The Resistance of PRESENT-80 Against Related-Key Differential Attacks

Sareh Emami[2], San Ling[1], Ivica Nikolić[1⋆], Josef Pieprzyk[2] and Huaxiong Wang[1]

[1] Nanyang Technological University, Singapore
[2] Macquarie University, Australia

Abstract. We examine the security of the 64-bit lightweight block cipher PRESENT-80 against related-key differential attacks. With a computer search we are able to prove that no related-key differential characteristic exists with probability higher than 2^{-64} for the full-round PRESENT-80. To overcome the exponential (in the state and key sizes) computational complexity we use truncated differences, however as the key schedule is not nibble oriented, we switch to actual differences and apply early abort techniques to prune the tree-based search. With a new method called extended split approach we are able to make the whole search feasible and we implement and run it in real time. Our approach targets the PRESENT-80 cipher however, with small modifications can be reused for other lightweight ciphers as well.

1 Introduction

PRESENT [5] is a lightweight block cipher and a current ISO standard. Arguably it is one of the most popular lightweight ciphers, and the first among 64-bit ciphers to reach the bound of around 1000 GE in a hardware implementation. The submission document of PRESENT gives a thorough security analysis of the cipher against various types of attacks. The initial analysis has been extended with several attacks on round-reduced PRESENT [24, 15, 18, 6, 7, 4, 25]. In this work, we present another complementary security analysis of PRESENT-80 against related-key differential attacks. This analysis model gives the attacker the most freedom, and in many cases leads to attacks on a larger number of rounds of the cipher or even to attacks on the full cipher. For example, the standard AES-256 [9] based on the wide-trail strategy is provably resistant against single-key differential attacks, however a high probability related-key differential characteristic exists on all 14 rounds [1]. Note that most of the recently published (lightweight) ciphers LED [10], Piccolo [20], TWINE [22], and CLEFIA [21] already have upper bounds on probabilities of such characteristics, and their designers have proved that no high probability related-key characteristic exists for the full-round ciphers. However, each of these ciphers is word-oriented (more precisely all[3] of the operations are nibble-oriented), and finding the upper bound on the probability is much simpler task due to the availability of automatic search tools for such characteristics [2]. Moreover some ciphers were specially designed to be resistant against related-key attacks, for example LED has round key additions only after every four AES-like rounds, while CLEFIA has highly non-linear key schedule.

In this work, we try to find related-key differential characteristics for the 64-bit cipher PRESENT-80 that hold with probability higher than 2^{-64} for all 31 rounds. As the cipher is not nibble-oriented due to the rotations in the key schedule, neither theory nor the existing tools can provide such characteristics. To achieve this goal we design an ad-hoc automated search approach and find an upper bound on all high probability related-key characteristics. To make the search feasible, we use a novel technique, which we call extended split approach. We combine it with a special representation for the difference in the state, which is truncated in most of the parts and can be switched to actual difference when it is needed. Also we show how to discard most of the invalid characteristics found at

⋆ The researcher is supported by the Singapore National Research Foundation Fellowship 2012 NRF-NRFF2012-06.

[3] CLEFIA has non-nibble oriented operations in the key schedule, however they come only after the non-linear operations which are used for proving the upper bound.

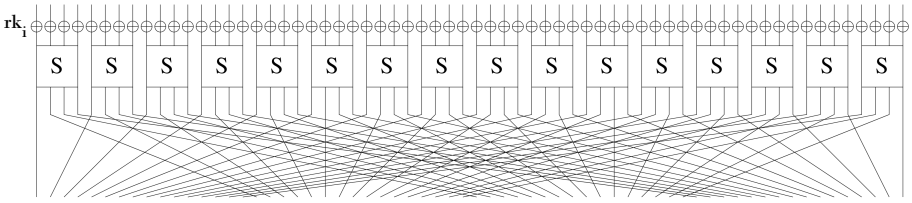

Fig. 1. PRESENT-80 round function.

the early stage of search by using several different filters. Based on the output of the search, we are able to provide and prove an upper bound on the probability of the best related-key characteristic for 8 rounds of PRESENT-80 – in the submission paper the designers obtain the same bound (see Theorem 1 in [5]), but for 5 rounds and single-key characteristic. Furthermore, we conclude that no related-key differential characteristic exists for full-round PRESENT-80 with probability higher than 2^{-64}. Thus this cipher does not have any obvious weakness against related-key attacks and along with the other competitive lightweight ciphers LED, Piccolo, TWINE enjoys provable security against related-key differential attacks.

The paper is structured as follows. We start with a description of PRESENT-80 in Section 2. A brief review of the previous results in automatic search for optimal differential characteristics is given in Section 3. Our new approach is described in Section 4. In Section 5, we show how this approach can be applied to PRESENT-80. Finally Section 6 concludes our work.

2 Description of PRESENT-80

PRESENT [5] is a 31-round SP-network with the block size of 64 bits and supports 80-bit and 128-bit secret keys. We analyze only the 80-bit version denoted further as PRESENT-80. The round function of PRESENT consists of the round key addition, substitution and permutation layers (see Fig. 1). At each round the input state is XORed with the 64-bit round key. Then 16 identical S-boxes are applied in parallel to the state. Each S-box substitutes a 4-bit input with a 4-bit output. Finally a bit permutation is performed to the whole 64-bit state.

The 80-bit secret key is stored in a key register K represented by $k_{79}k_{78}...k_0$. The key schedule of PRESENT-80 generates the round keys as follows: For $i = 0...30$:

1. The 64 most significant bits of the current register are extracted as the round key: $rk_i = k_{79}k_{78}...k_{16}$.
2. The contents of the key register is rotated to the left by 61 bits: $(K <<< 61)$.
3. The S-box is applied to the 4 most significant bits: $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$.
4. The 5 least significant bits of the round counter i are XORed with 5 bits $[k_{19}k_{18}k_{17}k_{16}k_{15}]$ of the key register so $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus [i_4i_3i_2i_1i_0]$.

For more details about PRESENT-80 refer to [5]. Further in our analysis, we call a nibble the sequence of 4 consecutive bits. Thus the state and all the round keys have 16 nibbles enumerated from left to the right as 1,2,...,16.

