Speeding Up the Search Algorithm for the Best Differential and Best Linear Trails

Zhenzhen Bao(✉), Wentao Zhang, and Dongdai Lin

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{baozhenzhen,zhangwentao,ddlin}@iie.ac.cn

Abstract. For judging the resistance of a block cipher to differential cryptanalysis or linear cryptanalysis it is necessary to establish an upper bound on the probability of the best differential or the bias of the best linear approximation. However, getting a tight upper bound is not a trivial problem. We attempt it by searching for the best differential and the best linear trails, which is a challenging task in itself. Based on some previous works, new strategies are proposed to speed up the search algorithm, which are called *starting from the narrowest point*, *concretizing and grouping search patterns*, and *trialling in minimal changes order* strategies. The efficiency of the resulting improved algorithms allows us to state that the probability (bias) of the *best* 4-round differential (linear) trail in NOEKEON is 2^{-51} (2^{-25}) and the probability (bias) of the best 10-round (11-round) differential (linear) trail is at most 2^{-131} (2^{-71}). For SPONGENT, the *best* differential trails for certain number of rounds in the permutation functions with width $b \in \{88, 136, 176, 240\}$ are found. That allows us to update some results presented by its designers.

Keywords: Differential cryptanalysis · Linear cryptanalysis · Differential trail · Linear trail · Search algorithm · Optimization · NOEKEON · SPONGENT

1 Introduction

Differential cryptanalysis (DC) [1] and linear cryptanalysis (LC) [2] are two of the most powerful attacks against modern block ciphers in which an adversary exploits good differentials or good linear approximations. The first step in a differential or a linear attack consists in finding differentials or linear approximations of the cipher with probabilities or bias as high as possible. In most cases, differential trails with highest probability and linear approximation trails with largest bias can be used to estimate the power of the corresponding attack. Differential (linear) trails consist of a sequence of differences (approximations) through the rounds of the primitive and those with the highest probability (the largest bias) are called the best. However, the problem of searching best trails is not trivial, because of the great cardinality of the set of candidates [3,4].

D. Lin et al. (Eds.): Inscrypt 2014, LNCS 8957, pp. 259–285, 2015.
DOI: 10.1007/978-3-319-16745-9_15

For many block ciphers, such as AES, NOEKEON and PRESENT, researchers prefer counting the minimal number of active S-Boxes to get the upper bound of the best probability (bias) of differential (linear) trails [5–8]. In this method, concepts such as branch number and structures of the linear layer are used. Furthermore, tools using MILP are developed [9]. However, those approaches could only provide a kind of differential trails without the instantiated actual differences or without the knowledge of exact probabilities of those trails. Remarkably, authors of [10] use a variant of Dijkstra's algorithm which is essentially a breadth-first search to efficiently find all best truncated differential trails with minimal number of active S-Boxes and instantiate them with actual differences. This method is very powerful, however, on one hand, it may fail to find the best differential trail which does not have the minimal number of active S-Boxes, and on the other hand it is powerless in the case of bit-oriented ciphers. Specifically, although this breadth-first approach is in polynomial time in the number of rounds, it is exponential in the state. Thus, for ciphers using large number of small S-Boxes, which is typically 4 bits wide, and have weak alignment, an intermediate state tends to large, and a PC cannot store all the intermediate state, that is, we cannot choose a breadth-first strategy. Thus, we seek for depth-first method to find the best trails.

In 1994, Matsui proposed a branch-and-bound depth-first search algorithm making it possible to effectively find the best differential trails and linear approximation trails of DES [11]. Unfortunately, his method is not fast enough for some other cryptosystems like FEAL. Consequently, improvements on Matsui's algorithm were studied by Moriai et al. [3] and Aoki et al. [12]. The work in [3] was based on analyzing the dominant factor of search complexity and it introduced the concept of search patterns in order to reduce unnecessary search candidates. The authors successfully obtained new results on best linear approximations for FEAL by applying the proposed search algorithm. In [12], Aoki et al. further optimized the search algorithm. They presented good results of the search for the best differential trails of FEAL using a pre-search strategy.

Recently, automatic tools for searching for differential trails in ARX ciphers are relatively mature [13,14]. One of them [14] is also extended from Matsui's algorithm. However, due to the fact that it uses a partial, rather than the full DDT, their algorithm is not guaranteed to find the best differential trail. To the best of our knowledge, there is no application of those tools which are designed for ARX ciphers to Sbox-based ciphers.

For modern Sbox-based ciphers, expanding block size and good diffusion cause the probabilities of the best trails of very short rounds to be tiny. Generally, the smaller the probability of a best trail, the longer the time of a search will be. The heuristic search algorithm described in [4] might be helpful, but could hardly satisfy the cryptographers to fully estimate the vulnerability of a modern cipher to DC and LC. For designers who need to repeatedly apply the search algorithm to their draft ciphers to choose the best possible components and to decide a proper number of rounds, and for attackers who want to obtain large sets of trails with probabilities as high as possible and with rounds as many as possible, it is profitable to further optimize the search algorithm.

In this paper, we focus on this problem and aim to speed up the depth-first search algorithm for the best actual differential and linear trails.

The target objects are of Sbox-based iterated block cipher [5] in which all intermediate rounds use the same round transformation. We only consider those iterated ciphers with round keys being added to the state by means of XOR operation which is very common in modern block ciphers.

1.1 Our Contributions

We present three new optimization strategies to speed up the search algorithm for the best trails, which are called *starting from the narrowest point*, *concretizing and grouping search patterns*, and *trialling in minimal changes order* strategies.

- *Starting from the narrowest point* is very helpful to reduce complexity to a great extent by raising the threshold to the candidates at the earliest phases of the search procedure and maximizing shareable work at those phases.
- *Concretizing and grouping search patterns* further maximizes the scope of shared works and collects more information on search patterns to filter out invalid ones, while keeping the memory requirement appropriate.
- *Trialling in minimal changes order* utilizes the locality of the nonlinear layer and the linearity of the linear layer, to tame the brute force search to behave in a systematical and efficient manner.

Experimental results show that, the first two strategies bring a speed up by a factor of around 740–2800, which can be seen in Table 2. Considering the profit brought by the third strategy, the resulting improved algorithm has around 1500–5000 speedup ratio for the experimental subject.

Our final improved algorithm has been applied to search for the best differential and best linear trails in a block cipher named NOEKEON which was designed by Joan Daemen et al. [6]. The efficiency of the improved algorithm allows us to find out the *best* trails, thus to state that probability (bias) of the best 4-round differential (linear) trail in NOEKEON is 2^{-51} (2^{-25}). Additionally, probability (bias) of the best 5-round and 6-round trails are $\leq 2^{-65}$ ($\leq 2^{-32}$) and $\leq 2^{-80}$ ($= 2^{-40}$) respectively. That allows us to claim that the probability (bias) of the best 10-round (11-round) differential (linear) trail in NOEKEON is at most 2^{-131} (2^{-71}). The results are summarized in Table 3. Besides, we found out the longest linear trail holding with bias larger then 2^{-65}, which is a 9-round trail with bias 2^{-62}. These improved positive results contribute to the estimation of the security of NOEKEON against differential (linear) cryptanalysis.

We have also used this final improved algorithm to search for the best differential trails in the permutation functions of SPONGENT, a hash function. We found out the *best* differential trails for variants with width $b \in \{88, 136, 176, 240\}$ for certain number of rounds, and update some results presented by its designers in [16]. Some of the results are summarized in Table 4.

These strategies are also useful for us to search for the best differential and best linear trails for other primitives and helpful to search for best multiple differential (multi-dimensional linear) distinguishers.

By the way, all of the experiments and results in this paper are timed and obtained on a PC with Intel(R) Core(TM) i5-4570S 2.90 GHz CPU, and 4 GB RAM, using single-thread program in C.

1.2 Organization

The paper is organized as follows. Some preliminaries and symbolic conventions are presented in Sect. 2. In Sect. 3, we introduce and briefly discuss three previous works including Matsui's algorithm, Moriai et al.'s algorithm and Aoki et al.'s algorithm. Section 4 establishes our overall strategies and basic principles. The three optimization strategies *starting from the narrowest point*, *concretizing and grouping search patterns* and *trialling in minimal changes order* are covered in Sects. 5–7, which provide the justification and experimental results on the efficiency. Finally, new results on best trails in a block cipher NOEKEON and in the permutation functions of a hash function SPONGENT are shown in Sect. 8. In Sect. 9, we conclude our algorithm and prospect for further improvement.

2 Notations and Preliminaries

For convenience, we will explain the optimization strategies with SPN ciphers with non-linear layer being a parallel execution of 4×4 - S-Boxes in our mind. While, proposed strategies are also applicable to ciphers of Feistel structure and with larger S-Boxes.

Because of the duality between the search for the best differential trails and the search for the best linear trails [11], we will explain the optimization strategies from the perspective of differential.

