Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others

Alex Biryukov and Ivica Nikolić*

University of Luxembourg
{alex.biryukov,ivica.nikolic}uni.lu

Abstract. While differential behavior of modern ciphers in a single secret key scenario is relatively well understood, and simple techniques for computation of security lower bounds are readily available, the security of modern block ciphers against related-key attacks is still very ad hoc. In this paper we make a first step towards provable security of block ciphers against related-key attacks by presenting an efficient search tool for finding differential characteristics both in the state and in the key (note that due to similarities between block ciphers and hash functions such tool will be useful in analysis of hash functions as well). We use this tool to search for the best possible (in terms of the number of rounds) related-key differential characteristics in AES, byte-Camellia, Khazad, FOX, and Anubis. We show the best related-key differential characteristics for 5, 11, and 14 rounds of AES-128, AES-192, and AES-256 respectively. We use the optimal differential characteristics to design the best related-key and chosen key attacks on AES-128 (7 out of 10 rounds), AES-192 (full 12 rounds), byte-Camellia (full 18 rounds) and Khazad (7 and 8 out of 8 rounds). We also show that ciphers FOX and Anubis have no related-key attacks on more than 4-5 rounds.

Keywords: Cryptanalysis tool, search for best differential characteristics, related-key attack, open key, AES, Camellia, Khazad, Anubis, FOX.

1 Introduction

Proving security of modern block ciphers against differential [6] and linear cryptanalysis [28] has become a well understood and relatively simple task. Many of the modern ciphers are constructed as so-called substitution-permutation networks (SPN) — they consist of layers of non-linear substitution boxes (S-boxes) and diffusion layers built from linear or affine functions. The designer simply has to use diffusion layers with high (or maximal) branch number which is typically achieved by using maximum distance separable matrices [13]. Using such diffusion layers one can prove lower bounds on the number of active S-boxes for a certain number of internal rounds. The designer then picks the number of rounds for which the probability of the best differential or linear characteristic is lower than 2^{-k} where k is the key size of a cipher. The resultant cipher is then provably secure against standard differential and linear attacks.

Such reasoning however holds only in the single key model, and does not extend to the case of related-key attacks [4]. In this class of cryptanalytic attacks the attacker knows or chooses the

* This author is supported by the Fonds National de la Recherche Luxembourg grant TR-PHD-BFR07-031.

relation between several keys and is given access to encryption/decryption functions with all these keys. The goal of the attacker is to find the actual keys. The relation between the secret keys is a function chosen by the attacker with some extra care taken to avoid trivial attacks, and quite often it is just a XOR with a chosen constant. Security of most modern block ciphers against related-key attacks still relies on heuristic and ad hoc arguments. This situation is very similar to the heuristic security that we have for the modern hash functions, which is due to a lack of proper tools and methodologies for the analysis of differentials of non-bijective functions.

In this paper we make a step in the direction of provable security of modern block ciphers (and by analogy of modern hash functions), by presenting an efficient tool that can evaluate and help to prove bounds for the security of block-ciphers (hash functions) against differential related-key (open-key or chosen message) attacks.

Automatic search for best differential characteristics and linear approximations in a single key scenario was first performed by Matsui [29] for DES. Algorithms for automatic search of differential characteristics for MD4 were presented in [34, 15], and for MD5 in [35]. De Cannière and Rechberger in [11] described a method that finds characteristics in SHA-1 in an automatic way and produced the best known collision trails for SHA-1. A typical problem that arises when trying to construct a tool for automatic search of characteristics is the size of the search space. The search space is exponential in the size of the block and the key which makes straightforward approaches infeasible for 128-bit block 128-256 bit key ciphers[1]. Therefore often the most important task for producing an efficient tool is the reduction in the size of the internal state by using some equivalent representation of the state, but with smaller size.

In that respect it is natural to look at byte (or word)-oriented ciphers which constitute a large fraction of modern ciphers. A natural compact (sometimes known as truncated) representation would shrink each byte into a single bit, representing by 0 a byte without difference and by 1 a byte with a difference. In such representation 16-byte block state, 16-32 byte key would translate into 16 and 16-32 bits respectively. These numbers are low, and give hope that a search of the whole $2^{32} - 2^{48}$ space of related-key differential characteristics might be possible. The main problem with this representation is a very heavy branching which will happen in the linear diffusion layers and on the XORs. Such representation alone will only allow to search for the most basic and short characteristics which happen with probability close to 1.

Our Contribution. Our goal is to perform a full search for related-key differential characteristic and to be able to find or to prove the non-existence of characteristics similar to those that were used in the recent attack on AES-256 [10]. In this paper we achieve this goal for all versions of AES and for several other ciphers. At the basis of our related-key search algorithm, further denoted as a *tool*, lies Matsui's approach for search of the best differential characteristics, with several important modifications. Depending on the key schedule of a cipher, we differentiate three classes of block ciphers. This is done to improve the efficiency. For each of the classes we introduce a special modification to Matsui's algorithm to obtain the final tool. The internal representation of the difference in a cipher (state and subkeys) plays a very important role for constructing a feasible tool. Using only compact representation may lead to a high branching (caused by XORs or other linear-diffusion transforms such as MixColumns in AES) when trying to build all possible one round characteristics. We completely eliminate the branching in the state of a cipher by using a special

[1] Note that full search was feasible for 64-bit block cipher DES, due to its Feistel structure, which reduces the search space to about 2^{32}.

representation that takes into account the properties of the matrix used in the linear-diffusion layer. The related-key differential characteristics produced by our tool, fix only the positions of the active bytes[2]. To produce standard differential characteristics, i.e. characteristic with exact values of the differences in the active bytes, one has to fix the byte differences corresponding to the possible transitions of the differences trough the S-boxes.

We apply the tool to different byte-oriented block ciphers. The tool finds related-key characteristics for the full-round ciphers, or if such characteristics do not exist, for the maximal number of rounds for which they exist. We provide the best possible differential characteristics for all the versions of AES. For AES-128 it is on 5 rounds out of 10 (this also means that AES-128 is secure against straightforward related-key attacks after 6 rounds). For AES-192 it is on 11 rounds – just one round short of the total 12 rounds. The characteristic for AES-256 is on all 14 rounds, and it is the same characteristic (and the only one on 14 rounds) that was given in [10]. Then we present boomerang attack on AES-128 reduced to 7 rounds, and improve the complexity of the best attack on AES-192 by a factor 2^7. We analyze the version of Camellia without the FL functions, and where the rotation constants in the key schedule are multiplies of 8. For this "byte-Camellia", the best related-key differential characteristic is on 8 rounds (out of 18). Additionally, we launch a chosen-key attack and find a characteristic on all 18 rounds of byte-Camellia. For Khazad first we find a related-key characteristic on 7 rounds (out of 8), and then we show a boomerang attack on 7 rounds, and a chosen-key attack on the full-round Khazad. For the ciphers FOX and Anubis, we show that related-key differential characteristics cannot exist on more than 4-5 rounds. The summary of our results is given in Tables 1,2. Due to space limitations, we will not describe the ciphers that we analyze, we refer the reader to [13, 1, 3, 20, 2], and will use the original notation proposed by the designers in these papers.

2 A Tool for Search of Related-Key Differential Characteristics

Related-key differentials, introduced by Biham in [4], unlike the traditional single-key differentials that have difference only in the plaintext, have difference in the key as well. A related-key differential is specified with two input differences: Δ_P in the plaintext and Δ_K in the key, and an output difference Δ_C in the ciphertext. A pair of plaintexts (P_1, P_2) and a pair of keys (K_1, K_2) follow the related-key differential in the cipher $E_K(P)$, if $P_1 \oplus P_2 = \Delta_P, K_1 \oplus K_2 = \Delta_K$ and $E_{K_1}(P_1) \oplus E_{K_2}(P_2) = \Delta_C$. A popular technique to find lower bounds on the probability of differentials is via finding probabilities of the best differential characteristics. A related-key differential characteristic besides the differences in the key, plaintext and ciphertext, also fixes the differences in the state and the subkeys after each round of the cipher.