3 Automatic Approaches for Search of Characteristics

The invention of differential analysis was followed by several design methods and theories that allow us to construct ciphers resistant against this type of attacks [17, 8]. However, this is in regards to the security of the cipher in the single-key model only. Interestingly, almost 20 years after these approaches have been proposed, no similar theory has been found to

design ciphers that are resistant against related-key attacks. Therefore, to prove bounds on probability of related-key characteristics, the designers either have to use some ad-hoc methods such as key schedules with a high number of non-linear operations (S-boxes), or to build search algorithms that can enumerate all high probability characteristics.

The first automatic approach for finding the best differential characteristic was proposed by Matsui [12]. He showed that the simple algorithm based on trees can be heavily pruned with the knowledge of previously found single-key characteristics on smaller number of rounds, thus finding the best differential characteristic for the full-round DES became feasible. Matsui's idea was later used in the search approaches for finding the best (or the upper bounds on the best) related-key differential characteristics in byte-oriented block ciphers [2, 16]. The main idea of the authors is to use a special truncated-like representation of the differences in the state and the subkeys with slightly modified way of extending each additional round.

Another approach for search of related-key characteristics in DES-like ciphers was proposed in [3]. As this type of ciphers are bit oriented, the authors used fully defined differences in the state and the subkeys. The way of extending characteristics is in line with [2]. Two approaches were used: one based on the original Matsui's approach, and the second based on so-called split approach. The main idea of the split approach is to divide r-round characteristic into t parts. Then if each of the $\frac{r}{t}$ characteristics holds with a probability upper bounded by 2^{-s}, then the upper bound on the probability of the whole r-round characteristic is $2^{-s \cdot t}$. It is mentioned in [2] that, this approach can only be used for finding high-probability characteristics (up to 2^{-20}), and for any 64-bit cipher with a linear key schedule. Therefore it cannot be applied to `PRESENT-80` when the probability of the related-key characteristic is significantly below 2^{-20}.

In addition, we note two approaches by Mouha et al. that provide upper bounds on the best characteristics: the first is based on mixed-integer linear programming [14], while the second on SAT solvers [13].

4 The Extended Split Approach

In this section we present our *extended split approach* (ESA) for search of all related-key differential characteristics in block ciphers. We note that this approach is used to provide the upper bound on the probability of all such characteristics.

The basic idea of ESA is related to the split approach. First note that the success of the split approach depends on the probability of the round-reduced characteristics. Assume we want to find the upper bound on $R = r_1 \cdot t_1$ rounds of a cipher. With the split approach, first we find the upper bound 2^{-p_1} on the probability of the characteristics for r_1 rounds and then argue that the probability of $r_1 \cdot t_1$ rounds is upper bounded by $2^{-p_1 \cdot t_1}$. To use this bound for an n-bit cipher, the inequality $2^{-p_1 \cdot t_1} < 2^{-n}$ has to hold. When this is not the case, then we can split the R rounds of the cipher into different equal parts, i.e. $R = r_2 \cdot t_2$, find the upper bound 2^{-p_2} on all characteristics for r_2 rounds, and check if $2^{-p_2 \cdot t_2} < 2^{-n}$. When r_1, r_2 are small, the probabilities $2^{-p_1}, 2^{-p_2}$ are usually high, and the upper bounds in both cases are quite large which does not allow to prove that the cipher is free of related-key differential characteristics. Thus in general one would want to increase the number of rounds r_i of the short characteristics. This however is not always possible as the complete search of all characteristics on r_i should be feasible. For example, assume we have 30-round `PRESENT-80` and we split the cipher into three parts of 10 rounds each, i.e. $R = 30, r = 10, t = 3$. Then we should be able to find the upper bound on the probability of all characteristics for 10 rounds. This might not be possible as the number of such characteristics is huge.

In ESA we pick smaller r thus making the complete search of all characteristics on r rounds possible. If 2^{-p} is small such that $2^{-p \cdot t} < 2^{-n}$ then we obtain the required upper bound (and we end up with a simple split approach). However, when 2^{-p} is larger then we take different approach. Assume we have 24-round `PRESENT-80` and we split the cipher into

4-round parts, i.e. $R = 24, r = 4, t = 6$. If each 4-round characteristic has at least 5 active S-boxes, then the characteristic on 24 rounds would have $5 \cdot 6 = 30$ active S-boxes. Thus we can prove that the probability of 24 rounds of PRESENT-80 is at most $2^{-30 \cdot 2} = 2^{-60}$ (the differential probability of each S-box is at most 2^{-2}). Now assume that among all possible 4-round characteristics *there is only a small part* that has less than 5 active S-boxes (say only 4 active). Due to these characteristics, when using the split approach we can prove an upper bound of only $2^{-4 \cdot 6 \cdot 2} = 2^{-48}$. Let S_1 be the set of all 4-round characteristics that have at least 5-active S-boxes, and S_2 be the set of the ones that have less than 5. The idea of the ESA is to treat the characteristics from S_1 and S_2 differently. Each 4-round characteristic from S_2 is *extended* to all possible 8-round characteristics, and an upper bound of all such 8-round characteristics is found. If this upper bound is 10 active S-boxes then we get the following:

- All 8-round characteristics composed of two 4-round characteristics from S_1 have at least 10 active S-boxes since each of them has at least 5 active S-boxes.
- All 8-round characteristics that have at least one 4-round characteristic from S_2 have at least 10 active S-boxes.

Therefore the probability of any 8-round characteristic is at most $2^{-10 \cdot 2} = 2^{-20}$ and for 24 rounds is $2^{-20 \cdot 3} = 2^{-60}$, i.e. we end up with the same bound as in the case when all 4-round characteristic have at least 5 active S-boxes.

The ESA can be seen as an efficient way of launching the split approach with parameters $(R, 2r, t)$ from the split approach $(R, r, 2t)$, e.g. in the example above we could use the split approach $(24, 8, 3)$ from the split $(24, 4, 6)$.

