Cryptanalysis of SPEEDY

Jinliang Wang[1,2], Chao Niu[1,2], Qun Liu[1,2], Muzhou Li[1,2(✉)], Bart Preneel[4], and Meiqin Wang[1,2,3]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
{jinliangwang,niuchao,qunliu,muzhouli}@mail.sdu.edu.cn
[2] School of Cyber Science and Technology, Shandong University, Qingdao, China
mqwang@sdu.edu.cn
[3] Quan Cheng Shandong Laboratory, Jinan, China
[4] imec-COSIC, KU Leuven, Leuven, Belgium
Bart.Preneel@esat.kuleuven.be

Abstract. SPEEDY is a family of ultra-lightweight block ciphers designed by Leander et al. at CHES 2021. There are three recommended variants denoted as SPEEDY-r-192 with $r \in \{5, 6, 7\}$. All of them support the 192-bit block and the 192-bit key. The main focus during its design is to ensure hardware-aware low latency, thus, whether it is designed to have enough security is worth to be studied. Recently, the full-round security of SPEEDY-7-192 is announced to be broken by Boura et al. at EUROCRYPT 2023 under the chosen-ciphertext setting, where a round-reduced attack on SPEEDY-6-192 is also proposed. However, no valid attack on SPEEDY-5-192 is given due to its more restricted security parameters. Up to now, the best key recovery attack on this variant only covers 3 rounds proposed by Rohit et al. at AFRICACRYPT 2022.

In this paper, we give three full-round attacks on SPEEDY-7-192. Using the divide-and-conquer strategy and other new proposed techniques, we found a 5.5-round differential distinguisher which can be used to mount the first chosen-plaintext full-round key recovery attack. With a similar strategy, we also found a 5-round linear distinguisher which leads to the first full-round attack under the known-plaintext setting. Meanwhile, the 5.5-round differential distinguisher also helps us slightly improve the full-round attack in the chosen-ciphertext setting compared with the previous result. Besides, we also present a 4-round differential attack on SPEEDY-5-192, which is the best attack on this variant in terms of the number of rounds so far. A faster key recovery attack covering the same rounds is also given using a differential-linear distinguisher. Both attacks cannot threaten the full round security of SPEEDY-5-192.

Keywords: Lightweight Cryptography · Low Latency · SPEEDY

1 Introduction

In lightweight cryptography, cryptographic primitives are required to be suitable for resource-constrained environments, such as low area, latency or energy

L. Simpson and M. A. Rezazadeh Baee (Eds.): ACISP 2023, LNCS 13915, pp. 124–156, 2023.
https://doi.org/10.1007/978-3-031-35486-1_7

consumption. Therefore, it's usually difficult to design such ciphers since one has to make trade-offs between implementation efficiency and security. Hence, the security of such ciphers is worth to be evaluated thoroughly.

In CHES 2021, Leander et al. proposed a family of ultra-low-latency block ciphers named as SPEEDY [9] to resolve the problem: How to design a secure encryption algorithm whose hardware implementation is fast. To gain such cipher, they first considered which type of logic gates and circuit topologies are suitable for ultra low-latency encryption. As a result, SPEEDY adopts a lightweight S-box, whose coordinate functions are realized as two-level NAND trees, and a hardware-friendly linear layer. After careful combination, SPEEDY achieves lower latency in hardware than most cryptographic primitives. SPEEDY contains three versions SPEEDY-5-192, SPEEDY-6-192 and SPEEDY-7-192, whose number of rounds are 5, 6, and 7, respectively. Note that, as the number of rounds decreases, the SPEEDY cipher gains better performance in hardware implementations but has a weak bound of security claim.

Recently, Boura et al. [5] announced that the full-round security of SPEEDY-7-192 is broken under the chosen-ciphertext setting, where a round-reduced attack on SPEEDY-6-192 is also proposed. However, due to the more restricted security parameters, no valid attack on SPEEDY-5-192 is given. The best key recovery attack on this variant only covers 3 rounds proposed by Rohit et al. [16].

In this paper, we aim to evaluate the security of SPEEDY thoroughly using differential cryptanalysis, linear cryptanalysis and differential-linear cryptanalysis. Proposed by Biham and Shamir in 1990, differential cryptanalysis [4] has been an important method in evaluating the security of symmetric-key ciphers. Using a differential distinguisher with high probability, one can recover the secret key after adding several rounds before or/and after it. Such an attack can be mounted only when the adversary is under the chosen-plaintext/ciphertext setting. Linear cryptanalysis [13], proposed by Matsui in 1992, only requires the adversary under the known-plaintext setting, which is more realistic than the differential attack. It exploits the linear distinguisher with high correlation considering the linear relation between some bits of plaintexts, ciphertexts and the secret key. Differential-Linear cryptanalysis [8] is another chosen plaintext/ciphertext method, which is proposed by Langford and Hellman in 1994. This method uses a distinguisher with a high correlation that consists of a differential distinguisher followed by a linear distinguisher. To evaluate its correlation more accurately, Achiya and Dunkelman introduced DLCT (short for Differential Linear Connectivity Table) [1] to connect the differential distinguisher and the linear one. This method works efficiently on ciphers whose differential probabilities and linear correlations are very high when the distinguisher covers small rounds, but decrease very quickly when the number of rounds increases.

1.1 Contributions

In this paper, we propose several key recovery attacks on full-round SPEEDY-7-192 and 4-round SPEEDY-5-192 by exploiting key-recovery friendly distinguishers constructed following the divide-and-conquer strategy. Detailed contributions are shown as follows.

Strategy of Searching Key-Recovery Friendly Differential and Linear Trail of SPEEDY. Our searching strategy includes two parts: (a) control the propagation of active pattern deduced from the input difference/mask and output difference/mask; (b) construct long distinguishers in the divide-and-conquer way. To fulfill (a), we introduce TDDT (short for truncated differential distribution table) to show the non-randomness of S-box with bit-wise truncated differential propagation. With this table, one can depict clearly in the searching algorithm on how to choose input/output differences of the target distinguisher. While for (b), we construct part of the distinguisher by concatenating two distinguishers covering the small number of rounds. For instance, the 5.5-round distinguisher is first built with a 4-round distinguisher. To gain a better 4-round distinguisher, we split it into two parts with each covering two rounds. The input difference of the MixColumn operation in the middle is chosen to minimize the total number of activated S-boxes involved in its two neighborhood S-box layers. For the other MixColumns contained in the 5.5-round distinguisher, no restriction on their input differences is necessary.

Improved Key Recovery Attacks Against Full-Round SPEEDY-7-192. By exploiting the above strategy, we gain a 5.5-round differential distinguisher, which has a higher probability than the one found by [5]. With this differential trail, we mount the first chosen-plaintext attack on its full-round variant. Besides, since it's key-recovery-friendly, it also leads to a full-round chosen-ciphertext attack, which requires slightly less complexities than the one proposed by [5]. Further, we also mount the first known-plaintext 7-round linear attack with a 5-round linear distinguisher by adopting a similar search strategy. Although this one costs slightly higher complexities than the differential attack, it is a known-plaintext attack which requires weaker capabilities of the adversary than the chosen-plaintext attack required in the differential cryptanalysis. Therefore, such linear key recovery attack is also valuable. All our attack results along with previous published ones are summarized in Table 1.

Improved Key Recovery Attack on Round-Reduced SPEEDY-5-192. We also provide two attacks on 4-round SPEEDY-5-192 using differential and differential-linear cryptanalysis, respectively. More precisely, the differential attack is mounted with a two-round distinguisher where one round is added at the top and bottom, respectively, while the differential-linear attack is given by exploiting a three-round distinguisher and adding one half-round at both sides. Note that they are the best valid attacks on it in terms of the number of rounds.

1.2 Organization

First, we briefly recall SPEEDY, as well as differential, linear and differential-linear cryptanalysis in Sect. 2. In Sect. 3, we show how to exploit the propagation properties of SB and MC operations, and introduce the TDDT (short for

Table 1. Summary of valid attacks on SPEEDY-7-192 and SPEEDY-5-192. KP, CP and CC separately denote the data collected in the known-plaintext, chosen-plaintext and chosen-ciphertext settings. Time complexities are evaluated in encryption units, while memory costs are evaluated in block size.

Version	Security Claim (Data, Time)	R	Data	Time	Memory	Type	Ref
SPEEDY-7-192	$(2^{192}, 2^{192})$	7	$2^{187.27}$CC	$2^{187.75}$	2^{42}	Diff.	[5]
		7	$2^{186.53}$CC	$2^{187.39}$	2^{36}	Diff.	Sect. 5.1
		7	$2^{186.53}$CP	$2^{187.39}$	2^{156}	Diff.	Sect. 5.1
		7	$2^{188.50}$KP	$2^{189.41}$	$2^{185.04}$	Linear	Sect. 5.2
SPEEDY-5-192	$(2^{64}, 2^{128})$	3	$2^{17.6}$CP	$2^{52.5}$	$2^{17.62}$	Integral	[16]
		4	2^{61}CC	$2^{119.69}$	2^{83}	Diff.	Sect. 5.3
		4	2^{61}CP	2^{105}	2^{105}	Diff.-Linear	Sect. 5.4

truncated differential distribution table). Using these properties, we introduce the automated search method oriented to key recovery for differential, linear and differential-linear cryptanalysis in Sect. 4. Detailed attack procedures are given in Sect. 5. Finally, we summarize our work in Sect. 6. Besides, the code for searching distinguishers in this paper is available at the following repository: https://github.com/Jin-liang-Wang/Cryptanalysis-of-SPEEDY.

2 Preliminaries

2.1 Brief Introduction of SPEEDY

SPEEDY [9] is a family of ultra-low latency block ciphers proposed at CHES 2021. SPEEDY uses a 6-bit bijective S-box and can be instantiated with different block sizes, key sizes, and numbers of rounds. For instance, SPEEDY-r-$6l$ is the version of block size and key size of $6l$-bits, and r is the number of iterated rounds. The internal state of SPEEDY-r-$6l$ can be viewed as an $l \times 6$ rectangle array of bits where $l = 32$. Following the notation of its design document [9], we use $x_{[i,j]}$ to denote the bit located at row i and column j of the state x where $0 \leq i < l, 0 \leq j < 6$.

The design document of SPEEDY suggests $l = 32$, and the number of rounds $r \in \{5, 6, 7\}$, which are called SPEEDY-5-192, SPEEDY-6-192, SPEEDY-7-192, respectively. As for the security claim, SPEEDY-r-192 achieves 128-bit security when iterated over $r = 6$ rounds and full 192-bit security when iterated over $r = 7$ rounds, while the $r = 5$ rounds variant provides 2^{128} time complexity and 2^{64} data complexity. To make it clear, we denote SPEEDY-r-192 as SPEEDY in this paper, and briefly recall the cipher as follows.

Initialization. SPEEDY receives a 192-bit plaintext and initializes the internal state with a two-dimensional matrix x. Specifically, it initializes the k-th MSB (short for the most significant bit) into $x_{[i,j]}$, where $k = 6i + j$.

Round Function. Its round function consists of five different operations: SubBox (SB), ShiftColumns (SC), MixColumns (MC), AddRoundKey (A_{k_r}) and AddRoundConstant (A_{c_r}). A high-level overview of its round function is shown in Fig. 1. Note that the last round function only has three operations.

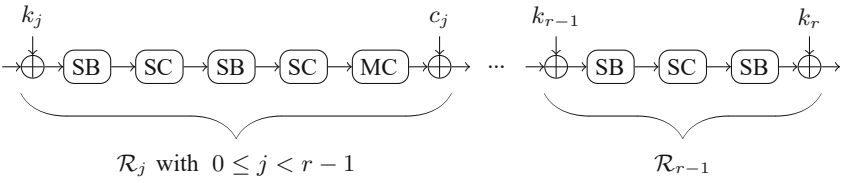

Fig. 1. r rounds of SPEEDY block cipher.

