Related-key Cryptanalysis of the Full AES-192 and AES-256

Alex Biryukov and Dmitry Khovratovich

University of Luxembourg
29 May 2009

Abstract. In this paper we present two related-key attacks on the full AES. For AES-256 we show the first key recovery attack that works for all the keys and has complexity 2^{119}, while the recent attack by Biryukov-Khovratovich-Nikolić works for a weak key class and has higher complexity. The second attack is the first cryptanalysis of the full AES-192. Both our attacks are boomerang attacks, which are based on the recent idea of finding *local collisions in block ciphers* and enhanced with the *boomerang switching* techniques to gain free rounds in the middle.

1 Introduction

The Advanced Encryption Standard (AES) [7] — a 128-bit block cipher, is one of the most popular ciphers in the world and is widely used for both commercial and government purposes. It has three variants which offer different security levels based on the length of the secret key: 128, 192, 256-bits. Since it became a standard in 2001 [1], the progress in its cryptanalysis has been very slow. The best results until 2009 were attacks on 7-round AES-128 [8,9], 10-round AES-192 [4,11], 10-round AES-256 [4,11] out of 10, 12 and 14 rounds respectively. The two last results are in the related-key scenario.

Only recently there was announced a first attack on the full AES-256 [5]. The authors showed a related-key attack which works with complexity 2^{96} for one out of every 2^{35} keys. They have also shown practical attacks on AES-256 in the chosen key scenario, which demonstrates that AES-256 can not serve as a replacement for an ideal cipher in theoretically sound constructions such as Davies-Meyer mode.

In this paper we further improve these results and present the first related-key attack on AES-256 that works for all the keys and has a better complexity (2^{119} data and time). We also develop the first related key attack on the full AES-192. In both attacks we minimize the number of active S-boxes in the key-schedule (which caused the previous attack on AES-256 to work only for a fraction of all keys) by using a boomerang attack [12] enhanced with *boomerang switching* techniques. We find our boomerang differentials by searching for *local collisions* in a cipher. Complexities of our attacks and comparison with the best previous attacks are given in Table 1.

This paper is structured as follows: In Section 2 we develop the idea of local collisions in the cipher and show how to construct optimal related-key differentials for AES-192 and AES-256 . In Section 3 we briefly explain the idea of a boomerang and an amplified boomerang attack. In Sections 5 and 6 we describe an attack on AES-256 and AES-192, respectively.

Attack	Rounds	# keys	Data	Time	Memory	Source
192						
Partial sums	8	1	$2^{127.9}$	2^{188}	?	[8]
Related-key rectangle	10	64	2^{124}	2^{183}	?	[4,11]
Related-key amplified boomerang	12	4	2^{123}	2^{176}	2^{152}	Sec. 6
256						
Partial sums	9	256	2^{85}	2^{226}	2^{32}	[8]
Related-key rectangle	10	64	2^{114}	2^{173}	?	[4,11]
Related-key differential	14	2^{35}	2^{96} *	2^{96} *	2^{65}	[5]
Related-key boomerang	14	4	2^{119}	2^{119}	2^{77}	Sec. 5

* — for each key.

Table 1. Best attacks on AES-192 and AES-256

2 Local collisions in AES

The notion of a local collision comes from the cryptanalysis of hash functions with one of the first applications by Chabaud and Joux [6]. The idea is to inject a difference into the internal state, causing a *disturbance*, and then to *correct* it with the next injections. The resulting difference pattern is spread out due to the message schedule causing more disturbances in other rounds. The goal is to have as few disturbances as possible in order to reduce the complexity of the attack.

In the related-key scenario we are allowed to inject difference into the key, and not only into the plaintext as in the pure differential cryptanalysis. However the attacker can not control the key itself and thus the attack should work for any key pair with a given difference.

Local collisions in AES-256 are best understood on a one-round example (Fig. 1). Here we need one active S-box and five non-zero byte differences in the two subkeys. These five bytes split into two parts: one-byte disturbance and four-byte correction.

Due to the key schedule the differences spread to other rounds. The AES key schedule is mostly linear, so a sequence of several consecutive subkeys can be viewed as a codeword of a linear code. This is the case, particularly, when a trail does not have active S-boxes in the key schedule, which we try to achieve.