5 Application to PRESENT-80

To find the upper bound on the probability of the best related-key characteristic in PRESENT-80, we start our analysis with the 24-round cipher. In the split approach, the number of short characteristics would be for 4 rounds, and thus the parameters for the ESA are $(24, 8, 3)$ (as above). Further we show how to find the probability of all characteristics for 4 rounds, and thus how to obtain the sets S_1, S_2. All 4-round characteristics that have at least 5 active S-boxes belong to S_1, and all with less than 5 active S-boxes belong to S_2. As any 8-round characteristic composed from two 4-round characteristics from S_1 already has at least 10 active S-boxes (each has at least 5 active), our task is to find only the characteristics that belong to S_2, i.e. *to find the 4-round characteristics that have at most 4 active S-boxes.*

5.1 General Approach for Producing S_2

A trivial way to produce S_2 is to start with a difference value for the input state at the beginning of the first round, and see how this difference propagates through 4 rounds. We have to try all such differences in the initial state, and all possible differences in the master key. This is infeasible, as there are 2^{64} possible differences in the state, and 2^{80} differences in the master key, thus in total 2^{144} possible starting points. A common strategy for reducing the complexity of the automatic search algorithms is instead of trying actual differences, to try and fix only position of active and inactive nibbles – this is a truncated representation of a difference in the state and round key. A nibble is called active if any of its 4 bits has difference, otherwise non-active or inactive. For example, the state in PRESENT-80 has 64 bits which is 16 nibbles and thus instead of trying 2^{64} values for the difference in state, now we can try only $2^{64/4} = 2^{16}$ values. However, even this strategy is not sufficient to make the search feasible as the amount of starting points now becomes 2^{16} for the plaintext and $2^{80/4} = 2^{20}$ for the master key, or in total 2^{36}. This number seems reasonable, however there is branching in the operations. To explain this assume we start with some initial state and master key differences and propagate them through 4 rounds.

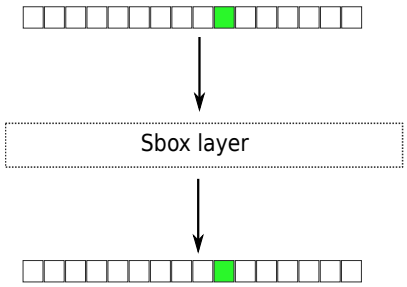

Fig. 2. Propagation of the truncated difference through the Sbox layer. The green nibble is active; the white nibbles are inactive.

Going through the round transformations (the bit permutation layer) produces different possible output differences for the same input difference, and thus instead of 2^{36}, we will end up with a much higher and infeasible complexity.

The reason we choose the improved search is that this method reduces the number of candidates significantly, and meanwhile we make sure that the search has checked all the possible combinations. This is achievable, since *we have limited ourselves to only 4 active Sboxes in 4 rounds*, hence we do not check the other combinations. Note, a similar strategy has been applied in the automatic approaches in [2, 3, 12]. Instead of fixing the truncated representation of the difference in the master key, we *fix the differences in the initial states of four consecutive rounds* and determine the master key difference from them. Note that we want to find all related-key (RK) characteristics which have no more than 4 active Sboxes, therefore the initial states of the four consecutive rounds can have at most 4 active Sboxes (as the S-box layer is at the beginning of each round). Thus, instead of trying all possible truncated differentials for these 4 states (i.e. $2^{4 \cdot 16} = 2^{64}$ differentials), we only have to check $C_{64}^1 + C_{64}^2 + C_{64}^3 + C_{64}^4 \approx 2^{20}$ differentials (i.e. respectively four states with only one active nibble out of 64 possible, four states with two active nibbles out of 64, and similarly for three and four active nibbles).

The next step is filtering the incorrect quartets. The 4 initial state differences of a quartet are not independent – each is produced from the previous in one round of `PRESENT-80`. As we have fixed the active nibbles of these 4 states randomly, some of them (actually most of them) cannot be produced in 4 steps. To filter the incorrect quartets we use the related round keys which are produced from the master key.

Thus our strategy is as follows: once we fix all possible truncated state differences at the beginning of the four consecutive rounds, we determine the differences in the round keys used in these 4 rounds and see if the round keys actually can be produced one from another with the key schedule. If none of the combinations of round key differences suggest the same master key, then the quartet of state differences is incorrect, otherwise we have found a 4-round related-key differential characteristic with at most 4 active nibbles.

Branching in a round. The truncated difference significantly reduces the complexity of the search, however it introduces branching. A branching occurs in some transformations if from the input difference (given as a truncated value) the output difference cannot uniquely be determined, and can have several different values.

The first transformation (i.e. the S-box layer) does not change the truncated representation of the state (see Fig. 2). More specifically, S-box layer keeps the position of differences in the state nibbles. Thus we can completely ignore this layer.

The second layer on the other hand introduces branching as the bit permutation is applied in this layer (see Fig. 3). Therefore, knowing the position of the active nibbles is not enough to determine the output active nibbles after the permutation. So to check all the possible trails, we need to try every possible output for a given truncated input. Note that in the permutation layer each four output nibbles depend on only four distinct

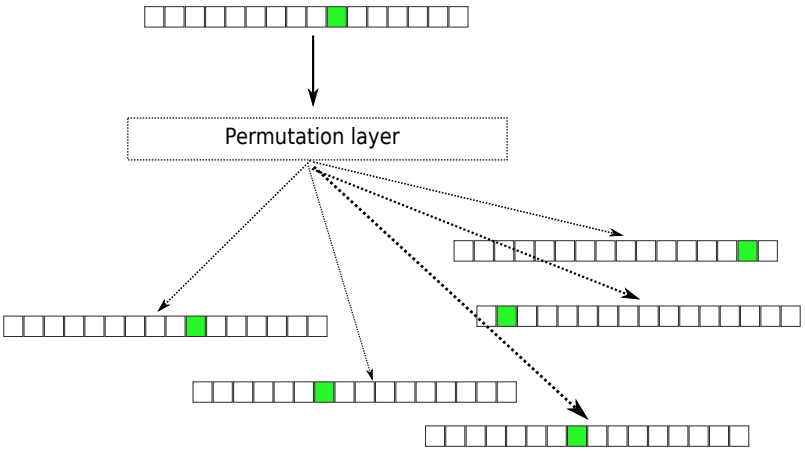

Fig. 3. Propagation of the truncated difference through the bit permutation layer. (The branching shown in the figure is not the actual branching of the permutation layer used in `PRESENT-80`).