There are a couple of approaches to construct a tool for search of the best round-reduced (related- or single-key) differential characteristics. One approach is using dynamic programming. Let $\Delta X_i, \Delta Y_j, \Delta Z_k$ be the differences only in the plaintext in the case of single-key, or in both the plaintext and the subkeys in case of related-key differentials. First, all one round characteristics $\Delta X_i \rightarrow \Delta Y_j$ are built, i.e. the attacker tries all possible starting differences X_i, and for each of them goes through one round of the cipher and obtains the differences Y_j. Distinct starting differences X_{i_1}, X_{i_2} can produce the same difference Y_j. For each Y_j only the characteristics, that have the

[2] Note that while our characteristics allow certain flexibility in the values due to the compact representation of differences – they are not *truncated differentials* since our goal is to find fully specified differential characteristics, rather than just truncated characteristics.

Table 1. Summary of attacks on the ciphers examined in the paper

Cipher	Attack/Result	Rounds	Data	Workload	Reference
AES-128	Collisions	7	2^{32}	2^{128}	[16]
	Partial sum	7	$2^{128} - 2^{119}$	2^{120}	[14]
	Impossible diff.	7	$2^{112.2}$	$2^{117.2}$	[26]
	Boomerang - RK	7	2^{97}	2^{97}	Section 3.2
AES-192	Rectangle - RK	9	2^{64}	2^{143}	[18]
	Rectangle - RK	10	2^{125}	2^{182}	[22]
	Boomerang - RK	12	2^{123}	2^{176}	[9]
	Boomerang - RK	12	2^{116}	2^{169}	Section 3.3
AES-256	Rectangle - RK	10	2^{114}	2^{173}	[5, 22]
	Subkey Diff.	10	2^{48}	2^{49}	[8]
	Differential - RK	14	2^{131}	2^{131}	[10]
	Boomerang - RK	14	$2^{99.5}$	$2^{99.5}$	[9]
Camellia-128	Impossible	11	2^{118}	2^{126}	[27]
byte-Camellia-128	Chosen-key dist.	18	$2^{6 \cdot 17}$	$2^{6 \cdot 17}$	Section 4.2
Khazad	Slide attack[a]	5	2^{98}	2^{104}	[7]
	Integral	5	2^{64}	2^{91}	[31]
	Boomerang[a] - RK	7	2^{50}	2^{50}	Section 5.2
	Chosen-key dist.	8	2^{55}	2^{55}	Section 5.3

[a] The attack works for a weak key class, and the workload includes the effort to find related keys from the class.

Table 2. The upper bounds on the probabilities of the related-key differential characteristics for full or round-reduced ciphers examined in the paper. The probabilities for a higher number of rounds are below 2^{-k}, where k is the key size.

Cipher	Rounds	Workload	Section
AES-128	5	$2^{6 \cdot 17}$	3.2
AES-192	11	$2^{6 \cdot 31}$	3.2
AES-256	14[b]	2^{131}	3.2
byte-Camellia-128	8	$2^{6 \cdot 19}$	4.1
Khazad	7	$2^{5 \cdot 19}$	5.1

[b] The same characteristics as in [10].

highest probability are left. Next, the attacker builds again all one round characteristics $Y_j \rightarrow Z_k$, for different Y_j (but only those Y_j that were obtained in the first step). Again, for each Z_j he selects only the characteristics that have the highest probability. As a result, he had built the optimal two-round characteristics $X_i \rightarrow Y_j \rightarrow Z_k$. This procedure is repeated until the target n-round differential characteristic is built. The time complexity of the dynamic programming approach is linear in the number of one round characteristics, but it requires a lot of memory for storing the intermediate round values $\Delta Y, \Delta Z$, etc. That is why we will use the second approach, similar to the one used by Matsui in [29] for finding the best differential and linear characteristics in DES. It requires relatively small memory and the time complexity highly depends on the probability of the best round-reduced differential characteristics. The algorithm works by induction: to find the

best n-round characteristic first it finds the best $1, 2, \ldots, n-1$ round characteristics. At some stage it requires building all one round characteristics – their number depends on the size of the search space. Thus a straightforward application of Matsui's search to modern ciphers would immediately fail but there is some hope for byte-oriented ciphers if one switches to compact representations in which each byte is replaced by a single bit: a byte with a difference, also called an *active* byte, is replaced by 1, a byte without a difference — by 0. This means that the difference in n-byte cipher can be represented as n-bit vector.

The compact representation seems optimal, yet several improvements to Matsui's algorithm are still required. Almost all byte-oriented ciphers are designed as substitution-permutations networks (SPN), i.e. they have a layer of S-boxes (S-layer) and a linear diffusion layer – a simple multiplication of the input by a matrix A (P-layer). When the S-boxes are bijective (a property common to most ciphers developed in the last 15 years), then an active byte stays active (and vice-versa) before and after the S-box. Hence, the S-layer does not alter the compact representation. On the other hand, the P-layer can change the number of the active bytes as well as their positions (depending on the exact values of the differences in the active bytes, and the branch number of the matrix A) and therefore it introduces a branching. Thus, besides the traditional compact representation, further denoted as S-value, we will introduce additional representation, called P-value. Indeed, the difference in n-byte cipher will be represented as $2n$-bit vector, where the first n coordinates (bits) are the S-value coordinates, and the next n are the P-value coordinates. The P-value of a difference is obtained when S-value goes through a P-layer and it is the same as the previous S-value (see Fig.1 for clarification). For example, in AES, if the value of a difference of some column is (0,1,0,0,0,0,0,0) (i.e. there is a difference only in the second byte of the column, the difference is of a type $(0, x, 0, 0)^T$) before the MixColumn, then after the MixColumn it is (0,0,0,0,0,1,0,0) meaning: $A(0, x, 0, 0)^T$, where A is the MixColumn matrix and x is an arbitrary non-zero byte value (i.e. it is a four-byte difference, obtained when some column with a difference only in the second byte was multiplied by the MixColumn matrix). Note that the representation can always be reduced to only S-value (although often not uniquely). For example, the above vector (0,0,0,0,0,1,0,0) can be represented as (1,1,1,1,0,0,0,0). The P-values reduce the branching as well: it is better to XOR two P-values, then to reduce them to only S-values and then XOR them. For example, if we XOR two differences (0,0,0,0,0,1,0,0) and (0,0,0,0,0,1,0,0) then the result can be (0,0,0,0,0,1,0,0) or (0,0,0,0,0,0,0,0). On the other hand, if we first reduce them to only S-values, then we will get the values (1,1,1,1,0,0,0,0) and (1,1,1,1,0,0,0,0). Obviously, XOR of these two values gives 2^4 possible outputs. In the states of the ciphers, after each transform, we will have either only non-zero S-value or only non-zero P-value of a difference (but never both), and hence we can effectively eliminate any branching in the state (the branching goes into the key). However even in such representations the search space of 128-bit block, 256-bit key cipher would be 16+32 bits, i.e. 2^{48}. Another complication is that if one would like to search for differential characteristics rather than truncated differentials one will need to pay in heavy branching at every XOR operation both in the state and in the key-schedule, which makes the search completely infeasible. Hence, depending on the key schedule, we would like to propose different variants of the tool to solve these problems:

1. The first variant is the original Matsui's approach itself. It applies to ciphers that *have minimal branching in the key schedule, with subkeys consecutively obtained one from another*. This means that once the difference in the subkey K_i is fixed, the difference in the subkey K_{i+1} can easily and almost uniquely be determined. Let $\Delta X \rightarrow \Delta Y$ be one round differential characteristic, where ΔX is the input difference in **both the state and the subkey**, and ΔY is the output

difference, and let $W(\Delta X \rightarrow \Delta Y)$ be the weight function of this characteristic – the probability cost required to produce a pair that follows the characteristic (the exact definition of W is given later). Let $W_1, W_2, \ldots, W_{n-1}$ be the weights of the best $1, 2, \ldots, (n-1)$-round characteristic found previously with the algorithm and let $\tilde{W}_n$ be the weight of some (not necessarily optimal) n-round characteristic D_n. The search for the best n-round characteristic in pseudo code is described in Alg. 13. In short, first the algorithm builds all possible one round characteristics with a weight at most $\tilde{W}_n - W_{n-1}$. This constraint is introduced to filter some of the one round characteristics: if the weight of the first round is more than $\tilde{W}_n - W_{n-1}$ then it can not be extended to an n-round characteristic because the weight of $n-1$ rounds is at least W_{n-1} so in total it will have a weight more than the previously found characteristic of weight $\tilde{W}_n$. Each of the good one round characteristics (the one that pass the filter) is extended (when possible) to n rounds by the NextRound procedure. One call of this procedure extends the characteristic by one additional round. Again, it extends only the characteristics that satisfy the weight condition, by checking if the sum of the weights of the r and $n-r$ characteristics is not greater than the weight $\tilde{W}_n$ of the already known differential D_n.