Let us figure out how to build an optimal trail for the key recovery attack. Typically, a trail is better if it has fewer active S-boxes. Disturbance differences thus form a codeword, which should have low weight. Simultaneously, correction differences also must form a codeword, and the key schedule codeword is the sum of the disturbance and the correction codewords. In further trails, the correction codeword is constructed from the former one by just shifting four columns to the right and applying the S-box-MixColumns expansion. Synchronization is simple since the injection is made to the first row, which is not rotated by ShiftRows. Otherwise, the task of synchronizing two codewords would have been much harder and would have lead to high-weight codewords.

An example of a good key-schedule pattern for AES-256 (see Section 4 for its formal description) is depicted in Figure 3 as a 4.5-round codeword. In the first four key-schedule rounds the disturbance codeword has only 9 active bytes, which is the lower bound. We want to avoid active S-boxes in the key schedule as long as possible, so we start with a difference in byte $b_{0,0}$ and go backwards. Due to a slow diffusion in the AES key schedule the difference affects only one more byte per key schedule round. The correction column should be positioned four columns to the right, and propagates backwards in the same way. The last column in the first subkey is active, so all S-boxes of the first round are active as well, which causes unknown difference in the first column. This "alien" difference should be canceled by the plaintext.

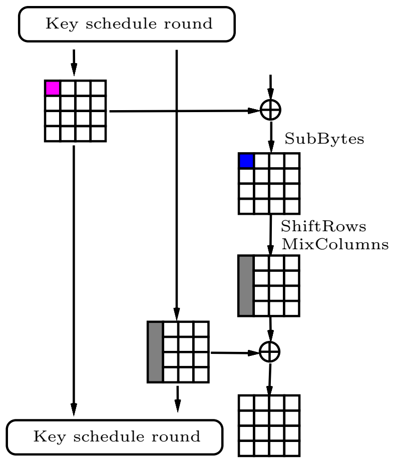


Fig. 1. A local collision.

3 Boomerang and amplified boomerang attacks

In this section we describe two types of boomerang attacks and their use in the related-key scenario.

A basic boomerang distinguisher [12] is applied to a cipher $E_K(\cdot)$ which is considered as a composition of two sub-ciphers: $E_K(\cdot) = E_1 \circ E_0$. The first sub-cipher is supposed to have a differential $\alpha \rightarrow \beta$, and the second one to have a differential $\gamma \rightarrow \delta$, with probabilities p and q, respectively. In the further text the differential trails of sub-ciphers are called *sub-trails*.

In the further text we denote plaintexts by X_i, their encryption on E_0 by Y_i, and the ciphertexts by Z_i. The attack works as follows. An attacker takes a pair of plaintexts (X_0, X_1) with the difference α and encrypts them. Then he adds the difference δ to both ciphertexts Z_0 and Z_1 and decrypts them, which results in new plaintexts X_2 and X_3. If $X_2 \oplus X_3 = \alpha$, the four plaintexts form a *quartet*. Differences δ in the two pairs (Z_0, Z_2) and (Z_1, Z_3) are converted by E_1^{-1} to a difference γ in pairs (Y_0, Y_2) and (Y_1, Y_3) with probability q^2. If $Y_0 \oplus Y_1 = \beta$, which has probability p, then the intermediate texts also form a quartet with differences β and γ. With probability p the pair (Y_2, Y_3) is finally decrypted to a pair with difference α. Therefore, a pair results in a quartet with probability p^2q^2. If $p^2q^2 > 2^{-n}$ then we have a boomerang distinguisher. This is a chosen plaintext – adaptive chosen ciphertext attack.

The amplified boomerang attack [10] (also called rectangle attack [2]) works in a chosen-plaintext scenario but relies on a birthday paradox to provide intermediate difference switching, which results in very high data complexity and complicated final filtering phase. In the amplified boomerang attack we compose a number of plaintext pairs (X_{2i}, X_{2i+1}) with the difference α, encrypt and store them. Out of N pairs, about pN pairs (Y_{2i}, Y_{2i+1}) come out of E_0 with the difference β. Due to the birthday paradox, there are $p^2N^2 \cdot 2^{-n}$ pairs (Y_{2i}, Y_{2j}) with the difference γ among those so they form quartets $(Y_{2i}, Y_{2i+1}, Y_{2j}, Y_{2j+1})$ with edges β and γ. With probability q^2 both pairs in a quartet follow the differential $\gamma \rightarrow \delta$ in E_1 and thus form a good quartet of ciphertexts (Z_0, Z_1, Z_2, Z_3) where $Z_0 \oplus Z_2 = Z_1 \oplus Z_3 = \delta$. We thus expect about $p^2q^2N^22^{-n}$ good quartets.