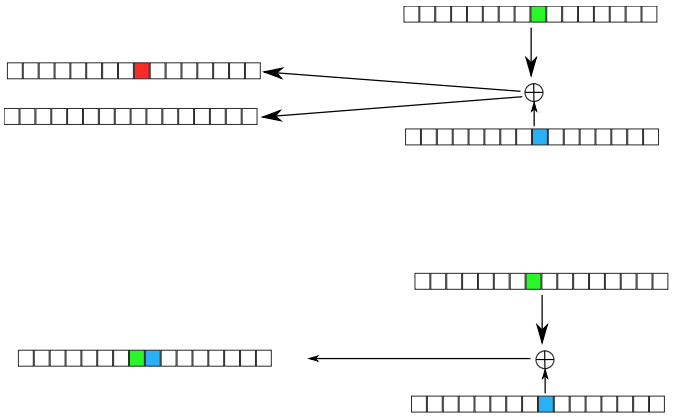

Fig. 4. Branching for XOR addition of two active nibbles at the same position (the top) and different positions (the bottom).

input nibbles, for example output nibbles $1, 5, 9, 13$ depend on the input nibbles $1, 2, 3, 4$. Thus the permutation layer is split into four independent parts and each part is processed separately. Therefore, existence of at least one active nibble in the input set (of four nibbles), results in all possible combinations of active nibbles in the output set. Due to the heavy branching in this layer, the complexity of the search increases significantly. For example, if there are four active nibbles at the input state of the permutation layer –each in a different set of four nibbles– then the output takes $(2^4 - 1)^4 \approx 2^{16}$ different values.

The XOR addition also introduces branching. As we work with the truncated differences, an XOR of two active nibbles can produce both active or inactive nibble. Thus we will have to branch on 2 for each XOR addition of two active nibbles at the same position, while no branching occurs when the active nibbles are at different positions (see Fig. 4).

Determining the round key. We should be able to determine the round key difference rk_i used in the round i from the differences s_i, s_{i+1} in two consecutive states. However, due to the truncated representation of the differences, the round key difference is not determined uniquely. So for a fixed s_i, s_{i+1}, there may be different values for rk_i. To determine the set of all round key differences $\{rk_i\}$, we start with s_i and go through the S-box layer. It is mentioned above that the S-box layer does not affect the active nibbles. So the output state difference is still s_i. Then, we obtain all possible state differences $\{s_i^j\}$ that are produced from s_i after the permutation layer branching. The number of such state differences may be larger than one. Finally, we XOR each of these state differences

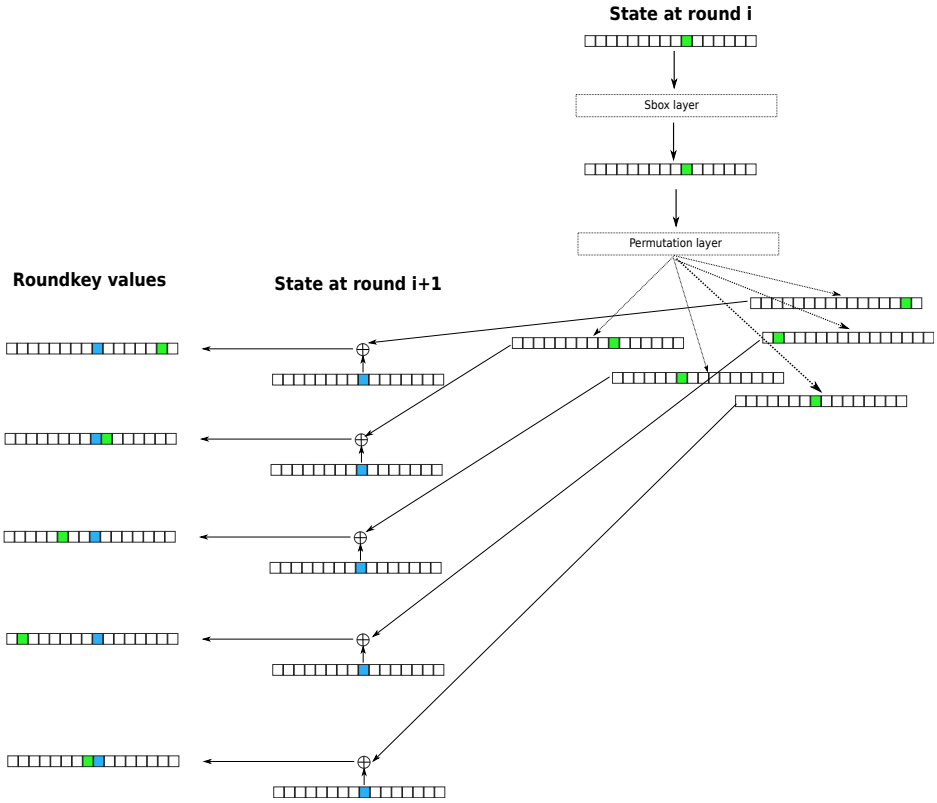

Fig. 5. Determine the differences in the round key from the differences of two consecutive initial states.

with s_{i+1} to produce the round key difference rk_i. Again, due to the branching of the XOR addition, for a single s_i^j, there may be more than one round key difference. This strategy is depicted in Fig. 5.

Eliminating the incorrect round keys. Given two truncated differences (s_i, s_{i+1}) of two consecutive round states, we can determine the set of all round key differences $\{rk_i\}$. Hence, when we fix the four truncated differences (s_0, s_1, s_2, s_3) (with no more than 4 active S-boxes), we can determine the sets $\{rk_0\}, \{rk_1\}, \{rk_2\}$. The next step is to filter out the incorrect round key differences. Recall that the round keys are not independent as they are produced from the same master key. Thus not all the found combinations of the round key differences are valid. In fact, due to the simple key schedule of `PRESENT-80`, the truncated representation is sufficient to filter out a vast amount of these combinations. We take all the (rk_0, rk_1, rk_2) combinations such that $rk_i \in \{rk_i\}, i = 0, 1, 2$ and check if rk_1 can be obtained from rk_0, and rk_2 can be obtained from rk_1 in one step of the key schedule. If we find one such triplet, then we have found a 4-round related-key differential characteristic that has no more than 4 active S-boxes, and thus this characteristic belongs to S_2. Therefore, by checking all possible combinations (s_0, s_1, s_2, s_3), and subsequently filtering out the combinations of the round keys, we can make sure that we have checked all 4-round characteristics.