SB is composed of 32 parallel 6-bit Sbox, where each Sbox takes the i-th row $(x_{[i,0]}, x_{[i,1]}, x_{[i,2]}, x_{[i,3]}, x_{[i,4]}, x_{[i,5]})$ as input and its output is placed in the same row. The detail of S-box is shown in Appendix A. SC rotates the j-th column of the state upwards by j bits. In other words, the output bit $y_{[i,j]}$ equals to the input bit $x_{[(i+j) \bmod 32,j]}$. MC also operates on each column, where seven specifically chosen input bits are XORed as a new bit of its output, as shown in Eq. (1). It also can be seen as multiplying each column by a cyclic matrix.

$$y_{[i,j]} = x_{[i,j]} \oplus x_{[i+1,j]} \oplus x_{[i+5,j]} \oplus x_{[i+9,j]} \oplus x_{[i+15,j]} \oplus x_{[i+21,j]} \oplus x_{[i+26,j]}, \quad \forall i, j. \tag{1}$$

The 192-bit round key k_r and 192-bit constant c_r are respectively XORed with the entire state in A_{k_r} and A_{c_r}.

Key Schedule. The 192-bit master key of SPEEDY is regarded as the first round key k_0. Relation between the r-th round key k_r and the $(r+1)$-the round key k_{r+1} is constructed by a bit permutation PB. To be clear, the j-th bit of k_{r+1} equals to the i-th bit of k_r, where $j \equiv (7i+1) \bmod 192$.

2.2 Differential, Linear and Differential-Linear Crytanalysis

Differential Cryptanalysis. Differential cryptanalysis is a chosen plaintext attack proposed by Biham and Shamir [4] in 1990. Starting from a well-chosen difference δ_{in}, the distribution of $E_k(x) \oplus E_k(x \oplus \delta_{in})$ is non-uniform. More precisely, there exits a specific δ_{out} such that $\Pr_{x,k}[E_k(x) \oplus E_k(x \oplus \delta_{in}) = \delta_{out}]$ is significantly higher than 2^{-n}, where n is the block size. In order to distinguish a cipher with a high probability differential $(\delta_{in} \to \delta_{out})$ from a random permutation, one can collect D ciphertexts corresponding to pairs of plaintexts $(P, P \oplus \delta_{in})$, and compute the number of pairs following the differential:

$$Q = \#\left\{P : E_k(P) \oplus E_k(P \oplus \delta_{in}) = \delta_{out}\right\}.$$

The expected value of Q for the cipher is $D \times \Pr[\delta_{in} \to \delta_{out}]$ and $D \times 2^{-n}$ for the random permutation. Thus, the distinguisher succeeds with high probability when $D = \mathcal{O}(1/\Pr[\delta_{in} \to \delta_{out}])$.

Linear Cryptanalysis. Linear cryptanalysis, proposed by Matsui at EUROCRYPT 1993 [13], exploits the linear approximations of the round function in order to obtain a biased approximation of the cipher. The linear approximation is a pair of masks (α, α') such that the distribution of the XORed masked plaintext and ciphertext $x \cdot \alpha \oplus E_k(x) \cdot \alpha'$ is biased. More precisely, we have $(|\Pr_x[x \cdot \alpha \oplus E_k(x) \cdot \alpha'] - \frac{1}{2}| \gg 2^{-n/2})$ for most keys k), where $x \cdot y = \bigoplus_i x_i y_i$ denotes the inner product. Then, we have the correlation which is expected to be zero when averaged over all keys is:

$$\mathrm{Cor}_k\left(\alpha \rightarrow \alpha'\right) = 2\Pr_x\left[x \cdot \alpha = E_k(x) \cdot \alpha'\right] - 1.$$

Similarly, to distinguish a cipher with a biased linear approximation (α, α') from a random permutation, we collect D known plaintexts/ciphertexts, and evaluate the experimental correlation:

$$Q = \left(\#\left\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 0\right\} - \#\left\{P, C : P \cdot \alpha \oplus C \cdot \alpha' = 1\right\}\right)/D.$$

The expected value Q is larger (in absolute value) for the cipher than for a random permutation and can be observed with high probability when $D = \mathcal{O}(\mathrm{Cor}(\alpha \rightarrow \alpha')^{-2})$.

Differential-Linear Cryptanalysis. Differential-linear cryptanalysis, proposed by Langford and Hellman in 1994 [8], is a technique to combine differential and linear cryptanalysis. Let E be a cipher which can be described as a cascade of three subciphers, E_0, E_m and E_1, i.e., $E = E_1 \circ E_m \circ E_0$. Let δ_{in} and δ_{out} denote the input and output difference of E_0, α and α' be the input and output linear mask of E_1. Asume that the differential trail $\delta_{in} \rightarrow \delta_{out}$ in E_0 is satisfied with probability p, and the linear trail $\alpha \rightarrow \alpha'$ in E_1 is satisfied with correlation q. The correlation of $\delta_{out} \rightarrow \alpha$ in E_m is satisfied with

$$r = \mathrm{Cor}\left(\delta_{out} \xrightarrow{E_m} \alpha\right) = 2\Pr_x\left[E_m(x) \cdot \alpha = E_m(x \oplus \delta_{out}) \cdot \alpha\right] - 1.$$

Then, the correlation of the trail $\delta_{in} \xrightarrow{E} \alpha'$ is satisfied with correlation $\mathrm{Cor}(\delta_{in} \xrightarrow{E} \alpha') = pq^2r$. One can collect D ciphertexts corresponding to pairs of plaintexts $(P, P \oplus \delta_{in})$ and evaluate the experimental correlation:

$$Q = \left(\#\left\{P : \alpha' \cdot (E(P) \oplus E(P \oplus \delta_{in})) = 0\right)\right\} - \#\left\{P : \alpha' \cdot (E(P) \oplus E(P \oplus \delta_{in})) = 1\right)\right\}\right)/D.$$

Similarly, the expected value is larger (in absolute value) for the cipher than for a random permutation and can be observed with high probability when $D = \mathcal{O}(\mathrm{Cor}(\delta_{in} \xrightarrow{E} \alpha')^{-2})$.

3 Further Study on Operations of SPEEDY

In general, the extended path before and after the distinguisher determines the number of guessed key bits, the attack rounds, and the time complexity. To

control the number of active S-boxes in the extended path, we add the constraint of the propagation through key recovery rounds and propose the concept of TDDT (short for truncated differential distribution table). Since the security of SPEEDY highly relies on the 6-bit S-box and the MC operation, we did in-depth research for SB operation with the TDDT in Sect. 3.1 and for MC operation in Sect. 3.2, respectively. With these newly observed properties, we can construct the automated search method for the differential and linear distinguishers by taking the key recovery into account in Sect. 4.

3.1 Truncated Differential Distribution Table

In general, the time complexity of differential cryptanalysis is affected by the number of active bits in plaintext and ciphertext. To control the number of active bits, we limit some bits being inactive before the second SB operation to reduce the number of active bits in plaintext, as shown in Sect. 4.

On average, each bit is limited inactive in rounds of key recovery with probability 2^{-1}. However, we have found that in a specific S-box, e.g., the S-box of SPEEDY, this probability is fluctuate considerably. Two examples are provided in Equations (2a) and (2b) below, although the input sets of these two situations are both one bit inactive, the probabilities of Eqs. (2a) and (2b) are much higher than 2^{-1}.

$$\texttt{**0***} \xrightarrow[p=2^{-0.54}]{\mathsf{SB}} \texttt{010000}\,, \tag{2a}$$

$$\texttt{0*****} \xrightarrow[p=2^{-0.83}]{\mathsf{SB}} \texttt{010000}\,, \tag{2b}$$

where $*$ denotes a arbitrary bit $\in \{0,1\}$. The probability is computed as follows. Taking Eq. (2b) as an example. The input set Δ_{in} contains 2^5 differences, i.e., {0b000000,0b000001,$\cdots$,0b011111}. We assume that each difference in Δ_{in} appears with equal probability. Besides, the output difference is $\delta_{out} =$ 0b010000. Then this probability is computed by $\Pr(\Delta_{in} \xrightarrow{\mathsf{SB}} \delta_{out}) = \frac{\sum_{\delta_{in}\in\Delta_{in}} \Pr(\delta_{in} \xrightarrow{\mathsf{SB}} \delta_{out})}{|\Delta_{in}|}$.

In order to get an optimal probability in rounds of key recovery, we need to consider how to select inactive bits. In this section, we introduce the TDDT and show how it can select inactive bits using the automatic solver in Sect. 4.

Compared to DDT (short for differential distribution table [4]) which contains the number of transition situations of each fixed input-output difference, for a N-bit S-box, the TDDT is a $2^N \times 2^N$ table which contains the number of transition situation from a set of input differences to a set of output differences. To clarify, we define the following notation to represent a set of differences named bit-string $\mathbf{k}$. For an n-bit bit-string $\mathbf{k} = (k_0, k_1, ..., k_{n-1}), \mathbf{k} \in \mathbb{F}_2^n$, we define $\mathbf{k}' \preceq \mathbf{k}$, if $k'_i \leq k_i$ for all i. Then, we denote $\mathcal{B}_{\Delta_{in}}/\mathcal{B}_{\Delta_{out}}$ as the bit-string representation of the input/output difference set Δ_{in}/Δ_{out}. For example, when a truncated input difference Δ_{in} is "00*00*", the bit-string representation $\mathcal{B}_{\Delta_{in}}$ is $\texttt{0b001001}_{(\mathcal{S})}$[1],

[1] To distinguish the set of differences from the single differential, we use numbers with subscript $_{(S)}$ to represent the set of differences.

i.e., $\mathcal{B}_{\Delta_{in}} = \texttt{0b001001}_{(\mathcal{S})}$ denotes the differential set `"00*00*"` which contains differences {`0b000000`,`0b000001`,`0b001000`,`0b001001`}. Thus, we can construct TDDT from the DDT using Algorithm 1 and the probability of the truncated difference propagation from Δ_{in} to Δ_{out} is:

$$\Pr(\Delta_{in} \xrightarrow{\mathsf{SB}} \Delta_{out}) = \frac{\mathsf{TDDT}[\mathcal{B}_{\Delta_{in}}][\mathcal{B}_{\Delta_{out}}]}{|\Delta_{in}| \cdot 2^N}, \tag{3}$$

where N is the size of S-box and $|\Delta_{in}|$ denotes the number of input differences δ_{in} that fulfills $\delta_{in} \preceq \Delta_{in}$.

Algorithm 1: Construct TDDT

Input: S-box
Output: TDDT; `// Truncated difference distribution table`
$\mathcal{L}[i][j] \leftarrow 0, (0 \le i, j < 2^N)$; `// Initial an empty list for TDDT`
$\mathcal{L}[0][0] \leftarrow 2^N$; `// Initialize the case` $0 \xrightarrow{\mathsf{SB}} 0$ `with probability 1`
Generate the DDT of S-box;
for $\mathcal{B}_{\Delta_{in}} \in [1_{(\mathcal{S})}, 2^N_{(\mathcal{S})})$ **do**
  for $\mathcal{B}_{\Delta_{out}} \in [1_{(\mathcal{S})}, 2^N_{(\mathcal{S})})$ **do**
    for $\delta_{out} \preceq \Delta_{out}$ **do**
      for $\delta_{in} \preceq \Delta_{in}$ **do**
        $\mathcal{L}[\mathcal{B}_{\Delta_{in}}][\mathcal{B}_{\Delta_{out}}] \leftarrow \mathcal{L}[\mathcal{B}_{\Delta_{in}}][\mathcal{B}_{\Delta_{out}}] + \mathsf{DDT}[\delta_{in}][\delta_{out}]$;
return $\mathcal{L}$ as TDDT of the S-box;

By exploiting the differential property of the transition from a set of differences that are inactive in certain bits to another set of differences with different inactive bits, TDDT can help us make a universal search model for key recovery. Moreover, we also consider the cases of differential propagation from the fixed input difference to a set of output differences. Then, we define FTDDT(short for fixed-input truncated difference distribution table) to describe the probability distribution of this transition and introduce it as follows.

