SPEEDY: Caught at Last

Christina Boura[1], Patrick Derbez[2], Baptiste Germon[2], Rachelle Heim Boissier[3], María Naya-Plasencia[4]

[1] IRIF, Université Paris Cité, France
christina.boura@irif.fr
[2] Univ Rennes, Inria, CNRS, IRISA, France
{patrick.derbez,baptiste.germon}@inria.fr
[3] UCLouvain, ICTEAM/ELEN/Crypto Group, Belgium
rachelle.heim@uclouvain.be
[4] Inria Paris, France
maria.naya-plasencia@inria.fr

Abstract. SPEEDY is a family of ultra-low-latency block ciphers designed by Leander et al. in 2021. In 2023, Boura et al. proposed a differential attack on the full 7-round variant, SPEEDY-7-192. However, shortly thereafter, Beyne and Neyt demonstrated that this attack was invalid, as the dominant differential characteristic it relied upon had probability zero. A similar issue affects another differential attack proposed the same year by Wang et al., which also targets SPEEDY-7-192 and suffers from the same flaw. As a result, although SPEEDY-7-192 was initially believed to be broken, it remained unbroken in practice, and the question of finding a valid attack on this cipher remained an open problem. In this work, we resolve this problem by presenting the first valid differential attack on SPEEDY-7-192. We verify the validity of our distinguisher using the quasidifferential framework. Moreover, our search for the differential distinguisher is significantly more rigorous than in the previous works, allowing us to explore a larger portion of the search space. We also fully exploit probabilistic extensions of the distinguisher to identify optimal parameters for the key recovery step. Our attack on SPEEDY-7-192 has data and time complexities of $2^{186.36}$ encryption calls and a memory complexity of 2^{84} 192-bit states. In addition, we present differential attacks on 4-round SPEEDY-5-192 and 5-round SPEEDY-6-192 which currently represent the best attacks against these smaller variants.

Keywords: SPEEDY · low-latency · differential cryptanalysis · quasidifferential trails

1 Introduction

The design of low-latency ciphers has been an active area of research in recent years, with many symmetric ciphers having been proposed, for example, PRINCE [11], MANTIS [4], QARMA [2] and QARMAv2 [3], BipBip [5], SCARF [17], Sonic [6], and Aradi [24], to cite just a few. Among them, the SPEEDY family of

block ciphers [27], introduced by Leander *et al.* in 2021, proposed an elegant design following the substitution-permutation network (SPN) construction, where all components were specifically chosen to ensure low latency in hardware implementations. A central element of this design was a new 6-bit S-box, optimized to achieve minimal hardware latency while maintaining good cryptographic properties.

As is customary, the authors provided, alongside the specifications and design rationale, a preliminary security analysis to demonstrate that `SPEEDY` resists a range of classical cryptanalytic techniques, such as differential, linear, higher-order differential, integral, and cube attacks. This analysis allowed the designers to determine the number of rounds for each variant of the `SPEEDY` family, explicitly choosing to leave only a very thin security margin in order not to needlessly hinder performance. More precisely, their analysis concluded that no distinguisher on the primitive could be extended by more than one round to result in a successful key-recovery attack. Based on this, they determined that a two-round security margin beyond the best-known distinguisher was sufficient to ensure the security of the design.

Among the attacks considered by the authors of `SPEEDY` to evaluate the security of their design was differential cryptanalysis [9], one of the oldest and most powerful techniques against block ciphers. This method exploits pairs of plaintexts with a fixed input difference such that after a certain number of rounds, an output difference occurs more frequently than expected under random behavior. All new block cipher proposals should include design arguments demonstrating resistance to this classical attack. The authors of `SPEEDY` provided such arguments by carefully analyzing the cryptographic properties of the S-box, selecting the branch number of the linear layer to ensure fast diffusion of differences, and estimating the minimum number of differentially active S-boxes over several rounds. They were able to estimate that the expected differential probability of a characteristic (EDPC) is upper bounded by 2^{-78}, 2^{-120}, 2^{-150}, and 2^{-192} for 3, 4, 5, and 6 rounds, respectively. This final bound, combined with the 192-bit block size of all variants and the designers' conviction that no distinguisher can be extended beyond one additional round, led the authors to conclude that the ciphers family is resistant to differential cryptanalysis.

Despite this reassuring analysis, Boura *et al.* presented in 2023 a differential attack against `SPEEDY-7-192`, the main variant of the family intended to provide 192-bit security, with a time complexity of $2^{187.84}$ [12]. They exploited, in particular, the presence of several high-probability 1-bit to 1-bit differentials in the `SPEEDY` S-box, as well as the limited diffusion of the linear layer, which only operates within columns, to derive a differential covering 5.5 rounds. The authors then exploited this differential to mount a successful key-recovery attack on the full 7-round variant. This result, in particular, invalidated the designers' claim that no distinguisher could be extended beyond one round.

However, a year later, Beyne and Neyt demonstrated in a short note [7] that the main characteristic used in [12] to break `SPEEDY-7-192` was actually invalid. More precisely, using the theory of quasidifferential trails [8], they showed that

the dominant characteristic had a probability of zero due to a local contradiction at the round level. This contradiction arises from the absence of a key addition between the two applications of the SB layer within a round.

Note also that a few months after the publication of the attack in [12], a separate work by Wang *et al.* [32] analyzed the security of SPEEDY against various statistical attacks. In particular, they presented a differential cryptanalysis of the full version of SPEEDY-7-192. While Beyne and Neyt focused on analyzing the validity of the distinguisher from [12], we verified that the dominant trail used in the differential attack of [32] also has probability zero, thereby invalidating this attack against SPEEDY-7-192 as well. Additionally, they proposed a 4-round differential attack against SPEEDY-5-192 and we have verified that the 2-round characteristic on which it relies is indeed valid. Finally, the same work also presented a linear cryptanalysis of SPEEDY-7-192. Although the linear distinguisher is valid, the associated key recovery attack is not. This is detailed in Section 2.4.

Given the invalidity of all known attacks against SPEEDY-7-192, this variant remains unbroken. In their note, Beyne and Neyt raised the open question of whether other valid, high-probability differential characteristics could be discovered for SPEEDY, and whether a valid differential attack could eventually be mounted against the full version of SPEEDY-7-192. They stated this as an interesting open problem for future research.

Our contributions. In this work, we present the first valid attack against the full version of SPEEDY-7-192. It is a differential attack, having both time and data complexities of $2^{186.36}$. In addition to being valid, the attack was discovered using a more systematic and optimized search method than those employed in previous works. Its main features are:

- We employ a new approach for the distinguisher search. Although our method is also based on the precomputation of one-round trails, as in [12], our strategy for generating these trails allows for significantly fewer constraints on their shape. This enables the exploration of a much more diversified portion of the search space.
- We use the theory of quasidifferential trails [8] to rigorously verify the validity of the one-round trails. Additionally, we exploit the differential dependencies between the two successive S-box layers. These dependencies actually *increase* the probabilities of our distinguishers, yielding up to a 3-bit gain on the distinguisher used in the attack targeting SPEEDY-7-192.
- We use dynamic programming to construct the best multi-round differential characteristics while simultaneously performing clustering. Finally, we compute a lower bound on the expected differential probability (EDP) of this differential using the quasidifferential framework extension from [15], and provide strong arguments supporting the validity of our attack for any key.
- We extend the distinguisher probabilistically to both the plaintext and the ciphertext and perform an exhaustive search over all possible such extensions to identify the one that minimizes the overall data and time complexity of

the attack. This step was only partially addressed in [12], where the authors did not explore the full space of possible extensions.

- In contrast to previous attacks on SPEEDY, where the key recovery step was performed manually, we use the automated tool KyRyDi developed by Boura *et al.* [13]. This tool automatically generates an optimal key recovery procedure given a differential distinguisher. We then verify *by hand* the tool's output and provide a detailed description of this step.

We apply the same methodology to derive attacks on round-reduced versions of the two other main variants in the SPEEDY family: SPEEDY-5-192 and SPEEDY-6-192. These variants are actually more challenging to attack, as their security claims impose stricter constraints on the attacker compared to the main variant. For instance, any attack on SPEEDY-5-192 must not use more than 2^{64} plaintexts or ciphertexts and the data limit is 2^{128} for SPEEDY-6-192. More precisely, we present a 4-round attack on SPEEDY-5-192 and a 5-round attack on SPEEDY-6-192. Both attacks respect the designers' stated security claims and, to the best of our knowledge, currently represent the best known results on these variants as well.

A summary of our attacks, along with all previously known attacks on SPEEDY that respect the stated security claims, is provided in Table 1. In this table, we mark as flawed the attacks that we, or others, have demonstrated to be incorrect. For differential attacks, invalidity comes from flaws in the distinguisher, while in the case of the linear attack from [32], the issue lies in the key recovery phase, as we explain later in this paper.

Version Security Claim ($\mathcal{D}/\mathcal{T}$)	Rounds	Type	Data	Time	Memory	Flaw Discovered	Source
SPEEDY-5-192 ($2^{64}/2^{128}$)	3	Integral	$2^{17.6}$	$2^{52.5}$	$2^{25.5}$	—	[30]
	4	Diff.	2^{61}	$2^{119.69}$	2^{83}	—	[32]
	4	Diff.-Linear	2^{61}	2^{105}	2^{105}	—	[32]
	4	Diff.	$2^{63.59}$	$2^{66.18}$	$2^{63.59}$	—	**This work**
SPEEDY-6-192 ($2^{128}/2^{128}$)	5.5	Diff.	$2^{121.65}$	$2^{127.8}$	2^{42}	✓	[12]
	5	Diff.	$2^{97.93}$	$2^{97.93}$	2^{96}	—	**This work**
SPEEDY-7-192 ($2^{192}/2^{192}$)	7	Diff.	$2^{186.53}$	$2^{187.39}$	2^{156} / 2^{36}	✓	[32]
	7	Linear	$2^{188.5}$	$2^{189.41}$	$2^{185.04}$	✓	[32]
	7	Diff.	$2^{187.28}$	$2^{187.84}$	2^{42}	✓	[12]
	7	Diff.	$2^{186.36}$	$2^{186.36}$	2^{84}	—	**This work**

Table 1: Overview of all known attacks, both from prior work and this paper, on the three variants of SPEEDY.

The rest of the article is organized as follows. Section 2 introduces the preliminary notions required for our work. In particular, Sections 2.1 and 2.2 provide a brief overview of differential cryptanalysis; Section 2.3 details the specifications of the SPEEDY family of block ciphers; and Section 2.4 summarizes both

the attack from [12] and the remark presented in [7]. Section 3 presents our methodology for identifying the best and valid differential characteristics. Section 4 describes the key-recovery phase of our attack against the full version of `SPEEDY-7-192`. Finally, Section 5 describes our attacks against `SPEEDY-5-192` and `SPEEDY-6-192`.