5.2 Efficient Round Key Filters

Since the differences are truncated and the key schedule is bit-oriented, it is hard to check if the two round key differences are produced one from another in one step of the key schedule. Hence efficient filtering is required to eliminate the incorrect round key differences.

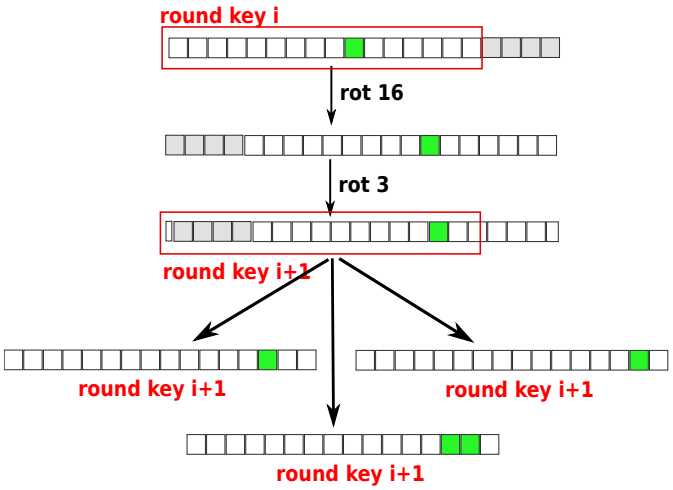

Fig. 6. The basic filter.

Basic Filter. First we start with a filter that checks only if the positions of the active nibbles of the differences rk_i, rk_{i+1} are consistent. Assume rk_i has a single active byte at position j (see Fig. 6). Note that we know only 16 nibble differences out of the possible 20 differences for the master key. This means that the last four nibbles are unknown (in the figure they are gray).

We apply the first operation of the key schedule, which is rotation by 61 bits to the left or equivalently by 19 bits to the right. Let us split the operation into two sequential rotations by 16 and by 3 bits to the right. Rotation by 16 bits to the right is equivalent to rotation by 4 nibbles, thus the active nibble at the position j would become active at the position $j + 4$. The rest of the nibbles would stay inactive. The rotation by 3 is not nibble-oriented, thus there will be branching. If the active nibble at the position $j + 4$ has a difference at the most significant bit (msb), then after rotation by 3, the difference would stay in the same nibble. If the difference is at some of the three least significant bits (lsb), then the nibble at the position $j + 5$ would become active. Finally, if the difference is in both msb and in some of the lsb, then both nibbles $j + 4$ and $j + 5$ will be active.

The second operation, i.e. the S-box, changes the truncated difference only in the first nibble after the rotation is active. This is equivalent to the last nibble being active (before the rotation) – in this case the result of the S-box makes the first nibble active or inactive. In the basic filter, we ignore this nibble, thus we do not explain further details. Similarly, the third operation, the XOR with the round counter, has no influence on the difference.

Taking into account all three operations we conclude that for an active nibble at the position j in rk_i, there should be one active nibble either at the position $j + 4$, or at $j + 5$ or at both $j + 4, j + 5$ in r_{i+1}. As rk_i and rk_{i+1} depend on different parts of the master key (due to the rotation), the value of j can go from 1 to 12 (1 is the most significant nibble in the difference). Converse filter applies as well, i.e. if $j + 4, j + 5$ are inactive in rk_{i+1} then j should be inactive in rk_i.

Filter based on Actual Values. In some cases in addition to the truncated differences of the round key, the precise values of some bits of the key difference can be determined as well. Recall that the permutation layer is split into four independent parts of 4 nibbles each and every output nibble of a part depends on all 4 input nibbles. Assume the input-output truncated difference for the first part has the form $(1001) \rightarrow (1000)$ (see Fig. 7), (i.e. at the input nibbles 1 and 4 are active, and at the output of the permutation layer only nibble 1 is active). Furthermore, let nibble 1 at the next state be inactive (it means the active nibble at position 1 has been canceled with the active nibble of the round key). Then the value of the active nibble of the round key is 9_h. This comes from the fact that

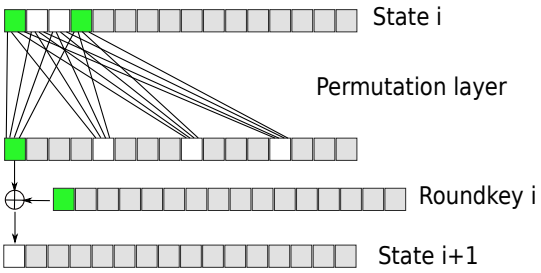

Fig. 7. Determine the actual values of the round key. The green nibbles are active, the white are inactive, and the gray can have any value.

the difference in the active nibble after the permutation layer has to be 9_h as the active nibbles at the positions 1,4 at the input have difference and this difference goes only to one nibble, thus both of most and least significant bits have to have difference.

The above observation can be generalized as follows. If at the output of any of the four sets of the permutation layer, there is a single active nibble, and the next state does not have difference at this nibble, then the value of the difference in the round key in the active nibble is equal to the truncated difference at the input set. Similar observation holds when the input has only one active nibble. Then each of the output active nibbles has an actual difference that equals to the truncated difference of the active input nibble.

Once we obtain the actual values of some active nibbles in the round keys, the filter becomes much stronger. Instead of comparing truncated differences, we can compare actual differences. Similar to the previous filter, when applying it for two consecutive round key differences rk_i, rk_{i+1}, we have to take into account the rotation by 19 bits to the left, and the application of the S-box to the first nibble. In practice, this filter allows elimination of many candidate round key differences. Although it seems the condition of having just one active nibble at the input or at the output of the part (there are 15 possible output differences) is rarely satisfied, most of the round keys have only a few active nibbles because of the low number of active S-boxes in the 4-round characteristics.

Filter based on Impossible 1-to-1 Transitions for S-box. The `PRESENT-80` S-box was specially chosen to prevent 1-to-1 transitions. That is if the input difference to the S-box is in a single *bit*, then the output difference cannot be in only one bit. We can use this property to eliminate some of the wrong candidates.