A more natural way will be used to characterize the power of trails - the weight of differential trail which is the sum of the weight of round differentials, where the latter is the negative of its binary logarithm of its probability [15, Chap. 5].

id_r: the input difference of the r-th round differential[1]
od_r: the output difference of the r-th round differential
p_r: the probability of the r-th round differential
w_r: the weight of the r-th round differential, $w_r = -\log_2 p_r$
$w_{(id_r, od_r)}$: the weight of the r-th round differential (id_r, od_r)
w^r: the weight of a r-round differential trail, $w^r = \sum_{i=1}^{r} w_i$, where w_i is the weight of the i-th round differential composing that r-round differential trail
Bw^r: the weight of the best r-round differential trail
Bwc^r: the candidate of Bw^r
ASN: the abbreviation of *Number of Active S-Boxes*
asn_r: ASN at the S-Layer of the r-th round[2]

[1] We index the rounds begin with 1, i.e. $1 \leq r \leq n$, where n is the number of rounds of a block cipher.

[2] When using the *starting from the narrowest point strategy*, we index the rounds relatively to the narrowest point.

3 Previous Works

3.1 Matsui's Algorithm

Matsui's algorithm [11] works by induction on the number of rounds n and derives the best n-round weight Bw^n from the knowledge of all best r-round weight Bw^r $(1 \leq r \leq n-1)$. The original search algorithm targets DES. Here, we summarize Matsui's algorithm for SPN ciphers. The framework consists of the recursive procedures described in Algorithm 1. In Algorithm 1, Bwc^n holds the temporary approximation of the value of Bw^n. It is an upper bound of Bw^n and improved in a decreasing manner during the search bounded by conditions $\sum_{i=1}^{r} w_i + Bw^{n-r} \leq Bwc^n$ $(1 \leq r \leq n-1)$. When all of the possible paths have been traversed, Bwc^n turns to be the exact value of Bw^n.

Algorithm 1. Matsui's Algorithm

```
1:  Bwc^n ← an upper bound of Bw^n
2:  procedure ROUND-1
3:      for all candidate of od_1 do
4:          w_1 ← min_{id_1}(w_{(id_1,od_1)})
5:          if w_1 + Bw^{n-1} ≤ Bwc^n then
6:              ROUND-i(2)
7:          end if
8:      end for
9:      Exit the program
10: end procedure
11: procedure ROUND-i(r) (2 ≤ r ≤ n)
12:     id_r ← od_{r-1}
13:     if r < n then
14:         for all candidate of od_r do
15:             w_r ← w_{(id_r,od_r)}
16:             if Σ_{i=1}^{r} w_i + Bw^{n-r} ≤ Bwc^n then
17:                 ROUND-i(r + 1)
18:             end if
19:         end for
20:     else
21:         w_n ← min_{od_n}(w_{(id_n,od_n)})
22:         if Σ_{i=1}^{n} w_i < Bwc^n then
23:             Bwc^n ← Σ_{i=1}^{n} w_i
24:         end if
25:     end if
26: end procedure
```

3.2 Moriai et al.'s Algorithm

Moriai et al.'s program [3] is based on Matsui's algorithm. The concept of *search patterns* was introduced to detecting the unnecessary and impossible search candidates.

Definition 1 (Search Pattern [3]). *An n-round search pattern used in the search for the best differential trail is a vector of n values of weights, which is denoted as $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$, where w_i is the weight of the i-th round differential $(1 \leq i \leq n)$. Let $|\mathbb{W}^n| \equiv \sum_{i=1}^{n} w_i$.*

Given n and Bwc^n which is a lower bound of Bw^n, their algorithm first generates all possible patterns[3] using Bw^r $(1 \leq r \leq n-1)$. It then examines whether there is a differential trail fitting one of the patterns. If none of the patterns has a real trail, another candidate for Bw^n is similarly trialled. Their algorithm is summarized as Algorithm 2. There are two improvements in Algorithm 2 compared with Algorithm 1. As for the first improvement, knowledge of all weights

[3] For simplicity, we sometimes address *search patterns* with the term *patterns*.

of best r-round trails is more sufficiently used by observing that $\forall r, i$ $(1 \leq r \leq n-1,\ 1 \leq i \leq n-r+1)$, $\sum_{j=i}^{i+r-1} w_j \geq Bw^r$. Thus it can delete more non-existent candidates. As for the second improvement which targets involutory ciphers, concept of search patterns is used and patterns are classified into two equivalent classes, the class which has more candidates is deleted and thus it can delete duplicate candidates and reduce the computation complexity.

Algorithm 2. Moriai et al.'s Algorithm

1: $Bwc^n \leftarrow$ a lower bound of Bw^n
2: **while** *true* **do**
3: Generate all search patterns $(w_1, w_2, \ldots, w_n)$ which make the following hold:
   1. $\sum_{i=1}^{n-r} w_i + Bw^r \leq Bwc^n$, for $1 \leq r \leq n-1$
   2. $\forall r, i$ $(1 \leq r \leq n-1,\ 1 \leq i \leq n-r+1)$, $\sum_{j=i}^{i+r-1} w_j \geq Bw^r$.
4: Discard either search pattern $(w_1, w_2, \ldots, w_n)$ or $(w_n, w_{n-1}, \ldots, w_1)$ whichever has more search candidates.
5: Search the differential trails corresponding to the search patterns as Algorithm 1 with all the inequalities replaced by equalities.
6: **if** find out a trail **then** $Bw^n \leftarrow Bwc^n$ and Exit
7: **end if**
8: $Bwc^n \leftarrow Bwc^n + 1$
9: **end while**

3.3 Aoki et al.'s Algorithm

In Algorithm 2 restrictions are merely based on weights of the best, while patterns of the best and patterns of r-round trails which are not the best cannot be used. Aoki et al.'s [12] considered the patterns themselves. They checked about the existence of the combined patterns using information about search patterns of differential trails with various weights by a pre-search strategy. Their algorithm is summarized as Algorithm 3.

Algorithm 3. Aoki et al.'s Algorithm

1: **procedure** Pre-Search
2: Search r-round $(r < n)$ differential trails with various weights, and compile information to the extent possible whether or not the search pattern exist for each round and weight.
3: **end procedure**
4: **procedure** Search
5: Do as Algorithm 2, but discard the search patterns which do not exist using the information from the pre-search phase.
6: **end procedure**

4 Overall Strategy and Basic Principle in Our Work

Based on the three previous works, our program works by inducting on the number of rounds n, using concept of the search patterns and doing pre-search. For n-round cipher, w^n is initialized with a lower bound of Bw^n and it is increased by an unit until exceeding the range of weight we considered. During this procedure, Bw^n is determined when the first time an n-round differential trail being found out. For each temporary value of w^n, we generate the corresponding search pattern set using Bw^r $(1 \leq r \leq n-1)$ as in Algorithm 2 and using the information about existence of r-round $(1 \leq r \leq n-1)$ search patterns collected during the preceding search phases. Within each search pattern set, the search traverses in a depth first manner. In Appendix A, framework of our search approach is shown in Algorithm 4. Algorithms 5–8 formalize the search procedure deploying the *starting from the narrowest point*, *concretizing and grouping search patterns*, and *trialling in minimal changes order* strategies which will be explained in the following Sects. 5–7.

5 Starting from the Narrowest Point Strategy

It has been shown in [3] that for Feistel ciphers the complexity of search for the best n-round trails is dominated by the number of candidates in procedures of the first two rounds. Similarly, for SPN ciphers, we found that the complexity of search is dominated by the number of candidates of the first two rounds (i.e., the first two layers in the depth-first search procedure). Generally, the number of candidates of the first two rounds greatly depends on the weight of the first round, the smaller the weight, the less the number of candidates. Thus, reducing the number of candidates at the first two layers will be much helpful. In this section, we propose our first strategy, i.e., *starting from the narrowest point* strategy. By using this strategy, the number of candidates at the first two layers can be reduced greatly.

5.1 Proposal and Justification of the Starting from the Narrowest Point Strategy

We organize the search patterns using a kind of balance trees, with roots starting from the *narrowest point* instead of the first point of the search patterns, see Definition 2 and Example 1 for clarify.

Definition 2 (Narrowest Point and Relative-Index Form). *Given an n-round search pattern $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$, suppose there are k minimal components $w_{x_1}, w_{x_2}, \ldots, w_{x_k}$, i.e. $w_{x_i} = w_{min} \equiv \min(w_1, w_2, \ldots, w_n)$ for $1 \leq i \leq k$. Let* $\mathrm{nxt}(x) \equiv \begin{cases} x+1 & 1 \leq x \leq n-1 \\ n-1 & x = n \end{cases}$, *and* $v_{min} \equiv \min(w_{\mathrm{nxt}(x_1)}, w_{\mathrm{nxt}(x_2)}, \ldots, w_{\mathrm{nxt}(x_k)})$. *Suppose w_m is the first component of $\mathbb{W}^n$ which satisfies $w_m = w_{min}$ and $w_{\mathrm{nxt}(m)} = v_{min}$, we call the index m the* ***narrowest point****, and say w_m lies*

at the ***narrowest point*** *of* $\mathbb{W}^n$*. If we index each component* w_x *of* $\mathbb{W}^n$ *with the relative distance between* x *and* m *(i.e.,* $x - m$*),* $\mathbb{W}^n$ *can be rewritten as* $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_{-1}, \breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m})$*, where* $\breve{w}_{x-m} = w_x$*. We call* $\breve{\mathbb{W}}^n$ *the* ***relative-index form*** *of* $\mathbb{W}^n$ *and define the relative index of* w_x *as* $rix(w_x) = rix(\breve{w}_{x-m}) = x - m$*,* $1 \leq x \leq n$*.*

A search pattern $\mathbb{W}^n$ is placed at the search tree in its relative-index form $\breve{\mathbb{W}}^n$ as depicted in Fig. 1.