We provide our source code at:

`https://anonymous.4open.science/r/speedy_caught_at_last-61D6/`

2 Preliminaries

2.1 Differential distinguishers

Differential cryptanalysis was introduced in 1990 by Biham and Shamir [9,10]. To distinguish a permutation $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ from a random permutation, this technique exploits the existence of high-probability *differentials*, that is, ordered pairs of differences $(\Delta_{in}, \Delta_{out}) \in (\mathbb{F}_2^n)^* \times (\mathbb{F}_2^n)^*$ such that the differential probability

$$\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) \coloneqq \frac{|\{X \in \mathbb{F}_2^n : f(X + \Delta_{in}) + f(X) = \Delta_{out}\}|}{2^n}$$

is significantly higher than 2^{-n}. Since most primitives are iterated, a classical technique to find differentials on a few rounds consists in chaining shorter (usually one-round) differentials to create differential *characteristics* or *trails*, that is, ordered vectors of $r_\Delta + 1$ intermediate differences $(\Delta_0, \Delta_1, \ldots, \Delta_{r_\Delta})$. The differential probability $\mathrm{DPC}_f(\Delta_0, \Delta_1, \ldots, \Delta_{r_\Delta})$ of a characteristic $(\Delta_0, \Delta_1, \ldots, \Delta_{r_\Delta})$ is then defined similarly to that of a differential, that is, as the proportion of X's such that the pair $(X, X + \Delta_0)$ propagates to a difference Δ_i after i applications of the round function. It then holds that

$$\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) = \sum_{\Delta_1, \ldots, \Delta_{r_\Delta - 1} \in \mathbb{F}_2^n} \mathrm{DPC}_f(\Delta_{in}, \Delta_1, \ldots, \Delta_{r_\Delta - 1}, \Delta_{out}) \,. \quad (1)$$

In the case of a keyed primitive $f : \mathbb{F}_2^\kappa \times \mathbb{F}_2^n \to \mathbb{F}_2^n, (k, X) \mapsto f(k, X) = f_k(X)$, a metric often used to evaluate the efficiency of a differential (resp. differential characteristic) is its expectancy, usually referred to as the *expected differential probability* EDP (resp. EDPC). This quantity is defined as

$$\mathrm{EDP}_f(\Delta_{in}, \Delta_{out}) \coloneqq \frac{1}{2^\kappa} \sum_{k \in \mathbb{F}_2^\kappa} \mathrm{DP}_{f_k}(\Delta_{in}, \Delta_{out}) \,.$$

Using Equation (1), we can show the EDP is related to the EDPC as follows:

$$\mathrm{EDP}_f(\Delta_{in}, \Delta_{out}) = \sum_{(\Delta_1, \ldots, \Delta_{r_\Delta - 1})} \mathrm{EDPC}_f(\Delta_{in}, \Delta_1, \ldots, \Delta_{r_\Delta - 1}, \Delta_{out}) \,. \quad (2)$$

A common assumption in differential cryptanalysis is that the differential probability for a fixed, randomly chosen key k is close to its average over all keys. This assumption, known as the *hypothesis of stochastic equivalence* [26], is not always valid in practice. Moreover, computing the fixed-key probability of a differential characteristic is generally difficult. For the so-called key-alternating block ciphers, where a round key is XORed with the state between each application of the round function, Lai, Massey, and Murphy showed in [26] that, if the round keys $k_1, \ldots, k_{r_\Delta}$ are drawn independently and uniformly at random, then the probability of a differential characteristic $(\Delta_0, \ldots, \Delta_{r_\Delta})$ satisfies

$$\mathrm{DPC}_{f_k}(\Delta_0, \ldots, \Delta_{r_\Delta}) = \prod_{i=1}^{r_\Delta} \mathrm{DP}_{R_{k_i}}(\Delta_{i-1}, \Delta_i)\,.$$

where R_{k_i} denotes the application of the round function R with the round key k_i. However, in practice, round keys are not chosen uniformly at random but are instead derived deterministically from a master key k via a key schedule algorithm. This can significantly affect the fixed-key probability of a characteristic, as demonstrated, for example, in [8,25,29].

In 2022, Beyne and Rijmen [8] presented the *quasidifferential framework* which provided for the first time an exact formula for the DPC. They introduced the concept of quasidifferential trails that are sequences of so-called mask-difference pairs $(u_0, \Delta_0), \ldots, (u_{r_\Delta}, \Delta_{r_\Delta})$. For each transition from one mask-difference pair to another through a round, a correlation coefficient can be computed. The correlation of the whole quasidifferential trail is then defined as the product of the correlations for each round. Intuitively, a quasidifferential trail represents a linear approximation over the valid pairs of plaintexts that follow the characteristic. In particular, the existence of quasidifferential trails with high absolute correlation indicates that certain intermediate values of valid input/output pairs are strongly linearly related, either positively or negatively correlated. Beyne and Rijmen [8] showed that the probability of a differential characteristic corresponds to the sum of the correlations of all quasidifferential trails that share the same intermediate differences.

Theorem 1 ([8]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function such that $f = f_{r_\Delta - 1} \circ \cdots \circ f_0$. The probability of a characteristic $(\Delta_0, \ldots, \Delta_{r_\Delta})$ is given by:*

$$\mathrm{DPC}_f(\Delta_0, \ldots, \Delta_{r_\Delta}) = \sum_{u_1, \ldots, u_{r_\Delta - 1}} \mathrm{Corr}\big((u_1, \Delta_1), \ldots, (u_{r_\Delta - 1}, \Delta_{r_\Delta - 1})\big)$$

with $u_0 = u_{r_\Delta} = 0$ and $\mathrm{Corr}\big((u_1, \Delta_1), \ldots, (u_{r_\Delta - 1}, \Delta_{r_\Delta - 1})\big)$ is the correlation of the quasidifferential trail $\big((u_0, \Delta_0), (u_1, \Delta_1), \ldots, (u_{r_\Delta - 1}, \Delta_{r_\Delta - 1}), (u_{r_\Delta}, \Delta_{r_\Delta})\big)$.

Their work was further extended by Boura, Derbez and Germon [15] to provide an exact formula for the EDPC. In the original framework, the DPC formula given by Theorem 1 corresponds to a function parametrized by the key. In contrast, in [15], the key is directly integrated into the quasidifferential trails, allowing the EDPC to be computed as the sum of the correlations of these trails.

Based on their experiments, the EDPC seems to be easier to compute than the DPC and can already detect fully impossible characteristics. During our search for distinguishers we used both the DPC and EDPC computations to guarantee that our differentials are valid (see Section 3).

2.2 Key recovery in differential cryptanalysis

Differential distinguishers on a reduced number of rounds of a block cipher E can be used to mount key recovery attacks. We let $(\Delta_{in}, \Delta_{out})$ be a differential of probability 2^{-p}, $p > 0$, over r_Δ rounds. The difference Δ_{in} can be propagated with probability $2^{-p_{in}}$ to a truncated difference in a affine subset $D_{in} \subset \mathbb{F}_2^n$ of dimension d_{in}, r_{in} rounds before. We similarly define p_{out}, D_{out}, d_{out} and r_{out} when propagating Δ_{out} to the ciphertext. Usually $p_{in} = p_{out} = 0$ but several works (e.g. [16,31,1]) have showed that probabilistic extensions of differential distinguishers can lead to much better key-recovery attacks and our new attacks on `SPEEDY` rely on such extensions. We let $\mathcal{K}_{in}$ (resp. $\mathcal{K}_{out}$) denote the set of key material candidates involved in the computation of Δ_{in} (resp. Δ_{out}), and define $\mathcal{K} = \mathcal{K}_{in} \cup \mathcal{K}_{out}$. Finally, let $p' = p + p_{in} + p_{out}$. Given a tuple $((P, C = E_k(P)), (P', C' = E_k(P')), \mathbf{k})$ such that $P \oplus P' \in D_{in}$, $C \oplus C' \in D_{out}$, $\mathbf{k} \in \mathcal{K}$, $\mathbf{k}$ is selected as a valid candidate for the key material if it partially encrypts to Δ_{in} (P, P') and decrypts (C, C') to Δ_{out}. Without loss of generality, we describe the attack in the encryption direction.

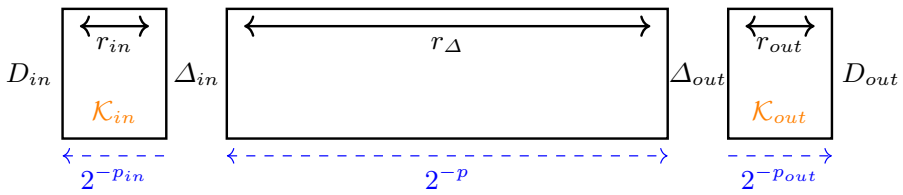


Fig. 1: Key recovery in differential cryptanalysis

Pair generation and data complexity. To generate and identify a valid plaintext P, valid in the sense that the corresponding input X to the distinguisher after r_{in} rounds satisfies that the pair $(X, X \oplus \Delta_{in})$ is mapped to a pair (Y, Y') with difference Δ_{out}, the attacker typically relies on *structures*. A structure is built by fixing a value on the $n - d_{in}$ non-active bits and considering all possible values in the remaining d_{in} bits. Structures are important because if a structure contains a plaintext P that belongs to a valid pair (P, P'), then necessarily P' also belongs to it. From a single structure, it is possible to generate approximately $\binom{2^{d_{in}}}{2} \approx 2^{2d_{in}-1}$ pairs. Hence, the number of chosen plaintexts

required to generate 2^x pairs following the inner differential i.e., the data complexity $\mathcal{D}$, is given by the following well-known formula [23], which distinguishes between two cases:

1. If $(x + p' + 1) \geq d_{in}$, then the data complexity is $\mathcal{D} = 2^{x+p'+1}$. This is the most common case in differential cryptanalysis [13], where a single structure does not permit to generate enough valid pairs for the attack.
2. If $(x+p'+1) < d_{in}$, then $\mathcal{D} = 2^{\frac{x+p'+d_{in}+1}{2}}$. In this case, one structure produces more pairs than necessary, and it is sufficient to query only a fraction of it.

Generating all the pairs from a structure is most often unnecessary since many of them cannot follow the differential, namely, those whose ciphertext difference does not lie in D_{out}. Thus, the attacker stores each plaintext/ciphertext pair in a hash table indexed by the value of the $n - d_{out}$ inactive bits on the ciphertext side. Only plaintexts with the same index can form a pair that may follow the differential, and only such *candidate pairs* are considered by the attacker. The number of candidate pairs is $N = 2^{p'+d_{in}+d_{out}-n}$.

Key recovery step. The *key recovery step* aims to associate each of the N candidate pairs with the round keys bits that allow it to be partially encrypted and decrypted to the input and output differences Δ_{in} and Δ_{out} of the distinguisher. If the recovered round key bits do not fully determine the master key, an additional exhaustive search over the remaining key bits might be required. There is no generic formula for this step as it is highly non-trivial. This step has recently been studied and formalized in [13], in the context of block ciphers whose linear layer consists of a bit-permutation. In that work, the authors introduced the tool `KyRyDi`, which automatically computes an efficient key recovery procedure for such ciphers. Although the round function of `SPEEDY` includes a `MixColumns` operation and therefore does not qualify as a cipher with a bit-permutation linear layer, this operation is not applied in the rounds involved in the key recovery step which allowed us to apply the tool nonetheless.

Remarks. It is worth noting that probabilistic extensions do not make the complexities formula more complicated than for deterministic extensions, in particular regarding the data complexity and the generation of candidate pairs. While it may affect the complexity of the key-recovery step, for all the attacks described in Sections 4 and 5 this step is never the bottleneck. We therefore would like to highlight how relevant are probabilistic extensions and we believe that many existing differential attacks could be improved using this technique.

2.3 The `SPEEDY` family of block ciphers

`SPEEDY` is an ultra-low-latency family of block ciphers designed by Leander *et al.* in 2021 [27]. Each instance in the family is denoted `SPEEDY-r-`n, where `r` is the number of rounds and n is the block size, with n being a multiple of 6.

While various instantiations are possible, the designers put forward three main variants: `SPEEDY-5-192`, `SPEEDY-6-192`, and `SPEEDY-7-192`, all operating on 192-bit blocks and using a 192-bit key. These variants differ only in the number of rounds and the corresponding security level. Notably, the security claims vary between the three versions, as discussed later.

The 192-bit internal state of `SPEEDY` is represented as a 32×6 rectangle of bits denoted by $x_{[i][j]}$, where $0 \leq i < 32$ and $0 \leq j < 6$. The index i refers to the row of the state, while j indicates the column. The bit (or word) at position 0 is considered the most significant. Additionally, note that computations within a row are performed modulo 6, whereas computations within a column are performed modulo 32.

We now describe the round function and the key schedule algorithm.