The S-box layer follows the XOR with the round key, and is followed by the permutation layer. Thus to find the 1-to-1 impossible differences for the active nibbles in the S-box layer, we have to find the active nibbles that have a single bit difference after the XOR addition with the round key, while the active nibbles have single bit differences at the input of the permutation layer. The later can be easily found using the same method as the filter based on actual values. This is where there is only one active nibble at the input set (of the four sets of the permutation layer) and one at the output, then the active nibble at the input has a single-bit difference. Then the task is finding single-bit active nibbles at the input of the S-box layer. There are two situations that we get these single-bit active nibbles. First, when the output nibble of the previous permutation layer is inactive and the round key XORed with this nibble has a single-bit difference. Second, when the round key nibble is inactive and the output of the permutation layer has a single-bit active nibble. Both cases are depicted in Fig. 8 and lead to the filtering of the states and the round keys that have such differences.

Filter on Three Consecutive Round Key Differences. The first two filters work exclusively based on values of differences (truncated or actual) of two consecutive round keys. They do not eliminate the incorrect characteristics that have some invalid conditions

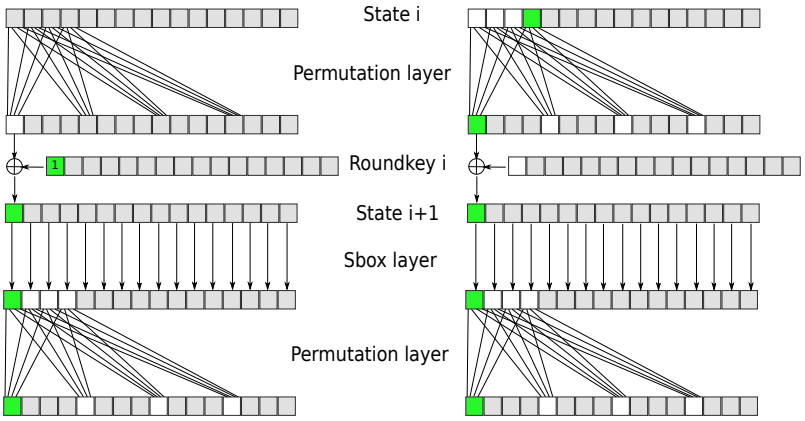

Fig. 8. Two instances of impossible values for state and round key differences that are eliminated by the filter.

when several round keys are considered. For example, let $rk_{i-1} = (0000000000001111)$, $rk_i = (0000000000000000)$, $rk_{i+1} = (0000000000000000)$. The basic filter analyzes the pairs $(rk_{i-1}, rk_i), (rk_i, rk_{i+1})$ and concludes that the pairs are correct. The first pair is valid because it cannot check the last 4 nibbles due to the rotation by 19 and the fact that the round key is smaller than the master key. Similarly, the second pair is valid as well. Clearly, rk_{i+1} cannot be produced from rk_{i-1} in two steps of the key schedule as rk_{i+1} should have at least one active nibble. Thus we need an additional filter that checks pairs of round key differences (rk_{i-1}, rk_{i+1}). This filter works as the basic one as well as the filter on actual values, and additionally performs the elimination using the same method as the first two filters. It works only on a small part of the round keys – the last 6 nibbles of rk_{i-1} and the first 6 nibbles of rk_{i+1}. We skip the precise description of the filter as the analysis is similar to the previous filters.

5.3 Generation of S_2

Once we have collected the outputs of all the filters, we can proceed with the generation of the set S_2 of all 4-round related-key differential characteristics with at most 4 active S-boxes. This procedure is given in a pseudo-code Algorithm 1. The branching tables $p[s]$ are sets of all the values of state differences which are obtained after the permutation layer applied to the state difference s. Similarly, $\oplus[a][b]$ denotes the set of all the differences obtained after the XOR addition of two truncated differences a, b. Finally, the $Filters(x_1, \ldots, x_t)$ are functions based on the previously described filters that return false when the round key (or state) differences $x_1, \ldots, x_t$ have contradictions and true otherwise.

Algorithm 1 *Generation of the set S_2*

Require: Branching tables $p[s], \oplus[a][b]$ for the permutation layer and XOR; the filters $F(x_1, \ldots, x_t)$
$S_2 \leftarrow \emptyset$
for all $s_1, s_2, s_3, s_4 | h_w(s_1) + h_w(s_2) + h_w(s_3) + h_w(s_4) \leq 4$ **do**
  $\{s_1\} \leftarrow p[s_1]$
  for all $s_1^i \in \{s_1\}, s_2$ **do**
    $\{rk_1\} = \oplus[s_1^i][s_2]$
    $\{s_2\} \leftarrow p[s_2]$
    for all $s_2^j \in \{s_2\}, s_3$ **do**
      $\{rk_2\} = \oplus[s_2^j][s_3]$
      for all $rk_1 \in \{r_1\}, rk_2 \in \{rk_2\} | Filters(rk_1, rk_2)$ **do**
        $\{s_3\} \leftarrow p[s_3]$

for all $s_3^k \in \{s_3\}, s_4$ **do**
$\{rk_3\} = \oplus[s_3^k][s_4]$
for all $rk_3 \in \{rk_3\} | Filters(rk_1, rk_2, rk_3)$ **do**
$S_2 \leftarrow S_2 \cup (s_1, s_2, s_3, s_4, rk_1, rk_2, rk_3)$
end for
end for
end for
end for
end for
end for

end

We have implemented and run the algorithm on PRESENT-80. The search took only a few minutes on a single PC and output 712 related-key differential characteristics, i.e. $|S_2| = 712$. Among them, 3 are with 2 active S-boxes, 73 with 3 active, and 636 with 4 active S-boxes.

5.4 The Complete Search and Results

Once we have the set S_2 we can extend each characteristic from this set to 8 rounds. We extend in both ways, i.e. first we extend for four additional rounds in the forward direction and obtain 8 rounds. Then we extend four rounds backward and again obtain 8 rounds. Note, the 8-round characteristics composed of two 4-round characteristics from the set S_1 have at least 10 active S-boxes. Thus, when extending the characteristics from S_2 to 8 rounds, we want to find if there exist 8-round characteristic that has less than 10 active S-boxes. If not, then all 8-round characteristics must have at least 10 active S-boxes.