Example 1. Considering a set of search patterns with $|\mathbb{W}^3| = 30$ of 3-round NOEKEON: $\mathbb{S}$ ={ (2, 14, 14), (14, 2, 14), (4, 13, 13), (13, 4, 13), (6, 11, 13), (13, 6, 11), (6, 12, 12), (12, 6, 12) }, Fig. 2 depicts two ways to organize the search patterns in $\mathbb{S}$. In Fig. 2a, search patterns in $\mathbb{S}$ are organized from the first point, and they are organized from the narrowest point in Fig. 2b.

There are three main reasons why *starting from the narrowest point strategy* can help to greatly reduce the search complexity:

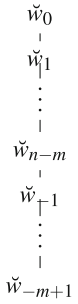
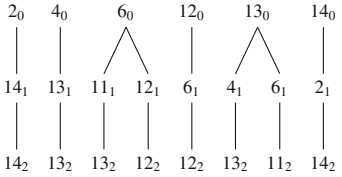
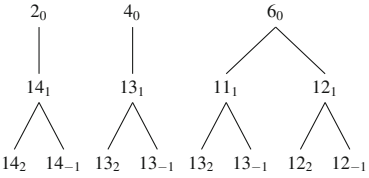


Fig. 1. Placing a search pattern at the search tree in its relative-index form



(a) Organizing from the first points(index components relative to the round-index of the first component)



(b) Organizing from the narrowest points (index components relative to the round-index of the narrowest point)

Fig. 2. Organizing the search patterns in Example 1 in two ways. Nodes in trees are elements representing one-round weights in search patterns. The subscript in each node represent the index of the point relative to the starting point.

1. Firstly, the range of the minimal value in a search pattern set is narrower than the range of an arbitrary value. In general, the range of the allowed minimal value which composes a sum is narrower than the range of the allowed arbitrary value. Here is a simple example. Assume we need to partition 11 to 4 positive numbers x_1, x_2, x_3, x_4, then the smallest number must equal to 1 or 2, while x_1 can be any number between 1 and 8. In our situation, take Example 1 again, for the search pattern set with $|\mathbb{W}^3| = 30$ of 3-round NOEKEON, set of values at the narrowest point is $\{2, 4, 6\}$, while set of values at the first point is $\{2, 4, 6, 12, 13, 14\}$. By organizing search patterns from the narrowest points, more nodes and longer prefix-paths can be shared. From Fig. 2 we can deduce that sharing is maximized among search patterns by organizing from the narrowest point:
   - There are only 7 nodes at the first two layers of the latter structure Fig. 2b, while 14 nodes at that of the former Fig. 2a.
   - More patterns can share search prefix-pathes in the latter structure Fig. 2b than in the former Fig. 2a.
2. The second reason which makes the *starting from the narrowest point strategy* work better is that the smaller the weight, the less the number of candidate round differentials. For most block ciphers, the number of candidate round differentials with larger weight is much more than that with smaller weight. That is because, for a single S-Box, take the 4×4-bit S-Box of NOEKEON for example, the number of differential pairs with weight 3 is 72 while the number of differential pairs with weight 2 is only 24. Moreover, round differentials with smaller weight usually have less active S-Boxes. In the starting round, the less the number of active S-Boxes, the less the number of candidate round differentials. Table 1 shows the explosive increase of number of one-round candidates with the increase of one-round weight.
3. The third reason is that by starting from the narrowest point, restriction are more stringent, backtracking in invalid path could arise early. This dues to the diffusion property within small number of rounds of the cipher. Take NOEKEON for example and limit the number of active S-Boxes of one round being less than 18, if the preceding round is with one active S-Box, there are 12 trails[4] propagating through the succeeding round, and if the preceding round is with two active S-Boxes, there are 981 trails. The number of trails that could propagate through the succeeding round increase explosively with the increase of the number of active S-Boxes in the preceding round, which can be seen in [6, Appendix A]. Combining with the fact that number of differential pairs with weight 3 is much more than that with weight 2 for active S-Boxes, smaller weight of the preceding round leads a narrower range of possible value of weight of the succeeding round, and less number of differential trails which could propagate through the succeeding rounds. Accordingly, for a given search pattern, if the search procedure starts from the narrowest point, there is much less candidate differential trails within the starting

[4] The number is up to rotation equivalence for NOEKEON.

two rounds and to search in the search patterns which do not exist in reality, the workload taken by the search procedure before it being blocked at some nodes in search trees will be much smaller than that of starting from an arbitrary point.

Table 1. Numbers of candidates for one-round differential under various weight

w^1	2	3	4	5
CN	24	72	$\binom{32}{2} \times 24^2$	$\binom{32}{2} \times 2 \times 24 \times 72$
	$\approx 2^{4.58}$	$\approx 2^{6.17}$	$\approx 2^{18.12}$	$\approx 2^{20.71}$
w^1	6	7	8	9
CN	$\binom{32}{2} \times 72^2 + \binom{32}{3} \times 24^3$	$\binom{32}{3} \times 3 \times 24^2 \times 72$	$\binom{32}{3} \times 3 \times 24 \times 72^2 + \binom{32}{4} \times 24^4$	$\binom{32}{3} \times 72^3 + \binom{32}{4} \times 4 \times 72 \times 24^3$
	$\approx 2^{26.08}$	$\approx 2^{29.20}$	$\approx 2^{33.68}$	$\approx 2^{37.08}$

CN: Numbers of one-round candidates.

Suppose there are 32 identical 4 × 4-bit S-Boxes in one round of the cipher and we ignore the rotation equivalence.

Suppose number of differential pairs with weight 3 is 72 and that with weight 2 is 24 in the DDT of an S-Box.

5.2 Experimental Results of Starting from the Narrowest Point Strategy

Table 2 includes the experimental results comparison between search with the *starting from the narrowest point strategy* and search without the strategy. It can be seen that hundreds of speed up ratio are reached by adopting the *starting from the narrowest point strategy* from column "Time-First", column "Time-Narrowest" and column "Ratio-(F/N)".

At the end of this section, we would like to point out the following.

When the starting point is the first round, it is sufficient to test only one of the input difference compatible with each output difference under the first round weight. Similarly, it is sufficient to test whether there exist one output difference compatible with the input difference under the weight of the last round. We call this the *free ends equivalent effect*.

When the narrowest point is internal round, *free ends equivalent effect* should also be considered. There are both forward and backward propagations in branches of search trees. For the last round of the forward propagation and last round of the backward propagation, which are the real last round and the first round of the current n-round cipher respectively, it is sufficient to test only one of the output differences compatible with the input difference. Besides, similar with the above discussed *free ends equivalent effect*, there is a *starting point equivalence*. Specifically, when output difference of the starting round is fixed, there are many input

differences of this round compatible with it. However, forward branches need to be traversed under only one of the compatible input differences of the starting round instead of all.

6 Concretizing and Grouping Search Patterns Strategy

It has been proved experimentally that the pre-search strategy is very powerful to filter out non-existent search patterns. However, as containers of results of pre-search, search patterns turn to be too abstract to store enough information on the underlying trails. To collect more information from preceding search phases, search patterns should be concretized to a proper extent while keeping the storage requirement appropriate. Thus, we propose the *concretizing and grouping search patterns strategy*. Besides, *starting point equivalence* can exist among various weights of narrowest point. That equivalence should be further considered to maximize shareable works at early phases of the procedure which dominate the search complexity.

6.1 Proposal and Justification of the Concretizing and Grouping Search Patterns Strategy

Concretizing: First, we append information of possible number of active S-Boxes at the narrowest point to each search pattern. Thus, a search pattern turns to be several concretized search patterns. For a search pattern $\mathbb{W}^n = (w_1, \ldots, w_m, \ldots, w_n)$ and its relative-index form $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_0, \ldots, \breve{w}_{n-m})$, its concretized search patterns are $\{\breve{W}^n\} = \{(\breve{w}_{-m+1}, \ldots, \binom{[asn]}{\breve{w}_0}, \ldots, \breve{w}_{n-m}) | asn \in [asn\_min, asn\_max]\}$ where $[asn\_min, asn\_max]$ is the range of possible ASN of round-differential at the narrowest round with round-weight $\breve{w}_0$.