2. The second variant of the tool is for ciphers that *have possibly high branching in the key schedule, with subkeys consecutively obtained one from another*. A good example of this type of key scheduling is the one in AES (subkey K_{i+1} is obtained from K_i in one iteration, but due to XORs in the key schedule, there is a lot of branching). If we try to apply the variant 1 of the tool to this type of ciphers we would have to build all one round characteristics (with differences in the state and the subkey). Yet, the high branching in the subkey, blows the number of characteristic out of proportion, and the search becomes infeasible. That is why we have to modify the tool for this special case of ciphers. Let ΔS_r be the difference in the state of round r after the XOR of the subkey K_r and let ΔK_r be the difference in this subkey. To add one more round to this characteristic one can proceed as follows:
   - take ΔS_r and go through all one round transformations of the state to build $\Delta \tilde{S}_{r+1}$ which is the difference in the state of round $r+1$ just before the XOR of the subkey K_{r+1}
   - take any ΔS_{r+1}
   - XOR $\Delta \tilde{S}_{r+1}$ and ΔS_{r+1} to produce ΔK_{r+1}
   - check if ΔK_{r+1} can be obtained from ΔK_r in one round.

   This way, instead of building all one round characteristics in the subkey, we only have to check if some subkey difference can be transformed to another difference in one subkey round (see Fig. 1). The number of these transitions that has to be checked is related to the size and branching of the state. Usually the state has smaller size than the subkey, and with the right representation it can have minimal or no branching, leading to a feasible search. Let ΔP be the plaintext difference, $\Delta S_r, \Delta K_r$ be the difference in the state and the subkey of round r, $\Delta \tilde{S}_r$ the difference in the state of round r just before the subkey XOR, $W(\Delta S_r \rightarrow \Delta \tilde{S}_{r+1})$ the probability of the characteristic $\Delta S_r \rightarrow \Delta \tilde{S}_{r+1}$. The notions of W_i are the same as in the previous variant. For the sake of clarity we assume there is no whitening key. The variant 2 of our search tool is described in Alg. 1.2.
3. The third variant of the tool applies to ciphers that *have key schedule with subkeys that are not successively obtained one from another*. Usually, the key schedule of these ciphers applies heavy transformations to the master key to obtain another key, and then combines these two keys (often with linear transformations) to get all subkeys. To build the tool we will use the following strategy: 1) from the master key, obtain all the subkeys, and 2) apply the first variant

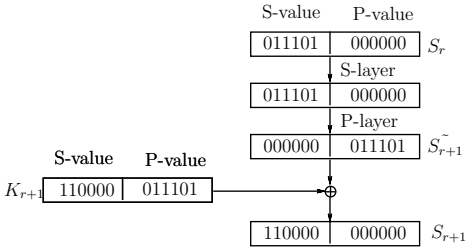

Fig. 1. The variant 2 of the tool with S- and P-value representations.

of the tool – build the characteristics for the state, but use the obtained subkeys (instead of building characteristics for subkeys). Let $\Delta X \xrightarrow{\Delta K} \Delta Y$ denote one round characteristic where $\Delta X, \Delta Y$ are the differences in the initial and final states, and ΔK is the subkey used in that round. The rest of the notions are the one used in variant 1. The pseudo code of the third variant is given in Alg. 1.3.

Algorithm 1.1. Search of n-round differential characteristics - Variant 1

for all $\{\Delta X \rightarrow \Delta Y | W(\Delta X \rightarrow \Delta Y) + W_{n-1} \leq \tilde{W_n}\}$ **do**
  Call NextRound(ΔY, $W(\Delta X \rightarrow \Delta Y)$, 2)
end for

NextRound(ΔY, w, r)
for all $\{\Delta Z | \Delta Y \rightarrow \Delta Z$ and $W(\Delta Y \rightarrow \Delta Z) + w + W_{n-r} \leq \tilde{W_n}\}$ **do**
  if $r = n$ **then**
    Update D_n
    $\tilde{W_n} \leftarrow w + W(\Delta Y \rightarrow \Delta Z)$
  else
    Call NextRound($\Delta Z, w + W(\Delta Y \rightarrow \Delta Z), r + 1$)
  end if
end for

Now, let us determine the weight function $W(\Delta_1 \rightarrow \Delta_2)$ of the one round characteristics $\Delta_1 \rightarrow \Delta_2$. In the attacks on AES [10, 9], the attacker pays only for the active S-boxes (active bytes that go through S-boxes) in the state and the subkey in each round of the cipher. Hence, we will use the same definition: $W(\Delta_1 \rightarrow \Delta_2)$ is defined as the number of active S-boxes in the state and the subkey in the one round characteristic $\Delta_1 \rightarrow \Delta_2$.

When searching for n-round differential characteristic, the upper bounds on the weight of these characteristics are limited by the maximal number of active S-boxes that a characteristic can have. These upper bounds, depend on the key size and the difference propagation probability of the S-boxes which is usually 2^{-7} (sometimes 2^{-6} or even 2^{-5}). The weight of the n-round differential characteristic for a cipher with k-bit key, and S-boxes with maximal difference propagation proba-

Algorithm 1.2. Search of n-round differential characteristics - Variant 2

for all $\Delta P, \Delta S_1$ **do**
  Obtain $\Delta\tilde{S_1}$ from ΔP
  $\Delta K_1 = \Delta\tilde{S_1} \oplus \Delta S_1$
  Call NextRound($\Delta S_1, \Delta K_1, W(\Delta P \rightarrow \Delta\tilde{S_1}), 2$)
end for

NextRound($\Delta S_{r-1}, \Delta K_{r-1}, w, r$)
Obtain $\Delta\tilde{S_r}$ from ΔS_{r-1}
if $W(\Delta S_{r-1} \rightarrow \Delta\tilde{S_r}) + w + W_{n-r} \leq \tilde{W_n}$ **then**
  for all ΔS_r **do**
    $\Delta K_r = \Delta\tilde{S_r} \oplus \Delta S_r$
    if $r = n$ **then**
      Update D_n
      $\tilde{W_n} \leftarrow w + W(\Delta S_{r-1} \rightarrow \Delta\tilde{S_r})$
    else
      Call NextRound($\Delta S_r, \Delta K_r, w + W(\Delta S_{r-1} \rightarrow \Delta\tilde{S_r}), r+1$)
    end if
  end for
end if

Algorithm 1.3. Search of n-round differential characteristics - Variant 3

for all master $\Delta K|$ obtain subkeys $\Delta K_1, \ldots \Delta K_n$ with weight W_K **do**
  for all $\{\Delta X \xrightarrow{\Delta K_1} \Delta Y | W(\Delta X \xrightarrow{\Delta K} \Delta Y) + W_K + W_{n-1} \leq \tilde{W_n}\}$ **do**
    Call NextRound(ΔY, $W(\Delta X \xrightarrow{\Delta K} \Delta Y)$, 2)
  end for
end for

NextRound(ΔY, w, r)
for all $\{\Delta Z | \Delta Y \xrightarrow{\Delta K_r} \Delta Z$ and $W(\Delta Y \xrightarrow{\Delta K_r} \Delta Z) + w + W_K + W_{n-r} \leq \tilde{W_n}\}$ **do**
  if $r = n$ **then**
    Update D_n
    $\tilde{W_n} \leftarrow w + W(\Delta Y \xrightarrow{\Delta K_r} \Delta Z) + W_K$
  else
    Call NextRound($\Delta Z, w + W(\Delta Y \rightarrow \Delta Z), r+1$)
  end if
end for

bility 2^{-l} is upper bounded by $\lfloor \frac{k}{l} \rfloor$. The related-key differential characteristics produced by the tool have fixed positions of the active bytes, while the exact values are undefined. To produce standard differential characteristics, one has to find the exact values of the active bytes (the differences in the active bytes). The probability of the standard characteristics may be lower than the one predicted by the tool, but never higher, because the tool assumed that all active S-boxes hold with maximal differential probability, while in practice (in the case of standard characteristic) some S-boxes may hold with lower probability.