The number of good ciphertext quartets is actually higher, since an attacker may consider other β and γ (with the same α and δ). As a result, the number Q of good quartets is expressed via *amplified probabilities* $\hat{p}$ and $\hat{q}$ as follows:

$$Q = \hat{p}^2\hat{q}^22^{-n}N^2,$$

where

$$\hat{p} = \sqrt{\sum_{\beta} P[\alpha \rightarrow \beta]^2}; \quad \hat{q} = \sqrt{\sum_{\gamma} P[\gamma \rightarrow \delta]^2}. \tag{1}$$

For a random permutation the expected number of good quartets is N^22^{-2n}, since every quartet should satisfy two n-bit conditions. Therefore, if $\hat{p}\hat{q} > 2^{-n/2}$ then we get a distinguisher.

Key recovery. The key-recovery process resembles that of a simple differential cryptanalysis. The (amplified) boomerang distinguisher is usually extended to a few more rounds in a truncated form. Then either a key is partially guessed so that the distinguisher could be applied, or right and wrong quartets are simply detected among the ciphertexts and then each quartet proposes some key candidates, which are then counted and ranked by their frequency of occurence.

3.1 Boomerang switch

Here we analyze the transition from the sub-trail E_0 to the sub-trail E_1, which we call the *boomerang switch*. Evidently, the shorter the sub-trail, the higher is the probability. Therefore, position of the switch is a tradeoff between the two probabilities, that should minimize the overall complexity of the distinguisher. Below we summarize the switching techniques that we use in the attacks.

Amplified probabilities. In the original boomerang attack paper by Wagner [12] it was noted that instead of considering two differentials $[\alpha \rightarrow \beta]$ and $[\gamma \rightarrow \delta]$ one may build quartets from a set of differentials $[\alpha \rightarrow \beta']$ and $[\gamma' \rightarrow \delta]$. The number of right quartets is then estimated via amplified probabilities $\hat{p}$ and $\hat{q}$ (1).

Related-key model. The boomerang attack can be carried on to the related-key model, where the difference appears not only in the plaintext, but also in the key. We thus consider four related keys K_A, K_B, K_C, K_D. The relation of the key quartet may be simple, such as a fixed difference, or more complicated (e.g., a fixed difference in a subkey, or even an arbitrary function of the key). If there is an active non-linear transformation in the key schedule (e.g. an active S-box) then either its input or output becomes undetermined, otherwise the attack is restricted to a weak key class.

Ladder switch. By default, a cipher is decomposed into rounds. However, such decomposition may not be the best for the boomerang attack. We propose not only to further decompose the round into simple operations but also to exploit the existing parallelism in these operations. For example some bytes may be independently processed. In such case we can switch in one byte before it is transformed and in another one after it is transformed, see Fig. 2 for an illustration.

An example is our attack on AES-192. Let us look at the differential trails (see Fig. 8). There is one active S-box in round 7 of the lower trail in byte $b_{0,2}$. On the other hand, the S-box in the same position is not active in the upper trail. If we would switch after ShiftRows in round 6, we would "pay" the probability in round 7 afterwards. However, we switch all the state except $b_{0,2}$ after MixColumns, and switch the remaining byte after the S-box application in round 7, where it is not active. We thus do not pay for this S-box.

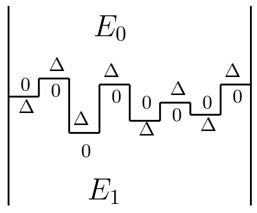


Fig. 2. Ladder switch.

Feistel switch. Surprisingly, a Feistel round with an arbitrary function (e.g., an S-box) can be passed for free in the boomerang attack (this was first observed in the attack on cipher Khufu in [12]). Suppose two internal variables X and Y are transformed to $Z = X \oplus f(Y)$ and Y. Suppose also that in the boomerang attack there is a differential before this transformation, which ends with a difference Δ_X in X and Δ_Y in Y, and a differential after this transformation, which starts with Δ_Z in Z and the same Δ_Y in Y. Let these differentials be used in the boomerang attack as follows.