Fixed-Input Truncated Difference Distribution Table. As aforementioned, FTDDT contains the number of transition situations of a fixed input difference to a set of output differences. For the special case, when the input difference is zero, the output difference must be zero, i.e., the set of output differences Δ_{out} contains only zero differences. More precise, when $\delta_{in} = 0$, we have

$$\mathsf{FTDDT}[0][0] = 2^N \quad \text{and} \quad \mathsf{FTDDT}[0][i] = 0, i \in [1, 2^N).$$

We show the construction of FTDDT in Algorithm 2. By modeling these difference distribution tables into our automated search model, we can find a key recovery-oriented differential which contains key recovery rounds as well

as distinguishers. Time complexity of Algorithm 2 is$(2^N - 1) \cdot \sum_{i=1}^{N} \binom{N}{i} 2^i = \mathcal{O}(2^{N \cdot \log_2 6})$, which is evaluated as the number of the look-up table operations.

Algorithm 2: Construct FTDDT

Input: S-box;
Output: FTDDT; `// Fixed input/output truncated difference distribution table`

1 $\mathcal{L}[i][j] \leftarrow 0$, $(0 \le i, j < 2^N)$; `// Initialize an empty list for FTDDT`
2 $\mathcal{L}[0][0] \leftarrow 2^N$; `// Initial the case` $0 \xrightarrow{\text{SB}} 0$ `with probability 1`
3 Generate the DDT of S-box;
4 **for** $\delta_{in} \in [1, 2^N)$ **do**
5 **for** $\Delta_{out} \in [1_{(\mathcal{S})}, 2^N_{(\mathcal{S})})$ **do**
6 **for** $\delta_{out} \preceq \Delta_{out}$ **do**
7 $\mathcal{L}[\delta_{in}][\Delta_{out}] \leftarrow \mathcal{L}[\delta_{in}][\Delta_{out}] + \text{DDT}[\delta_{in}][\delta_{out}]$;
8 **return** $\mathcal{L}$ as FTDDT of the S-box;

In summary, FTDDT shows the non-randomness of S-box with bit-wise truncated differential propagation. Considering the differential propagation through a random permutation, the fixed input difference δ_{in} will propagate to the output difference set Δ_{out} with probability 2^{-i} where the number of zero bits in $\mathcal{B}_{\Delta_{out}}$ is i. As mentioned before, due to the non-randomness of a specific S-box, the probability is changed to $\text{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]/2^N$. In order to verify the correctness of FTDDT, we have conducted an experiment in Appendix B. With this accurate propagation probability of differential, we can gain a better-extended path for a key recovery attack as shown in Sect. 4.

Inverse Probability of FTDDT. During key recovery, we need to partially encrypt (resp. decrypt) the plaintext pair (resp. ciphertext pair) to validate the input (resp. output) of the distinguisher. To filter the good pairs of plaintext more precisely in validating the input of the distinguisher, we utilize the inverse of FTDDT. We denote N_a as the number of active bits in Δ_{out}. For example, $N_a = 5$ if $\Delta_{out} =$ `0b110111`, i.e., `**0***`. Then, when considering a fixed difference $\delta_{out} \preceq \Delta_{out}$ propagating to δ_{in} which is the inverse of $\text{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]$, the average probability of this transition is $\text{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]/2^{N+N_a}$. The proof is shown in Appendix C. With the inverse probability of FTDDT, we will get the more precise time complexity of the key-recovery procedure.

3.2 Differential Propagation Property of MC

The MC operation of `SPEEDY` generates each new bit in a column by XORing seven current bits. It also can be seen as multiplying each column by a

cyclic matrix M. In this section, we research the MC operation in-depth by considering the matrix M. Here we traverse all possible inputs with HW (short for Hamming weight) from one to eight and find that their minimum output HW are $(7, 8, 7, 8, 7, 6, 5, 4)$. Table 2 summarizes which choices of active differential bits are associated with these minimal output HW cases. Specifically, we use $\{a_0, a_1, \cdots\}$ to denote that the a_i-th bit of the input difference is active. Moreover, due to the property of the MC operation of SPEEDY, we rotate the array and make the 0-th bit active to represent the rotation invariant array class. From Table 2, we can observe that when the input HW is one, there is a minimum sum of the input and output minimum HW. Meanwhile, when the input HW is two and three, the minimal sum of the input and output HW is relatively small.

Table 2. Summary of active differentials yielding low output HW from M. When the input HW is greater than 3, the cases of active bits are too large to list. We record their number to simplify. For the input differences with different HW, we list its output difference which have the minimum HW.

In HW	Out HW	Active Bits	Total HW
1	7	{0}	8
2	8	{0, 6}	10
		{0, 11}	10
3	7	{0, 4, 15}	10
4	8	10 cases	12
5	7	7 cases	12
6	6	8 cases	12
7	5	8 cases	12
8	4	5 cases	12

When building the automated search model, one can utilize the property of MC to control the diffusion of the differential, which can reduce the size of the search space. In this work, we utilize a divide-and-conquer method to search for a longer differential by dividing it into several parts. More precisely, for a four rounds differential, we first search several two rounds differentials with the input difference of the last MC is of relatively low HW as in Table 2. Then, we search the last two rounds of differential and connect them with the middle MC. By limiting the **sum** of input-output HW of the middle MC, we can control the diffusion of the 4-round differential, *i.e.* minimizing the total number of activated S-boxes involved in the middle, and achieve a higher probability.

We note here that Boura et al. [5] proposed a different strategy to utilize the properties of the MC operation. In their approach, a differential is divided into several one-round parts and they consider the case that the HW of **both** input and output are small.

4 Automated Search Method Oriented to Key Recovery

In this section, we introduce the automated search strategies of differential, linear and differential-linear distinguishers against SPEEDY following the divide-and-conquer strategy, as well as taking the key recovery procedure into consideration.

Usually, cryptanalysts perform key recovery attacks based on a strong distinguisher. By appending several rounds before and after the distinguisher, one can obtain an extended path containing both distinguisher and key recovery rounds. However, this two-step process of key recovery cannot promise optimal results as shown in [15,18]. Therefore, automated search oriented to key recovery can obtain better attack results. To find such key-recovery-friendly distinguishers, properties of SB and MC are introduced in Sect. 3 are utilized.

We implement our search algorithms using STP[2], which is a solver developed based on the SAT (short for boolean satisfiability problem) [12]/SMT(short for satisfiability modulo theories) [2] problem and has been widely used in the field of cryptography [6,10,11,14].

4.1 Differential Search Model Against SPEEDY

Since we aim to search for distinguishers in the single-key setting, we can omit A_{k_r} and A_{c_r} in our model. We denote S_j^i as the j-th intermediate state of the i-th round and set the first round indexed with zero as the key recovery round.

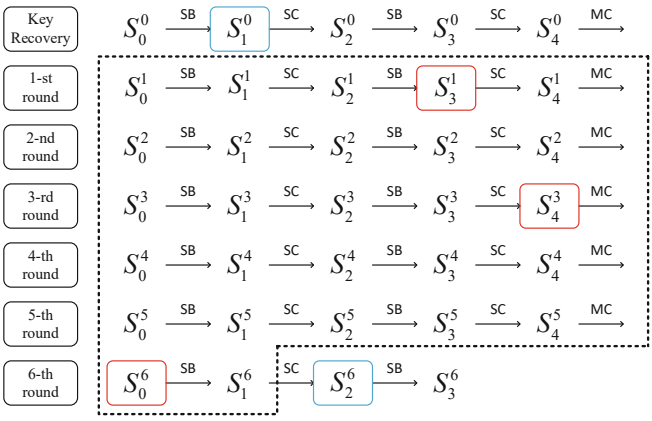

Fig. 2. 7-round differential attack of SPEEDY.

Generally, the variable number and the size of each variable will determine the total scale of the search model and affect the complexity of solving the SAT/SMT problem. As the number of rounds increases, the time and memory

[2] https://stp.github.io/.

complexity of solving the problem increases exponentially. Specifically, when the model exceeds four rounds, we cannot obtain a solution in a reasonable time with STP in the case of SPEEDY. For obtaining a longer distinguisher, we employ a divide-and-conquer [18] approach to search several models with a short round to form the entire differential. As shown in Fig. 2, our automated search model is divided into several steps. Motivated by [15], we also consider a key recovery-oriented search model where the distinguisher rounds and key recovery rounds are all included.

Model One: 7-Round Differential Contains 1.5-Round of Key Recovery. The 7-round differential consists of 5.5-round differential distinguisher $S_0^1 \rightarrow S_1^6$ and 1.5 key recovery rounds $S_0^0 \rightarrow S_0^1, S_1^6 \rightarrow S_3^6$. For convenience, we use backward FTDDT to represent the FTDDT of SPEEDY S-box and forward FTDDT to represent the FTDDT of the inverse of SPEEDY S-box.

Step 1: From S_3^1 to S_4^3, the search strategies are as follows.

Rule 1: Model SB to comply with DDT.

Rule 2: Model each active S-box in S_3^1 to comply with $2^{-3\times 2}$ probability. This strategy is mainly used to limit the active S-box in the S_3^1. More precisely, we measure this trail with probability $2^{-6\times N_a} \times p$, where N_a is the number of active S-box in S_3^1 and p is the probability generated from **Rule** 1 in **Step** 1. Indeed, N_a active S-boxes imply that there will have $2 \times N_a$ S-boxes in trail $S_0^1 \rightarrow S_3^1$. Since the maximum probability transition through the S-box is 2^{-3}, we measure each active S-box in S_3^1 with probability 2^{-6}.

Rule 3: Constrain only one of the columns to be activated in S_4^3. For the input before MC, consider only entries with low HW from Table 2.

Step 2: From S_4^3 to S_0^6, the strategies to reduce search space are as follows.

Rule 1: Due to the differential must be linked, we fix the input differences as those in S_4^3.

Rule 2: Model the active S-box in S_0^6 to comply with a maximum probability of 2^{-3}.

Step 3: After running the above two steps in parallel, we obtain several four rounds of differentials with high probability. Then, we choose the differential whose probability is higher than 2^{-170} and search the corresponding last 1.5-round differential of distinguisher and 1.5-round of key recovery as follows.

Rule 1: Model $S_0^1 \xrightarrow{\text{SB}} S_1^1$ and $S_2^1 \xrightarrow{\text{SB}} S_3^1$ to comply with DDT.

Rule 2: Model $S_1^0 \xrightarrow{\text{SB}^{-1}} S_0^0$ and $S_0^6 \xrightarrow{\text{SB}} S_1^6$ to comply with FTDDT.

Rule 3: Constrain the active S-box in S_1^0 and S_2^6 less than 32.

Rule 4: Constrain the active S-box number in S_1^0 less than 16.

Rule 5: Constrain the active S-box number in S_2^6 more than 6.

Here, rules 3 to 5 are used to constrain the active bits to reduce the time complexity of the key recovery phase as shown in Sect. 5.1.

Step 4: Following step one to three, we can get several 5.5-round distinguishers with 1.5-round of key recovery. Then, we choose the distinguisher with maximum probability and search an extended path for key recovery as shown in Algorithm 6 (Appendix G). To reduce the time complexity, we consider the differential rotation invariant of SPEEDY. More precisely, we use one of the 32 rotation-invariant distinguishers to implement the key recovery attack, and the chosen one makes the lowest time complexity.

With the above four steps, we find a differential trail with probability $2^{-185.53}$ which can be used to attack on the 7-round SPEEDY. More precisely, this trail consists of a 5.5-round differential distinguisher with probability $2^{-182.49}$ and a 1.5-round key-recovery phase with probability $2^{-3.04}$. We illustrate it as Fig. 3.

Model Two: 4-Round Differential Contains Two Rounds of Key Recovery. From **Step 2**, we can get several two rounds distinguisher from S_4^3 to S_0^6. Then, we use backward FTDDT and forward FTDDT to search 2-round of key recovery. We show this distinguisher with probability $2^{-59.35}$ in Appendix J. For the property of linear key relation between k_0 and k_4, the rotation invariant property does not affect the time complexity when attacking the 4-round SPEEDY.

4.2 Searching Linear Distinguishers Against SPEEDY

For linear cryptanalysis, we use a similar strategy used for differential search, which is a model oriented toward key recovery. Then, we search for a five-round distinguisher with two rounds of key recovery by following a four-step process. Due to the page number limit of the paper, we have placed the specifics in Appendix D.