Round function The internal state is first initialized with the 192-bit plaintext. Then, the round function $\mathcal{R}_r$ is applied to the state `r` times, where `r` equals 5, 6, or 7 depending on the variant. Each round consists of the `AddRoundKey` operation, which XORs the round key into the state, followed by the non-linear transformation `SubBox`, the linear transformation `ShiftColumns`, a second application of `SubBox` and `ShiftColumns`, and then the linear `MixColumns` operation. The round concludes with the `AddRoundConstant` step, which, as the name suggests, XORs a round constant into the state. This structure is depicted in Figure 2.

All rounds follow this structure, except for the final round, in which the second `ShiftColumns`, the `MixColumns`, and the `AddRoundConstant` operations are omitted. Finally, a whitening subkey is XORed with the state before producing the ciphertext.

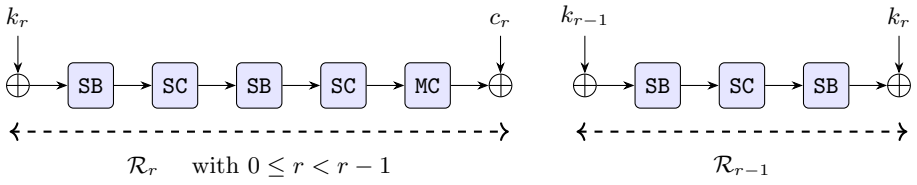


Fig. 2: Illustration of the `SPEEDY-r-192` round function: the left side shows the round structure of the first `r` $-$ 1 rounds, while the right side depicts the last round.

In the following, we assume that each operation takes as input a vector x and outputs a vector y, both from $\mathbb{F}_2^{32\times 6}$.

- `AddRoundKey (A`$_{k_r}$`)`: In this step, the 192-bit round key k_r is XORed into the internal state:

$$y_{[i,j]} = x_{[i,j]} \oplus k_{r[i,j]}, \quad \text{for all } 0 \le i < 32, 0 \le j < 6.$$

- `SubBox (SB)`: The same 6-bit S-box S is applied in parallel to each row of the state. More precisely,

$$(y_{[i,0]}, y_{[i,1]}, y_{[i,2]}, y_{[i,3]}, y_{[i,4]}, y_{[i,5]}) = S(x_{[i,0]}, x_{[i,1]}, x_{[i,2]}, x_{[i,3]}, x_{[i,4]}, x_{[i,5]}),$$

  for each row $0 \le i < 32$ of the state. This S-box was specifically designed for `SPEEDY`, with low-latency implementation as a primary criterion. Its table representation is provided in Table 4 of Supplementary Material A.1.
- `ShiftColumns (SC)`: Each column j of the state is rotated upwards by j positions. Formally, for each row $0 \le i < 32$ and each column $0 \le j < 6$, the output is defined as:

$$y_{[i,j]} = x_{[i+j,j]}.$$

- `MixColumns (MC)`: The `MC` operation in `SPEEDY` is applied column-wise and consists of multiplying each column of the state by a binary matrix. More precisely, each output bit is computed by XORing seven input bits from the same column:

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

  where all additions on the row index i are taken modulo 32.
- `AddRoundConstant (A`$_{c_r}$`)`: In this step, a 192-bit round constant c_r is XORed into the internal state. Formally,

$$y_{[i,j]} = x_{[i,j]} \oplus c_{r[i,j]}.$$

  Since the specific values of the constants are not relevant for our analysis, we omit their detailed description.

Key schedule The key schedule of `SPEEDY` is identical for all variants and is based on the repeated application of a fixed bit-permutation, denoted `PB`. All round keys are viewed as 32×6 bit matrices. The initial round key k_0 is set to the master key. Subsequent round keys are generated by applying `PB` to the previous round key:

$$k_{r+1} := \texttt{PB}(k_r), \quad \text{with} \quad k_{r+1[i',j']} = k_{r[i,j]},$$

and the pair (i', j') is defined by a fixed permutation P as $(i', j') := P(i, j)$. The definition of the permutation P is provided in Table 5 of Supplementary Material A.2.

2.4 Invalidity of previous attacks on SPEEDY [12,32]

In 2023, Boura *et al.* presented a differential attack on the full version of the SPEEDY-7-192 variant [12]. Improvements on this attack, along with a linear cryptanalysis of SPEEDY-7-192, were later proposed by Wang *et al.* in [32]. However, Beyne and Neyt showed in [7] that the first one-round segment of the main differential characteristic used in [12] had probability zero. This flaw arises primarily from the absence of a key addition between the two SB layers within a single round. While they provided a demonstration of this flaw using the quasidifferential framework, we include here an alternative demonstration,[5] which we believe to be slightly more accessible.

Furthermore, we show that the same issue affects the dominant differential characteristic used in the attack of [32] against SPEEDY-7-192, rendering that attack invalid as well. Finally, we demonstrate that the linear attack proposed in [32] against SPEEDY-7-192 is also flawed.

On the validity of the differential characteristics in [12] and [32]. Let us consider the characteristic over $\texttt{SB} \circ \texttt{SC} \circ \texttt{SB}$ shown in Figure 3, which forms part of the first one-round trail in the dominant characteristic of [12]. In the first SB, rows 4 and 5 must satisfy the differential $(\texttt{0,0,0,0,0,1}) \rightarrow (\texttt{0,1,0,0,0,0})$. The set of output values $y \in \mathbb{F}_2^6$ satisfying

$$S^{-1}(y) \oplus S^{-1}(y \oplus (\texttt{0,1,0,0,0,0})) = (\texttt{0,0,0,0,0,1})$$

(where S is the S-box) is of the form $y = \texttt{(1,*,*,*,*,1)}$. Thus, bit $(4, 0)$ at the input of SC has to be 1 in both states of the pair. Since SC does not modify column 0, bit $(4, 0)$ must also be 1 at its output. For the second SB, row 4 must meet the differential transition $(\texttt{0,1,0,0,0,0}) \rightarrow (\texttt{0,0,0,1,0,0})$. All solutions for the input x of this S-box are of the form $x = \texttt{(0,*,1,1,*,*)}$, requiring bit $(4, 0)$ to be 0 at the input of SC on both elements of the pair. Both conditions are contradictory, so the characteristic is impossible and has probability 0, as illustrated in Figure 3. Consequently, the characteristics in Figures 4 and 7 of [12] are invalid and so are the 5.5-round attack against SPEEDY-6-192 and the 7-round attack against SPEEDY-7-192. A similar incompatibility appears in the second round of the characteristic shown in Figure 3 of [32], which is used in their attack against SPEEDY-7-192. For completeness, we provide an explanation of this incompatibility in Supplementary Material B; however, it is essentially identical to the one discussed in this section.

On the validity of the linear attacks from [32]. The authors of [32] present, in the appendix of their paper, a 5-round linear distinguisher with correlation $2^{-93.014}$. To extend this distinguisher into a key-recovery attack, they claim to use the formulas from [19], applying the FFT or FWT speed-up. Unlike the

[5] Our demonstration is a direct consequence of their analysis; we do not claim this as a novel contribution.

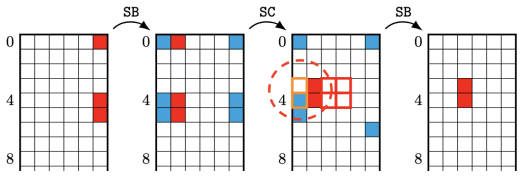


Fig. 3: An impossible characteristic over $\texttt{SB} \circ \texttt{SC} \circ \texttt{SB}$. White bits are inactive, red bits are active. Regarding constraints imposed by the first S-box application, the bits that must be equal to 1 in value because of the first S-box layer are blue. The bits that must be equal to 1 (resp. 0) in value on both elements of the pair because of the second S-box layer are framed in red (resp. orange).

previously discussed attack, where the distinguisher itself was invalid, the flaw here lies in the key-recovery step, as we detail below.

In Section 3.2 of [19], the authors discuss how key-schedule relations can be taken into account to optimize the FFT/FWT-based speed-up techniques. They consider a setting involving four round keys: two external keys, k_0 and k_3, which are XORed directly with the plaintext and ciphertext, and two internal keys, k_1 and k_2, which are applied to internal states. Notably, Formula (34) shows that the dominant term in the complexity expression, typically the final one, can be reduced by a factor of $2^{l_{1,2}}$, where $l_{1,2}$ represents the number of bits shared between the internal keys k_1 and k_2. However, no such reduction applies to external keys like the k_0 and k_3, since it is not possible to exploit in FFT-based techniques relations between the bits of the external keys [6]. Despite this, the authors of [32] divide their complexity by a quantity they denote as $l_{0,7}$, corresponding to the number of bits shared between the two external keys. As a result, the complexity formula they apply is incorrect. When the methodology from [19] is correctly applied, the actual complexity of their attack exceeds 2^{260}.

3 Finding good differential distinguishers

In this paper, we significantly improve the approach used in [12] to find a differential distinguisher over r_Δ rounds using a variety of advanced techniques. Our approach is divided into four steps, each described in a dedicated subsection. After describing the method used in [12] in Section 3.1, we explain how we searched for one-round differential trails under more relaxed constraints in Section 3.2. Next, we explain in Section 3.3 how we applied the quasidifferential framework to filter out all invalid one-round trails. This filtering step primarily removes potential incompatibilities within a round, as those discussed in Section 2.4. We then introduce in Section 3.4 our dynamic programming-based algorithm to

[6] It would be possible to slightly reduce this complexity using puncturing techniques, such as those proposed in [18,20,21]. However, even with such techniques, the overall complexity would remain significantly higher than that of exhaustive search.

chain one-round trails and to identify the best multi-round differential characteristics. Finally, Section 3.5 provides several strong arguments supporting the validity of the characteristics we use in our attacks. In particular, we analyzed key-dependency effects that may arise across multiple rounds and show that the probabilities of the characteristics that contribute the most to the differential, are only weakly dependent on the key. This result strongly supports our claim that our attacks hold for all possible keys.

3.1 Approach to find a distinguisher in [12] and comparison with our work

We describe here the approach used in [12] to identify a high-probability differential distinguisher, focusing on the case of `SPEEDY-7-192`, which was the main target of their analysis. The main idea in this work was to precompute all one-round differential characteristics satisfying certain predefined constraints and then to chain them together to construct longer characteristics. For `SPEEDY-7-192`, the authors use a 5.5-round distinguisher, specifically, a distinguisher over the composition $\texttt{SC} \circ \texttt{SB} \circ \mathcal{R}^5$. We borrow the notation used in this work to represent the intermediate states after each step of the round function. Specifically, `st[0]` denotes the input state before the `MC` operation, `st[1]` the state after applying `MC` to `st[0]`, `st[2]` the state after applying `SB` to `st[1]`, and so on, as follows:

$$\texttt{st[0]} \xrightarrow{\texttt{MC}} \texttt{st[1]} \xrightarrow{\texttt{SB}} \texttt{st[2]} \xrightarrow{\texttt{SC}} \texttt{st[3]} \xrightarrow{\texttt{SB}} \texttt{st[4]} \xrightarrow{\texttt{SC}} \texttt{st[5]} \xrightarrow{\texttt{MC}} \texttt{st[6]}.$$

The authors of [12] exhaustively enumerated all possible input/output pairs (`st[0]`, `st[6]`) that satisfy the following constraints:

1. `st[0]` contains a single active column c_0 such that both c_0 and $M(c_0)$ have at most seven active rows, where M denotes the matrix used in the `MC` operation;
2. `st[5]` contains a single active column c_5 such that both c_5 and $M(c_5)$ have at most seven active rows;
3. `st[2]` contains at most two active bits per row;
4. the probability of the differential characteristic (`st[0]`, `st[6]`) is strictly greater than 2^{-49}.

They obtain $1,565,536$ one-round trails satisfying those constraints. These one-round characteristics are then chained to obtain characteristics over r rounds with $r \leq 4$. In the case of the attack on `SPEEDY-7-192`, $r = 4$.