Grouping: We then group the concretized search patterns according to the number of active S-Boxes at the narrowest point. For two search patterns having same possible ASN at the narrowest point:

1. $\mathbb{W}_1^n = (w_{1,1}, \ldots, w_{1,m_1}, \ldots, w_{1,n})$ and its relative-index form $\breve{\mathbb{W}}_1^n = (\breve{w}_{1,-m_1+1}, \ldots, \breve{w}_{1,0}, \ldots, \breve{w}_{1,n-m_1})$, and one of its concretized pattern $\breve{W}_1^n = (\breve{w}_{1,-m_1+1}, \ldots, \binom{[asn]}{\breve{w}_{1,0}}, \ldots, \breve{w}_{1,n-m_1})$
2. $\mathbb{W}_2^n = (w_{2,1}, \ldots, w_{2,m_2}, \ldots, w_{2,n})$ and its relative-index form $\breve{\mathbb{W}}_2^n = (\breve{w}_{2,-m_2+1}, \ldots, \breve{w}_{2,0}, \ldots, \breve{w}_{2,n-m_2})$, and one of its concretized pattern $\breve{W}_2^n = (\breve{w}_{2,-m_2+1}, \ldots, \binom{[asn]}{\breve{w}_{2,0}}, \ldots, \breve{w}_{2,n-m_2})$,

suppose

- $w_{1,m_1} \neq w_{2,m_2}$ and
- $w_{1,\mathrm{nxt}(m_1)} = w_{2,\mathrm{nxt}(m_2)} = v$,
- $\mathrm{rix}(w_{1,\mathrm{nxt}(m_1)}) = \mathrm{rix}(w_{2,\mathrm{nxt}(m_2)}) = i$ where $i \in \{1, -1\}$ and
- $w_{1,\mathrm{nxt}(\mathrm{nxt}(m_1))} \neq w_{2,\mathrm{nxt}(\mathrm{nxt}(m_2))}$,

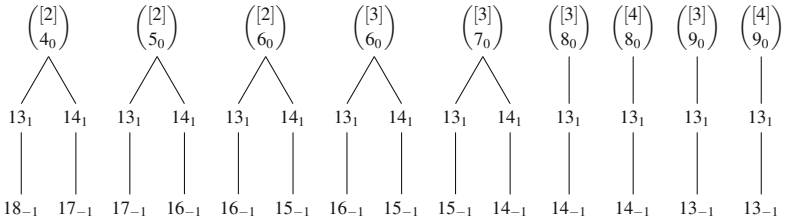
we group them as $\begin{pmatrix} [asn] \\ \{\check{w}_{1,0}, \check{w}_{2,0}\}_0 \end{pmatrix}$ | $v_i^{\{\check{w}_{1,0},\check{w}_{2,0}\}}$: $\check{w}_{1,-m_1+1}^{\{\check{w}_{1,0}\}}$: $\check{w}_{2,-m_2+1}^{\{\check{w}_{2,0}\}}$.

Figure 3a depicts the way how we concretize the search patterns by an example set of search patterns with $|\mathbb{W}^3| = 35$ of 3-round NOEKEON. Figure 3b depicts the way how we group the search patterns by the same example set of search patterns taken in Fig. 3a.

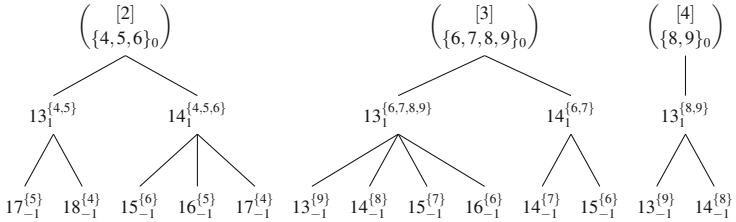
Then the search is processed group by group, instead of value by value of weight starting from the narrowest point. Grouping search patterns in such a way can bring two advantages.

1. Firstly, more specified knowledge of the search patterns will be learned during the pre-search phase. Information on allowed number of active S-Boxes at the narrowest point of a search pattern are stored in memory and can be used to filter search patterns in later process. For example, in Fig. 3b, search pattern $(16, 6, 13)$ is included in the tree with $[3]$[5] as the root node. Since a round differential with weight 6 is possible to have 2 active S-Boxes, $(16, 6, 13)$ should have been included in the tree with $[2]$ as the root node as well. However, during the former search of 2-round cipher, we learn that search pattern $(6, 13)$ does not exist when round differential with weight 6 has 2 active S-Boxes. Thus, $(16, 6, 13)$ can be deleted in the tree with $[2]$ as the root. Besides, since the allowed number of active S-Boxes at the narrowest point are usually small, memory requirement stays appropriate.
2. Secondly, once the search patterns are grouped according to the allowed number of active S-Boxes and starting from the output difference in the narrowest point, searches can share the forward propagation prefixes among different search patterns with various narrowest point weight. For example, in Fig. 3b search pattern $(13, 9, 13)$, $(14, 8, 13)$, $(15, 7, 13)$ and $(16, 6, 13)$ share the same prefix $([3], 13_1)$ when we search starting from an output difference with 3 active S-Boxes at the narrowest point.

[5] For simplicity, we use the number in square bracket to represent the root node (eg. $[3]$ is the shortening of $\begin{pmatrix} [3] \\ \{6, 7, 8, 9\}_0 \end{pmatrix}$).



(a) Concretizing search patterns by appending information of possible number of active S-Boxes at the narrowest point



(b) Grouping search patterns by number of active S-Boxes at the narrowest point. Superscript numbers in brace represent the narrowest point weight of the patterns in which the node belongs to. For example, search pattern $(14, 8, 13)$ is included in the branch with $\binom{[3]}{\{6,7,8,9\}_0}$ as the root node, $13_1^{\{6,7,8,9\}}$ as the first layer node, and $14_{-1}^{\{8\}}$ as the leaf. It is also included in the branch with $\binom{[4]}{\{8,9\}_0}$ as the root node, $13_1^{\{8,9\}}$ as the first layer node,and $14_{-1}^{\{8\}}$ as the leaf.

Fig. 3. Concretizing and grouping search patterns. Number in square bracket appended at root node of each tree represents the possible number of active S-Boxes at the narrowest point. Subscripts represent indices of the points relative to the starting points.

6.2 Experimental Results of Concretizing and Grouping Search Patterns

Table 2 summarizes the experimental results comparison between search with the *concretizing and grouping search patterns strategy* and searches without the strategy. Column "Ratio-(N/C)" shows a tens of times speedup ratio. A hundreds of speedup ratio are achieved combined with the efficiency brought by the first strategy shown as in column "Ratio-(F/C)".

7 Trialling in Minimal Changes Order Strategy

Once the set of search patterns is created, to obtain a differential trail, we only need to simply generate and concatenate round differentials under fixed round weights, if there exist any. By looking up the differential distribution table (DDT) of S-Box in the S-Layers and by executing the P-Layers between continuous rounds on the differences, round differentials can be constructed and connected to a differential trial. However, for NOEKEON and SPONGENT, execution of the

Table 2. Experimental results comparison between search *starting from the first point* (abbr. as "First" or "F"), search *starting from the narrowest point* (abbr. as "Narrowest" or "N"), and search *starting from the narrowest point* with the *concretizing and grouping search patterns strategy* (abbr. as "Concretize" or "C")

w^3	Time (mins)			Ratio		
	First	Narrowest	Concretize	F/N	N/C	F/C
28	7.43	0.11	0.01	67.55	11.00	743.00
29	8.01	0.98	0.01	8.17	98.00	801.00
30	375.60	1.10	0.44	341.45	2.50	853.64
31	375.88	1.11	0.45	338.63	2.47	835.29
32	2398.50	15.99	0.85	150.00	18.81	2821.76
33	-	16.65	0.91	-	18.30	-
34	-	16.77	1.08	-	15.53	-
35	-	165.54	1.56	-	106.12	-
36	-	172.82	30.97	-	5.58	-
37	-	177.73	33.70	-	5.27	-

Ratio: Ratio between two kinds of time.
Rows are separated by weight of 3-round trails in NOEKEON.
All of the experiments are done with the *trialling in minimal changes order strategy.*
Weight range of pre-search information of 2-round cipher is $[8, 31]$.

P-Layer will become the most costly part of the search process. That is because executing the P-Layer by looking up big tables which is a more suitable way of implementation in the case of searching for differential trails, the cost of P-Layer can be up to 10 times of the cost of generating a new candidate differential at the S-Layer. What is more, each replacement of candidate differential in a single S-Box at the S-Layer, calls for the replacement of differential at the P-Layer in full scope.

We avoid the full execution of the P-Layer considering that there is locality of individual S-Box within S-Layer and linearity of P-Layer, which makes local calculation feasible when generate round differentials. Further more, we propose the *trialling in minimal changes order strategy* to minimize the number of local calculation, thus to minimize the cost of generating round differentials. The following are explicit explanations.

If we can pre-calculate all 128-bit outputs corresponding to local nonzero inputs (let us take 4-bit for example hereafter) of P-Layer, we can get the output differences corresponding to the input differences which is locally active by simple XOR operations instead of the costly execution of P-Layer. Linear operation (XOR) on 128-bit (256-bit) data can resorting to the SSE and AVX instructions.