Finally, we found a 5-round distinguisher combined with two key recovery rounds with correlation $2^{-93.01}$. We show the detail of the linear distinguisher in Appendix I. In this distinguisher, the bits of k_7 which is derived from k_0 by linear key schedule is 139. We reduce the guessed key bits to 185 bits in the key recovery phase.

4.3 Searching Differential-Linear Distinguishers Against SPEEDY

Similarly, we search for distinguishers against 4-round SPEEDY. More specifically, we search for three rounds of differential-linear distinguishers and extend half a round forward and backward for key recovery. The detailed strategy is shown in Appendix E.

Finally, the distinguisher is shown in Fig. 5 of Appendix F. The correlation of this 3-round distinguisher is $2^{-23.095}$ and the probability of rounds of key-recovery is $2^{-6.972}$. The bits of k_7 deduced from k_0 by linear key schedule are 29.

5 Key Recovery Attack Against SPEEDY

In this section, we first implemented both differential and linear cryptanalysis on **full** SPEEDY with the newly obtained distinguishers in Sect. 4. Then, the key recovery attack against other versions of SPEEDY are also provided.

5.1 7-Round Differential Attack Against Full SPEEDY

Using the newly obtained 7-round differential containing distinguishers and key recovery rounds, we can perform key recovery attacks on **full** SPEEDY. Before the attack, we first introduce some techniques that will be used in this section as follows.

Deduce k_0 from k_7 by Linear Key Schedule. Due to the linear key schedule, we can get the relationship between k_0 and k_7 and deduce the key bits of k_0 from k_7. We call these bits of k_0 deduced key bits from k_7. The detailed relationship between k_0 and k_7 is shown as the purple squares in Fig. 3.

Function to Sieve Possible Key Guesses. The FirstSBoxSieve function uses the key bits of k_7 to deduce the six bits of each row in k_0 using the linear key schedule. After deducing the six bits of k_0 from k_7, we can filter the corresponding key guessing of the plaintext pair with the zero difference in S_1^0. Then, for each plaintext pair and the corresponding six bit keys of each row, there will leave 2^{6-d-n} key guesses after sieving by the first S-box, where d is the bit number deduced from k_7 and n is the inactive bit number after the first S-box. We show the details of FirstSBoxSieve in Algorithm 3. The SecondSBoxSieve is used to combine the sets L_i of possible key bits in a bigger set. The function can sieve the elements of the bigger set using the difference in S_3^0. The detail is shown in Algorithm 4.

Distinguisher with Rounds of Key Recovery. The 5.5-round differential distinguisher is shown in Fig. 3, where the black square and the blue square denote the active bits from S_4^1 to S_4^5 and the active bits from S_3^0 to S_3^1, respectively. To distinguish the differential propagation of the key recovery round, we use different colors to indicate the propagation of each active bit in the extended rounds, where the gray square indicates the active bits whose differences are forced zero. Further, we use the shaded square to denote the bits random from $\{0, 1\}$. Then, we use the red square to denote the active bits in S_0^0 and S_3^6. To represent the bits in k_0 that are derived from k_7 using the linear key schedule, we use the purple square to denote the deduced bits in k_0 and the green square to denote the guessed key bits when performing the key recovery attack. Note that 180 key bits in k_0 are involved in the key recovery attack.

The attack procedure contains three parts, i.e., data collection, key recovery and brute force the master key.

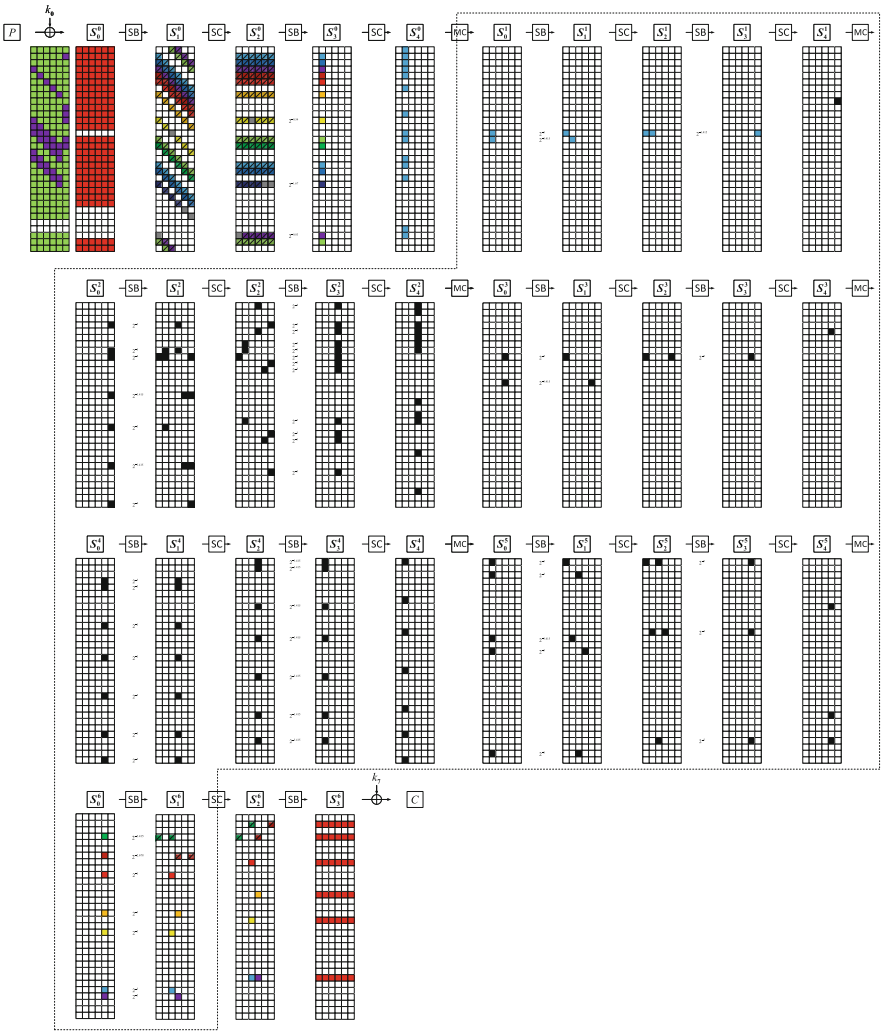

Fig. 3. Key recovery attack on 7-round `SPEEDY` with differential cryptanalysis. When the output set of a FTDDT has only one single bit, it is shown as a single difference in this figure since it has one possible case for such an output set, e.g., `0b00*000` is shown as `0b001000` since the active difference is impossible to transit to the `0b000000` through the S-box of `SPEEDY`.

Algorithm 3: Sieving keys with the first S-box

Input: (m_0^i, m_1^i); `// The` i`-th row of the plaintext pair`
Input: Δ_k; `// The set of 6-bit keys`
Input: δ_{out}; `// The output difference for right pair`
Output: Possible 6-bit keys for the i-th row with the corresponding compressed internal state of pairs in S_1^0;

function FirstSboxSieve($m_0^i, m_1^i, \Delta_k, \delta_{out}$)
  $\mathcal{L} \leftarrow \emptyset$;
  for *all* $k_i \in \Delta_k$ **do**
    $x_0 \leftarrow$ S-box$(m_0 \oplus k_i)$;
    $x_1 \leftarrow$ S-box$(m_1 \oplus k_i)$;
    if $x_0 \oplus x_1 == \delta_{out}$ **then**
      Compress x_0 by the active bits in δ_{out};
      Compress x_1 by the active bits in δ_{out};
      $\mathcal{L} \xleftarrow{\cup} (x_0, x_1, k_i)$;
  if $\mathcal{L} == \emptyset$ **then**
    Discard the 6-bit last round key for this (m_0^i, m_i^j) pair;
  else
    return $\mathcal{L}$;

Data Collection. Consider structure S which consists of 2^{156} plaintexts and for each plaintext pair $P_0, P_1 \in S$, $P_0 \oplus P_1 = (N, N, N, N, N, N, N, N, N, N, N, N, N, 0, N, N, N, N, N, N, N, N, N, N, N, 0, 0, 0, 0, 0, 0, N, N)$, where N denotes any 6-bit cell. We require the oracle to encrypt those 2^{156} plaintexts and collect the ciphertexts by the *hash* table which is indexed by the 156-bit inactive bits in ciphertexts. From the *hash* table, we expect to get 2^{155} pairs of plaintext-ciphertext pair $(P_0, C_0), (P_1, C_1)$ that satisfy $C_0 \oplus C_1 = (0, N, 0, N, 0, 0, 0, 0, N, 0, 0, 0, 0, 0, N, 0, 0, 0, N, 0, 0, 0, 0, 0, 0, 0, 0, 0, N, 0, 0, 0, 0, 0, 0, 0)$.

Next, we briefly introduce the key recovery and brute force phases, the detail of the pseudocode is shown in Appendix G. We use δ_{in}^i to denote the i-th row of the input difference of the 5.5-round distinguisher.

Key Recovery. For each pair obtained from the data collection phase, we first use $C_0 \oplus C_1$ to deduce the 36-bit k_7 from DDT. Note that, since we evaluate the propagation probability of key-recovery rounds by FTDDT, the number of deduced 36-bit k_7 should be calculated by the inverse probability of FTDDT as shown in Sect. 3.1. For each deduced 36-bit k_7, we fix the deduced keys in k_0 from k_7 due to the linear key schedule and guess other 144-bit k_0 as follows.

Firstly, we use the early abort technique which is guessing the key-bits of row `14,15,17` and partial encrypt to run FisrtSboxSieve. If there are no satisfied key guesses to validate the output difference of the SubBox operation, we consider the 36-bit deduced k_7 is wrong and discard it. If all the

Algorithm 4: Sieving keys with the second S-box

Input: $[\mathcal{L}_i, ..., \mathcal{L}_{i+n-1}]$; `// The sets obtained from` FisrtSboxSieve `or` SecondSboxSieve

Input: δ_{out}; `// The output difference for right pair in` S_3^0

Output: The set of possible keys with the corresponding compressed internal state of pairs in S_3^0.

function SecondSboxSieve($m_0^i, m_1^i, \Delta_k, \delta_{out}$)
  $\mathcal{L} \leftarrow \emptyset$;
  for *all* $(x_0^i, x_1^i, k_i) \in \mathcal{L}_i$ **do**
    for *all* $(x_0^{i+n-1}, x_1^{i+n-1}, k_{i+n-1}) \in \mathcal{L}_i$ **do**
      $x_0 \leftarrow$ ComActBit$(x_0^i, \cdots x_0^{i+n-1})$;
      $x_1 \leftarrow$ ComActBit$(x_1^i, \cdots x_1^{i+n-1})$;
      /* ComActBit `function is used to combine the bits propagated by the` δ_{out} `in` $(x_t^i, \cdots x_t^{i+n-1})$ `into` x_t */
      $y_0 \leftarrow$ S-box(x_0);
      $y_1 \leftarrow$ S-box(x_1);
      if $y_0 \oplus y_1 == \delta_{out}$ **then**
        $key \leftarrow k_i||...||k_{i+n-1}$;
        $x_0 \leftarrow$ SaveBit$(x_0^i, \cdots x_0^{i+n-1})$;
        $x_1 \leftarrow$ SaveBit$(x_1^i, \cdots x_1^{i+n-1})$;
        /* SaveBit `function is used to save the bits used in the following algorithm` */
        $\mathcal{L} \xleftarrow{\cup} (x_0, x_1, key)$;
  if $\mathcal{L} == \emptyset$ **then**
    Discard the 6-bit last round key for this (m_0^i, m_i^j) pair;
  else
    return $\mathcal{L}$;

deduced k_7 are wrong, we consider this pair of plaintext-ciphertext is wrong and discard it. Next, for each plaintext-ciphertext pair, we sieve k_0 in row `11,12,13,16` with FisrtSboxSieve and partial encrypt to δ_{in}^{11} with SecondSboxSieve. After this step, the plaintext-ciphertext pair and corresponding k_0 can validate the input difference δ_{in}^{11}. Similarly, we discard the deduced k_7 if there are no satisfied k_0 guesses to validate δ_{in}^{11} and discard the pair if no deduced k_7 is left. Then, we employed the same method to those deduced keys in the row of k_0 with FisrtSboxSieve and partial encrypt plaintext pairs to $\delta_{in}^{14}, \delta_{in}^{15}, \delta_{in}^{18}, \delta_{in}^{19}, \delta_{in}^{7}, \delta_{in}^{5}, \delta_{in}^{4}, \delta_{in}^{3}, \delta_{in}^{2}, \delta_{in}^{1}, \delta_{in}^{30}, \delta_{in}^{29}, \delta_{in}^{21}$ to further sieve with SecondSboxSieve.