The algorithm for extension uses exactly the same approach as the algorithm for producing S_2. Thus we can find all 8-round characteristics that are extended from 4-round characteristics that belong to S_2. Once we find such characteristic, we confirm it is correct and the round key differences comply. Note, this check is necessary because even after using all the filters, we still accept some incorrect round key differences. We have produced all 8-round characteristics by implementing this procedure. The search took around a day on 32 PC's. Although it produced 869 characteristics, by a careful examination we have found that none of them actually are valid as there were inconsistencies in the round key differences. We note that the inconsistencies followed around 30 different patterns – once we found all the patterns, we were able to quickly test all of the characteristics. In the Appendix we point out one such characteristic and we show how the inconsistency was found. The remaining characteristics were tested similarly.

The PRESENT-80 designers have proven a lower bound for the number of active S-boxes in the single-key case:

Theorem 1 ([5]). *Any 5-round (single-key) differential characteristic of* PRESENT *has a minimum of 10 active S-boxes.*

We have analyzed the case for related-key characteristics:

Theorem 2. *Any 8-round related-key differential characteristic of* PRESENT-80 *has a minimum of 10 active S-boxes.*

Proof. Any 8-round characteristic can be seen as a composition of two 4-round characteristics. When both of the 4-round characteristics have at least 5 active S-boxes (belong to S_1), then the 8-round related-key characteristic has at least 10 active S-boxes. When one of them has at most 4 active S-boxes (belongs to S_2), then extending it for 4 rounds (as we have seen above) would always result in an 8-round characteristic that has at least 10 active S-boxes. □

Any 24-round related-key characteristic in PRESENT-80 has at least 30 active S-boxes, and 29-round[4] has at least 33 active. As the differential propagation probability of an active S-box is at most 2^{-2}, *all 29-round related-key differential characteristics on* PRESENT-80 *have a probability at most* 2^{-66}. Therefore PRESENT-80 is secure against related-key differential attacks that exploit a single related-key differential characteristic.

It is also worth mentioning that the boomerang attack [23] is not a threat as well. If one of the characteristics in the boomerang (the top or the bottom one) is on 16 rounds, then it has at least 20 active S-boxes, thus the probability of this part only (computed as p^2) is $2^{-20 \cdot 2 \cdot 2} = 2^{-80}$, hence it is below the required 2^{-64}. Similar probability is obtained when both of the characteristics have at least 8 rounds.

The security of PRESENT-80 against related-key attacks based on differentials (rather than differential characteristics) is still an open problem as it is for any other cipher. However, by applying the standard conjecture (used in the security proofs of ciphers against differential attacks) that the high probability differentials have only one high probability characteristic, we can deduce that PRESENT-80 is secure against related-key attacks. The experimental results presented in the submission paper of PRESENT-80 confirm the conjecture in the case of single-key differentials. Performing similar experiments in the related-key case is difficult as no obvious high-probability related-key characteristics exist [5].

6 Conclusions

We have derived a bound on the probability of the best related-key differential characteristic in PRESENT-80 which confirms that the cipher is secure against basic related-key differential attacks.

Interestingly, our approach is a hybrid of the two previous methods published in [2, 3]. The first method uses strictly truncated representation of the difference, while the second approach works with actual differences. Our approach on the other hand uses both which makes the search feasible. More precisely, we take advantage of the truncated representation as it significantly reduces the number of possible state differences. Once we obtain the truncated round key differences, we switch to actual differences (whenever possible) and use them to filter out incorrect characteristics. The extended split approach provides more computationally efficient search compared to the simple split approach. We only have to compare the number of starting initial state differences to find the real advantage. If we use the split approach for 8 rounds, we have to start with around $C^9_{8 \cdot 16} \approx 2^{44}$ initial differences. On the other hand, the extended split approach performs the same task with only $C^4_{4 \cdot 16} + 3 \cdot C^7_{4 \cdot 16} + 73 \cdot C^6_{4 \cdot 16} + 636 \cdot C^5_{4 \cdot 16} \approx 2^{33}$ truncated differences in the initial states.

The advantage of using nibble (or byte) oriented ciphers with non-nibble oriented key schedule is still unclear and remains an open problem if such ciphers can be secure with some nibble oriented key schedule. However it is clear that the search for related-key differential characteristics when all the operations are nibble oriented is much simpler and (almost) always feasible, whereas the same search in non-nibble ciphers is very complex. We have shown in this paper that for PRESENT-80 the search can be performed although the key schedule is non-nibble oriented. Our method can be used to obtain upper bounds on probabilities of related-key characteristics for some other lightweight ciphers as well. One only has to recompute the branching tables for the permutation layer and to find filters for the round key differences. The filters will have the biggest impact on the efficiency of the

[4] By taking into account that the best previously found characteristics in S_2 have 2 active S-boxes, and extending each of them for an additional round always gives at least 3 active S-boxes (we have checked this fact).

[5] Our approach only finds an upper bound, but does not provide valid characteristics on extended number of rounds, i.e. 8. On the other hand, the output of testing the conjecture on low number of rounds (i.e. 4) does not seem to be a valid measure.

search. We have shown in the case of PRESENT-80 that efficient filters can be found when the key schedule is relatively simple – this property is common for most of the lightweight ciphers, thus our method could be applied. The upper bound found with our approach not necessarily would be low, however it would contribute towards achieving the initial trust in the security of the cipher against related-key differential attacks.