A new class of attacks, presented in [23, 10], called *open-key* attacks, gives the attacker the full freedom of *knowing* or even *choosing* the key. In return, the attacker has to demonstrate some non-trivial property of the cipher which differentiates it from an ideal cipher. The motivation behind these attacks is that ciphers are often used as building blocks for some other cryptographic primitives, such as hash functions. There, the attacker has a full freedom of choosing all input parameters. An interesting approach is applicable to all ciphers in the chosen-key attack model. We call this approach *divide-and-conquer* technique. Let us have some related-key differential characteristic for a cipher. Since we control both the key and the state (it is a chosen-key attack), we can find a good pair of keys and states that follow the characteristic by the following method: 1) first find a good pair of keys that follow the differential characteristic only in the key, 2) once the subkeys are fixed, find a good pair of plaintexts that follow the differential characteristic in the state. It means we can split the whole characteristic in two halves: the one in the key, and the one in the state, and *instead of multiplying their probabilities, we can add them.* We will launch chosen related-key differential attacks on the full-round ciphers, in the cases when (secret) related-key characteristics do not exist. Note that proving the resistance against the chosen related-key differential attacks is still an open problem because it is unclear how to estimate the upper bound on the weighs of these characteristics since the number of rounds that can be covered for free varies from 1 in the rebound attack [30], 2 in the Super-Sbox [17], and even more in the tool of Khovratovich et al [21].

3 AES

The 128-bit block version of Rijndael[13] has been standardized by NIST as Advanced Encryption Standard (AES) in November 2001 [32]. It supports three different key sizes: 128, 192, and 256 bits, denoted as AES-128, AES-192, and AES-256, respectively. Various cryptanalytic results were published on AES, and until recently, the best attacks presented non-random properties of 7/10/10 rounds (out of 10/12/14 rounds) of AES-128/192/256 [14, 16, 23, 18, 22]. A breakthrough in analysis of AES have been the results [10, 9]. In [10] a related-key attack on all 14 rounds of AES-256 was presented. In [9], boomerang attacks on full-round AES-192 and AES-256 were shown.

AES is an SPN cipher. The subkeys are generated consecutively one from another, but there is a lot of branching caused by the XORs of columns in the key schedule. Hence, *we will use variant 2 of the tool.* The state goes through four transformations: S-box layer, ShiftRows, linear-diffusion layer called MixColumns, and XOR of the key. In the tool we will use the following optimal representation of the state trough one round: the beginning state (before the S-boxes) can have non-zero only S-value (but zero P-value), after the S-box layer and after ShiftRows has again only non-zero S-value, after the MixColumns has non-zero only P-value, and after the subkey XOR again it has only non-zero S-value. This way, there is no branching in the state. The subkeys then can be determined as a XOR of a state of only P-value (the one after MixColumns) and a state of only S-value (the next round state, just before the S-boxes), hence they have columns that can have both non-zero S- and P-values. To use variant 2 of the tool we would have to be able to determine if the difference in the subkey K_{i+1} can be obtained from the difference in K_i. One subkey round consists of XOR of columns, application of S-boxes, and rotation of a column. If we represent the columns of the subkeys simply with only S-value, all of the above transforms can be easily checked. Therefore, each column of the subkeys is reduced only to S-value: 1) convert P-value into S-value, 2)XOR the obtained S-value with the initial S-value. Note, reduction to S-value as well as the XOR introduce branching, but the search is still feasible.

3.1 Best Round-Reduced Differential Characteristics for AES

We have applied the tool to all three versions of AES. The maximal difference propagation of the S-box in AES is 2^{-6}. Since the key sizes are 128,192, and 256, we can allow no more than 21, 31, and 42 active S-boxes in the characteristics for AES-128, AES-192, and AES-256, respectively. For AES-128 we found differentials characteristics on 4 rounds with 13 active S-boxes, and on 5 rounds with 17 active S-boxes. *In AES-128 there are no 6-round related-key differential characteristics.* For AES-192 we found differential characteristics up to 11 rounds out of 12. The characteristic on 11 rounds has 31 active S-boxes (20 in the state, and 11 in the key). For AES-256 we found unique differential characteristic on all 14 rounds, but this is the same characteristic that was presented in [10]. The characteristic from [10] is optimal for 9-14 rounds, but we have found better characteristic on 8 rounds (10 active S-boxes instead of 14). The 5-round differential characteristic for AES-128, and 11-round for AES-192 are presented in Fig. 2 in the Appendix. Regarding chosen-key attacks, in all versions of AES, there are no differential characteristic on 10 rounds with 21 or less active S-boxes in the state.

3.2 Related-key Boomerang Attack on 7-round AES-128

Let us show a boomerang attack on 7 rounds of AES-128. We will use two 3-round differential characteristics: a top 3-round truncated differential characteristic (4-1-4-16) with no key difference, and a 3-round related-key bottom differential characteristic with 5 active S-boxes in the state and 1 in the subkeys. They are presented in the Figure 4 in the Appendix. Note that if we extend the bottom characteristic for one additional round then the difference in the ciphertexts is fixed in 9 bytes, 4 bytes have equal difference and 3 bytes have a random difference. The difference δ between these two ciphertexts can have $2^{4 \cdot 7} = 2^{28}$ distinct values (1-bit of freedom is lost for each of the 4 active S-boxes, since given a fixed input difference only 2^7 output differences are possible). Let Δ be the difference between K^4 and let the key schedule transform this difference into Δ' in K^7. Instead of guessing 2^{28} possibilities of the bottom difference for each ciphertext (which would increase the pressure on our filters) we guess 31 bit of the key: seven bits of $k^6_{1,3}$ and full $k^7_{1,1}, k^7_{1,2}, k^7_{1,3}$. This guess allows us to work on both faces of the bottom characteristic, since unlike in most related-key boomerang attacks our boomerang has only two related keys, instead of four.

The attack works as follows: For each guess of 2^{31} bits of the key

1. Prepare a structure of plaintexts P_i with all the possible 2^{32} four byte values on the main diagonal and the other bytes fixed.
2. Enrypt all the plaintexts P_i with the secret key K and obtain ciphertexts C_i.
3. For each ciphertexts C_i compute the correct difference δ using the 31-bit key guess, and obtain $D_i = C_i \oplus \delta$.
4. Decrypt all D_i with the key which is computed from the last subkey: $K^7 \oplus \Delta'$ and obtain plaintexts Q_i.
5. Sort all Q_i by 12 non-diagonal bytes. Pick only the pairs (Q_i, Q_j) that have zero difference in these 12-bytes. If none are found then goto 1.
6. Check the candidate quartet against 8 active S-boxes at the top (four on both sides of the boomerang) which gives an 8-bit filter.
7. Do the key counting step with the remaining quartet candidates.