Finally, it will leave $2^{135.46}$ pairs and $2^{6.5}$ 180-bit keys for each pair on average, and we will get $2^{141.96}$ candidate keys of this 180-bit master key for each structure. Then, we use the brute force procedure to sieve the wrong candidate keys.

Brute Force. For each candidate key from the key recovery procedure, we guess the remaining 12-bit of the master key to get the complete 192-bit candidate master key k^c. Then we use a test pair (m_t, c_t) from encrypt oracle to check if $E_{k^c}(m_t) = c_t$ to verify if k^c is the right key.

Complexity and Success Probability. Since the memory access of storing the ciphertexts to the *hash* table is negligible compared to the time for collecting the ciphertexts, the time complexity of data collection for each structure is 2^{156}. Since the *hash* table storing 2^{155} pairs is very large, we treat the time to access this table as one encryption. Further, since the 7-round SPEEDY encryption has $7 \times 2 \times 32 = 448$ SubBox operations, the time complexity of the key recovery phase is less than $2^{155} + (70 \times 2^{155})/448$ for each structure (details are shown in Appendix H). Then, the time complexity of Brute Force phase is $2^{151.96}$ times 7-round SPEEDY encryption for each structure. Overall, the total time complexity of each structure is $2^{156} + 2^{155} + (70 \times 2^{155})/448 + 2^{153.96} \approx 2^{156.86}$ 7-round SPEEDY encryptions.

The signal-to-noise-ratio is Using $2^{185.53}/2^{155} \approx 2^{30.53}$ structures, we can ensure the success rate of 79.15% with 7.51-bit advantage according to [17]. Hence, the data complexity is $2^{186.53}$, the time complexity is $2^{30.53} \times 2^{156.86} \approx 2^{187.39}$ and the memory required is 2^{156}.

Reducing Memory Requirement. The above attack require 2^{156} bits in memory, which is too large. To obtain lower memory complexity, we can use the chosen ciphertext attack mode and select the active 36-bit structure at the ciphertext. After using the inactive 36 bits in the plaintext as the index, we get 2^{35} pairs, and then one can follow the key recovery procedure used in the above attack to obtain the right key.

5.2 7-Round Linear Attack Against Full SPEEDY

With the 5-round linear distinguisher shown in Appendix I, we compute the attack parameter on SPEEDY-7-192 using the symbols borrowed from [7]. With the $\mathcal{FWT}$ approach in [7], the time complexity is $2(\rho_M + \rho_A)2^{|k_0|+|k_7|-l_{07}}$, where the $|k_i|$ means the bit number guessed in k_i, and l_{07} means the key bits of k_7 deduced from k_0. Following the notation in [7], ρ_A is the cost of an addition, and ρ_M is the cost of a multiplication. Assuming that two multiplications correspond to roughly one evaluation of the cipher, we have the time complexity of key recovery phase 2^{186} encryptions. The memory required is $2^{144} + 2^{180} + 2^{185} \approx 2^{185.04}$. According to [17], we need $2^{188.50}$ data and the advantage $a = 4$ to achieve the success rate 69.12%. Finally, we exhaustively search the remaining 7 key bits. The total time complexity of the attack is $2^{188.50} + 2^{186} + 2^{188} \approx 2^{189.41}$.

5.3 4-Round Differential Attack Against SPEEDY.

As shown in Appendix J, we choose ciphertexts in S_3^3 to make a 60-bit structure and filter the corresponding plaintext pairs in S_0^2. After running this step, it will

leave $\binom{2^{60}}{2}/2^{36} \approx 2^{83}$ pairs. Then, we sieve the key bits of the 108-bit involved k_4 according to the 2^{83} pairs and leave $2^{43.692}$ possible k_4 for each pair. Then, for the combination of these $2^{83} \times 2^{43.692} = 2^{126.692}$ plaintext-ciphertext pairs and corresponding key guesses, we guess the 156-bits k_0 using the method in Sect. 5.1.

Due to the differential probability being $2^{-59.35}$ and one structure can only provide 2^{60} data, we construct two structures to collect the data. The data complexity of this attack is 2^{61}, the time complexity is $2^{119.692}$ and the memory required is 2^{83}. With above attack parameters, the success rate is 94.17%.

5.4 4-Round Differential-Linear Attack Against SPEEDY

With the distinguisher searching in Sect. 4.3, we obtain a 4-round differential-linear attack against SPEEDY with 2^{61} data complexity 2^{105} time complexity and 2^{105} memory requirement. The detail of this attack is shown in Appendix F.

6 Conclusion

In this paper, we first show how to find key-recovery friendly distinguishers for SPEEDY by following the divide-and-conquer strategy as well as some other new techniques. With such strategies, we found a 5.5-round differential distinguisher which is key-recovery friendly and with higher probability than the one used in the previous result. Using this distinguisher, we are able to mount the first chosen-plaintext full-round attack on SPEEDY-7-192. Besides, with the same distinguisher, we can also mount a full-round attack under the chosen-ciphertext setting, which slightly reduces the attack complexities compared with the previous one proposed in the same setting. Meanwhile, using a similar search strategy, we also found a 5-round linear distinguisher which leads to the first known-plaintext full-round attack. Although the full-round security of this variant has recently been announced to be broken, our attacks here need a weaker ability requirement for the adversary. Besides, for SPEEDY-5-192, which requires more restricted attack parameters, we also propose two 4-round key recovery attacks using a differential distinguisher and a differential-linear one, respectively. According to our best knowledge, both attacks are the best ones on this variant in terms of the number of rounds. However, its full-round security is not threatened.

Acknowledgements. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025). The corresponding author is also supported by Qingdao Innovation project (Grant No. QDBSH20230101008).

A The 6-Bit S-Box of SPEEDY

(See Table 3).

Table 3. The 6-bit S-box of SPEEDY. All elements are expressed in hexadecimal.

x_0x_1	$x_2x_3x_4x_5$															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	08	00	09	03	38	10	29	13	0c	0d	04	07	30	01	20	23
1	1a	12	18	32	3e	16	2c	36	1c	1d	14	37	34	05	24	27
2	02	06	0b	0f	33	17	21	15	0a	1b	0e	1f	31	11	25	35
3	22	26	2a	2e	3a	1e	28	3c	2b	3b	2f	3f	39	19	2d	3d

B Experiment to Verify the Correctness of FTDDT

In this section, we conducted an experiment to calculate the probability of propagation to verify the correctness of FTDDT. For several random-chosen fixed-input differences δ_{in}, we tested the transfer probability from a single difference δ_{in} (shown in the left-hand side of Eq. 4) to a set of differences Δ_{out} (shown in the right-hand side of Eq. 4) in the S-box used in SPEEDY.

More precisely, we randomly choose 2^{20} 42-bit plaintexts p and denote the number of p that satisfies $\{\text{S-box}(p) \oplus \text{S-box}(p \oplus \delta_{in}) \in \Delta_{out}\}$ as $\mathcal{N}$. Then the probability obtained from this test is $\mathcal{N}/2^{20}$.

$$\begin{aligned}
010000 &\xrightarrow{\text{SB}} **0*** \\
010000 &\xrightarrow{\text{SB}} ****00 \\
010000 &\xrightarrow{\text{SB}} 0***** \\
001000 &\xrightarrow{\text{SB}} 00**** \\
001000 &\xrightarrow{\text{SB}} ****00 \\
000100 &\xrightarrow{\text{SB}} ***0** \\
000100 &\xrightarrow{\text{SB}} 0**0**
\end{aligned} \tag{4}$$

Our validation algorithm is written in python code and we run it on an Intel(R) Core(TM) i7-8700 CPU. Finally, the tested probability is $2^{-5.9671}$ while the transfer probability caculated from FTDDT is $2^{-5.9622}$. Therefore, we consider that the accuracy obtained by FTDDT is verified.

C Inverse Probability of FTDDT

Theorem 1. When considering a fixed difference $\delta_{out} \preceq \Delta_{out}$ propagate to δ_{in} which is the inverse of $\mathsf{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]$, the average probability of this transition is

$$\mathsf{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]/2^{N+N_a}.$$

Proof. Since the S-box is bijective, the inverse probability of FTDDT is equal to the average probability of $\delta_{in} \xrightarrow{\mathsf{SB}} \delta_{out}$ (averaged over all $\delta_{out} \in \Delta_{out}$). Next, remember that the FTDDT describes the probability of a group of differential transition. Thus, with this probability, we can obtain the average probability of $\delta_{in} \xrightarrow{\mathsf{SB}} \delta_{out}, \delta_{out} \preceq \Delta_{out}$ by dividing by the size of the output differential set $|\Delta_{out}|$. The probability of the case $\delta_{in} \xrightarrow{\mathsf{SB}} \Delta_{out}$ is $\mathsf{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]/2^N$ and the number of difference in Δ_{out} is 2^{N_a}. Thus, the probability of the case $\delta_{in} \xrightarrow{\mathsf{SB}} \delta_{out}$ is $\mathsf{FTDDT}[\delta_{in}][\mathcal{B}_{\Delta_{out}}]/2^{n+N_a}$ on average if $\delta_{out} \in \Delta_{out}$. □

D Searching Linear Distinguishers Against SPEEDY

Since the MC operation can be seen as $\Gamma_{in} = \mathsf{M}^{\mathsf{T}} \cdot \Gamma_{out}$ in linear attack where Γ_{in} is the input mask and Γ_{out} is the output mask of matrix M. Following the method in Sect. 3.2, we get the linear mask transition properties for MC, precisely, the correspondence between the input mask Γ_{in} and the output mask Γ_{out} of the minimum HW.

For linear cryptanalysis, we use a similar strategy used for differential search, which is a model oriented toward key recovery. Then, we search for a five-round distinguisher with two rounds of key recovery by following four steps.

Step 1: We found the 4-round distinguishers from S_3^0 to S_0^5 using the same method in searching differentials.

Step 2: For each of the 4-round linear distinguishers, we found the distinguishers with two rounds of key recovery from S_0^5 to S_2^6 according to the following conditions: the probability from S_0^5 to S_0^6 is as high as possible and after spreading from S_0^6 to S_2^6 with probability one, the active S-box number in S_2^6 is less than 32.

Step 3: Then, the linear mask propagates from S_3^0 to S_1^0 with probability one and limits the number of active S-box at S_1^0 is less than 32.

Step 4: Finally, we consider the linear rotation invariant of the 7-round distinguishers combined with key recovery and choose the case where the number of bits of k_7 deduced from k_0 by linear key schedule is maximized.

Following step one to four, we found a 5-round distinguisher combined two key recovery rounds with correlation $2^{-93.01}$. We show the detail of the linear distinguisher in Appendix I. In this distinguisher, the bits of k_7 which is derived from k_0 by linear key schedule is 139. We reduce the guessed key bits to 185 bits in the key recovery phase.

E Searching Differential-Linear Distinguishers Against SPEEDY

We show the four rounds differential-linear search model in Fig. 4. More specifically, we search for three rounds of differential-linear distinguishers from S_4^0 to S_0^4 and extend half a round forward and backward for key recovery. Detailed strategy is as follows.

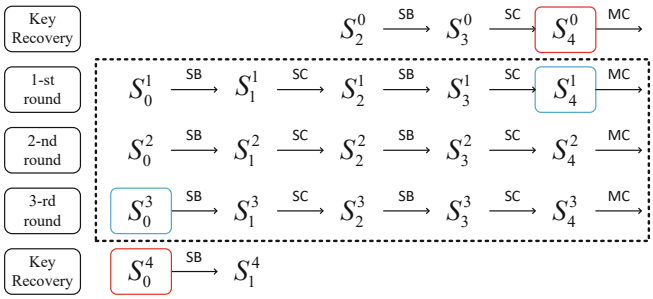

Fig. 4. 4-round differential-linear attack model of SPEEDY.