P-Layer operations can be further removed completely by planting the above 4×128-bit differences tables of P-Layer to the DDT of S-Box. We call each 128-bit SP-Layer output difference caused by a 4-bit (located at a single S-Box) S-Layer input difference the "128-bit contribution difference" to the 128-bit

round output difference. We generate table of 4-bit input differences and their 128-bit contribution differences which is called *contribution differential distribution table* (CDDT).

To minimize the cost of generating new candidate differences, we generate the new from the old with minimal local changes by removing and adding 128-bit contribution differences which can be done by looking up the CDDT and by simple XOR instructions. The following shows how we achieve the least number of looking up table and XOR instructions.

Candidate round differentials are characterized by the *weight patterns of active S-Boxes* and *indices* of their 4-bit candidate differences within each active S-Box. By *weight patterns of active S-Boxes*, we mean the possible compositions of partition one-round weight into weights of active S-Boxes. Take one-round differential weight as 10 and number of active S-Boxes as 4 for example, weight patterns of active S-Boxes are (3322),(2323),(2332),(3223),(3232) and (2233), which are restricted 4-compositions of 10. We then run through all the candidate output differences by enumerate the weight patterns of active S-Boxes with a light algorithm extended from [17] and run through all the indices within each weight pattern with an algorithm named "Loopless reflected mixed-radix Gray generation" in [18] to achieve minimal changes and least XOR operations. Generally, generating a new candidate only cost two XOR-operations. An example can be seen in Example 2.

Example 2. Following is an example of *trialling in minimal changes order strategy*. Assume there are 4 active S-Boxes in the round input difference, and input differences of the four active S-Boxes are 0x2, 0x1, 0x4 and 0x8. We need to run through all the compatible round output difference with round weight equals to 10. To minimize the cost, we try the weight patterns of 4 active S-Boxes (3322),(2323),(2332),(3223),(3232) and (2233) in an order as show in Fig. 4. Within a weight pattern, take (2233) for example, assume $\mathrm{num}(0x2, 2) = 3$, $\mathrm{num}(0x1, 2) = 2$, $\mathrm{num}(0x4, 3) = 3$ and $\mathrm{num}(0x8, 3) = 2$, where $\mathrm{num}(id, w)$ denotes the number of candidate output differences given input difference id and weight w of a single S-Box. We run through candidates by the indices in an order as shown in Fig. 5.

That bring us at least double times speedup. An intuition understanding for the *trialling in minimal changes order strategy* is that, we utilize the small change effect and large scale effect. Small change effect means that generating the new from the old might be much cheaper than generate from nothing if the changes are subtle. Large scale effect means that doing things in large scale can be more economical and efficient. The cost to systemically finish the whole is much less than the sum of cost to separately finish each individual.

$$(3322) \Rightarrow (3\underline{\mathbf{23}}2) \Rightarrow (\underline{\mathbf{23}}32) \Rightarrow (23\underline{\mathbf{23}}) \Rightarrow (\underline{\mathbf{32}}23) \Rightarrow (\underline{2}2\underline{3}3)$$

Fig. 4. An example of trialling weight patterns of active S-Boxes in minimal changes order. Bold numbers with underbreves are items exchanged from the former weight pattern. There are only 2 changes at each step.

1111 ⇒ 111**2** ⇒ 11**2**2 ⇒ 112**1** ⇒ 11**3**1 ⇒ 113**2** ⇒ 1**2**32 ⇒ 123**1**
⇓
222**1** ⇐ 22**2**2 ⇐ 221**2** ⇐ **2**211 ⇐ 121**1** ⇐ 12**1**2 ⇐ 122**2** ⇐ 12**2**1
⇓
22**3**1 ⇒ 22**3**2 ⇒ 2**1**32 ⇒ 213**1** ⇒ 21**2**1 ⇒ 212**2** ⇒ 21**1**2 ⇒ 211**1**
⇓
323**1** ⇐ 3**2**32 ⇐ 313**2** ⇐ 31**3**1 ⇐ 312**1** ⇐ 31**2**2 ⇐ 311**2** ⇐ **3**111
⇓
32**2**1 ⇒ 322**2** ⇒ 32**1**2 ⇒ 321**1**

Fig. 5. An example of trialling candidates within a weight pattern in the mixed-radix Gray code order. Bold numbers with a underbreve are the unique item changed from the former.

8 Results on Best Trails of Noekeon and Spongent

8.1 Object Cipher - A Block Cipher NOEKEON

NOEKEON is a self-inverse block cipher with a block and key length of 128-bit. It is a 16 rounds iterated cipher with a round transformation composed of transformations Theta, Pi1, Gamma, Pi2 and XORing a Working Key. The round transformation can be split into two parts - nonlinear part Gamma and linear part Lambda = (Pi1 ∘ Theta ∘ Pi2). Gamma can be specified as the S-layer which is a parallel execution of 32 4×4-bit identical S-Boxes. Lambda can be seen as the P-layer. A full description of NOEKEON can be found in [6]. By searching the complete space of 4-round trails (both linear and differential) with less than 24 active S-Boxes, the designers can guarantee that there are no 4-round differential (linear) trails with a predicted probability (bias) above 2^{-48} (2^{-25}).

In this work, by adopting the proposed three optimization strategies and making use of the symmetry properties, these statements are confirmed and further refined. They turn to be as follow: of all 4-round differential (linear) trails, the *best* has a probability (bias) equals to 2^{-51} (2^{-25}). Figure 6 in Appendix B shows one of the best 4-round differential trails. It takes 21 (1.2) hours to systematically investigate whether 4-round differential (linear) trails of weight up to 51 (25) exist on the formerly mentioned PC. Table 3 summarizes more results about differential (linear) trails of NOEKEON we have achieved.

Besides, *best* 6-round and 9-round linear trails with bias 2^{-40} and 2^{-62} are found out and Fig. 7 in Appendix B shows one of them. The 9-round linear trail is the longest one holding with bias larger than 2^{-65}. By observing that the internal part in the best 6-round linear trail is iterative on a 2-round trail which is also a sub-trail in the best 9-round, a 10-round linear trail with bias 2^{-68} can be constructed.

According to the results in Table 3, the probability (bias) of best 10-round (11-round) differential (linear) trails in NOEKEON is at most 2^{-131} (2^{-71}). As mentioned in [6], for a DC attack to exist, there must be a predictable difference propagation over all but a few rounds with a probability significantly larger than 2^{-127}, and LC attacks are possible if there are predictable input-output

correlation values (2 times of bias) over all but a few rounds significantly larger than 2^{-64}, thus we can benefit from these results that exclude classical DC (LC) attacks on NOEKEON.

8.2 Object Cipher - A Hash Function SPONGENT

SPONGENT is a lightweight hermetic sponge hash function with a PRESENT-type permutation [16]. There are 13 variants with 11 kinds of permutation width. In this work, variants with permutation width $b \in \{88, 136, 176, 240\}$ are considered.

By applying the three proposed optimization strategies, *best* differential trails corresponding to the number of rounds in [16, Table 3] are searched. The results suggest that finding out the unconditional best trails could help to establish more tight upper bound on the probability of the best differential than that provided by finding the trails with minimal number of active S-Boxes. New results are

Table 3. Comparison between results from specification of NOEKEON and results from this work. Entries with * are the updates due to this work. Note that the trails we found are confirmed to be the best.

#Rounds	NOEKEON-differential				NOEKEON-linear			
	Spec.		This		Spec.		This	
	ASN	Prob.	ASN	Prob.	ASN	Bias	ASN	Bias
1	1	2^{-2}	1	2^{-2}	1	2^{-2}	1	2^{-2}
2	4	2^{-8}	4	2^{-8}	4	2^{-5}	4	2^{-5}
3	-	-	*13	$^*2^{-28}$	-	-	*13	$^*2^{-14}$
4	-	$\leq 2^{-48}$	*22	$^*2^{-51}$	-	$\leq 2^{-25}$	*21	$^*2^{-25}$
5	-	-	-	$^*\leq 2^{-65}$	-	-	-	$^*\leq 2^{-32}$
6	-	-	-	$^*\leq 2^{-80}$	-	-	*33	$^*2^{-40}$

Table 4. Comparison between results from specification of SPONGENT and results from this work. Entries with * are the updates due to this work.

