c88c	96.00	2
			010c200c0000, 000024000400	a0a000008880, 81a02004c88c	96.00	3
			010c20040000, 000024000400	a0a000008880, 81a02004c88c	94.99	4
			010C60040000, 000024000400	a0a000008880, 81a02004c88c	95.95	5
			0104e0040000, 000024000400	a0a000008880, 81a02004c88c	95.95	6
			0104200c0000, 000024000400	a0a000008880, 81a02004c88c	94.99	7
			010420040000, 000024000400	a0a000008880, 81a02004c88c	93.98	8
			0104201c0000, 000024000400	a0a000008880, 81a02004c88c	96.00	9
			0104600c0000, 000024000400	a0a000008880, 81a02004c88c	95.95	10

Continued on next page

Table 18 – Continued from previous page

ID	Variant 2n	Rounds	Input difference	output difference	weight	Number
			010460040000, 000024000400	a0a000008880, 81a02004c88c	94.94	11
			011c20040000, 000024000400	a0a000008880, 81a02004c88c	96.00	12
			014c20040000, 000024000400	a0a000008880, 81a02004c88c	96.00	13
			0144200c0000, 000024000400	a0a000008880, 81a02004c88c	96.00	14
			014420040000, 000024000400	a0a000008880, 81a02004c88c	94.99	15
			014460040000, 000024000400	a0a000008880, 81a02004c88c	95.95	16
			017c20040000, 000024000400	a0a000008880, 81a02004c88c	96.00	17
			030c20040000, 000024000400	a0a000008880, 81a02004c88c	96.00	18
			0304200c0000, 000024000400	a0a000008880, 81a02004c88c	96.00	19
			030420040000, 000024000400	a0a000008880, 81a02004c88c	94.99	20
			030460040000, 000024000400	a0a000008880, 81a02004c88c	95.95	21
			034420040000, 000024000400	a0a000008880, 81a02004c88c	96.00	22
			070420040000, 000024000400	a0a000008880, 81a02004c88c	96.00	23
9	128	20	0124000c00000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	1
			0124000400000000, 0801042004000000	8004000080000124, 8420040080000801	123.17	2
			0124400400000000, 0801042004000000	8004000080000124, 8420040080000801	124.15	3
			0164000400000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	4
			0324000400000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	5
			012c000400000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	6
30	128	20	0124400400000000, 0801042004000000	8004000080000124, 8420040080000801	124.15	1
			0164000400000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	2
			0324000400000000, 0801042004000000	8004000080000124, 8420040080000801	124.19	3
			012c000400000000, 0801042004000000	8004000080000124 8420040080000801	124.19	4
			0124000c00000000,	8004000080000124,	124.19	5

Continued on next page

Table 18 – Continued from previous page

ID	Variant 2n	Rounds	Input difference	output difference	weight	Number
			0801042004000000	8420040080000801		
			0124000400000000,	8004000080000124,	123.17	6
			0801042004000000	8420040080000801		