References

1. A. Biryukov, D. Khovratovich, and I. Nikolic. Distinguisher and related-key attack on the full AES-256. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.
2. A. Biryukov and I. Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 322–344. Springer, 2010.
3. A. Biryukov and I. Nikolic. Search for related-key differential characteristics in DES-like ciphers. In Joux [11], pages 18–34.
4. C. Blondeau and B. Gérard. Multiple differential cryptanalysis: Theory and practice. In Joux [11], pages 35–54.
5. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
6. J. Y. Cho. Linear cryptanalysis of reduced-round PRESENT. In J. Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2010.
7. B. Collard and F.-X. Standaert. A statistical saturation attack against the block cipher PRESENT. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2009.
8. J. Daemen and V. Rijmen. The wide trail design strategy. In B. Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.
9. J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.
10. J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED block cipher. In Preneel and Takagi [19], pages 326–341.
11. A. Joux, editor. *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*. Springer, 2011.
12. M. Matsui. On correlation between the order of s-boxes and the strength of DES. In A. D. Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.
13. N. Mouha and B. Preneel. A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:328, 2013.
14. N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In C. Wu, M. Yung, and D. Lin, editors, *Inscrypt*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.
15. J. Nakahara, P. Sepehrdad, B. Zhang, and M. Wang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In J. A. Garay, A. Miyaji, and A. Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2009.
16. I. Nikolic. Tweaking AES. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 198–210. Springer, 2010.
17. K. Nyberg and L. R. Knudsen. Provable security against differential cryptanalysis. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 566–574. Springer, 1992.
18. O. Özen, K. Varici, C. Tezcan, and Çelebi Kocair. Lightweight block ciphers revisited: Cryptanalysis of reduced round PRESENT and HIGHT. In C. Boyd and J. M. G. Nieto, editors, *ACISP*, volume 5594 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2009.
19. B. Preneel and T. Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
20. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An ultra-lightweight blockcipher. In Preneel and Takagi [19], pages 342–357.
21. T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In A. Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.
22. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE : A lightweight block cipher for multiple platforms. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.

23. D. Wagner. The boomerang attack. In L. R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.
24. M. Wang. Differential cryptanalysis of reduced-round PRESENT. In S. Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.
25. M. Wang, Y. Sun, E. Tischhauser, and B. Preneel. A model for structure attacks, with applications to PRESENT and Serpent. In A. Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 49–68. Springer, 2012.

A An Example of Invalid 8-round Related-Key Characteristic

Here we give an example of one related-key differential characteristic for 8-round `PRESENT-80` found by search that has less than 10 active S-boxes, passes all the filters and is invalid (see Fig. 9). Recall that our search algorithm outputs 869 such characteristics and each of them is invalid. In the analysis given below, we would like to point out the inconsistency found in this characteristic. Other characteristics can be analyzed similarly.

We focus on the differences in the round keys. Note that they pass all of our 5 filters. However our filters are not sufficient to test completely the validity of round key differences, i.e. they might miss to filter out some invalid characteristics. One might create stronger filters but then the computational efficiency of such filters might slow down the search. The differential characteristics in the round keys is (0020) $\rightarrow$ (0003) $\rightarrow$ (0000) $\rightarrow$ (0800) $\rightarrow$ (0040) $\rightarrow$ (0002) $\rightarrow$ (0000) (see Fig. 9). Further we try to find the exact value of the difference in the round keys from the truncated ones. We analyze each round key separately, assuming that the nibbles are enumerated 1-16 plus the four nibbles 17-20 that are not used in the round key but are part of the master key. Recall that each consecutive round key is produced from the previous (along with 4 more nibbles) by a rotation by 19 bits to the right. For the differences in the round keys, we get:

- **Round key 1 - difference** 0020 - There is only one active nibble at the position 11. Let the value of the difference be $xyzt$, where x, y, z, t are single bit differences.
- **Round key 2 - difference** 0003 - The active nibbles at the positions 15 and 16 have the values $000x$ and $yzt0$. As both are active, it follows that $x = 1$ and one of y, z, t is 1 (otherwise these two nibbles would not be active).
- **Round key 3 - difference** 0000 - No active nibbles in this round key. However, the active nibbles from the previous round key, with the rotation went to one of the nibbles 17-20. In fact, the nibble 20 has the difference $00xy = 001y$ while the nibble 21, which is in fact the nibble 1, has the value $zt00$. As the nibble 1 is not active in this round key it follows that $z = t = 0$.
- **Round key 4 - difference** 0800 - Only one nibble at the position 5 is active. This has to correspond to the nibble at position 20 from the previous round key. Thus the value of the difference in this nibble is $01y0$.
- **Round key 5 - difference** 0040 - Similarly, we have only one active nibble at the position 10, with the difference $1y00$.
- **Round key 6 - difference** 0002 - Only one active nibble at the position 15 is active. However, if we rotate by 19 the previous round key difference $1y00$, we will end up with the difference 0001 in the nibble 14, and the difference $y000$ in the nibble 15. As the nibble 14 is inactive, we get a contradiction.

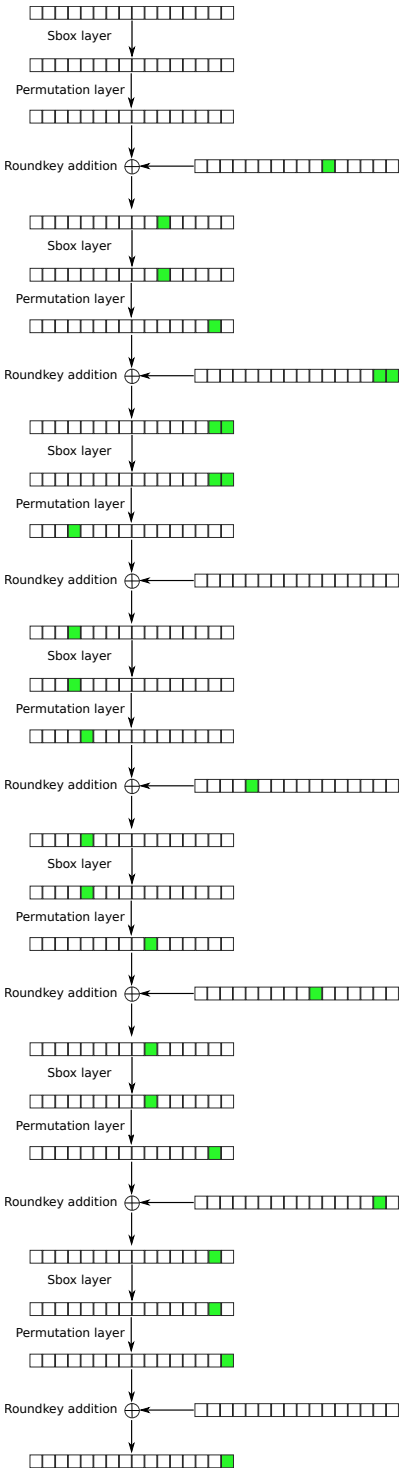

Fig. 9. An example of invalid 8-round characteristics found by the search.