#Rounds	$b = 88$				$b = 136$			
	Spec.		This		Spec.		This	
	ASN	Prob	ASN	Prob	ASN	Prob	ASN	Prob
5	10	2^{-21}	10	$^*2^{-20}$	10	2^{-22}	10	$^*2^{-20}$
10	20	2^{-47}	20	2^{-47}	24	2^{-60}	*22	$^*2^{-55}$
15	30	2^{-74}	30	2^{-74}	40	2^{-101}	*43	$^*2^{-96}$

#Rounds	$b = 176$				$b = 240$			
	Spec.		This		Spec.		This	
	ASN	Prob	ASN	Prob	ASN	Prob	ASN	Prob
5	10	2^{-21}	10	$^*2^{-20}$	10	2^{-21}	10	$^*2^{-20}$
10	20	2^{-50}	20	$^*2^{-46}$	20	2^{-43}	20	2^{-43}
15	30	2^{-79}	30	2^{-79}	30	2^{-66}	30	2^{-66}

listed in Table 4. Figures 8 and 9 in Appendix B depicts two of the updated best trails. Besides, longer differential trails are found, which could correct and update the results listed in [16, Table 4]:

- For variants with $b = 88$, probability of *best* 17-round (18-round) differential trail is 2^{-86} (2^{-94}), which was found out within 1 min, and shown in Fig. 10 in Appendix B.
- For variants with $b = 240$, probability of *best* 44-round differential trail is 2^{-196}, which was found within 1 min. By observing the results up to 44-round, we can conclude the following: $Bw^6 = 30$ and for $r \geq 7$, $Bw^r = \begin{cases} Bw^{r-1} + 4 & if\ r\ is\ even \\ Bw^{r-1} + 5 & if\ r\ is\ odd \end{cases}$. An observation is that there is a 2-round iterative trail with weight pattern (4, 5) composing the best trails, as shown in Fig. 11 in Appendix B.
- For variants with $b = 176$, *best* 17-round (18-round) differential trail with weight 91 (99) was found within 50 min, and shown in Fig. 12 in Appendix B.

9 Conclusion and Future Work

We improved the search algorithm for the best differential and best linear trails by introducing three optimization strategies. Those strategies reduce the complexity to a great extent by organizing candidate search patterns properly, collecting more information during preceding procedures and trialling in good order, which allowed us to find out best trails more efficiently. At the end, we briefly overview future work.

- We trial a search pattern in an order of $(\breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m}, \breve{w}_{-1}, \ldots, \breve{w}_{-m+1})$. As an anonymous reviewers suggested, it might also be interesting to consider the order $(\breve{w}_0, \breve{w}_1, \breve{w}_{-1}, \breve{w}_2, \breve{w}_{-2}, \ldots)$.
- Experimental result on NOEKEON shows that search patterns that cannot be filtered out by the information collected in preceding procedures are usually with fat paunches, while the existing search patterns are usually with narrow waists. That can be understood considering the diffusion property of the target cipher. How to use this empirical knowledge to add heuristics to the search algorithm remains unclear.
- By avoiding detailed properties of the target ciphers, our algorithm is general to some extent, while remaining space for further improvement by utilizing more special properties of the object ciphers.
- Strategies proposed in this paper are also helpful to generate all the trails up to a given weight. Thus, they can be adopted when search for the best multiple differential (multi-dimensional linear) distinguishers. While, we haven't adopted them to the case of related-key differential.

Acknowledgement. Many thanks go to the anonymous reviewers for many useful comments and suggestions. The research presented in this paper is supported by the National Natural Science Foundation of China (No.61379138), the "Strategic Priority Research Program" of the Chinese Academy of Sciences (No.XDA06010701).

A Our Search Algorithm

Algorithm 4. Our Search Approach Part 1 - Framework of Our Search Approach

for $n \leftarrow 1$, $RoundN$ **do** ▷ n is the current number of round and $RoundN$ is the total of that considered for the cipher. All of the following variables are global which can be directly accessed in each procedure. Superscript n on the shoulders of those variables is changed with the value of n, thus for different values of n, they are different variables.
  $w\_l^n \leftarrow$ a lower bound for Bw^n
  $n\_w^n \leftarrow$ the number of extra values of weights larger than Bw^n ▷ Determine the amount of pre-search information of n-round cipher, there is no pre-search information of n-round cipher if $n\_w^n = -1$
  $w\_u^n \leftarrow 0$ ▷ w_u^n will be update as the upper bound for values of weights at the end of procedure SearchForRounds, and thus $[w\_l^n, w\_u^n]$ is the range of values of w^n under which we will completely examine the existence of the search patterns.
  $Bw^n \leftarrow \infty$
  $w^n \leftarrow w\_l^n$ ▷ w^n is a global temp variable holding current weight of n-round
  $ExistentPatterns^n \leftarrow \varnothing$ ▷ Trees of existent search patterns under $w^n \in [Bw^n, w\_u^n]$
  $ExistentPatternsCG^n \leftarrow \varnothing$ ▷ Trees of existent concretized patterns under $w^n \in [Bw^n, w\_u^n]$
  $SearchPatterns^n \leftarrow \varnothing$ ▷ Temporary trees of search patterns
  $SearchPatternsCG^n \leftarrow \varnothing$ ▷ Temporary trees of concretized search patterns
  SearchForRounds(n)
  Next-n: ▷ A tag for long jump
end for

procedure SearchForRounds(n)
  $i \leftarrow -1$
  while $i \leq n\_w^n$ **do**
    GenerateSearchPatterns ▷ Generate, filter and formalize search patterns
    OrganizeSearchPatterns ▷ Concretize and group search patterns
    SearchFromTheNarrowest ▷ Search in trees of organized search patterns
    $SearchPatterns^n \leftarrow \varnothing$ ▷ Clear trees of search patterns
    $SearchPatternsCG^n \leftarrow \varnothing$ ▷ Clear trees of concretized search patterns
    $w^n \leftarrow w^n + 1$
    if $Bw^n \neq \infty$ **then** $i \leftarrow i + 1$
    end if
  end while
  $ExistentPatterns^n \leftarrow_{gather} ExistentPatternsCG^n$ ▷ Gather information on existence of search patterns $\breve{\mathbb{W}}^n$ according to the information on existence of corresponding concretized search patterns $\breve{W}^n$.
  $w\_u^n \leftarrow Bw^n + n\_w^n$
end procedure

Algorithm 5. Our Search Approach Part 2 - Generate and Filter, Concretize and Group Search Patterns

procedure GENERATESEARCHPATTERNS
  while partition of w^n could generate a new n-composition **do**
    Partition w^n into n components to form a new possible search pattern $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$, which make the following hold:
    1. $|\mathbb{W}^n| = \sum_{i=1}^{n} w_i = w^n$,
    2. $\sum_{i=1}^{n-r} w_i + Bw^r \leq w^n$ for $\forall r$ $(1 \leq r \leq n-1)$, and
    3. $w^r \geq Bw^r$ and $\mathbb{W}^r = (w_i, w_{i+1}, \ldots, w_{i+r-1}) \in ExistentPatterns^r$ if $w^r \in [Bw^r, w\_u^r]$, for $\forall r, i$ $(1 \leq r \leq n-1,\ 1 \leq i \leq n-r+1)$, where $w^r = \sum_{j=i}^{i+r-1} w_j$.
    Find the narrowest point of $\mathbb{W}^n$, let it be m. ▷ If the cipher is involution, let $\bar{\mathbb{W}}^n = (w_n, w_{n-1}, \ldots, w_1)$ and $\bar{w}_{\bar{m}}$ be the narrowest point of $\bar{\mathbb{W}}^n$. If $w_{\mathrm{nxt}(m)} = \bar{w}_{\mathrm{nxt}(\bar{m})}$, let $\breve{\mathbb{W}}^n$ be the relative-index form of $\bar{\mathbb{W}}^n$, if $\breve{\mathbb{W}}^n \in SearchPatterns^n$, discard $\mathbb{W}^n$ and continue. Else, if $w_{\mathrm{nxt}(m)} > \bar{w}_{\mathrm{nxt}(\bar{m})}$, discard $\mathbb{W}^n$ and $\mathbb{W}^n \leftarrow \bar{\mathbb{W}}^n$.
    Turn $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$ into its relative-index form $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_0, \ldots, \breve{w}_{n-m})$.
    $SearchPatterns^n \leftarrow_{insert} \breve{\mathbb{W}}^n$
  end while
end procedure

procedure ORGANIZESEARCHPATTERNS
  for all $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_0, \ldots, \breve{w}_{n-m}) \in SearchPatterns^n$ **do**
    $asn\_min \leftarrow$ the minimal possible ASN determined by $\breve{w}_0$
    $asn\_max \leftarrow$ the maximal possible ASN determined by $\breve{w}_0$
    for $asn \leftarrow asn\_min, asn\_max$ **do**
      for all $r, \dot{r}, (2 \leq r < n, 1 \leq \dot{r} \leq r)$ **do**
        if $\breve{w}_0$ is at the narrowest point of reduced search pattern $(\breve{w}_{-\dot{r}+1}, \ldots, \breve{w}_0, \ldots, \breve{w}_{r-\dot{r}})$ **then**
          Let $\breve{W}^r \leftarrow (\breve{w}_{-\dot{r}+1}, \ldots, \binom{[asn]}{\breve{w}_0}, \ldots, \breve{w}_{r-\dot{r}})$ and $\breve{w}^r \leftarrow \sum_{i=-\dot{r}+1}^{r-\dot{r}} \breve{w}_i$, then
          if $\breve{w}^r \in [Bw^r, w\_u^r]$ and $\breve{W}^r \notin ExistentPatternsCG^r$ **then** goto next asn
          end if
        end if
      end for
      $SearchPatternsCG^n \leftarrow_{groupingInto} \breve{W}^n = (\breve{w}_{-m+1}, \ldots, \binom{[asn]}{\breve{w}_0}, \ldots, \breve{w}_{n-m})$
    end for
  end for
end procedure