The 4-round characteristic was then extended to a 5.5-round characteristic by prepending a one-round trail and appending a half-round one. These extensions were performed separately, without an exhaustive exploration of all possible options. There are two main reasons why it was difficult for the authors of [12] to automatically construct a full characteristic. First, the prepended one-round trail does not satisfy the constraints they had fixed, specifically, Condition 1, as it activates more than seven rows in `st[0]`. Given the limited scalability of their one-round trails search algorithm, relaxing the constraints to include a broader

set of trails would have made the search too slow. Second, such relaxation would have significantly increased the number of candidate one-round trails, and their chaining algorithm was not efficient enough to handle chaining this many trails over 5 rounds. In this work, we improve upon both of these algorithms, which allows us to exhaust a larger search space (see Section 3.2).

The authors of [12] also looked for clustering effects by searching for additional characteristics sharing the same input and output differences as the selected 5-round characteristic (excluding the final half-round trail from this search). We do not detail how the authors found such characteristics, but highlight the main difference with our own algorithm: whilst they needed to find new 'ad-hoc' one-round trails and chain them separately, our algorithm does the clustering and the chaining at the same time (see Section 3.4).

Finally, for the 0.5 round (i.e., $\texttt{SC} \circ \texttt{SB}$) extension at the end of the distinguisher, the authors considered a trade-off between a fully deterministic extension—guaranteeing a probability of 1 but potentially activating too many rows in the ciphertext—and a specified characteristic, which activates less rows but has a higher probability cost. This extension was performed in an *ad hoc* manner, disconnected from the final filtering step, and no systematic search for optimal transitions was carried out. In our approach, we observed that to obtain optimal extensions, the first and last rounds should be treated symmetrically, as they play equivalent roles. As a result, it is more effective to restrict the distinguisher to 5 rounds and then search for optimal probabilistic extensions in both directions simultaneously. This allows us to optimise the number of non-active rows on both sides at the same time and to reduce the cost of the filtering step.

As described in Section 4, we fully automate this step, whereas in [12] it was done by hand for a single, pre-selected 5-round characteristic. In our case, the automatisation of this step allowed us to process all the best 5-round characteristics and to find the one that leads to the best extension, rather than doing it by hand for a single pre-selected 5-round characteristic. Interestingly, the best attacks did not originate from the 5-round distinguishers with the highest probability (see Sections 4 and 5).

3.2 Efficient search for one-round differential trails

As described in Section 3.1, the authors of [12] identified differential characteristics by precomputing one-round trails that satisfy certain constraints, and then chained them to build longer characteristics. We adopt a similar methodology but extend the search space while keeping the computation feasible. This is important for a deep understanding of the security of SPEEDY.

For a state $x \in \mathbb{F}_2^{192}$, we denote by $w(x)$ the number of non-zero rows in x. In a first step, we pre-computed the set of states

$$\mathcal{T} = \{x \in \mathbb{F}_2^{192} \quad \text{s.t.} \quad w(\texttt{SC}^{-1} \circ \texttt{MC}^{-1}(x)) + w(x) \leq 14\}\,.$$

We found 1177 such states, where states that are column-wise rotations of each other are counted only once. We then searched for all one-round trails satisfying the following three conditions: $\texttt{st}[1] \in \mathcal{T}$, $\texttt{st}[6] \in \mathcal{T}$ and the estimated

probability of the trail is higher than 2^{-46}.[7] To exhaustively find the one-round trails that satisfy these three conditions, we built a first MILP program that searched directly over a full round. Unfortunately, it did not run in reasonable time. Therefore, we built a second MILP model that takes as input a state $\texttt{st}[1]$ and returns one-round trails that satisfy the last two constraints. To exhaust the full space of one-round trails that satisfy our constraints, we then fed every state in $\mathcal{T}$ as well as every column-wise rotation of these states to our model. This allowed us to highly parallelize the search, as we ran up to 16 MILP models in parallel. We also parallelised the MILP model itself, with a maximum of 8 threads per model.

Our model is less restrictive than the one from [12], except regarding the probability bound. The choice of our constraints results from the following observation. In [12], the authors limit the number of rows to 7 before *and* after $\texttt{MC}$ - and, since they only allow for one active column, before and after $\texttt{MC} \circ \texttt{SC}$. This choice is justified by the fact that fewer active rows result in a higher differential probability. However, whilst their constraint implies that at most 14 S-boxes are active over $\texttt{SB} \circ \texttt{MC} \circ \texttt{SC} \circ \texttt{SB}$, the reverse implication is not true. On the other hand, our constraints are equivalent to limiting the number of active S-boxes to 14 over $\texttt{SB} \circ \texttt{MC} \circ \texttt{SC} \circ \texttt{SB}$. Note that we do not require $\texttt{st}[0]$ and $\texttt{st}[5]$ to have only one active column and that we do not limit the number of active bits per row in $\texttt{st}[2]$. These relaxed constraints were critical: the first and last rounds of the main characteristic used in our attack against `SPEEDY-7-192` (see Figure 5) do not respect the criteria of [12]. However, they came at a small cost, namely, a higher constraint on the probability (2^{-46} instead of 2^{-49} in [12]). We did not manage to lower this probability bound: even though it was highly parallelised, our program ran for close to two weeks. We obtained $37,829,776$ *distinct* one-round trails that satisfied our conditions.

However, we noticed that the first round of the 5.5-round characteristic used in [12] does not satisfy our constraints. Indeed, the state x^* at position $\texttt{st}[1]$ in the first round is such that $w(\texttt{SC}^{-1} \circ \texttt{MC}^{-1}(x^*)) + w(x^*) = 16$. Yet, this first round has very low probability because x^* has a very low number of active rows. Based on this observation, we aimed at adding such one-round trails to our list. More precisely, we defined

$$\mathcal{T}' = \{x \in \mathbb{F}_2^{192} \quad \text{s.t.} \quad w(\texttt{SC}^{-1} \circ \texttt{MC}^{-1}(x)) + w(x) \leq 16\}\,,$$

and used a MILP model to exhaustively search for one-round trails satisfying the following three conditions: $\texttt{st}[1] \in \mathcal{T}'$, $\texttt{st}[6] \in \mathcal{T}'$, and the estimated probability of the trail is greater than 2^{-25}. After adding those one-round trails, we obtained a total of $59,118,080$ *distinct* one-round characteristics. In comparison, [12] identified only $1,565,536$ one-round trails satisfying their constraints. Whilst it turns out that none of these additional one-round trails are part of the dominant characteristic used in our attack against `SPEEDY-7-192` (see Figure 5),

[7] By estimated probability, we refer to the probability computed with the DDT, assuming round-independence.

one of them is used in our attack against SPEEDY-5-192 reduced to 4 rounds (see Figure 11).

3.3 Validity filter using the quasidifferentials

The dependencies on the state values that can arise from the application of two S-box layers within a round can sometimes cause incompatibilities, as illustrated in Section 2.4. In this step, we therefore used the quasidifferential framework to recompute the exact probability of our one-round trails and to remove the ones with probability 0.

Inspired by [15], we developed a MILP model that computes all the quasidifferential trails associated with a given differential characteristic. Since we only considered one-round characteristics at this point, it was practical to run this code on our one-round trails. Furthermore, several trails are equal up to column-wise rotations, which is why we actually ran our filter on representatives of those equivalence classes. This means that we applied our filter on only $1,854,534$ trails instead of $59,118,080$. As the correlation of the quasidifferential trails over one round does not depend on the key, computing the exact probability can simply be done by adding numerically all the correlations of the quasidifferential trails. We also used parallelization to reduce the time taken by this step, and we were able to process all our one-round trails in less than 5 hours. We obtained $40,162,928$ ($1,262,136$ up to rotations) valid differential trails, which means that around one third of the trails actually had probability 0. Furthermore, the exact probability of the remaining ones is on average slightly better than naively computed with the independence hypothesis.

3.4 Efficient chaining and clustering

To compute, for example, a 5-round characteristic, the strategy described in [12] consisted of exhaustively building and storing all 2-round and 3-round characteristics derived from the one-round trails, and then chaining them. Applying the same approach was actually suboptimal and not feasible in our case, as our pool of one-round trails is significantly larger: $40,162,928$ trails compared to $1,565,536$ in [12].

Our improvements are mainly achieved through the use of *dynamic programming*. This technique has previously been applied to find (truncated) characteristics, for example in [22] and [14]. Dynamic programming is naturally suited to this problem: if an r-round differential characteristic $(\Delta_0, \ldots, \Delta_r)$ with final difference Δ_r has maximal probability, then the corresponding $(r-1)$-round characteristic $(\Delta_0, \ldots, \Delta_{r-1})$ must also be optimal for its final difference Δ_{r-1}. Otherwise, if there existed an alternative $(r-1)$-round characteristic $(\Delta'_0, \ldots, \Delta'_{r-2}, \Delta_{r-1})$ with a higher probability, then substituting it would yield an r-round characteristic $(\Delta'_0, \ldots, \Delta'_{r-2}, \Delta_{r-1}, \Delta_r)$ with a higher overall probability, contradicting the optimality of the original.

This initial idea of using dynamic programming to chain trails forms the basis of our algorithm. However, we also aimed to compute the clustered probability

for each pair $(\Delta_{in}, \Delta_{out})$, rather than just for each Δ_{out} to be as accurate as possible when computing the probability of the differentials. In other words, our goal was to design an algorithm that, for each input/output pair $(\Delta_{in}, \Delta_{out})$ over r rounds, computes the sum of the probabilities of all possible characteristics constructed from our one-round trails. To achieve this, we proposed building a map where each index Δ_{out} points to a vector of pairs $(\Delta_{in}, 2^{-p})$ such that:

1. there exists at least one characteristic, built from our one-round trails, with input difference Δ_{in} and output difference Δ_{out};
2. 2^{-p} is the sum of the DPC values of all such characteristics.

Then, to compute clusters over r rounds, one step of our algorithm takes as input:

- a list of one-round trails;
- a map $\texttt{map}_{r-1}$ of $(r-1)$-round trails, constructed as described above.

The algorithm then iterates over each one-round trail and all possible column-wise rotations.[8] Given a one-round trail with input/output difference $(\Delta_{in}, \Delta_{out})$ and probability 2^{-p}, it proceeds as follows. If Δ_{in} appears as an index in the map $\texttt{map}_{r-1}$, i.e., there exists at least one $(r-1)$-round trail ending with the difference Δ_{in}, then for each pair $(\Delta'_{in}, 2^{-p'})$ in the vector associated to this index, the algorithm inserts the pair $(\Delta'_{in}, 2^{-p-p'})$ into a new map $\texttt{map}_r$, under the index Δ_{out}. If a pair with the same input difference Δ'_{in} but different probability $2^{-p'}$ is already present at index Δ_{out}, the pair $(\Delta'_{in}, 2^{-p'})$ is replaced by $(\Delta'_{in}, 2^{-p}+2^{-p'})$ to account for clustering directly.

Best differential characteristics identified. The running time of our program was approximately one hour, and the best differentials we found for 1, 2, 3, 4 and 5 rounds are reported in Table 2. However, we would like to highlight that we did not use any of them to mount an attack against `SPEEDY` as there are better choices. For instance, the 5-round differential used in our attack on `SPEEDY-7-192` (see Figure 5) has a probability of $2^{-174.63}$, about 2^9 times lower than that of the best 5-round characteristic, but leads to a more efficient key-recovery attack, as we will demonstrate in Section 4.

3.5 Validity of our clustered trails

We describe in detail in Section 4 the process by which we select the differentials used in our attack. We dedicated a significant effort to verifying the validity of each selected differential, an aspect that is often overlooked in prior cryptanalysis works. To this end, we reconstructed the full characteristics from our set of one-round trails and checked their validity. Since all one-round trails are valid thanks

[8] Note that two rotations of the same initial state could yield the same state, which could create some artificial clustering. We thus added a condition in our code to avoid this phenomenon.