Let us calculate the data and time requirements of the attack. A pair of plaintexts passes the first round with a probability 2^{-22} (MixColumns from four to one active byte, the position and the value of the active byte is irrelevant). The next two rounds are passed with probability 1, so in the third round with have a pair of states with all bytes active. In the second characteristic, from the bottom up assume that the initial 31-bit guess was correct, then in the next three rounds we have five active S-boxes which hold with probability 2^{-30} (when each is 2^{-6}). Yet for the two pairs of ciphertexts we only need the same difference after the fourth round (from bottom up) and therefore the two S-boxes of this round can be counted only once (on the one side of the boomerang the pair passes the layer of S-boxes with one of the 2^{14} possible differences, on the parallel side of the boomerang the pair matches this difference with probability 2^{-14}). So the two pairs of ciphertexts pass the second characteristic with a probability of $2^{-(3\cdot 6+3\cdot 6+2\cdot 7)} = 2^{-50}$. Now that we have passed with low cost the 4th round layer of S-boxes where the top-down characteristic had 16 active S-boxes we switch to the last phase of the boomerang attack, where the effect of the mixing of the third round can be undone for free because are guaranteed to have the same difference as in the forward direction. In the second round we pay again 2^{-24} for four to one active byte of MixColumns(2^{-22} if we do not require the boomerang to return in exactly the same 4 bytes). The next round is done for free so we obtain two plaintexts with a difference only in four diagonal bytes. Hence the total probability of the boomerang is $2^{-22-50-24} = 2^{-96}$. Each structure of 2^{32} plaintexts contains 2^{63} pairs with a difference in the four diagonal bytes. Hence, to find two good boomerang quartets we need $2^{96-63+1} = 2^{34}$ structures or $2^{34+32} = 2^{66}$ chosen plaintexts and $2^{31} \cdot 2^{66} = 2^{97}$ adaptive chosen ciphertexts. The average amount of false quartets for all 2^{31} key guesses which satisfy our 96+8 = 104-bit filtering condition is $2^{31+34+63-104} = 2^{24}$. Note that each boomerang quartet suggest 31-bit value for the key guess at the bottom as well as 16 guesses for 64 bits at the top (corresponding to 4 active S-boxes in the plaintext at each side). Since we requested two good boomerang quartets they will vote together for the correct keys while the remaining 2^{24} false quartets would vote randomly. We expect that none of the false quartets survive this 91-bit key voting step.

At this point the attacker can either finish the attack with an exhaustive search of about 2^{96} steps or by repeating the boomerang attack starting from another 4 active S-boxes in the plaintext.

3.3 Related-key Boomerang Attacks on AES-192 and AES-256

We tweaked our tool to produce the optimal differential characteristics for a boomerang attack on AES-192. The tool produced a top differential characteristic other than the one presented in [9], with the same bottom characteristic. The two characteristics are shown at Fig.3 in the Appendix. The ladder switch between the two characteristics in round 6 is simpler: due to the switch there are no active S-boxes in this round. The top characteristic has 2 active in round 3, and 1 in round 4, while the bottom characteristic has 1 active in round 7, 8, and 10, and 2 active in round 9. Hence, the probability of the boomerang is $2^{-6\cdot(2+1+1+1+2+1)} = 2^{-48}$ compared to the boomerang in [9] with a probability 2^{-55}. A rough estimate between these two attacks gives us a speed-up of 2^7: the new boomerang attack requires 2^{116} data, and 2^{169} time.

For AES-256, on 6 and 7 rounds there are only two characteristics with 5 active S-boxes (and no characteristics with less active S-boxes), and these are the exact characteristics used in the boomerang attack on AES-256 in [9]. On the other hand, 8-round characteristic has at least 10 active S-boxes, hence using it in a boomerang attack will blow up the complexity above the best known attack. Therefore, we believe that the characteristics used for the attack on AES-256 in [9] are optimal.

4 Camellia

Camellia [1] is a 128-bit SPN block cipher with 128, 192, and 256-bit keys. We will analyze Camellia with 128-bit keys, without the FL functions. This version has 18 rounds, and so far, the best cryptanalytical results are truncated differential of 8 rounds [25], and impossible differential on 11 rounds [27]. The key schedule of Camellia is not byte oriented because the rotation constants are not a multiple of 8. In order to test our tool we will make it byte oriented, by using the following rotation constants for rounds 1-18: 0, 0, 16, 16, 16, 16, 48, 48, 48, 64, 64, 64, 96, 96, 96, 96, 112, 112. We call this version — byte-Camellia. Note that, since we choose the rotation constants as close as possible to the original constants, a differential characteristic in the key schedule of byte-Camellia, may be suitable for the original Camellia. For that to happen, the positions of the active *bits* in an active byte have to be invariant of small rotations. On the other hand, trying all possible combinations of active bits, i.e. building all possible differential characteristics in the key schedule for the original version of Camellia, seems too much time consuming. Hence, we will analyze only byte-Camellia.

The key schedule of Camellia-128 applies transforms (4 rounds) to the master key K_L, to produce another key K_A and it uses these two values to generate the subkeys in a linear way. Therefore, *we will use variant 3 of the tool*. Internally in the tool, in all steps we will use only the S-type representation.

4.1 Best Round-reduced Differential Characteristics for byte-Camellia

The maximal difference propagation probability of the S-boxes in Camellia is 2^{-6}. Therefore, we can allow no more than $\lfloor \frac{128}{6} \rfloor = 21$ active S-box in the characteristic of the key and the state. With this type of limitations, the tool produced the best related-key differential characteristic. It is on 8 rounds, and it has 20 active S-boxes.

4.2 Chosen-key Attack on Full-round byte-Camellia

When searching for chosen related-key characteristics in byte-Camellia, we can spend 21 active S-box in each, the key and the state (using the divide-and-conquer technique). With these weight limitations our tool was able to produce a good characteristic on all 18 rounds of byte-Camellia. The characteristic has 17 active S-boxes in the key, and 15 in the state (see Fig. 4 in the Appendix). The characteristic can be used to show that 256-bit double-block-length [19] hash function construction initiated with byte-Camellia-128 cipher, can be distinguished from a random function.

5 Khazad

Khazad [3] is a 64-bit block cipher with a key size of 128 bits. It is an SPN with 8 rounds. The best attacks go only up to 5 rounds: an integral attack [31] with 2^{91} complexity and a class of 2^{64} weak keys which can be attacked in 2^{40} steps using a slide attack [7].

The subkeys in the key schedule of Khazad are obtained consecutively from one another using a Feistel function. Therefore, *we will use variant 2 of the tool*. The small key and block sizes, in addition to the low branching in the key schedule allows to use the variant 1 as well. The optimal representation is similar to the one used in the tool for AES. In the state, after the S-boxes (γ) we will have non-zero only S-value, and after the linear-diffusion layer (θ) only P-value.

5.1 Best Round-reduced Differential Characteristics for Khazad

The maximal difference propagation of the S-boxes in Khazad is 2^{-5}. Hence, a differential characteristic for Khazad cannot have more than 25 active S-boxes, at most 12 can be in the state. With this type of limitations, the tool was able to produce interesting results. The best related-key differential characteristics for 4,5,6, and 7 rounds have 9, 10, 19, and 20 active S-boxes, respectively. The related-key attacks based on such characteristics would be the new best attacks on Khazad up to 7 rounds. The 7-round characteristic is presented at Fig.5 in the Appendix.