Algorithm 6. Our Search Approach Part 3 - Search in Trees of Organized Search Patterns

procedure SEARCHFROMTHENARROWEST
  for all $rootnode \leftarrow \binom{[asn]}{\{\breve{w}_{1,0}, \breve{w}_{2,0}, \ldots, \breve{w}_{k,0}\}_0} \in SearchPatternsCG^n$ **do**
    $asn_0 \leftarrow asn$
    $asn\_min_1 \leftarrow$ the allowed minimal ASN determined by the minimal weight of the first forward rounds succeeding the current *rootnode*
    $asn\_max_1 \leftarrow$ the allowed maximal ASN determined by the maximal weight of the first forward rounds succeeding the current *rootnode*
    $asn\_min_{-1} \leftarrow$ the allowed minimal ASN determined by the minimal weight of the first backward rounds succeeding the forward branches of the current *rootnode*
    $asn\_max_{-1} \leftarrow$ the allowed maximal ASN determined by the maximal weight of the first backward rounds succeeding the forward branches of the current *rootnode*
    while ∃ new candidate round-output-differential od_0 **do**
      Generate a new od_0 with the *trailling in minimal changes order strategy*, which makes the following hold:
      1. there are asn_0 active S-Boxes at the narrowest round, and
      2. $asn_1 \in [asn\_min_1, asn\_max_1]$, where asn_1 is ASN at the first succeeding forward round, which is computed from od_0.
      while ∃ new candidate round-input-differential id_0 **do**
        Generate a new id_0 with the *trailling in minimal changes order strategy*, which makes the following hold:
        1. id_0 is compatible with od_0 with $w_{(id_0,od_0)} \in \{\breve{w}_{1,0}, \breve{w}_{2,0}, \ldots, \breve{w}_{k,0}\}$,
        2. $asn_{-1} \in [asn\_min_{-1}, asn\_max_{-1}]$, where asn_{-1} is ASN at the first backward round, which is computed from id_0.
        $idListAtNarrowestPoint^{w_{(id_0,od_0)}} \leftarrow_{insert} id_0$
      end while
      if ∃ succeeding forward branches under $node_r$ **then**
        SEARCHFORWARD(od_0, 1)
      end if
      for all $\breve{w}_{i,0} \in \{\breve{w}_{1,0}, \breve{w}_{2,0}, \ldots, \breve{w}_{k,0}\}$ **do**
        if ∃ succeeding backward branches with $\breve{w}_{i,0}$ as the narrowest point under $node_r$ **then** ▷ For simplicity henceforth, we say *branches with* $\breve{w}_{i,0}$ *as the narrowest point* if there are nodes carry superscript including $\breve{w}_{i,0}$ on their shoulders.
          for all $id_0 \in idListAtNarrowestPoint^{\breve{w}_{i,0}}$ **do**
            SEARCHBACKWARD($\breve{w}_{i,0}$, id_0, -1)
          end for
        end if
      end for
    end while
  end for
end procedure

Algorithm 7. Our Search Approach Part 4 - Search in Forward Branches

86: **procedure** SEARCHFORWARD(id_r, r)
87: **for all** $node_r \leftarrow \breve{w}_r^{\{\breve{w}_{1,0},\ldots,\breve{w}_{j,0}\}} \in$ forward branches succeeding the preceding node in $SearchPatternsCG^n$ **do**
88: **if** $\exists$ succeeding forward branches under $node_r$ **then**
89: **while** $\exists$ new candidate od_r **do**
90: Generate a new od_r with the *trailling in minimal changes order strategy*, which makes the following hold:
91: 1. od_r is compatible with id_r with $w_{(id_r,od_r)} = \breve{w}_r$,
92: 2. $asn_{r+1} \in [asn\_min_{r+1}, asn\_max_{r+1}]$, where asn_{r+1} is ASN at the succeeding forward round, which is computed from od_r, and asn_min_{r+1} (or asn_max_{r+1}) is determined by the minimal (or maximal) weight of nodes on the succeeding forward branches of $node_r$.
93: SEARCHFORWARD(od_r, $r+1$)
94: **end while**
95: **end if**
96: **if** $\exists$ succeeding backward branches under $node_r$ or $\nexists$ succeeding branches under $node_r$ **then**
97: **if** $\exists$ an od_r compatible with id_r with $w_{(id_r,od_r)} = \breve{w}_r$ **then**
98: **if** $\exists$ succeeding backward branches under $node_r$ **then**
99: **for all** $\breve{w}_{i,0} \in \{\breve{w}_{1,0}, \ldots, \breve{w}_{j,0}\}$, $1 \le i \le j$ **do**
100: **if** $\exists$ succeeding backward branches with $\breve{w}_{i,0}$ as the narrowest point under $node_r$ **then**
101: **for all** $id_0 \in idListAtNarrowestPoint^{\breve{w}_{i,0}}$ **do**
102: SEARCHBACKWARD($\breve{w}_{i,0}$, id_0, -1)
103: **end for**
104: **end if**
105: **end for**
106: **else**
107: **if** this is the first time get an n-round differential trail **then** $Bw^n \leftarrow w^n$
108: **end if**
109: **if** $n\_w^n = -1$ **then** goto Next-n
110: **end if**
111: $\breve{W} \leftarrow_{delete} SearchPatternsCG^n$ ▷ Delete current concretized search pattern from trees of concretized search patterns under current w^n
112: $ExistentPatternsCG^n \leftarrow_{insert} \breve{W}$ ▷ Save current concretized search pattern to trees of actually existent concretized search patterns under $w^n \in [Bw^n, w\_u^n]$
113: **end if**
114: **end if**
115: **end if**
116: **end for**
117: **end procedure**

Algorithm 8. Our Search Approach Part 5 - Search in Backward Branches

118: **procedure** SEARCHBACKWARD($\breve{w}_{i,0}$, od_r, r)
119: **for all** $node_r \leftarrow \breve{w}_r^{\{\breve{w}_{1,0},\ldots,\breve{w}_{j,0}\}} \in$ backward branches succeeding the preceding node in $SearchPatternsCG^n$ **do**
120: **if** $\breve{w}_{i,0} \in \{\breve{w}_{1,0}, \ldots, \breve{w}_{j,0}\}$ **then**
121: **if** $\exists$ succeeding backward branches under $node_r$ **then**
122: **while** $\exists$ new candidate id_r **do**
123: Generate a new id_r with the *trailling in minimal changes order strategy* making the following hold:
124: 1. id_r is compatible with od_r with $w_{(id_r,od_r)} = \breve{w}_r$,
125: 2. $asn_{r-1} \in [asn\_min_{r-1}, asn\_max_{r-1}]$, where asn_{r-1} is ASN at the succeeding backward round, which is computed from id_r, and asn_min_{r-1} (or asn_max_{r-1}) is determined by the minimal (or maximal) weight of nodes on the succeeding backward branches of $node_r$.
126: SEARCHBACKWARD($\breve{w}_{i,0}$, id_r, $r-1$)
127: **end while**
128: **else**
129: **if** $\exists$ id_r compatible with od_r with $w_{(id_r,od_r)} = \breve{w}_r$ **then**
130: **if** this is the first time get an n-round differential trail **then** $Bw^n \leftarrow w^n$
131: **end if**
132: **if** $n\_w^n = -1$ **then** goto Next-n
133: **end if**
134: $\breve{W} \leftarrow_{delete} SearchPatternsCG^n$ ▷ Delete current concretized search pattern from trees of concretized search patterns under current w^n
135: $ExistentPatternsCG^n \leftarrow_{insert} \breve{W}$ ▷ Save current concretized search pattern to trees of actually existent concretized search patterns under $w^n \in [Bw^n, w\_u^n]$
136: **end if**
137: **end if**
138: **end if**
139: **end for**
140: **end procedure**

B Examples of Best Trails

R	*id* of Gamma	*od* of Gamma	w_r
1	0x00000000000000001000020210c280001	0x0000000000000000c0000408e0182000c	17
2	0x0000000000000000c0000008e0000000c	0x00000000000000001000000210000000 1	8
3	0x00000000004000000400800004000000	0x00000000002000000020040000300000	12
4	0x00000000002000400021040000300000	0x00000000008000200008c020000800000	14

Fig. 6. A best 4-round differential trail with weight 51 in NOEKEON

R	*im* of Gamma	*om* of Gamma	Bias
1	0x000000000000000800004*c*0100000001	0x000000000000000500009201000000001	2^{-6}
2	0x000000000000000500009*a*2100000001	0x000000000000000500009*a*2100000001	2^{-9}
3	0x000000000000000500009201000000001	0x000000000000000500009201000000001	2^{-7}
4	0x000000000000000500009*a*2100000001	0x000000000000000500009*a*2100000001	2^{-9}
5	0x000000000000000500009201000000001	0x000000000000000500009201000000001	2^{-7}
6	0x000000000000000500009*a*2100000001	0x000000000000000800004*cc*100000001	2^{-7}