Rounds	1	2	3	4	5
Probability p	$2^{-9.415}$	$2^{-49.24}$	$2^{-78.58}$	$2^{-120.64}$	$2^{-165.46}$

Table 2: Summary of the best differential probabilities for 1- to 5-round characteristics.

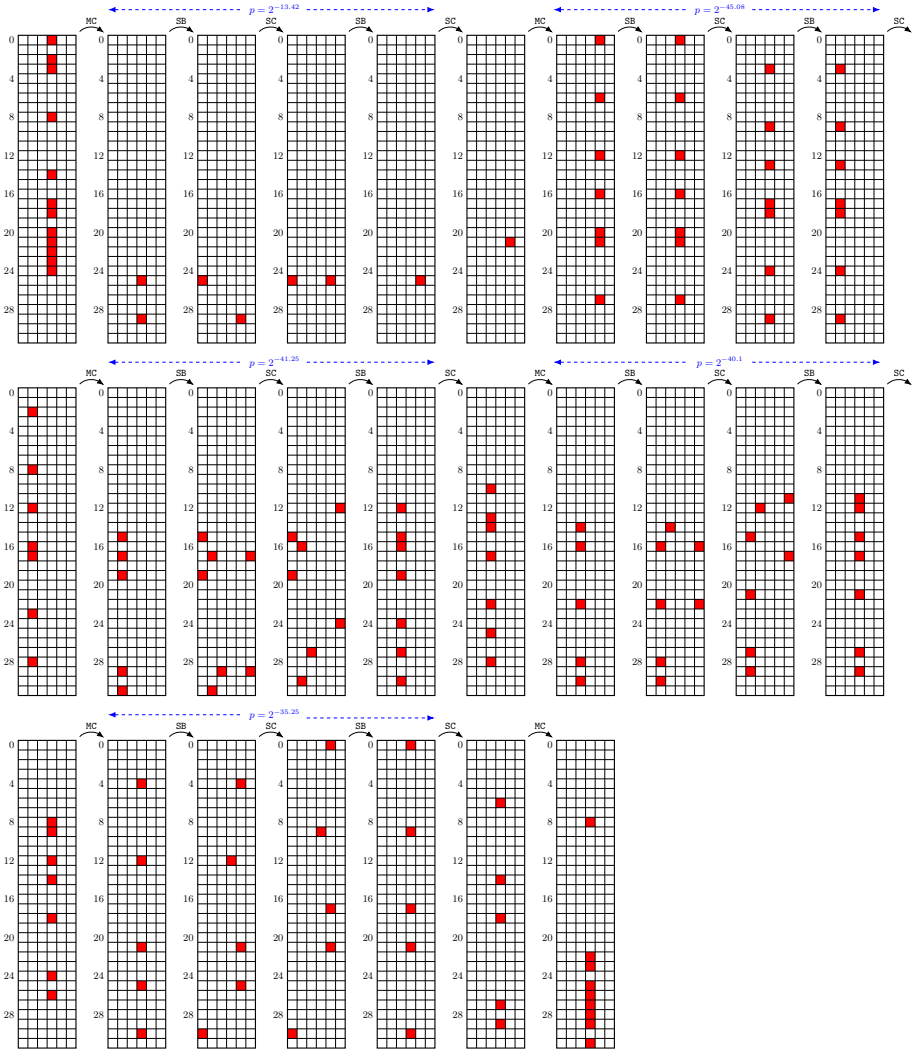


Fig. 4: 5-round dominant trail of the distinguisher used for the attack against SPEEDY-7 of probability $2^{-174.63}$.

to the validity filter described in Section 3.3, we extended our analysis to examine potential key dependencies over multiple rounds. Our analysis was twofold:

1. *Computation of the EDPC.* First, we estimated the value of the expected differential probability of each characteristic.
2. *Analysis of the fixed-key probability.* Then, we evaluated the impact of the key on the differential probability of each characteristic.

Computation of the EDPC. First, we verified that the EDPC of the characteristics is non-zero using the extended quasidifferential framework from [15]. In that work, Boura *et al.* proposed estimating the EDPC of a given characteristic by modeling the block cipher as a single function comprising both the key schedule and the data path. The EDPC can then be computed by finding all the quasi-differential trails over this function. Therefore, we developed another MILP model that searches for those quasidifferential trails and as done in [15], we set a bound on the absolute correlation of the searched trails for our computation to be feasible. More precisely, if 2^{-p} is the estimated probability of the studied characteristic, we lower bounded the absolute correlation by 2^{-p-15}. This bound was selected exclusively for efficiency reasons. We then estimated the EDPC of a given characteristic as the sum of the correlation of such trails. Interestingly, our estimated values of the EDPCs are equal to the product of the DPCs of each round, suggesting that rounds are in fact independent. We ran this model for all the characteristics that we reconstructed and found no impossibility results at this stage.

Analysis of the fixed-key probability. To support the fact that our attack is valid for all the keys, we conducted an analysis of the fixed-key probability of our characteristics. We consider a key-alternating function used with a fixed key k, and denote by k_i the key added at the end of round i. To explain our experiments in more detail, we rely on an informal reformulation of Theorem 4.3 from [8] which states that the differential probability of the characteristic $(\Delta_0, \ldots, \Delta_{r_\Delta})$ is given by:

$$\sum_{u_1,\ldots,u_{r_\Delta-1}} (-1)^{\sum_{i=0}^{r_\Delta-1} u_{i+1}^T k_i} \mathrm{Corr}\big((u_1, \Delta_1), \ldots, (u_{r_\Delta-1}, \Delta_{r_\Delta-1})\big)\,.$$

The sum in this formula can be split into two terms: a first term that does not depend on the key (which corresponds to having all masks u_{i+1} equal to zero), and a term that is key-dependent (when at least one u_{i+1} is non-zero). The first term corresponds to the product of the DPCs of each round and has the same value for all keys. This term can be easily calculated using the exact probabilities of the one-round trails, which we computed as described in Section 3.3. The second term does depend on the key: if, for a given key, this term is positive (resp. negative) the characteristic will have a higher (resp. lower) probability. In particular, the probability for a given key can only be zero if this second term is equal to the opposite of the first term.

We implemented a MILP model to search for trails exhibiting at least one non-zero linear mask before one key addition step (i.e., with at least one non-zero u_{i+1}). To ensure that the model runs in reasonable time, we constrained the search to trails with an absolute correlation of at least $2^{-p-12.5}$. Our results show that such trails always have a correlation below 2^{-p-11} and are extremely rare. In fact, for every characteristic we analysed, we found at most 8 such trails. While this does not entirely rule out the existence of clustering effects with trails of smaller correlation (i.e., below $2^{-p-12.5}$), the rarity of the trails found during our research strongly suggests that the fixed-key correlation variations are negligible and occur only for a marginal subset of keys.

Additional rationale. Whilst it is impossible to draw a definite conclusion due to computational limitations, our experiments strongly indicate that our differentials are valid and that their probability marginally depends on the choice of key. To mitigate this limitation and support the validity of our differential attacks across the entire key space, we present a strong argument based on the sparsity of our characteristics. Indeed, key dependencies can only emerge if all bits involved around the key addition are constrained. More formally, the bit $y_{[i,j]}$ at the input of the first S-box layer can also be expressed as:

$$y_{[i,j]} = x_{[i,j]} \oplus x_{[i+1,j]} \oplus x_{[i+5,j]} \oplus x_{[i+9,j]} \oplus x_{[i+15,j]} \oplus x_{[i+21,j]} \oplus x_{[i+26,j]} \oplus k_{r[i,j]}$$

where x denotes the state before the `MC` operation. Thus, a constraint on $k_{r[i,j]}$ can only arise if all the other bits in the expression are also constrained. Based on this reasoning, we conclude that for the differentials we analysed, the fixed-key probabilities are close to their corresponding EDP values, with at most negligible variation for some keys.

4 Extending the distinguisher and key recovery

After identifying a distinguisher, the next step in a differential attack is to exploit it in a key recovery phase. To do so, the distinguisher is typically extended on both sides, as explained in Section 2.2. In most attacks, this extension is deterministic: the input difference Δ_{in} extends to the plaintext with probability one, and similarly for the output difference Δ_{out} to the ciphertext.

In [12], the authors partially used probabilistic extensions; however, these were selected manually, and their potential was not fully exploited. In this work, we perform an automated search for the best such extensions, as described in the following subsection.

4.1 Probabilistic extensions

The main limitation of a deterministic extension of the distinguisher is that a large part of the plaintext or ciphertext may become active, significantly reducing the effectiveness of pair sieving and leaving the attacker with too many

pairs to begin the key recovery step with. Probabilistic extensions help address this problem: the idea is to constrain specific transitions (or sets of transitions) through the S-box layer, accepting a reduction in probability but maintaining control over the number of inactive rows in both the plaintext and the ciphertext. As shown in Section 2.2, the data complexity as well as the number of candidate pairs for the key recovery depend only on the total number of inactive rows, denoted by $\ell_{in}+\ell_{out}$. This allowed us to automatize the search for the best extension configurations for optimizing the complexities. Given a required number of inactive rows $\ell_{in}+\ell_{out}$, the goal becomes minimizing the combined cost $p_{in}+p_{out}$, where p_{in} and p_{out} are the probabilities of the backward and forward extensions from Δ_{in} to D_{in} and from Δ_{out} to D_{out}, respectively. Importantly, this is independent of the distinguisher's probability 2^{-p}.

The cost of the pre-sieving step being

$$\max(2^{p+p_{in}+p_{out}+192-6(\ell_{in}+\ell_{out})},\ 2^{p+p_{in}+p_{out}+1}),$$

it was reasonable to begin with $\ell_{in}+\ell_{out}=32$ inactive rows in total to find the best attacks on both `SPEEDY-6-192` and `SPEEDY-7-192`. However for `SPEEDY-5-192`, the data complexity should not exceed 2^{64} but 2^{128} are allowed for the time complexity. Hence, for this particular variant we had to set $\ell_{in}+\ell_{out}$ to 31 to keep the overall probability low enough.

To exhaust all the possible probabilistic extensions, it is important to notice that both sides are independent. We therefore can compute the best probabilistic extension to the plaintext from the input difference Δ_{in} of the differential and to the ciphertext from the output difference Δ_{out} in parallel, before merging the results. Since there are only 32 rows on the plaintext (resp. on the ciphertext), there are at most 2^{32} configurations to check for each side. In practice there are much less configurations since some rows are already inactive and there is a limit on the number of rows that can be made inactive.

To find the best differentials together with their optimal probabilistic extensions we thus proceeded in two steps. First we exhausted all the best differentials through the multi-step procedure described in Section 3 up to a bound on the probability (2^{-64} for `SPEEDY-5-192`, 2^{-128} for `SPEEDY-6-192` and 2^{-192} for `SPEEDY-7-192`). Then, for each of them we searched for the best probabilistic extension such that the overall probability stays within the required range. To accelerate the search, we noticed that the set of possible Δ_{in} as well as the set of possible Δ_{out} formed from the best differentials is quite small and the optimal probabilistic extension only has to be computed once per element of these sets. Overall, it took less than 3 days on a 128-core server to find the best differentials to mount attacks on the three variants of `SPEEDY`.