Let cipher E_0 cover the first differential and the Feistel transformation, while E_1 covers the second differential. Let two plaintexts P_A and P_B be encrypted under E_0 and their internal variables X_A, X_B, Y_A, Y_B satisfy the equations:

$$X_A \oplus X_B = \Delta_X; \quad Y_A \oplus Y_B = \Delta_Y.$$

The ciphertexts C_A and C_B are modified to C_C and C_D, respectively, and then decrypted. Then after E_1^{-1} we get a quartet of Y: (Y_A, Y_B, Y_C, Y_D) and a quartet of Z: (Z_A, Z_B, Z_C, Z_D). If the differential holds the following equations hold:

$$Z_A \oplus Z_C = Z_B \oplus Z_D = \Delta_Z; \quad Y_A \oplus Y_B = Y_B \oplus Y_D = Y_C \oplus Y_A = \Delta_Y. \qquad (2)$$

Then we obtain that $Y_B = Y_C$ and $Y_A = Y_D$. Let us denote $f(Y_A)$ by f_1 and $f(Y_B)$ by f_2. Then we get the following system of equations for Z and X:

$$\begin{aligned} Z_A &= X_A \oplus f_1; \\ Z_B &= X_B \oplus f_2; \\ Z_C &= X_C \oplus f_2; \\ Z_D &= X_D \oplus f_1; \end{aligned}$$

Let us substitute these equations to (2), then we obtain:

$$X_A \oplus f_1 \oplus X_C \oplus f_2 = X_B \oplus f_1 \oplus X_D \oplus f_2 \;\Leftrightarrow\; X_A \oplus X_B = X_C \oplus X_D.$$

Therefore, we get difference Δ_X in X again for free. This trick is used in the switch in the subkey in the attack on AES-192.

S-box switch. This is similar to the Feistel switch, but costs probability only in one of the directions. If the output of an S-box in a cipher has difference Δ and if the same difference Δ comes from the lower trail, then propagation through this S-box is for free on one of the faces of the boomerang. The other direction can use amplified probability since specific value of the difference Δ is not important for the switch.

4 AES description

We expect that most of our readers are familiar with the description of AES and thus point out only the main features of AES-256 that are crucial for our attack.

We denote the i-th 192-bit subkey (do not confuse with the 128-bit round key) by K^i, i.e. the first (whitening) subkey is the first four columns of K^0. The last subkey is K^7 in AES-256 and K^8 in AES-192. The difference in K^i is denoted by ΔK^i. Bytes of a subkey are denoted by $k^l_{i,j}$, where i, j stand for the row and column index in the standard matrix representation of AES and l stands for the number of the subkey. Bytes of the plaintext are denoted by $p_{i,j}$, and bytes of the internal state after the SubBytes transformation in round r are denoted by $a^r_{i,j}$. Let us also denote by $b^r_{i,j}$ byte in position (i, j) after the r-th application of MixColumns. In boomerang trails, the difference between subkeys in the upper trail is denoted by ΔK^i, and in the lower part by ∇K^i.

Features of AES-256. AES-256 has 14 rounds and a 256-bit key, which is two times larger than the internal state. Thus the key schedule consists of only 7 rounds. One key schedule round consists of the following transformations:

$$\begin{aligned}
k_{i,0} &\leftarrow S(k_{i+1,7}) \oplus K_{i,0} \oplus C_r, && 0 \le i \le 3;\\
k_{i,j} &\leftarrow k_{i,j-1} \oplus K_{i,j}, && 0 \le i \le 3,\ 1 \le j \le 3;\\
k_{i,4} &\leftarrow S(k_{i,3}) \oplus K_{i,4}, && 0 \le i \le 3;\\
k_{i,j} &\leftarrow k_{i,j-1} \oplus K_{i,j}, && 0 \le i \le 3,\ 5 \le j \le 7,
\end{aligned}$$

where $S()$ stands for the S-box, and C_r — for the round-dependant constant. Therefore, each round has 8 S-boxes.

Features of AES-192. AES-192 has 12 rounds and a 192-bit key, which is 1.5 times larger than the internal state. Thus the key schedule consists of 8 rounds. One key schedule round consists of the following transformations:

$$\begin{aligned}
K_{i,0} &\leftarrow S(K_{i+1,5}) \oplus K_{i,0} \oplus C_r, && 0 \le i \le 3;\\
K_{i,j} &\leftarrow K_{i,j-1} \oplus K_{i,j}, && 0 \le i \le 3,\ 1 \le j \le 5.
\end{aligned}$$

Notice that each round has only four S-boxes.

5 Attack on AES-256

In this section we present a related key boomerang attack on AES-256.