5.2 Related-key Boomerang Attacks on 7 rounds of Khazad

Let us improve the probability of the 7-round attack by using a boomerang attack. We will use two 4 round characteristics (See Fig.5 in the Appendix). The four related keys K^A, K^B, K^C, and K^D, are obtained as follows: 1) fix any K^A, i.e. (K^A_{-2}, K^A_{-1}), 2) produce (K^A_0, K^A_1) from K^A, and fix K^B such that $K^B = (K^A_0, K^A_1) \oplus (\Delta K_0, \Delta K_1)$, 2) obtain (K^A_6, K^A_7) and (K^B_6, K^B_7) and then fix $K^C = (K^A_6, K^A_7) \oplus (\Delta K_6, \Delta K_7)$, $K^D = (K^B_6, K^B_7) \oplus (\Delta K_6, \Delta K_7)$. The pink difference was chosen such that after γ and θ it could produce gray difference with a probability 2^{-5}. Let us find the complexity of the attack. We start with the same one byte difference (the pink byte) in the plaintext and the subkey K_0, hence there are no active bytes in the state in round 1 and 2. The difference in the subkey K_3, as well as in the state, (denoted with the grey bytes) is obtained when the pink byte goes through γ and θ, and hence it happens with 2^{-5}. At the end of round 4 we can switch to the bottom characteristic. The ciphertext difference is fully determined. We pay 2^{-5} in round 7 so the blue byte in the state after the inverse S-box will become pink (and then cancel with the pink difference in the key). To get a zero difference in the subkey K_5 we pay additional 2^{-5}. In round 4 we switch the state to the top characteristic. An important moment is the switch in the keys. When the gray difference in the top characteristic between K^A and K^B in the subkey K_3 is the same as the gray difference in the bottom characteristic between K^A and K^C (and K^B and K^D) in the subkey K_3, then the switch in the key is for free (this is due to the Feistel switch[3], See [9]). Then not only the difference in K_3 between K^C and K^D will be the same as between K^A and K^B, but their value will be equal to the values of K^A_3 and K^B_3 and hence will go through the S-boxes producing the same values. Therefore, we pay additional 2^{-5} for each of the differences in subkey K_3 (instead of a switching cost of 2^{-64}!). After the switch to the top characteristic, we pay 2^{-5} in the state of round 3 to get the same pink difference which will cancel after the key XOR. We pay additional 2^{-5} for the zero difference in the subkey K_1. The rest of the characteristic holds with probability 1. The probability of the whole boomerang attack is $2^{-(2\cdot5+2\cdot5+2\cdot5+2\cdot5+5+5)} = 2^{-50}$. This translates into a boomerang attack in a class of weak keys that works for 1 out of 2^{30} related-key quartets, with a complexity 2^{20} encryptions/decryptions. Moreover if we relax constraints on the difference in the key we can increase the size of the weak key class to 1 out of every 2^8 keys which can be attacked with complexity of 2^{49} encryptions and analysis steps. In both cases when the boomerang returns we know the plaintext difference and thus we have a 64 bit filter which allows us to filter out all the wrong quartets. Returning boomerang provides us with 7 bits of information about the key byte K^0_2 since we know the input and output difference for the active S-box of the key schedule and similarly about 7 bits of the key of K^0_6. One can extend this attack into a full key recovery attack via auxiliary techniques.

[3] We have tested the Feistel switch in the key schedule of Khazad, and a related-key quartet, following the whole 7-round differential characteristic, was found.

5.3 Chosen-key Attack on Full-round Khazad

The 7-round related-key differential characteristic can easily be extended at the top for an additional round and then used in a chosen-key attack (See Fig. 5 in the Appendix). Since we control the exact values of the key and the state, we will use the divide-and-conquer technique, and first fix the keys satisfying the characteristic in the key, and then find a proper pair of plaintexts that follow the characteristic in the state. We can use the rebound attack [30] and fix one round for free in both the key and in the state. For the key, we can fix the round for ΔK_4 (or ΔK_6), and obtain a characteristic in the key that holds with probability 2^{-55}. In the state, we will fix the values in the first round, hence the characteristic in the state holds with 2^{-10}. The S-boxes are non-injective regarding the difference, i.e. if we fix the input and output difference of the S-box, then there is a solution with probability $\frac{1}{2}$. Therefore, we introduce a possible one bit difference in each byte of the plaintext so that there will always be a solution for the S-box input/output differences. The total complexity of the chosen-key distinguisher is bounded by the probability of the characteristic in the key and is 2^{55}. The input difference is fixed in 56 bits, while the output is fixed in all 64-bits. This shows that full Khazad has properties which are not present in an ideal cipher. This also means, for example, that 256-bit Tandem-DM[24] hash function construction initiated with Khazad cipher, can be distinguished from a random function.

6 FOX and Anubis

The ciphers FOX [20] and Anubis[2] have highly non-linear key schedule, leading to a potentially low key agility. This property becomes important when the key of the cipher is frequently changed, for example when the cipher is used in some hash function construction. Yet, with the respect to related-key differentials, the key schedule of these ciphers is exceptionally resistant. The large number of S-boxes in the schedule ensures that a related-key differential attack cannot be launched on more than some very modest number of rounds.

We have analyzed FOX64 – the 64-bit block version of FOX with 128-bit key and 16 rounds. Each round key of this cipher is produced from the master key with a sequence of transformations $NL64$. We have found empirically, by checking all the possible key differential characteristics, that the minimal number of active S-boxes in $NL64$ is 7. This means that in any related-key differential characteristic, for each round of FOX64 one has to spend at least 7 S-boxes *only* for producing the round key. The maximal difference propagation probability of the S-box in FOX is 2^{-4}, while the key is 128 bits. Hence, we can conclude that for FOX64 there is no related-key differential characteristic on more than $\lfloor \frac{128}{4 \cdot 7} \rfloor = 4$ rounds.

Anubis, a 128-bit block cipher, supports variety of key sizes from 128 bits to 320 while it has $8 + \frac{keysize}{32}$ rounds. We will focus on key sizes up to 256 bits. The maximal difference probability of the S-boxes is 2^{-5}, hence the maximal number of active S-boxes in a characteristic can not be greater than $\lfloor \frac{256}{5} \rfloor = 51$. The key schedule of Anubis is SPN with an additional S-box layer at the end of each round (as well as ω and τ transformations). This means that in a characteristic of the key schedule, each active S-box should be counted twice, except for the S-boxes in the first round (which are counted only once). The branch number of the linear-diffusion layer is 5, hence in four consecutive rounds there are at least $5^2 = 25$ active S-boxes. Therefore, in 5 rounds of the key schedule, there are at least $2 \cdot 25$ active S-boxes in the last 4 rounds, and at least 1 in the first round, or in total at least 51 active S-boxes in the 5-round characteristic of the key schedule. Hence, in Anubis there are no related-key differential characteristics on more than 5 rounds.

7 Conclusions and Future Research

We presented a tool for search of related-key differential characteristics in various ciphers. It produced the best round-reduced differential characteristics, which helped to improve the best known attacks on AES-128, AES-192, byte-Camellia-128, and Khazad. It also allowed to prove security bounds against simple related-key attacks for AES-128, FOX and Anubis. The tool runs in a range of few hours (Khazad, byte-Camellia) to several days (AES-128) and weeks (AES-192) and takes from several megabytes to 25 Gbytes (byte-Camellia) of memory for the transition lookup tables in the key schedule.

The tool was implemented as described in the paper and while it produced a lot of interesting results, a couple of open problems emerged. The first one is how to deal with ciphers that have small part that is not byte oriented such as Camellia which has rotations in the key schedule. Second, it is very interesting to adapt the tool to hash functions. This would mean that the problem of very large internal state of some modern hash functions has to be solved. The idea of applying a similar tool for finding related-key differential characteristics in DES and in non-byte oriented ciphers also seems attractive. Producing chosen-key differential characteristics for 9 and 10 rounds of AES-128 is still an open problem.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In D. R. Stinson and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2000.
2. P. Barreto and V. Rijmen. The Anubis Block Cipher. In *Submission to the NESSIE Project*, 2000.
3. P. Barreto and V. Rijmen. The Khazad Legacy-Level Block Cipher. In *Submission to the NESSIE Project*, 2000.
4. E. Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.
5. E. Biham, O. Dunkelman, and N. Keller. Related-key boomerang and rectangle attacks. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 507–525. Springer, 2005.
6. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
7. A. Biryukov. Analysis of involutional ciphers: Khazad and Anubis. In T. Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 45–53. Springer, 2003.
8. A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, , and A. Shamir. Key recovery attacks of practical complexity on AES variants with up to 10 rounds. In *EUROCRYPT 2010, to appear*, 2010.
9. A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
10. A. Biryukov, D. Khovratovich, and I. Nikolić. 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.
11. C. D. Cannière and C. Rechberger. Finding SHA-1 characteristics: General results and applications. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.