4.2 Attack against `SPEEDY-7-192`

Our new attack against `SPEEDY-7-192` relies on a differential distinguisher of probability $2^{-174.63}$ combined to a probabilistic extension of probability $2^{-6.71-4.02}$

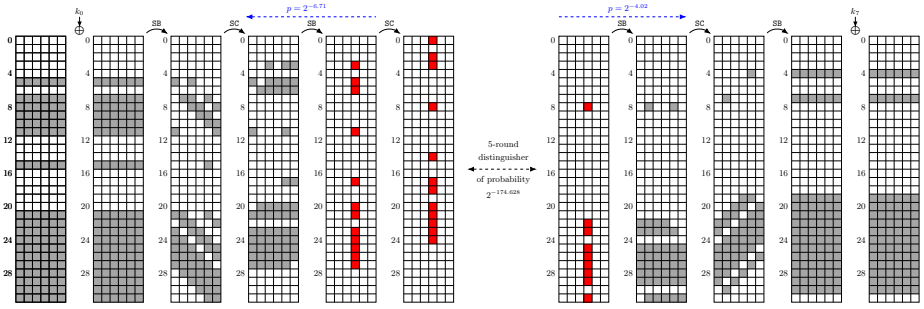
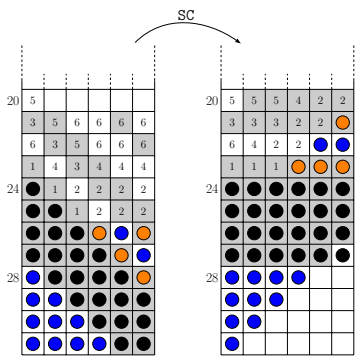


Fig. 5: Attack against `SPEEDY-7-192`.

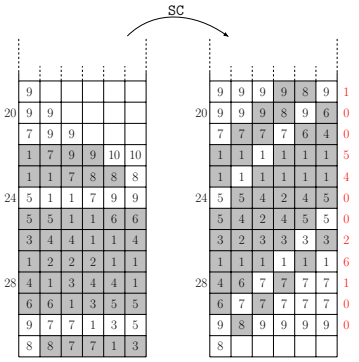
$= 2^{-10.73}$, resulting in an overall differential probability of $2^{-185.36}$, as illustrated in Figure 5. According to the formula given in Section 2.2, approximately $2^{186.36}$ chosen plaintexts are required to obtain a single pair that follows the full differential. This gives around $2^{185.36}$ candidate pairs for the key recovery step, since the total number of active rows in both the plaintext and ciphertext is 32. As in [12], the first step is to filter the pairs that cannot follow the differential because of incompatible differences before the first S-box layer and after the last S-box layer. The filter is non-marginal as only $2^{185.36-11.94} = 2^{173.42}$ pairs remain after this step. The detail of the computation of this filter is given in Table 6 in Supplementary Material C. We used the tool `KyRyDi` from [13] to find an efficient way to perform the key-recovery, and it found a procedure with a time complexity below 2^9 simple operations that can be summarized as follows:

1. The first step is to precompute a table containing all the possible values for the S-boxes 24 to 27 of the second S-box layer as well as the S-boxes 26 to 31 of the first S-box layer. We have $6 \times 4 = 24$ bits to guess for the first ones (which are reduced by a factor $2^{0.19}$ because of the probabilistic transition on row 27) and $3+2+1+1+2+3+4 = 16$ bits to complete the second ones. There are also 4 bits with unknown difference that have to be guessed as well. They are represented in orange on Figure 6a. Hence, in total we have only $2^{24-0.19+16+4} = 2^{43.81}$ possibilities for the values and the differences on rows 26 to 31 right after the first key addition instead of the naive 2^{64}. From a candidate pair, we know the differences on these 6 rows and thus we expect around $2^{43.81-36+f} = 2^{7.81+f}$ matches, where $f = 0.17 + 0.51 = 0.68$ represents the inverse of the filter already used on these S-boxes. The time complexity of this step is bounded by the $2^{173.42+7.81+0.68} = 2^{181.81}$ triplets of pair of data and partial key that we keep, while the memory complexity is $2^{43.81}$ 44-bit values. This is the most costly step of the key recovery and it is negligible compared to the cost of generating the original $2^{185.36}$ pairs.
2. Then we retrieve step by step and with no additional cost the possible values and differences of S-boxes of rows 21 to 25 on the first S-box layer and of S-boxes of rows 23, 21 and 20 of the second S-box layer as indicated in Figure 6a. We start by determining the bits marked with a 1, next with a

2, and so on until the ones with a six. It is easy to see how the row 23 of the second S-box layer will fully determine the three bits of values and of differences marked with a '1', as the pairs of the three remaining orange bits are already fully determined, and so is the output difference (6+6 determined bits in total). For the rest of the rows, we can see how step by step they all become overdetermined and therefore filtering. The number of solutions per triplet at the end of this stage is $2^{3\times6-1-2\times4+12+4-5\times6+f} = 2^{-5+f}$ with $f = 0.12+0.05+0.07 = 0.24$ represents the inverse of the filter already used on S-boxes from rows 21, 22 and 23. Indeed, there are $2^{18-1} = 2^{17}$ possible values for rows 20, 21 and 23 before the second S-box layer (the -1 comes from the transition on row 20, with a difference bit to 0) but 8 values are already known on these rows. Then 12 inactive bits as well as 2 active ones are missing to complete rows 21 to 25 before the first S-box layer. Hence, we are now left with $2^{181.81-5+0.24} = 2^{177.05}$ candidate triplets.



(a) On the first step, black and blue (resp. orange) bits are guessed for one message only (resp. both messages). On the second step, even indexes are obtained through the first S-box layer using the plaintext while odd indexes are computed using the second S-box layer.



(b) Third step of the key recovery attack. Even indexes are obtained through the penultimate S-box layer while odd indexes are computed using both the ciphertexts and the known key bits (in red).

Fig. 6: Description of the main steps of the key recovery attack against `SPEEDY-7-192`. The numbers indicate the order in which each bit is recovered.

3. At this point it is more efficient to retrieve S-boxes of the last round than the missing ones of the first round. Indeed, as the key schedule of `SPEEDY` is a permutation, the knowledge of the 66 key bits of k_0 we just obtained during steps 1 and 2, directly implies the knowledge of 25 key bits of k_7 that are involved in the final active S-box transitions, as it can be observed in Table 3, represented by the bits in blue. In particular, we know respectively 6, 5, 4 and 6 key bits related to the S-boxes of rows 4, 22, 23 and 27. In a similar

way as what we did in step 2, we apply guess and determine techniques that will not increase the complexity. At the end of this step, we decrease the number of candidates by $2^{8\times6-2.02+34+7-13\times6-25+5.61} = 2^{-10.41}$ and so only $2^{177.05-10.41} = 2^{166.64}$ of triplets remain.

4. We can go back to the missing rows of the first S-box layer. We now know 2 key bits of k_0 for rows 7 and 10, 3 for rows 5, 8, 9 and 4 for rows 11 and 15. As a consequence we can obtain for free the values of these S-boxes and then filter them through the second S-box layer by checking whether they are compatible with the input difference of the distinguisher. We finally obtain the missing active rows. We obain in the end $2^{185.36-35.73} = 2^{149.63}$ remaining triplets.
5. After this key recovery achieved, each of the $2^{149.63}$ triplets determines a possible value for the $192 - 46 = 146$ key bits involved. Hence, it seems that the attack does not directly filter any key but associates pairs of data to key guesses, which provides more information than just the 146 key bits and will allow us to reduce the number of candidates further. In addition, using a probabilistic extension trades some of the key bits that would have normally been involved into the knowledge of several state bits that we did not use yet. Namely, for the remaining triplets, we know the values of each active S-boxes in the first and the last round, and therefore the value of many inactive bits. In total, after the first S-box layer and before the last one, there are 19 inactive rows for which we know at least one bit: 1 for which we know 3 bits, 2 with 2 known bits and 12 with 1 known bit. The idea is thus to guess the missing bits for few of these rows to obtain more key material. By guessing $3 + 2 \times 4 + 4 \times 5 = 31$ state bits we retrieve $7 \times 6 = 42$ key bits. Let x be the number of these key bits we already know. In total, the remaining triplets now suggest $2^{149.63+31-x} = 2^{180.63-x}$ values for $146 + 42 - x = 188 - x$ key bits and it is simple to exhaust the missing ones for an overall complexity of $2^{180.63-x+192-(188-x)} = 2^{184.63}$. Finally, each key can be tested against a plaintext/ciphertext pair to check whether it is the right one or not.

5 Attacks against SPEEDY-5-192 and SPEEDY-6-192

We also tried to attack both other variants for SPEEDY, however we did not manage to fully break them. Still we propose the first valid attack against 5-round SPEEDY-6-192 as well as the best attack against 4-round SPEEDY-5-192.

5.1 SPEEDY-5-192

Our attack against SPEEDY-5-192 is based on a 2-round distinguisher of probability at least $2^{-51.66}$. Its dominant trail of probability $2^{-55.61}$ is shown in Figure 11.

This distinguisher is then extended probabilistically by one round in both directions, as illustrated in Figure 7. The parameters of this attack are: $p_{in} =$

Row number	Master key bits						# guessed
4	137	144	151	158	165	172	6
7	71	78	85	92	99	106	2
19	191	6	13	20	27	34	2
20	41	48	55	62	69	76	4
21	83	90	97	104	111	118	1
22	125	132	139	146	153	160	5
23	167	174	181	188	3	10	4
24	17	24	31	38	45	52	3
25	59	66	73	80	87	94	3
26	101	108	115	122	129	136	2
27	143	150	157	164	171	178	6
28	185	0	7	14	21	28	1
29	35	42	49	56	63	70	6
30	77	84	91	98	105	112	1

Table 3: Key bits of the round key k_7 that are involved in the attack against full SPEEDY-7-192. Coloured bits are also involved in k_0 and blue ones are known at the end of the second step of the key recovery.

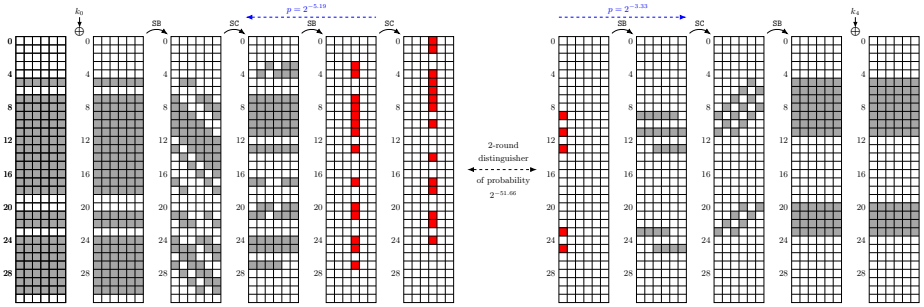


Fig. 7: Attack against 4-round SPEEDY-5-192.

5.19, $p_{out} = 3.33$, and $p' = p + p_{in} + p_{out} = 60.18$, obtained with $\ell_{in} = 10$ and $\ell_{out} = 21$ inactive rows in the plaintext and ciphertext, respectively.

Using the formula given in Section 2.2, we obtain that the data complexity of the corresponding attack is $2^{\frac{p'+6\times(32-\ell_{out})+1}{2}} = 2^{63.59}$ chosen ciphertexts and the number of candidate pairs to deal with is $2^{p'+192-6\times(\ell_{in}+\ell_{out})} = 2^{66.18}$. We used the tool KyRyDi which was able to find a key recovery attack associated to our differential distinguisher with complexity $2^{6.5}$ lower than the pairs generation process, thanks to the filter of $2^{-11.30}$ coming from the possible differences for both the plaintext and the ciphertext. At the end of the key recovery step, we still have $2^{66.18-21.53} = 2^{44.65}$ pairs, each of them suggesting a value for the $33\times 6 - 29 = 169$ key bits involved. Guessing the 23 missing key bits would make the final exhaustive search the bottleneck of the attack and it is more efficient to

apply the same technique used to finalize the attack against SPEEDY-7-192. For instance, we know 2 inactive bits on rows 4, 20 and 23 right after the first S-box layer. We can therefore guess 12 bits and retrieve 18 new bits of k_0, making the final exhaustive search negligible compared to the rest of the attack. Overall, the time complexity is dominated by the generation of the pairs and is lower $2^{66.18}$ encryptions. The memory complexity is around $2^{63.59}$ 192-bit blocks, required to store the data.

5.2 SPEEDY-6-192

Our attack against SPEEDY-6-192 is based on a 3-round distinguisher of probability at least $2^{-83.91}$. Its dominant trail of probability $2^{-83.91}$ is shown in Figure 10.