Step 1: We search the differential part contains key recovery rounds from S_3^0 to S_4^1, the search strategies are as follows.

- **Rule 1:** Constrain the active bit numbers in S_3^0 less than 64.
- **Rule 2:** Model $S_3^0 \xrightarrow{\mathsf{SB}^{-1}} S_2^0$ to comply with FTDDT.
- **Rule 3:** Model $S_0^1 \xrightarrow{\mathsf{SB}} S_1^1, S_2^1 \xrightarrow{\mathsf{SB}} S_3^1$ to comply with DDT.
- **Rule 4:** Constrain only one bit active in S_4^1.

Step 2: Meanwhile, we search the linear part contains key recovery rounds from S_0^3 to S_0^4, the search strategies are as follows.

- **Rule 1:** Constrain only one bit active in S_0^3.
- **Rule 2:** Model $S_0^3 \xrightarrow{\mathsf{SB}} S_1^3, S_2^3 \xrightarrow{\mathsf{SB}} S_3^3$ to comply with LAT (short for linear approximation table).
- **Rule 2:** Model the active S-box in S_0^4 to comply with a maximum correlation of $2^{-1.415}$.

Step 3: We search the connect round from S_4^1 to S_0^3 as follows.

- **Rule 1:** Constrain only one bit active in S_4^1 and S_0^3.
- **Rule 2:** Model $S_0^2 \xrightarrow{\mathsf{SB}} S_1^2$ with backward FTDDT.
- **Rule 3:** Model $S_2^2 \xrightarrow{\mathsf{SB}} S_3^2$ to comply with LAT.
- **Rule 4:** Constrain the difference of the i-th row in S_2^2 is zero if the difference of corresponding row in S_2^2 is active.

Step 4: From Step one to three, we get several distinguishers. Then, we choose the one with the highest correlation and run an experiment for obtaining the estimated correlation from S_2^1 to S_1^3. After testing 128 random keys with 2^{30} plaintexts under a certain key, we obtained the correlation estimated as $2^{-8.85}$.

Step 5: Finally, we consider the rotation invariant property of SPEEDY such that the bits of k_7 deduced from k_0 by linear key schedule are maximized.

Finally, the distinguisher is shown in Fig. 5. The correlation of this 3-round distinguisher is $2^{-23.095}$ and the probability of rounds of key-recovery is $2^{-6.972}$. The bits of k_7 deduced from k_0 by linear key schedule are 29.

F 4-Round Differential-Linear Attack Against SPEEDY

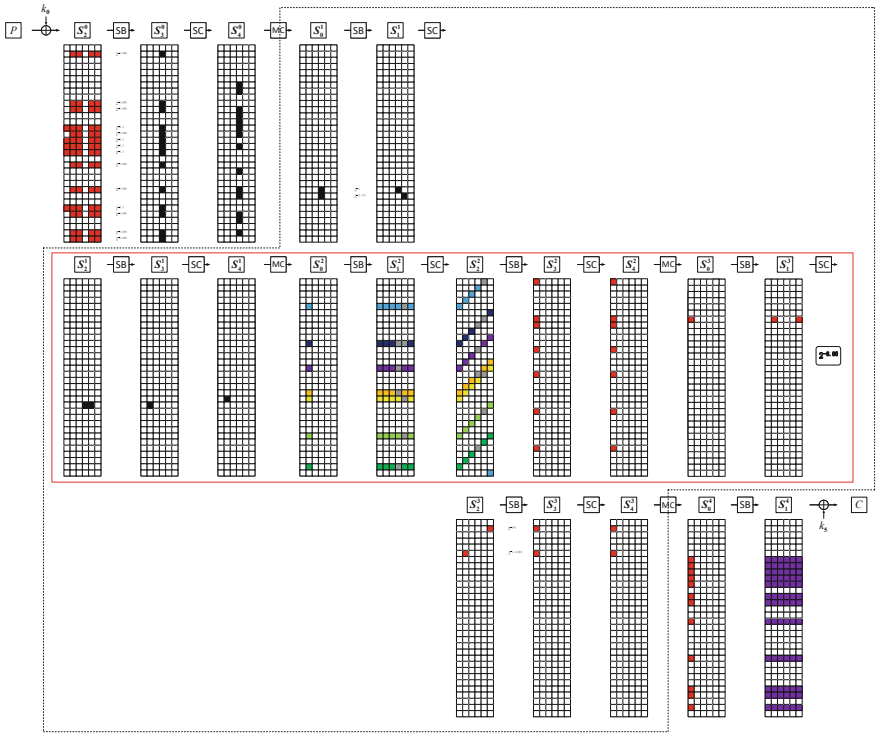

Fig. 5. Key recovery attack on 4-round SPEEDY with differential-linear cryptanalysis.

With the 3-round differential-linear distinguisher shown in Fig. 5, we can implement the differential-linear attack against 4-round SPEEDY. Firstly, we choose the active bits in plaintext to make a 61-bit structure and obtain corresponding

ciphertexts. After guessing the 61-bit k_0, we use Algorithm 5 to put the pairs that can validate the input difference of the 3-round differential-linear distinguisher into the set $\mathcal{S}$. Then we recover the master key by applying the FWHT (short for Fast Walsh-Hadamard Transform) which has already been used in [3].

Since the time complexity of the Algorithm 5 is negligible compared to those of FWHT, we use the same computing method for our attack. Then, we choose the advantage $a = 88$, use one structure to collect data, the time complexity of this attack is $2^{61} + 2^{104} + 2^{104} \approx 2^{105}$. The data complexity is 2^{61} and the memory required is $2^{61} + 2^{72} + 2^{105} \approx 2^{105}$. We denote $|\mathcal{S}|$ as the number of pairs in $\mathcal{S}$, according to [17] the distribution of $|\mathcal{S}|$ is

$$|\mathcal{S}| \sim \frac{1}{2} N(2^{61} \cdot 2^{-6.972}, 2^{61} \cdot 2^{-6.972}) = N(2^{53.028}, 2^{52.028}).$$

Then, the success rate is

$$\begin{aligned} P_S &= \int_0^{+\infty} P(|\mathcal{S}| = x) \cdot P_{S'}(|\mathcal{S}| = x) dx \\ &\geq P(|\mathcal{S}| \geq 2^{53.027}) \cdot P_{S'}(|\mathcal{S}| = 2^{53.027}) \\ &\approx 45.31\%, \end{aligned}$$

where $P_{S'}(|\mathcal{S}| = n)$ is the success rate for this attack, if $|\mathcal{S}| = n$.

Algorithm 5: Sieving pairs for differential-linear attack

Input: $\mathcal{P}$; // The set of plaintexts
Input: $\mathcal{C}$; // The set of ciphertexts indexed by plaintexts
Input: k; // The guessed 61-bit k_0
Input: δ_{in}; // The begin of differential-linear distinguisher
Output: $\mathcal{S}$; // The set of pairs can encrypt to the begin of differential-linear distinguisher by k

1 $\mathcal{S} \leftarrow \emptyset$;
2 **for** $p \in \mathcal{P}$ **do**
3 $\quad p' \leftarrow D(E(p \oplus k) \oplus \delta_{in}) \oplus k$;
$\quad$ /* $E(\cdot)$ is the function encrypt plaintext to the begin of distinguisher */
$\quad$ /* $D(\cdot)$ is the decrypt function of $E(\cdot)$ */
4 $\quad c \leftarrow \mathcal{C}[p]$;
5 $\quad c' \leftarrow \mathcal{C}[p']$;
6 $\quad \mathcal{S} \xleftarrow{\cup} ((p, c), (p', c'))$;
7 Delete duplicate pairs in $\mathcal{S}$;
8 **return** $\mathcal{S}$;

G Pseudocode of Key Recovery and Brute Force Parts on the 2^{155} Pairs for Each Structure in the 7-Round Differential Attack

Algorithm 6: Deduce key from the data set $\mathcal{P}$

Input: $\mathcal{P}$; // The set of 2^{155} pairs, each pair consist of two plaintexts with their ciphertexts
Input: δ_{out}; // The output difference of the 5.5-round distinguisher
Input: δ_{in}^{i}; // The input difference of the i-th row of 5.5-round distinguisher
Input: (m_t, c_t); // c_t is the ciphertext of m_t. At the end of this algorithm, we use it to test if the candidate keys are the right key.
Output: $\mathcal{K}$; // The set of possible full master key

$\mathcal{K} \leftarrow \emptyset$;
for $((m_0, c_0), (m_1, c_1)) \in \mathcal{P}$ **do**
  $m_0^0||...||m_0^{31} \leftarrow$ DividePlain(m_0);
  $m_1^0||...||m_1^{31} \leftarrow$ DividePlain(m_1);
  /* DividePlain(m_i) is use to divide plaintext m_i into 32 row, m_i^j is the 6-bit j-th row of m_i */
  $\mathcal{K}^7 \leftarrow$ Deduce the 36-bit k_7 by $m_0 \oplus m_1$ and δ_{out} according to DDT; // $|\mathcal{K}^7| = 4$ on average for each pair, details are shown in Appendix H.
  for $k^7 \in \mathcal{K}^7$ **do**
    Deduce the k_0 from k^7 due to the linear key schedule;
    For i-th row of k_0, fix the deduced bits and traverse other bits to get set $\mathcal{K}_i^0, i = 0, \cdots, 26, 29, 30, 31$;
    $\mathcal{L}_{14} \leftarrow$ FirstSboxSieve$(m_0^{14}, m_1^{14}, \mathcal{K}_{14}^0$,0b*00*00);
    $\mathcal{L}_{15} \leftarrow$ FirstSboxSieve$(m_0^{15}, m_1^{15}, \mathcal{K}_{15}^0$,0b**00*0);
    $\mathcal{L}_{17} \leftarrow$ FirstSboxSieve$(m_0^{17}, m_1^{17}, \mathcal{K}_{17}^0$,0b00**00);
    $\mathcal{L}_{11} \leftarrow$ FirstSboxSieve$(m_0^{11}, m_1^{11}, \mathcal{K}_{11}^0$,0b*000*0);
    $\mathcal{L}_{12} \leftarrow$ FirstSboxSieve$(m_0^{12}, m_1^{12}, \mathcal{K}_{12}^0$,0b0*000*);
    $\mathcal{L}_{13} \leftarrow$ FirstSboxSieve$(m_0^{13}, m_1^{13}, \mathcal{K}_{13}^0$,0b******);
    $\mathcal{L}_{16} \leftarrow$ FirstSboxSieve$(m_0^{16}, m_1^{16}, \mathcal{K}_{16}^0$,0b0**00*);
    $\mathcal{L} \leftarrow$ SecondSboxSieve$([\mathcal{L}_{11}, \mathcal{L}_{12}, \mathcal{L}_{13}, \mathcal{L}_{14}, \mathcal{L}_{15}, \mathcal{L}_{16}], \delta_{in}^{11})$;
    $\mathcal{L}_{18} \leftarrow$ FirstSboxSieve$(m_0^{18}, m_1^{18}, \mathcal{K}_{18}^0$,0b*00**0);
    $\mathcal{L}_{19} \leftarrow$ FirstSboxSieve$(m_0^{19}, m_1^{19}, \mathcal{K}_{19}^0$,0b**000*);
    $\mathcal{L} \leftarrow$ SecondSboxSieve$([\mathcal{L}, \mathcal{L}_{17}, \mathcal{L}_{18}, \mathcal{L}_{19}], \delta_{in}^{14})$;
    $\mathcal{L}_{20} \leftarrow$ FirstSboxSieve$(m_0^{20}, m_1^{20}, \mathcal{K}_{20}^0$,0b0**00*);
    $\mathcal{L} \leftarrow$ SecondSboxSieve$([\mathcal{L}, \mathcal{L}_{20}], \delta_{in}^{15})$;