5.1 The trail

The boomerang trail is depicted in Figure 7, and the actual values are listed in Tables 3 and 2. It consists of two identical 7-round trails: the first one covers rounds 1–7, and the second one covers rounds 8–14. The switching state is the state A^7 (internal state after the SubBytes in round 7) and a special key state K_S, which is the concatenation of the last four columns of K^3 and the first four columns of K^4. Although there are active S-boxes in the first round of the key schedule, we do not impose conditions on them. As a result, the difference in column 0 of K^0 is unknown yet.

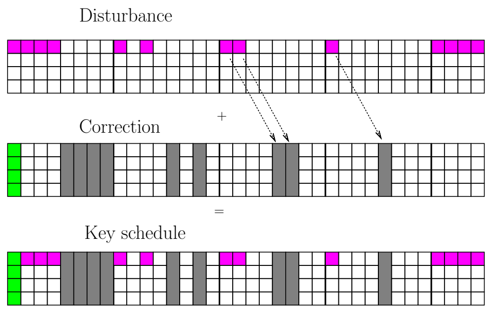


Fig. 3. AES-256 key schedule codeword (4.5 key-schedule rounds).

Related keys We define the relation between four keys as follows (see Figure 4 for the illustration). For a secret key K_A, which the attacker tries to find, compute its second subkey K_A^1 and apply the difference ΔK^1 to get a subkey K_B^1, from which the key K_B is computed. The switch into the keys K_C, K_D happens between the 3rd and the 4th subkeys in order to avoid active S-boxes in the key-schedule using the *Ladder switch* idea described above. We compute subkeys K^3 and K^4 for both K_A and K_B. We add the difference ∇K^3 to K_A^3 and compute the upper half (four columns) of K_C^3. Then we add the difference ∇K^4 to K_A^4 and compute the lower half (four columns) of K_C^4. From these eight consecutive columns we compute the full K_C. The key K_D is computed from K_B in the same way.

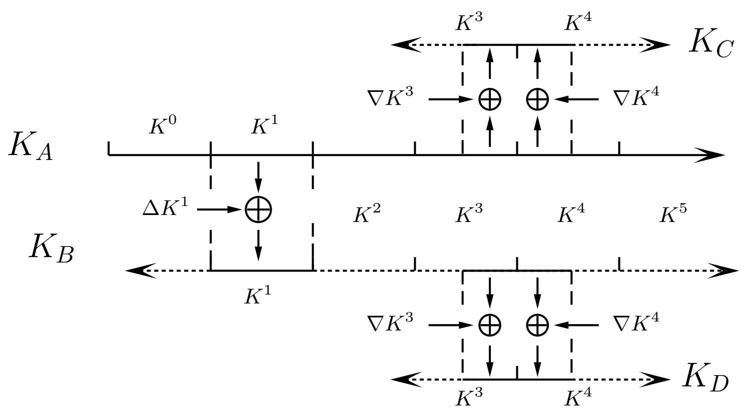


Fig. 4. AES-256: Computing K_B, K_C, and K_D from K_A.

Finally, we point out that difference between K_C and K_D can be computed in the backward direction deterministically since there would be no active S-boxes till the first round. The secret key K_A, and the three keys K_B, K_C, K_D computed from K_A as described above form a proper related key quartet. Moreover, due to a slow diffusion in the backward direction, as a bonus we can compute some values in ∇K^i even for $i = 0, 1, 2, 3$ (Table 2). Hence given the byte value $k^l_{i,j}$ for K_A we can partly compute K_B, K_C and K_D.

Internal state The plaintext difference is specified in 9 bytes. We require that all the active S-boxes in the internal state should output the difference `0x1f` so that the active S-boxes are passed with probability 2^{-6}. The only exception is the first round where the input difference in seven active bytes is not specified.

Let us start a boomerang attack with a random pair of plaintexts that fit the trail after one round. Active S-boxes in rounds 3 and 5 are passed with probability 2^{-6} each so the overall probability is 2^{-24}.

For the last S-box (round 7) we pay the probability only when going in the forward direction due to an *S-box switch*. Indeed, assume that it gets a and a+`0x01` as input and outputs b and b+`0x1f`. When the boomerang comes back, each of the two pairs has the same difference `0x1f` in $A^7_{0,0}$. Therefore, the S-box get as output values b and b+`0x1f` and produces `0x01` as the input difference with probability one.

The second part of the boomerang trail is quite simple. The top four active S-boxes of the round 7 in E_1 trail switch for free due to the *Ladder switch* (zero difference coming from E_0 in three places and the fourth one covered by an S-box switch that we have just explained. Therefore, only five S-boxes contribute to the probability, which is thus equal to 2^{-30}. Finally we get one boomerang quartet after the first round with probability $2^{-24-6-30-30-24} = 2^{-114}$.