This distinguisher is then extended probabilistically by one round in both directions, as illustrated in Figure 8. The parameters of this attack are: $p_{in} = 8.93$, $p_{out} = 4.09$, and $p' = p + p_{in} + p_{out} = 96.93$, obtained with $\ell_{in} = \ell_{out} = 16$ inactive rows in both the plaintext and ciphertext.

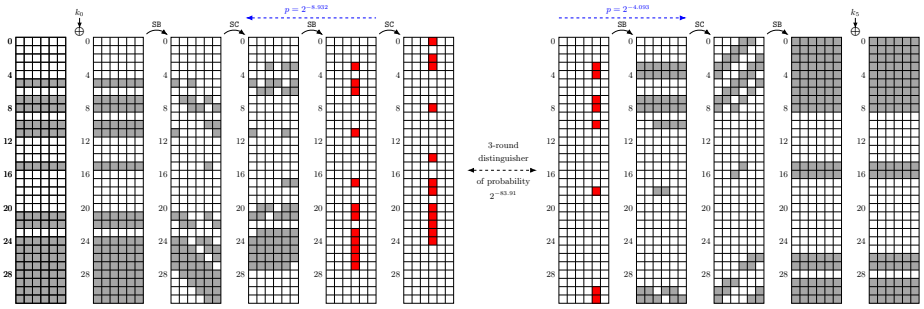


Fig. 8: Attack against 5-round SPEEDY-6-192.

The data complexity of our attack is $2^{97.93}$ and the memory is dominated by storing the structures of size 2^{96}. The KyRyDi tool found a key recovery with a complexity negligible compared to the pairs generation process. As for our other attacks, to keep the complexity as low as possible it is necessary to complete the key recovery by guessing some extra inactive state bits in order to retrieve the value of more key bits before performing a final exhaustive search.

Attacking more rounds The complexity of our attack on 5-round SPEEDY-6-192 is well below the designers' security claim, suggesting the potential to break more rounds. While the best distinguisher we identified for attacking 6 rounds has a probability significantly lower than 2^{-128}, we did find a 4-round distinguisher that can be extended to almost cover 5.5 rounds—that is, all but the final S-box layer. This distinguisher has a probability of $2^{-127.38}$, resulting in a data

and time complexity of approximately $2^{128.38}$, which only slightly exceeds the designers' 2^{128} bound. Still, this result indicates that only a small improvement would be needed to mount a valid attack on 5.5 rounds.

6 Conclusion

In this work, we present a cryptanalysis of the SPEEDY family of block ciphers. Our most significant contribution is the first full break of SPEEDY-7-192, achieved through a differential attack with a data and time complexity equivalent to $2^{186.36}$ encryption calls, and a memory complexity of 2^{84} states. We also introduce attacks on reduced-round versions of other instances in the family, specifically SPEEDY-5-192 reduced to 4 rounds and SPEEDY-6-192 reduced to 5 rounds. To the best of our knowledge, these results are the best attacks published to date against all instances of SPEEDY.

A core contribution of our work is the multi-step algorithm we deployed to search for high probability differential characteristics while taking into account their validity as well as their accurate probability through the quasidifferential framework. This made the search for differential distinguishers more tedious, which perhaps explains why it is often overlooked in cryptanalysis works. However, we believe that several recent contributions [8,29,15,28] have laid the path for this step to become a central part of the analysis of differential attacks.

Another central element of our approach is the use of probabilistic extensions, which we explore in a systematic and exhaustive manner. We believe that this technique could significantly optimize other differential attacks in the literature and represent a promising direction for future research.

References

1. Ahmadian, Z., Khalesi, A., M'foukh, D., Moghimi, H., Naya-Plasencia, M.: Improved differential meet-in-the-middle cryptanalysis. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 280–309. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_10
2. Avanzi, R.: The QARMA block cipher family. IACR Trans. Symm. Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/tosc.v2017.i1.4-44
3. Avanzi, R., Banik, S., Dunkelman, O., Eichlseder, M., Ghosh, S., Nageler, M., Regazzoni, F.: The QARMAv2 family of tweakable block ciphers. IACR Trans. Symm. Cryptol. **2023**(3), 25–73 (2023). https://doi.org/10.46586/tosc.v2023.i3.25-73
4. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 123–153. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_5
5. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: BipBip: A low-latency tweakable block cipher with small dimensions. IACR TCHES **2023**(1), 326–368 (2023). https://doi.org/10.46586/tches.v2023.i1.326-368

6. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: Introducing two low-latency cipher families: Sonic and SuperSonic. Cryptology ePrint Archive, Report 2023/878 (2023), https://eprint.iacr.org/2023/878
7. Beyne, T., Neyt, A.: Note on the cryptanalysis of speedy. Cryptology ePrint Archive, Report 2024/262 (2024), https://eprint.iacr.org/2024/262
8. Beyne, T., Rijmen, V.: Differential cryptanalysis in the fixed-key model. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 687–716. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_23
9. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO'90. LNCS, vol. 537, pp. 2–21. Springer, Berlin, Heidelberg (Aug 1991). https://doi.org/10.1007/3-540-38424-3_1
10. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology **4**(1), 3–72 (Jan 1991). https://doi.org/10.1007/BF00630563
11. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knežević, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_14
12. Boura, C., David, N., Boissier, R.H., Naya-Plasencia, M.: Better steady than speedy: Full break of SPEEDY-7-192. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 36–66. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30634-1_2
13. Boura, C., David, N., Derbez, P., Boissier, R.H., Naya-Plasencia, M.: A generic algorithm for efficient key recovery in differential attacks - and its associated tool. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 217–248. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_8
14. Boura, C., Derbez, P., Funk, M.: Related-key differential analysis of the AES. IACR Trans. Symm. Cryptol. **2023**(4), 215–243 (2023). https://doi.org/10.46586/tosc.v2023.i4.215-243
15. Boura, C., Derbez, P., Germon, B.: Extending the quasidifferential framework: From fixed-key to expected differential probability. IACR Trans. Symmetric Cryptol. **2025**(1), 515–541 (2025). https://doi.org/10.46586/TOSC.V2025.I1.515-541, https://doi.org/10.46586/tosc.v2025.i1.515-541
16. Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. Journal of Cryptology **31**(1), 101–133 (Jan 2018). https://doi.org/10.1007/s00145-016-9251-7
17. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A low-latency block cipher for secure cache-randomization. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 1937–1954. USENIX Association (Aug 2023)
18. Flórez-Gutiérrez, A.: Optimising linear key recovery attacks with affine Walsh transform pruning. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 447–476. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_16
19. 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, Part I. LNCS, vol. 12105, pp. 221–249. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_9

20. Flórez-Gutiérrez, A., Todo, Y.: Improving linear key recovery attacks using Walsh spectrum puncturing. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 187–216. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_7
21. Flórez-Gutiérrez, A., Todo, Y.: Improved cryptanalysis of chacha: Beating pnbs with bit puncturing. In: Fehr, S., Fouque, P. (eds.) EUROCRYPT 2025, Part I. LNCS, vol. 15601, pp. 427–457. Springer (2025). https://doi.org/10.1007/978-3-031-91107-1_15, https://doi.org/10.1007/978-3-031-91107-1_15
22. Fouque, P.A., Jean, J., Peyrin, T.: Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 183–203. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_11
23. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-13858-4_21
24. Greene, P., Motley, M., Weeks, B.: ARADI and LLAMA: Low-latency cryptography for memory encryption. Cryptology ePrint Archive, Report 2024/1240 (2024), https://eprint.iacr.org/2024/1240
25. Knudsen, L.R.: Iterative characteristics of DES and s^2-DES. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 497–511. Springer, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_35
26. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 17–38. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_2
27. 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. IACR TCHES **2021**(4), 510–545 (2021). https://doi.org/10.46586/tches.v2021.i4.510-545, https://tches.iacr.org/index.php/TCHES/article/view/9074
28. Nageler, M., Ghosh, S., Jüttler, M., Eichlseder, M.: Autodiver: Automatically verifying differential characteristics and learning key conditions. IACR Trans. Symmetric Cryptol. **2025**(1), 471–514 (2025). https://doi.org/10.46586/TOSC.V2025.I1.471-514, https://doi.org/10.46586/tosc.v2025.i1.471-514
29. Peyrin, T., Tan, Q.Q.: Mind your path: On (key) dependencies in differential characteristics. IACR Trans. Symm. Cryptol. **2022**(4), 179–207 (2022). https://doi.org/10.46586/tosc.v2022.i4.179-207
30. Rohit, R., Sarkar, S.: Cryptanalysis of reduced round SPEEDY. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT 22. LNCS, vol. 2022, pp. 133–149. Springer, Cham (Jul 2022). https://doi.org/10.1007/978-3-031-17433-9_6
31. Song, L., Yang, Q., Chen, Y., Hu, L., Weng, J.: Probabilistic extensions: A one-step framework for finding rectangle attacks and beyond. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 339–367. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_12
32. Wang, J., Niu, C., Liu, Q., Li, M., Preneel, B., Wang, M.: Cryptanalysis of speedy. In: Simpson, L., Rezazadeh Baee, M.A. (eds.) Information Security and Privacy. pp. 124–156. Springer Nature Switzerland, Cham (2023)

Supplementary material

A SPEEDY detailed specifications

A.1 Table representation of the SPEEDY S-box

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

Table 4: Table representation of the 6-bit S-box S

A.2 The bit-permutation P of SPEEDY

Table 5 describes the bit-permutation P used to instantiate the key schedule of SPEEDY-r-192 for r = 5, 6, 7.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	1	8	15	22	29	36	43	50	57	64	71	78	85	92	99	106
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	113	120	127	134	141	148	155	162	169	176	183	190	5	12	19	26
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	33	40	47	54	61	68	75	82	89	96	103	110	117	124	131	138
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	145	152	159	166	173	180	187	2	9	16	23	30	37	44	51	58
i	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$P(i)$	65	72	79	86	93	100	107	114	121	128	135	142	149	156	163	170
i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$P(i)$	177	184	191	6	13	20	27	34	41	48	55	62	69	76	83	90
i	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
$P(i)$	97	104	111	118	125	132	139	146	153	160	167	174	181	188	3	10
i	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
$P(i)$	17	24	31	38	45	52	59	66	73	80	87	94	101	108	115	122
i	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
$P(i)$	129	136	143	150	157	164	171	178	185	0	7	14	21	28	35	42
i	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
$P(i)$	49	56	63	70	77	84	91	98	105	112	119	126	133	140	147	154
i	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
$P(i)$	161	168	175	182	189	4	11	18	25	32	39	46	53	60	67	74
i	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
$P(i)$	81	88	95	102	109	116	123	130	137	144	151	158	165	172	179	186

Table 5: The bit-permutation P for SPEEDY-r-192.

B Impossible characteristic from [32]

As in Section 2.4, we consider the characteristic over `SB` ∘ `SC` ∘ `SB` displayed in Figure 9. It represents a part of the second round in the dominant characteristic of the attack in `SPEEDY-7` in [32]. Looking at the first S-box layer on row 7 one can see that the differential transition `(0, 0, 0, 0, 0, 1)` → `(0, 1, 0, 1, 0, 0)` must be satisfied. The set of output values $y \in \mathbb{F}_2^6$ that actually allow this transition are of the form `(1, *, 1, *, 0, 0)`. Thus, the leftmost bit must be equal to 1. This bit isn't moved by the `SC` operation and therefore must also be a valid input for the differential transition `(0, 1, 0, 0, 0, 0)` → `(0, 0, 0, 1, 0, 0)` through the second S-box layer. However, this transition is only satisfied for inputs $x \in \mathbb{F}_2^6$ of the form `(0, *, 1, 1, *, *)`. This explains why this characteristic is actually invalid. A similar constraint arises from the third bit of the 8-th row in the same round. As it can be expected, those two constraints can be fully explained by the existence of 4 quasidifferential trails with absolute correlation equal to the correlation of the zero-masks trail.