22

23

24 $\mathcal{L}_{21} \leftarrow$ FirstSboxSieve($m_0^{21}, m_1^{21}, \mathcal{K}_{21}^0$,0b*0**00);

25 $\mathcal{L}_{22} \leftarrow$ FirstSboxSieve($m_0^{22}, m_1^{22}, \mathcal{K}_{22}^0$,0b0*0**0);

26 $\mathcal{L}_{23} \leftarrow$ FirstSboxSieve($m_0^{23}, m_1^{23}, \mathcal{K}_{23}^0$,0b00*0**);

27 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}, \mathcal{L}_{21}, \mathcal{L}_{22}, \mathcal{L}_{23}], \delta_{in}^{18}$);

28 $\mathcal{L}_{24} \leftarrow$ FirstSboxSieve($m_0^{24}, m_1^{24}, \mathcal{K}_{24}^0$,0b000*0*);

29 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}, \mathcal{L}_{24}], \delta_{in}^{19}$);

30 $\mathcal{L}_7 \leftarrow$ FirstSboxSieve($m_0^7, m_1^7, \mathcal{K}_7^0$,0b*0***);

31 $\mathcal{L}_8 \leftarrow$ FirstSboxSieve($m_0^8, m_1^8, \mathcal{K}_8^0$,0b0*0***);

32 $\mathcal{L}_9 \leftarrow$ FirstSboxSieve($m_0^9, m_1^9, \mathcal{K}_9^0$,0b00*0**);

33 $\mathcal{L}_{10} \leftarrow$ FirstSboxSieve($m_0^{10}, m_1^{10}, \mathcal{K}_{10}^0$,0b000*0*);

34 $\mathcal{L} \leftarrow$SecondSboxSieve($[L_7, L_8, L_9, L_{10}, \mathcal{L}], \delta_{in}^7$);

35 $\mathcal{L}_5 \leftarrow$ FirstSboxSieve($m_0^5, m_1^5, \mathcal{K}_5^0$,0b*****0);

36 $\mathcal{L}_6 \leftarrow$ FirstSboxSieve($m_0^6, m_1^6, \mathcal{K}_6^0$,0b0*****);

37 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_5, \mathcal{L}_6, \mathcal{L}], \delta_{in}^5$);

38 $\mathcal{L}_4 \leftarrow$ FirstSboxSieve($m_0^4, m_1^4, \mathcal{K}_4^0$,0b****00);

39 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_4, \mathcal{L}], \delta_{in}^4$);

40 $\mathcal{L}_3 \leftarrow$ FirstSboxSieve($m_0^3, m_1^3, \mathcal{K}_3^0$,0b***00*);

41 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_3, \mathcal{L}], \delta_{in}^3$);

42 $\mathcal{L}_2 \leftarrow$ FirstSboxSieve($m_0^2, m_1^2, \mathcal{K}_2^0$,0b**00**);

43 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_2, \mathcal{L}], \delta_{in}^2$);

44 $\mathcal{L}_1 \leftarrow$ FirstSboxSieve($m_0^1, m_1^1, \mathcal{K}_1^0$,0b*00**0);

45 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_1, \mathcal{L}], \delta_{in}^1$);

46 $\mathcal{L}_{30} \leftarrow$ FirstSboxSieve($m_0^{30}, m_1^{30}, \mathcal{K}_{30}^0$,0b**0000);

47 $\mathcal{L}_{31} \leftarrow$ FirstSboxSieve($m_0^{31}, m_1^{31}, \mathcal{K}_{31}^0$,0b0**000);

48 $\mathcal{L}_0 \leftarrow$ FirstSboxSieve($m_0^0, m_1^0, \mathcal{K}_0^0$,0b00**00);

49 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_{30}, \mathcal{L}_{31}, \mathcal{L}_0, \mathcal{L}], \delta_{in}^{30}$);

50 $\mathcal{L}_{29} \leftarrow$ FirstSboxSieve($m_0^{29}, m_1^{29}, \mathcal{K}_{29}^0$,0b******);

51 $\mathcal{L} \leftarrow$SecondSboxSieve($[\mathcal{L}_{29}, \mathcal{L}], \delta_{in}^{29}$);

52 $\mathcal{L}_{25} \leftarrow$ FirstSboxSieve($m_0^{25}, m_1^{25}, \mathcal{K}_{25}^0$,0b******);

53 $\mathcal{L}_{26} \leftarrow$ FirstSboxSieve($m_0^{26}, m_1^{26}, \mathcal{K}_{26}^0$,0b******);

54 $\mathcal{L} \leftarrow$SecondSboxSieve($\mathcal{L}, \mathcal{L}_{25}, \mathcal{L}_{26}], \mathcal{L}_{21}$);

55 **for** $k' \in \mathcal{L}$ **do**

56 **for** k'' *traverse unguessed 12-bit keys* **do**

57 Combine k' and k'' to get the 192-bit keys k;

58 **if** $E(m_t, k) == c_t$ **then**

59 $\mathcal{K} \xleftarrow{\cup} k$;

60 **return** $\mathcal{K}$ as the set of candidate keys;

H Details for the Time Complexity of the Key Recovery on the 2^{155} Pairs for Each Structure in the 7-Round Differential Attack

In this section, we mainly consider the time of operating the Algorithm 6 in Appendix G. In order to make it easier to understand, we show the details of each step in Algorithm 6 in Table 4.

Position in Table 4 denotes the location in Algorithm 6, e.g., position 9 denotes the operation that generates $\mathcal{L}_{14}$ at line 9 in Algorithm 6. The filtering probability of FirstSboxSieve is determined by the number of inactive bits in the output of the S-box, e.g., when generating the $\mathcal{L}_{14}$, the output different set of the S-box is `0b*00*00` so that the filtering probability is 2^{-4} while the filtering probability of SecondSboxSieve is calculated by the inverse probability of FTDDT.

Besides, to facilitate the calculation of the time complexity, we see the 2^{155} pair of each structure as a group. Then, we compute the time complexity of each operation in Algorithm 6 in this group. More precisely, we introduce two new parameters, i.e., the number of the remaining pairs and the number of remaining keys for each pair. Further, for an operation with filtering probability p, the number of the remaining pairs and remaining keys before this operation is N_p and N_k, respectively. Besides, we use N_a to denote the number of imported keys in this operation. For the FisrtSboxSieve operation, I_k is the number of keys in the input key set, e.g., $|\mathcal{K}_{14}^0|$ in position 9. For the SecondSboxSieve operation, I_a is zero. Then, the complexity of this operation for the whole group in this operation, the remaining pairs and the remaining keys after this operation is calculated by Algorithm 7.

In fact, the sum of the time complexity in Table 4 is about 68.8×2^{155} times SubBox operation. Since there are other operations that have not been evaluated, we give a generous estimate for the complexity of the key recovery phase, i.e., 70×2^{155} times SubBox operation.

Evaluation the Number of Elements in $\mathcal{K}^7$. In this section, we use $S_j^i[k,:]$ to denote the k-th line in the S_j^i, i.e., the $6k$-th bit to $(6k+5)$-th bit of S_j^i. As shown in Fig. 3, there are only two active differences transitioning to a set of differences through the process $S_0^6 \xrightarrow{\mathsf{SB}} S_1^6$, i.e., the $S_0^6[3,:] \xrightarrow{\mathsf{SB}} S_1^6[3,:]$ ($\mathtt{0b000010} \xrightarrow{\mathsf{SB}} \mathtt{0b*0*000}$) and $S_0^6[6,:] \xrightarrow{\mathsf{SB}} S_1^6[6,:]$ ($\mathtt{0b000010} \xrightarrow{\mathsf{SB}} \mathtt{0b000*0*}$). Other active differences propagate to a single difference.

Therefore, when deducing the key bits of k_7 by DDT (position 5 in Algorithm 6), on average, there is only one 6-bit partial key will be deduced for the states $k_7[i,:]$, where $i \in \{7, 2, 16, 25\}$ since the corresponding difference in S_2^6 is a specific single difference. However, $k_7[1,:]$ and $k_7[3,:]$ will have several possibilities since the corresponding state $S_2^6[1,:]$ and $S_2^6[3,:]$ are sets of differences. Thus, the average number of elements in $\mathcal{K}^7$ is determined by the average number of deduced $k_7[1,:]$ and $k_7[3,:]$.

Algorithm 7: Compute parameters in Table 4.

Input: I_k: the number of imported keys in this operation.
Input: N_p: the number of remaining pairs before this operation.
Input: N_k: the number of remaining keys before this operation.
Input: p: the number of remaining keys before this operation.
Output: T: time complexity of this operation.
Output: N_p: the number of remaining pairs after this operation.
Output: N_k: the number of remaining keys after this operation.

if This operation is FirstSboxSieve **then**
  $T \leftarrow 2 \times I_k \times N_p$;
  `// Each pair has two plaintexts so that need two SubBox operation for an imported key.`
  if $I_k \times p \geq 1$ **then**
    $N_k \leftarrow N_k \times I_k \times p$;
    $N_p \leftarrow N_p$;
  if $I_k \times p < 1$ **then**
    $N_k \leftarrow N_k$;
    $N_p \leftarrow N_p \times I_k \times p$;
if This operation is SecondSboxSieve **then**
  $T \leftarrow 2 \times N_k \times N_p$;
  if $N_k \times p \geq 1$ **then**
    $N_k \leftarrow N_k \times p$;
    $N_p \leftarrow N_p$;
  if $N_k \times p < 1$ **then**
    $N_k \leftarrow 1$;
    $N_p \leftarrow N_p \times N_k \times p$;
return T, N_p, N_k;

Note that the state $S_2^6[1,:]$ and $S_2^6[3,:]$ are propagated by $S_1^6[3,:]$ and $S_1^6[6,:]$ while the state $S_1^6[i,:]$ is propagated by $S_0^6[i,:]$ through the S-box. Thus, we did an in-depth study of the propagation $S_0^6[i,:] \xrightarrow{\mathsf{SB}} S_0^6[i,:]$, where $i \in \{3,6\}$.

- In our trail, the $S_0^6[3,:] \xrightarrow{\mathsf{SB}} S_0^6[3,:]$ will have two cases since the propagation $\mathtt{0b000010} \xrightarrow{\mathsf{SB}} \mathtt{0b*0*000}$ have two possible propagations, i.e., $\mathtt{0b000010} \xrightarrow{\mathsf{SB}} \mathtt{0b100000}$ and $\mathtt{0b000010} \xrightarrow{\mathsf{SB}} \mathtt{0b001000}$ (indeed, there are four cases for this propagation, but other two cases have zero probability).
- For the same reason, the $S_0^6[6,:] \xrightarrow{\mathsf{SB}} S_0^6[6,:]$ also has two cases.

Above all, there are $2 \times 2 = 4$ possible cases for the propagation of the two active S-boxes and each case will deduce one possible 12-bit keys for k_7. Besides, each case needs two Subox operations. Thus, the average number of elements in $\mathcal{K}^7$ is and the time complexity of deducing 36-bit k_7 is $4 \times 2 + 4 = 12$ times SubBox operation.

Table 4. Details for the Time Complexity of Algorithm 6. The set parameter is the output of the operation, e.g., $\mathcal{L}_{14}$ for position 9. The time complexity is measured by the SubBox operation. I_k is the number of imported keys in this operation. N_p and N_k are the number of the remaining pairs and remaining keys after the operation calculated by Algorithm 7.