Fig. 7. A best 6-round linear trail with bias 2^{-40} in NOEKEON

R	S-Layer *id*	S-Layer *od*	w_r
1	0x000000000000000000000303000030003	0x000000000000000000000020200000020002	8
2	0x000000000000000000000504400000000	0x00000000000000000000040*cc*00000000	6
3	0x000000*c*0000000*b*0000000000000000000	0x0000008000000080000000000000000000	6
4	0x020200000000000000000000000000000	0x0*a*0*a*0000000000000000000000000000	4
5	0x500000000000000005000000000000000	0x400000000000000004000000000000000	4
6	0x000000002000100000000000000000000	0x000000005000500000000000000000000	5
7	0x000000000022000000000000000002200000	0x0000000000*aa*0000000000000005500000	10
8	0x003000000000006000300000000000060	0x002000000000000200020000000000040	10
9	0x000000000000000022001100000000000	0x000000000000000055005500000000000	10
10	0x000000000000330000000000000033000	0x000000000000220000000000000022000	8
11	0x00000000000000000000*c*0006000000000	0x000000000000000000010001000000000	5
12	0x000000000000000000000000000002200	0x000000000000000000000000000000*aa*00	4
13	0x000000030000000000000003000000000	0x000000020000000000000002000000000	4
14	0x000000000000000001000080000000000	0x000000000000000003000090000000000	4
15	0x000001000000000000000020000000008400	0x000003000000000000000*a*000000009*c*00	8

Fig. 8. A best 15-round differential trial with weight 96 in SPONGENT with $b = 136$

R	S-Layer *id*	S-Layer *od*	w_r
1	0x000000000050005000000000000000000000000000000000	0x000000000040004000000000000000000000000000000000	4
2	0x000000000000022000000000000000000000000000000000	0x000000000000066000000000000000000000000000000000	6
3	0x000000000000006000000000060000000000000000000000	0x000000000000001000000000010000000000000000000000	4
4	0x000000000000000000000000000000000020040000	0x000000000000000000000000000000000060060000	6
5	0x000000000000000000090000000009000000000000	0x000000000000000000008000000008000000000000	4
6	0x000008010000000000000000000000000000000000000000	0x000003030000000000000000000000000000000000000000	5
7	0x000000000000000000000500000000005000000000	0x000000000000000000000400000000004000000000	4
8	0x000000000000000010020000000000000000000000000000	0x000000000000000050050000000000000000000000000000	5
9	0x000000000000009000000000000000000009000000	0x000000000000000800000000000000000008000000	4
10	0x000100000400000000000000000000000000000000000000	0x000300000*c*0000000000000000000000000000000000	4

Fig. 9. A best 10-round differential trial with weight 46 in SPONGENT with $b = 176$

R	S-Layer *id*	S-Layer *od*	w_r
1	0x900900000000000900900	0x800800000000000800800	8
2	0x900090000000000000000	0x800080000000000000000	4
3	0x880000000000000000000	0x990000000000000000000	4
4	0xc00000000000000300000	0x10000000000000200000	5
5	0x00000000000000008200000	0x00000000000000005500000	6
6	0x00000000060000000060	0x00000000010000000010	4
7	0x00000000000000001002	0x00000000000000005005	5
8	0x00000000009000000009	0x00000000002000000004	6
9	0x00000000100200000000	0x00000000005005000000	5
10	0x00000009000000000900	0x00000008000000000800	4
11	0x00801000000000000000	0x00505000000000000000	5
12	0x000000*a*0000000000*a*0000	0x00000010000000010000	6
13	0x00000000000000008010	0x00000000000000005050	5
14	0x0000000000*a*0000000000*a*	0x00000000001000000001	6
15	0x00000000000000000801	0x00000000000000000505	5
16	0x00000000005000000005	0x00000000004000000004	4
17	0x00000000801000000000	0x00000000903000000000	4

Fig. 10. A best 17-round differential trial with weight 86 in SPONGENT with $b = 88$

R		Trail	w_r
1	S-Layer *id*	0x000000000000000000000000000000000009000000000009000000000000	4
	S-Layer *od*	0x000000000000000000000000000000000008000000000008000000000000	
2	S-Layer *id*	0x000000001001000000000000000000000000000000000000000000000000	4
	S-Layer *od*	0x000000003003000000000000000000000000000000000000000000000000	
3	S-Layer *id*	0x000000000000000000000000000000000009000000000009000000000000	4
	S-Layer *od*	0x000000000000000000000000000000000008000000000008000000000000	
4	S-Layer *id*	0x000000008001000000000000000000000000000000000000000000000000	5
	S-Layer *od*	0x000000003003000000000000000000000000000000000000000000000000	
⋮	⋮	2-round iterative trails as (R3, R4) where R3 and R4 are the differential of the third and the fourth round	⋮
43	S-Layer *id*	0x000000000000000000000000000000000009000000000009000000000000	4
	S-Layer *od*	0x000000000000000000000000000000000008000000000008000000000000	
44	S-Layer *id*	0x000000008001000000000000000000000000000000000000000000000000	4
	S-Layer *od*	0x000000009003000000000000000000000000000000000000000000000000	

Fig. 11. A best 44-round differential trial with weight 196 in SPONGENT with $b = 240$

R	S-Layer *id*	S-Layer *od*	w_r
1	0x00000000000000000000000033000000000000033	0x00000000000000000000000022000000000000022	8
2	0x00000000000000000000000030003000000000000	0x00000000000000000000000020002000000000000	4
3	0x00000000000000000000000088000000000000000	0x00000000000000000000000099000000000000000	4
4	0x00000006000000000000000000000000000000006000	0x00000001000000000000000000000000000000002000	5
5	0x00000000000000000000000008010000000000	0x00000000000000000000000003030000000000	5
6	0x0000000000000000000000000000*a*0000000000*a*00	0x00000000000000000000000000001000000000100	6
7	0x00000000000000000000000000000000000000002004	0x00000000000000000000000000000000000000006006	6
8	0x00000000000000000009000000000900000000000	0x00000000000000000004000000000400000000000	6
9	0x00000000000000040080000000000000000000000	0x00000000000000030030000000000000000000000	6
10	0x00000000000000000000000009000000000900000	0x00000000000000000000000020000000002000000	6
11	0x00000000000000000000000002004000000000000	0x00000000000000000000000006006000000000000	6
12	0x00000000000000000900000000090000000000000	0x00000000000000000800000000080000000000000	4
13	0x00002004000000000000000000000000000000000000	0x00006006000000000000000000000000000000000000	6
14	0x00000000000090000000009000000000000000000	0x00000000000020000000002000000000000000000	6
15	0x00000000000000000000000801000000000000000	0x00000000000000000000000505000000000000000	5
16	0x00000000000000005000000000000000000000050000	0x00000000000000004000000000000000000000040000	4
17	0x00000000000000040000100000000000000000000000	0x0000000000000000*c*0000300000000000000000000000	4
18	0x00010000000001000000000008000000000800000	0x00030000000003000000000009000000009000000	8

Fig. 12. A best 18-round differential trial with weight 99 in SPONGENT with $b = 176$

References

1. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
2. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
3. Ohta, K., Moriai, S., Aoki, K.: Improving the search algorithm for the best linear expression. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 157–170. Springer, Heidelberg (1995)
4. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improved and multiple linear cryptanalysis of reduced round serpent. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Information Security and Cryptology. LNCS, vol. 4990, pp. 51–65. Springer, Heidelberg (2008)
5. Daemen, J., Rijmen, V.: The Design of Rijndael - AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
6. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: Nessie Proposal: The Block Cipher NOEKEON. Nessie submission (2000)
7. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

8. Biryukov, A., Nikolić, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
9. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012)
10. 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, Heidelberg (2013)
11. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
12. Aoki, K., Kobayashi, K., Moriai, S.: Best differential characteristic search of FEAL. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 41–53. Springer, Heidelberg (1997)
13. Leurent, G.: Construction of differential characteristics in ARX designs application to skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 241–258. Springer, Heidelberg (2013)
14. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 227–250. Springer, Heidelberg (2014)
15. Daemen, J.: Cipher and hash function design strategies based on linear and differential cryptanalysis. Doctoral Dissertation, March 1995, K.U.Leuven (1995)
16. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: SPONGENT: the design space of lightweight cryptographic hashing. IEEE Trans. Comput. **62**(10), 2041–2053 (2013)
17. Ehrlich, G.: Loopless Algorithms for Generating Permutations, Combinations, and Other Combinatorial Configurations. Journal of the ACM **20**(3), 500–513 (1973)
18. Knuth, D.E.: The Art of Computer Programming. Introduction to Combinatorial Algorithms and Boolean Functions, vol. 4. Addison Wesley, Upper Saddle River (2008)