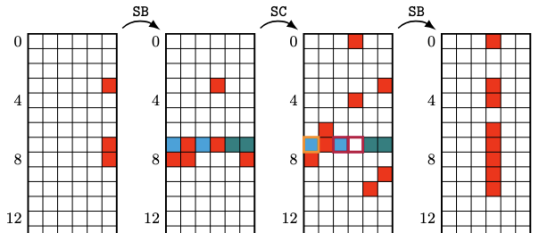


Fig. 9: Impossible characteristic over `SB` ∘ `SC` ∘ `SB`. We represent the inactive bits of the characteristic in white and active bits in red. Furthermore, for the first S-box layer we represent the bits fixed to 1 (resp. 0) in cyan (resp. teal). Regarding, the second S-box layer the bits fixed to 1 (resp. 0) are framed in purple (resp. orange). The impossibility is visually explained by the fact that the leftmost bit of row 7 is both colored in cyan and framed in orange after the `SC` operation.

C Pairs filtering

Table 6 provides the filter per active S-box obtained from the first S-box application of the first round and the last S-box application of the last round of the attack. We explain how this filtering was computed with one example. Let's look at row 7 before and after the first S-box application. By looking at the DDT of SPEEDY's S-box, we see that the input differences `0x7`, `0x19` and `0x1b` never propagate to an output difference of the form (0,*,*,0,*,0). Thus, any pair with one of those three plaintext differences at row 7 can be filtered out. This gives us a filter of log2(61/64) = 0.0069, as shown in Table 6.

First-round S-boxes				Last-round S-boxes			
Row	Filter	Row	Filter	Row	Filter	Row	Filter
5	0.415	23	0.069	4	2	26	0
7	0.069	24	0	7	2.093	27	0
8	0.142	25	0	19	0.642	28	0.069
9	0.541	26	0	20	0.046	29	0.752
10	0.508	27	0	21	0	30	2.093
11	0.415	28	0	22	0		
15	1.245	29	0	23	0		
21	0.117	30	0.167	24	0		
22	0.046	31	0.508	25	0		
Total filter: 11.94							

Table 6: Filtering in the active S-boxes of the first S-box and the last S-box application during the attack against `SPEEDY-7-192`.

Table 7 and Table 8 present the same information for the attacks against `SPEEDY-5-192` and `SPEEDY-6-192` respectively.

First-round S-boxes				Last-round S-boxes	
Row	Filter	Row	Filter	Row	Filter
5	0.476	17	0.385	5	2
7	0.069	18	0.476	6	0.508
8	0.023	21	0.069	7	0.791
9	0	22	0.046	8	0.046
10	0	24	0.069	9	0.117
11	0	25	0	10	0.752
12	0	26	0.046	11	0.752
13	0	27	0.069	20	0.508
14	0	28	0.117	21	0.791
15	0.167	29	0.167	22	0.752
16	0.642	30	0.642	23	0.752
Total filter: 11.115					

Table 7: Filtering in the active S-boxes of the first S-box and the last S-box application during the attack against `SPEEDY-5-192`.

First-round S-boxes				Last-round S-boxes			
Row	Filter	Row	Filter	Row	Filter	Row	Filter
5	0.415	24	0	0	0.642	8	0.752
7	0.069	25	0	1	0.445	15	1.541
8	0.142	26	0	2	0.046	16	1.678
10	0.508	27	0	3	0	26	0.956
11	0.415	28	0	4	0.023	27	0.642
15	1.245	29	0	5	0.046	29	0.752
21	0.117	30	0.167	6	0.167	30	0.046
22	0.046	31	0.508	7	0.069	31	0.093
Total filter: 11.53							

Table 8: Filtering in the active S-boxes of the first S-box and the last S-box application during the attack against `SPEEDY-6-192`.

D Dominant trails of our attacks against `SPEEDY-5-192` and `SPEEDY-6-192`

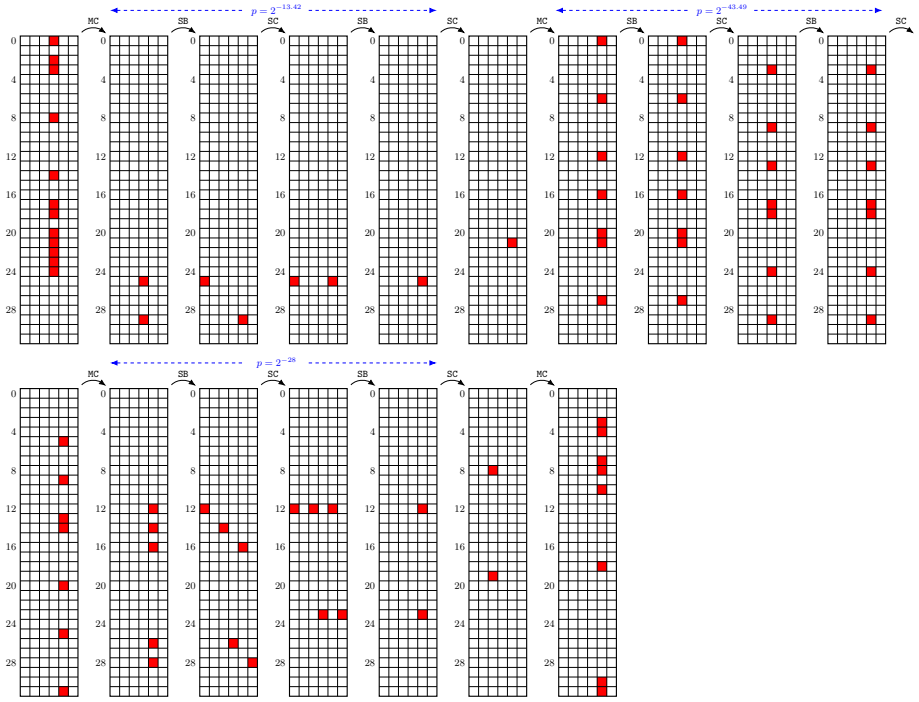


Fig. 10: 3-round trail in the differential distinguisher used to attack 5 rounds of `SPEEDY-6-192`.

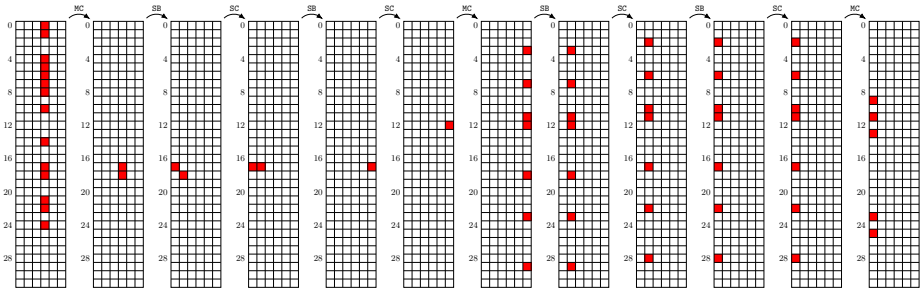


Fig. 11: Dominant 2-round trail in the differential distinguisher used to attack 4 rounds of SPEEDY-5-192.

E Relations between round keys for our attacks against SPEEDY-5-192 and SPEEDY-6-192

Row number	Master key bits						# guessed
0	25	176	135	94	53	12	3
1	163	122	81	40	191	150	3
2	109	68	27	178	137	96	3
3	55	14	165	124	83	42	2
4	1	152	111	70	29	180	3
5	139	98	57	16	167	126	2
6	85	44	3	154	113	72	2
7	31	182	141	100	59	18	2
8	169	128	87	46	5	156	4
15	175	134	93	52	11	162	4
16	121	80	39	190	149	108	2
26	157	116	75	34	185	144	4
27	103	62	21	172	131	90	3
29	187	146	105	64	23	174	4
30	133	92	51	10	161	120	3
31	79	38	189	148	107	66	3

Table 9: Key bits of the subkey k_5 involved in the attack against 5-round SPEEDY-6-192. In red, bits of k_0 that are also involved in the attack.

Row number	Master key bits						# guessed
5	14	111	16	113	18	115	2
6	20	117	22	119	24	121	0
7	26	123	28	125	30	127	2
8	32	129	34	131	36	133	5
9	38	135	40	137	42	139	3
10	44	141	46	143	48	145	4
11	50	147	52	149	54	151	6
20	104	9	106	11	108	13	3
21	110	15	112	17	114	19	2
22	116	21	118	23	120	25	0
23	122	27	124	29	126	31	2

Table 10: Key bits of the subkey k_4 involved in the attack against 4-round SPEEDY-5-192. In red, bits of k_0 that are also involved in the attack.

F KyRyDi graph of the key recovery for SPEEDY-7-192

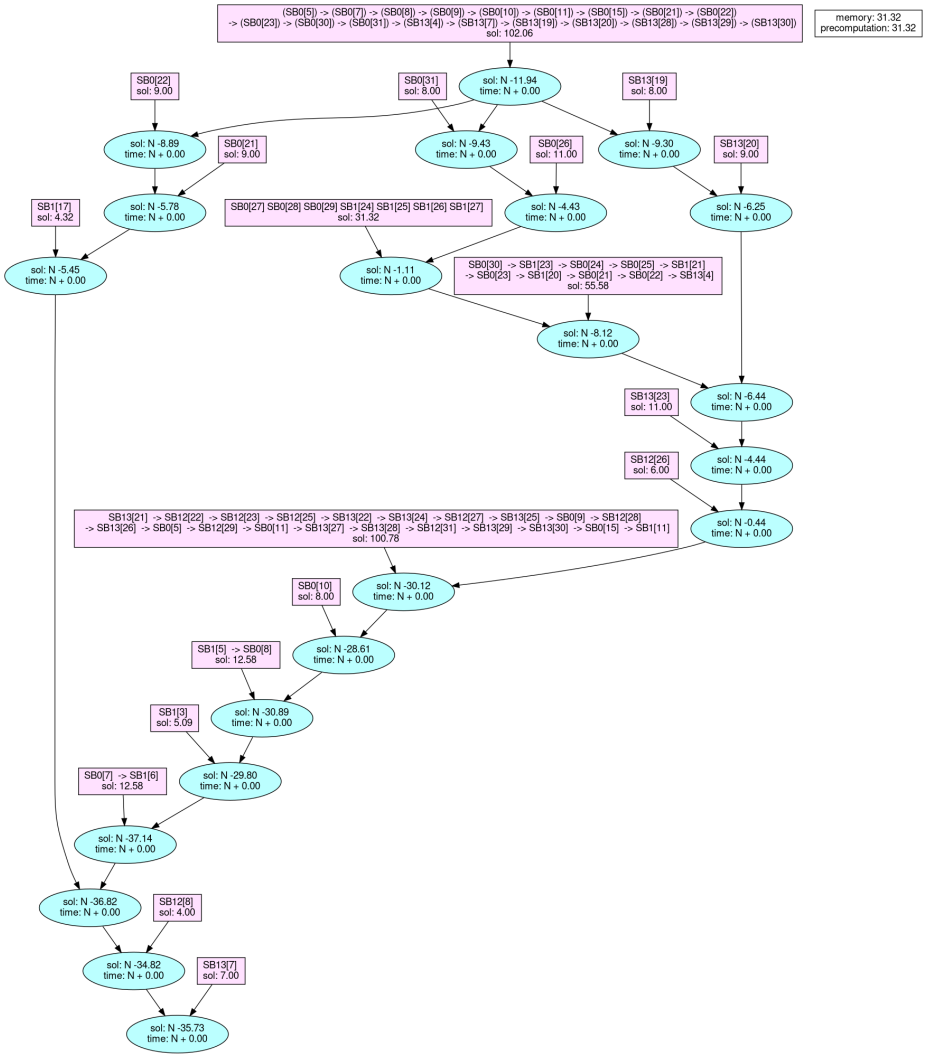


Fig. 12: Key recovery outputted by the `KyRyDi tool` for our attack against `SPEEDY-7-192`.