Position	set	I_k	Time complexity	Filtering probability	N_p	N_k
5	$\mathcal{K}^7$	–	12×2^{155}	1	2^{155}	4
9	$\mathcal{L}_{14}$	2^2	2^{158}	2^{-4}	2^{155}	1
10	$\mathcal{L}_{15}$	2^2	2^{158}	2^{-3}	2^{154}	1
11	$\mathcal{L}_{17}$	2^3	2^{158}	2^{-4}	2^{153}	1
12	$\mathcal{L}_{11}$	2^4	2^{158}	2^{-4}	2^{153}	1
13	$\mathcal{L}_{12}$	2^4	2^{158}	2^{-4}	2^{153}	1
14	$\mathcal{L}_{13}$	2^3	2^{157}	1	2^{153}	2^3
15	$\mathcal{L}_{16}$	2^3	2^{157}	2^{-3}	2^{153}	2^3
16	$\mathcal{L}$	0	2^{157}	$2^{-5.54}$	$2^{150.46}$	1
17	$\mathcal{L}_{18}$	2^4	$2^{155.46}$	2^{-3}	$2^{150.46}$	2
18	$\mathcal{L}_{19}$	2^4	$2^{155.46}$	2^{-2}	$2^{150.46}$	2^3
19	$\mathcal{L}$	0	$2^{154.46}$	2^{-6}	$2^{147.46}$	1
20	$\mathcal{L}_{20}$	2^4	$2^{152.46}$	2^{-3}	$2^{147.46}$	2
21	$\mathcal{L}$	0	$2^{149.46}$	2^{-6}	$2^{142.46}$	1
24	$\mathcal{L}_{21}$	2^5	$2^{148.46}$	2^{-3}	$2^{142.46}$	2^2
25	$\mathcal{L}_{22}$	2^6	$2^{149.46}$	2^{-3}	$2^{142.46}$	2^5
26	$\mathcal{L}_{23}$	2^6	$2^{149.46}$	2^{-3}	$2^{142.46}$	2^8
27	$\mathcal{L}$	0	$2^{151.46}$	2^{-6}	$2^{142.46}$	2^2
28	$\mathcal{L}_{24}$	2^6	$2^{149.46}$	2^{-4}	$2^{142.46}$	2^4
29	$\mathcal{L}$	0	$2^{147.46}$	2^{-6}	$2^{140.46}$	1
30	$\mathcal{L}_7$	2^5	$2^{146.46}$	2^{-1}	$2^{140.46}$	2^4
31	$\mathcal{L}_8$	2^6	$2^{147.46}$	2^{-2}	$2^{140.46}$	2^8
32	$\mathcal{L}_9$	2^5	$2^{146.46}$	2^{-3}	$2^{140.46}$	2^{10}

(*continued*)

Table 4. (*continued*)

Position	set	I_k	Time complexity	Filtering probability	N_p	N_k
33	$\mathcal{L}_{10}$	2^5	$2^{146.46}$	2^{-4}	$2^{140.46}$	2^{11}
34	$\mathcal{L}$	0	$2^{152.46}$	2^{-6}	$2^{140.46}$	2^5
35	$\mathcal{L}_5$	2^5	$2^{146.46}$	2^{-1}	$2^{140.46}$	2^9
36	$\mathcal{L}_6$	2^5	$2^{146.46}$	2^{-1}	$2^{140.46}$	2^{13}
37	$\mathcal{L}$	0	$2^{154.46}$	2^{-6}	$2^{140.46}$	2^7
38	$\mathcal{L}_4$	2^5	$2^{146.46}$	2^{-2}	$2^{140.46}$	2^{10}
39	$\mathcal{L}$	0	$2^{151.46}$	2^{-6}	$2^{140.46}$	2^4
40	$\mathcal{L}_3$	2^5	$2^{146.46}$	2^{-2}	$2^{140.46}$	2^7
41	$\mathcal{L}$	0	$2^{148.46}$	2^{-6}	$2^{140.46}$	2
42	$\mathcal{L}_2$	2^6	$2^{147.46}$	2^{-2}	$2^{140.46}$	2^5
43	$\mathcal{L}$	0	$2^{146.46}$	2^{-6}	$2^{139.46}$	1
44	$\mathcal{L}_1$	2^5	$2^{145.46}$	2^{-3}	$2^{139.46}$	2^2
45	$\mathcal{L}$	0	$2^{142.46}$	2^{-6}	$2^{135.46}$	1
46	$\mathcal{L}_{30}$	2^6	$2^{142.46}$	2^{-4}	$2^{135.46}$	2^2
47	$\mathcal{L}_{31}$	2^6	$2^{142.46}$	2^{-4}	$2^{135.46}$	2^4
48	$\mathcal{L}_0$	2^6	$2^{142.46}$	2^{-4}	$2^{135.46}$	2^6
49	$\mathcal{L}$	0	$2^{142.46}$	2^{-6}	$2^{135.46}$	1
50	$\mathcal{L}_{29}$	2^6	$2^{142.46}$	1	$2^{135.46}$	2^6
51	$\mathcal{L}$	0	$2^{142.46}$	$2^{-5.83}$	$2^{135.46}$	$2^{0.17}$
52	$\mathcal{L}_{25}$	2^6	$2^{142.46}$	1	$2^{135.46}$	$2^{6.17}$
53	$\mathcal{L}_{26}$	2^6	$2^{142.46}$	1	$2^{135.46}$	$2^{12.17}$
54	$\mathcal{L}$	0	$2^{148.63}$	$2^{-5.67}$	$2^{135.46}$	$2^{6.5}$

I Key Recovery Attack Against 7-Round SPEEDY with Linear Cryptanalysis

(See Fig. 6).

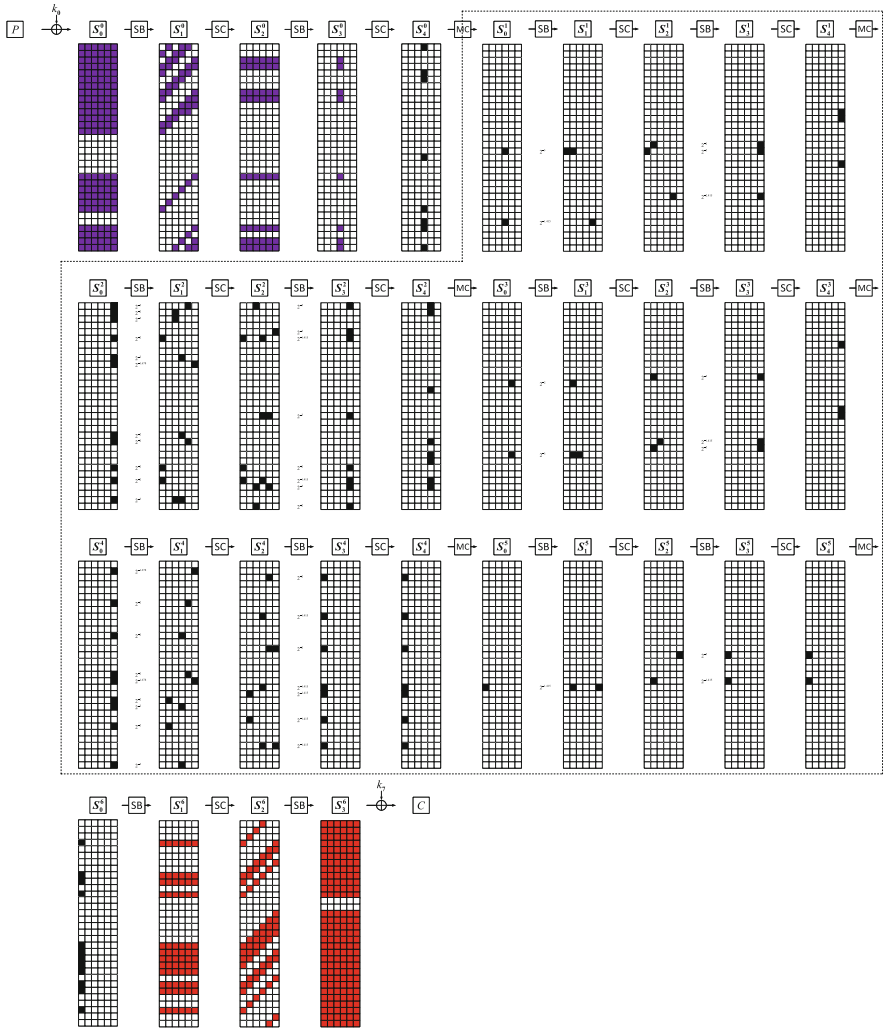

Fig. 6. Key recovery attack on 7-round SPEEDY with linear cryptanalysis.

J Key Recovery Attack against 4-Round SPEEDY with Differential Cryptanalysis

(See Fig. 7).

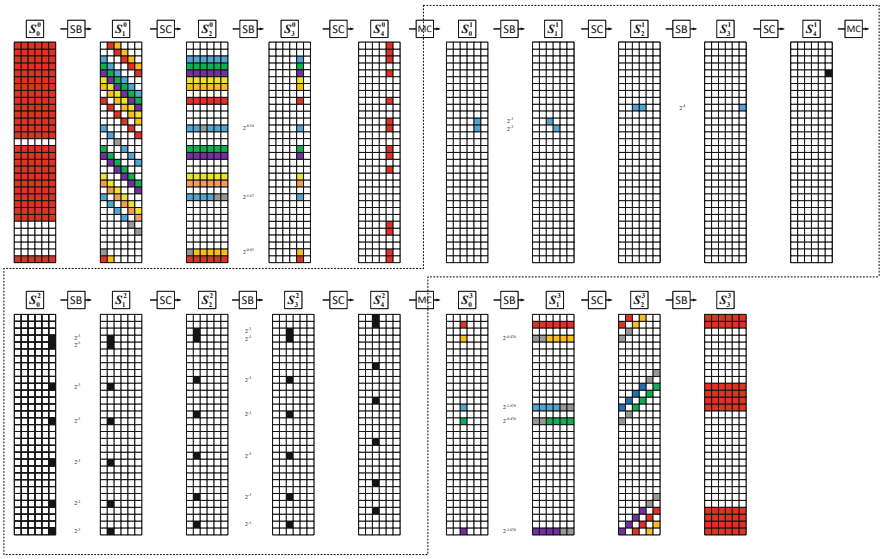

Fig. 7. Key recovery attack on 4-round SPEEDY with differential cryptanalysis.

References

1. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: a new tool for differential-linear cryptanalysis. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 313–342. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_11
2. Barrett, C., Tinelli, C.: Satisfiability modulo theories. In: Clarke, E., Henzinger, T., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 305–343. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_11
3. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 329–358. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_12
4. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)
5. Boura, C., David, N., Boissier, R.H., Naya-Plasencia, M.: Better steady than speedy: full break of speedy-7-192. Cryptology ePrint Archive, Paper 2022/1351 (2022)

6. Fan, Y., Li, M., Niu, C., Lu, Z., Wang, M.: Related-tweakey impossible differential attack on reduced-round SKINNY-AEAD M1/M3. In: Galbraith, S.D. (ed.) CT-RSA 2022. LNCS, vol. 13161, pp. 247–271. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95312-6_11
7. Flórez-Gutiérrez, A., Naya-Plasencia, M.: Improving key-recovery in linear attacks: application to 28-round PRESENT. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 221–249. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_9
8. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 17–25. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_3
9. Leander, G., Moos, T., Moradi, A., Rasoolzadeh, S.: The speedy family of block ciphers - engineering an ultra low-latency cipher from gate level for secure processor architectures. Cryptology ePrint Archive, Paper 2021/960 (2021)
10. Li, M., Hu, K., Wang, M.: Related-tweak statistical saturation cryptanalysis and its application on QARMA. Cryptology ePrint Archive (2019)
11. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and chaskey. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 485–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_26
12. Longo, G., Zilli, M.V.: Complexity of theorem-proving procedures: some general properties. Revue française d'automatique inf. recherche opé. Inf. théorique **8**(R3), 5–18 (1974)
13. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33
14. Niu, C., Li, M., Sun, S., Wang, M.: Zero-correlation linear cryptanalysis with equal treatment for plaintexts and tweakeys. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 126–147. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75539-3_6
15. Qin, L., Dong, X., Wang, X., Jia, K., Liu, Y.: Automated search oriented to key recovery on ciphers with linear key schedule: applications to boomerangs in SKINNY and ForkSkinny. IACR Trans. Symmetric Cryptol. **2021**(2), 249–291 (2021)
16. Rohit, R., Sarkar, S.: Cryptanalysis of reduced round speedy. Cryptology ePrint Archive (2022)
17. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. J. Cryptol. **21**(1), 131–147 (2008)
18. Zhou, C., Zhang, W., Ding, T., Xiang, Z.: Improving the MILP-based security evaluation algorithm against differential/linear cryptanalysis using a divide-and-conquer approach. Cryptology ePrint Archive (2019)