12. D. R. Chowdhury, V. Rijmen, and A. Das, editors. *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*. Springer, 2008.
13. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
14. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved cryptanalysis of Rijndael. In B. Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2000.
15. P.-A. Fouque, G. Leurent, and P. Nguyen. Automatic search of differential path in MD4. *Cryptology ePrint Archive, Report 2007/206*.
16. H. Gilbert and M. Minier. A collision attack on 7 rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.
17. H. Gilbert and T. Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations. *Fast Software Encryption, 2010, to appear.*, 2010.
18. M. Gorski and S. Lucks. New related-key boomerang attacks on AES. In Chowdhury et al. [12], pages 266–278.
19. S. Hirose. Some plausible constructions of double-block-length hash functions. In Robshaw [33], pages 210–225.
20. P. Junod and S. Vaudenay. FOX : A new family of block ciphers. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2004.
21. D. Khovratovich, A. Biryukov, and I. Nikolić. Speeding up collision search for byte-oriented hash functions. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 164–181. Springer, 2009.
22. J. Kim, S. Hong, and B. Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In A. Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2007.
23. L. R. Knudsen and V. Rijmen. Known-key distinguishers for some block ciphers. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007.
24. X. Lai and J. L. Massey. Hash function based on block ciphers. In R. A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.
25. S. Lee, S. Hong, S. Lee, J. Lim, and S. Yoon. Truncated differential cryptanalysis of Camellia. In K. Kim, editor, *ICISC*, volume 2288 of *Lecture Notes in Computer Science*, pages 32–38. Springer, 2001.
26. J. Lu, O. Dunkelman, N. Keller, and J. Kim. New impossible differential attacks on AES. In Chowdhury et al. [12], pages 279–293.
27. J. Lu, J. Kim, N. Keller, and O. Dunkelman. Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In T. Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 370–386. Springer, 2008.
28. M. Matsui. Linear cryptoanalysis method for DES cipher. In T. Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.
29. 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.
30. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In O. Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.
31. F. Muller. A new attack against Khazad. In C.-S. Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 347–358. Springer, 2003.
32. National Institute of Standards and Technology. Advanced encryption standard (AES). FIPS 197, November 2001.

33. M. J. B. Robshaw, editor. *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*. Springer, 2006.
34. M. Schläffer and E. Oswald. Searching for differential paths in MD4. In Robshaw [33], pages 242–261.
35. M. Stevens. Fast collision attack on MD5. *Cryptology ePrint Archive, Report 2006/104*.

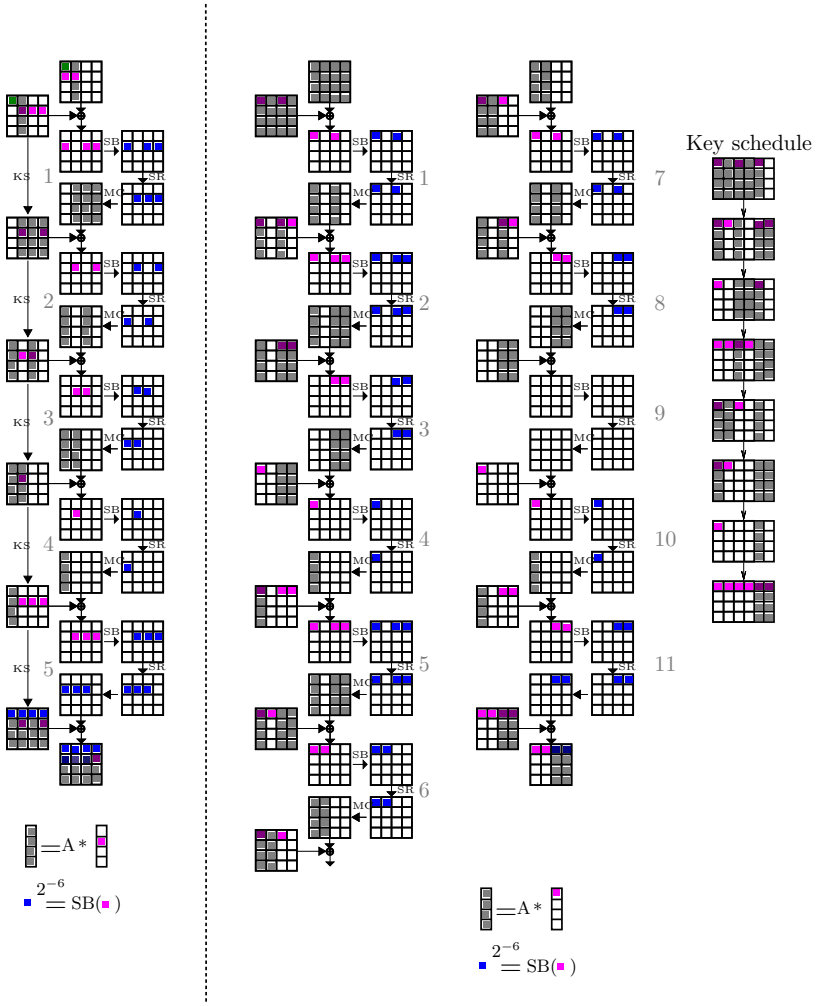

Fig. 2. The best characteristics for AES-128(left) and AES-192(right). The first one is on 5 rounds, while the second one is on 11 rounds.

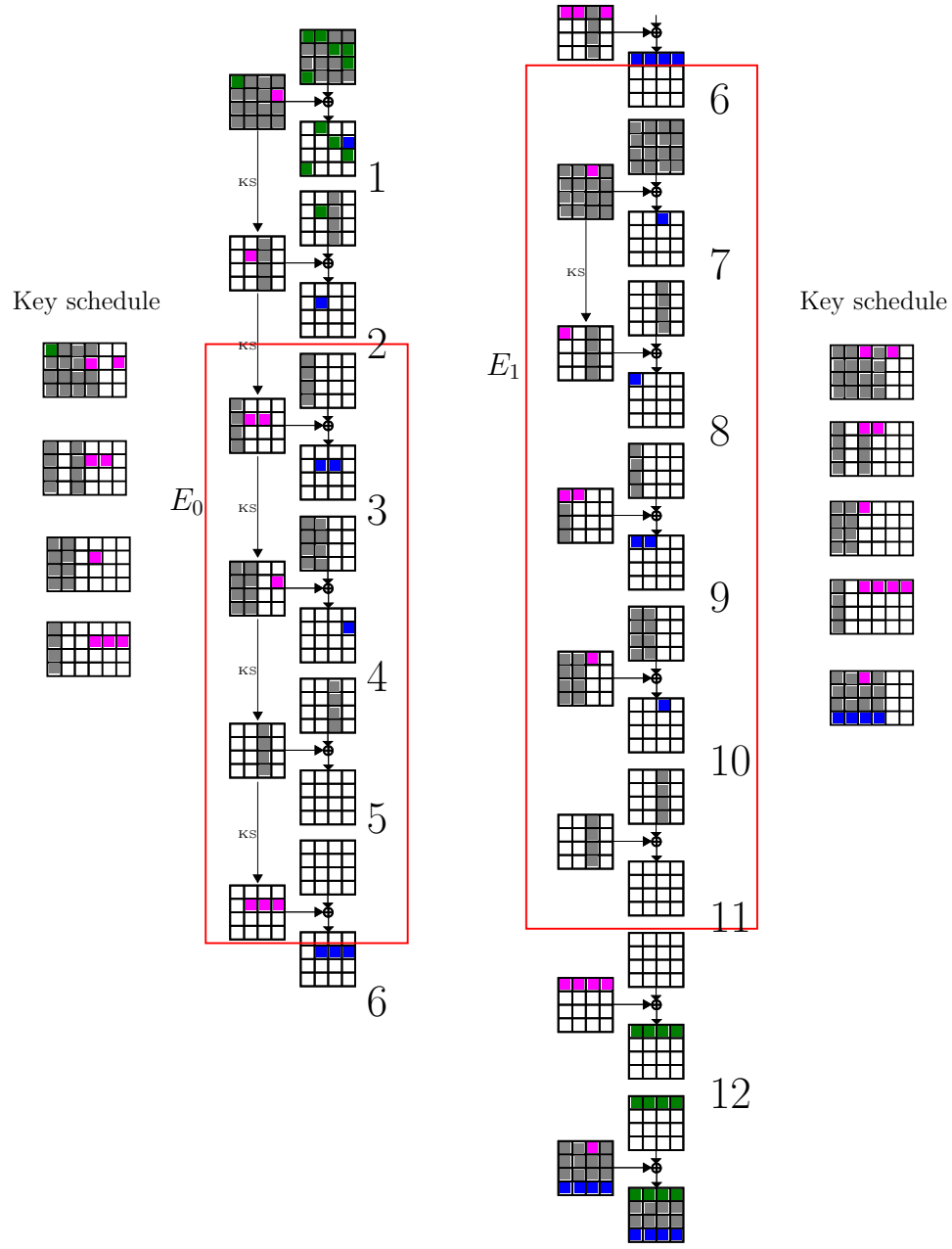

Fig. 3. A related-key boomerang attack on AES-192. The bottom characteristic is the same as in [9].

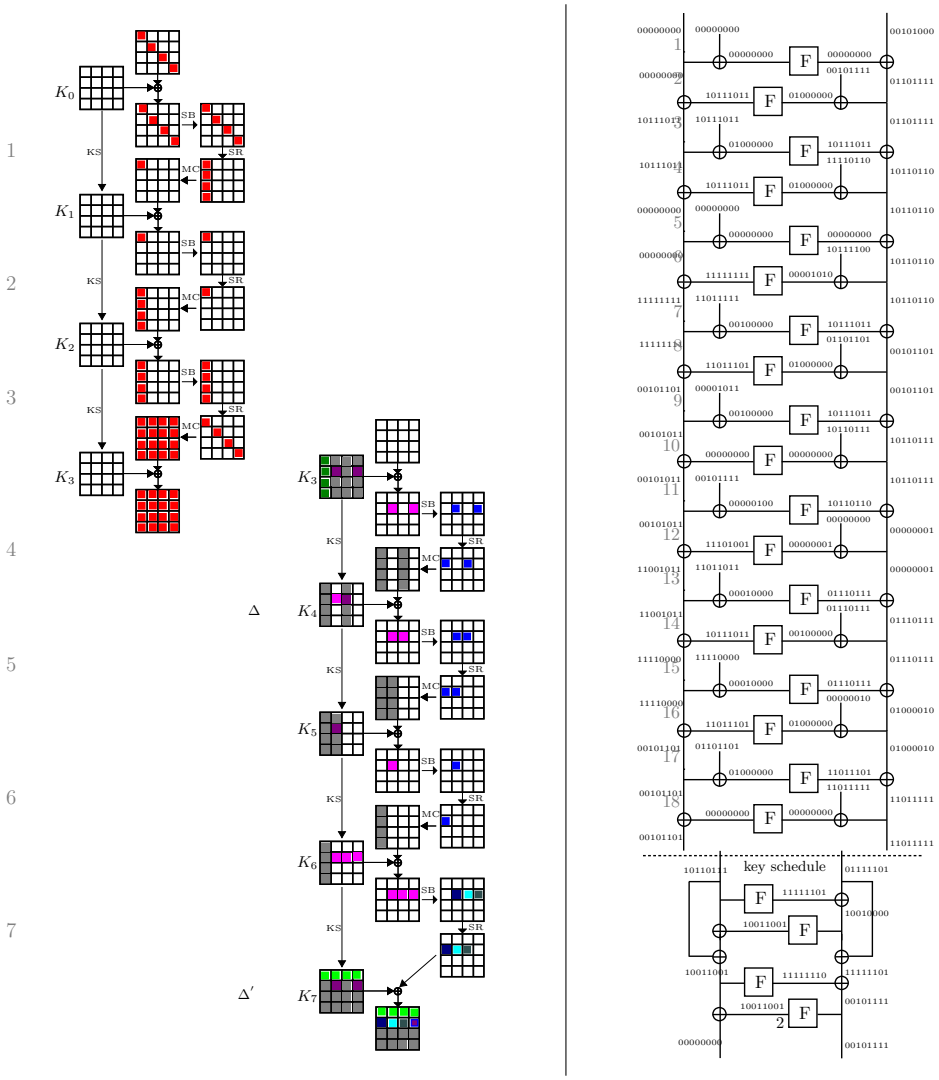

Fig. 4. On the left, two characteristics for the related-key boomerang attack on AES-128 reduced to seven rounds. On the right, the related-key differential characteristic (top in the state, bottom in the key) on full-round byte-Camellia-128 for the chosen-key distinguisher. The characteristic has compact representation, the actual differences are to be fixed. The key XOR is depicted separately from the function F.

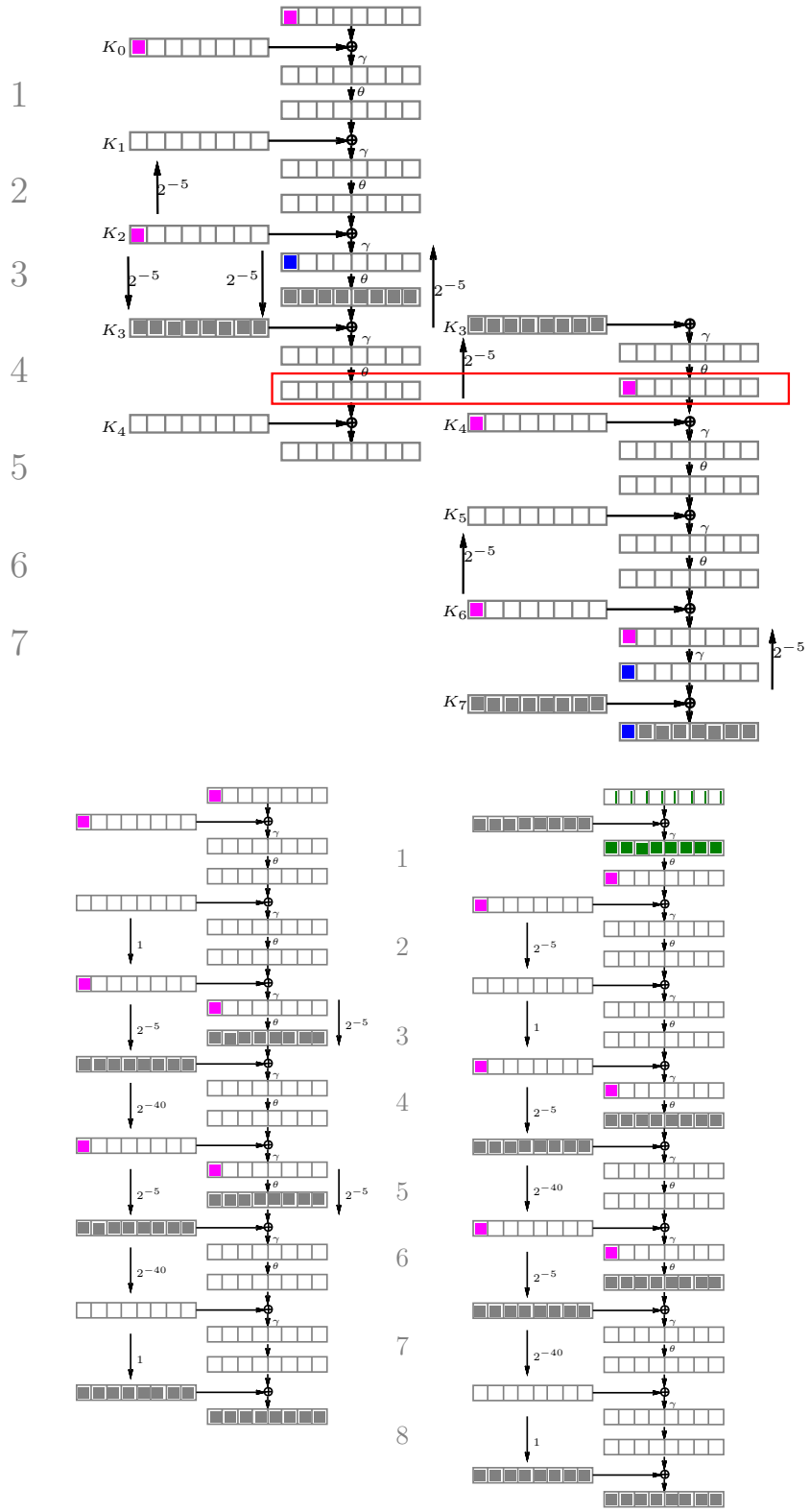

Fig. 5. Characteristics for the related-key boomerang attack on 7-round Khazad (top), related-key differential characteristic on 7 rounds (bottom-left), and related-key differential characteristic on 8 rounds used for a chosen-key distinguisher (bottom-right).