5.2 The attack

The attack works as follows. Do the following steps 2^{61} times:

1. Prepare a structure of plaintexts as specified below.
2. Encrypt it on keys K_A and K_B and keep the resulting sets S_A and S_B in memory.
3. XOR Δ_C to all the ciphertexts in S_A and decrypt the resulting ciphertexts with K_C. Denote the new set of plaintexts by S_C.
4. Repeat previous step for the set S_B and the key K_D. Denote the set of plaintexts by S_D.
5. For each guess of the key byte $k^0_{1,7}$ of K_A:
   (a) Compute the key byte $k^0_{1,7}$ of K_B, K_C and K_D with Table 2. The key difference $\Delta k^0_{0,0}$ is now completely determined.
   (b) Compose from S_C and S_D all the possible pairs of plaintexts which are equal in 72 bits $p_{i,j}$, $i, j > 0$.

(c) For every remaining pair check if the difference in $p_{i,0}, i > 0$ is equal on both sides of the boomerang quartet (24-bit filter). Note that $\nabla k^0_{i,7} = 0$ so $\Delta k^0_{i,0}$ should be equal for both key pairs (K_A, K_B) and (K_C, K_D).

(d) Filter out the quartets whose difference can not be produced by active S-boxes in the first round (one-bit filter per S-box per key pair) and active S-boxes in the key schedule (one-bit filter per S-box), which is a $4 \cdot 2 + 3 = 11$-bit filter.

(e) The remaining quartets propose candidates for several key bytes (see details below).

Each structure has all possible values in column 0 and row 0, and constant values in the other bytes. Of 2^{56} texts per structure we can compose 2^{112} ordered pairs. Of these pairs $2^{112-8\cdot 7} = 2^{56}$ pass the first round. Thus we expect one right quartet per $2^{114-56} = 2^{58}$ structures, and eight right quartets out of 2^{61} structures.

Let us now compute the number of noisy quartets. About 2^{40} pairs come out of step 5b. The next two steps apply a 24+11 = 35-bit filter, so we get $2^{61+40-35} = 2^{66}$ candidate quartets in total. Each quartet proposes 2^4 candidates for bytes $k_{0,j}, j < 4$ of K_A and K_C each, 2^3 candidates for bytes $k_{i,7}, i = 0, 2, 3$ of K_A; i.e. 2^{11} candidates for the 88 key bits. The probability that eight false quartets propose the same candidate is $2^{(66+11)\cdot 8 - 88\cdot 7}/8! \approx 2^{-15}$, while the eight right quartets propose correct candidates for all the key bytes. Even after we repeat this loop 2^7 times (for each guess of $k^0_{1,7}$) with high probability we are left only with a single correct key candidate. We estimate the complexity of the counting step as 2^{77} time and memory. Hence the total time complexity is bounded by the amount of data and is equal to 2^{119}. Every quartet gives us 8 bits of information on each of the bytes $k_{0,j}, j < 4$ and 7 bits of information on each of the bytes of $k_{i,7}, i = 0, 2, 3$. Two guesses of $k^0_{1,7}$, that give the right difference, are also indistinguishable. We thus recover 60 bits of K_A (and 60 bits of K_C) with 2^{119} data and time and 2^{77} memory.

The remaining part of the key can be found with many approaches. One is to relax the condition on one of the active S-boxes in round 3 thus getting four more active S-boxes in round 2, which in turn leads to a full-difference state in round 1. The condition can be actually relaxed only for the first part of the boomerang (the key pair (K_A, K_B)) thus giving a better output filter. For each candidate quartet we use the key bytes, that were recovered at the previous step, to compute ΔA^1 and thus significantly reduce the number of keys that are proposed by a quartet. We then rank candidates for the first four columns of K^0_A and take the candidate that gets the maximal number of votes. Since we do not make key guesses, we expect that the complexity of this step does not exceed the complexity of the previous step (2^{119}). The remaining 100 bits of K_A can be found with the exhaustive search.

6 Attack on AES-192

The key schedule of AES-192 has better diffusion, so it is hard to avoid active S-boxes in the subkeys. We construct an amplified-boomerang attack with two sub-trails of 6 rounds each. However, the switching process and the key relations are more complicated than for AES-256, so we describe them in more details.

6.1 The trail

The trail is depicted in Figure 8, and the actual values are listed in Tables 4 and 5.

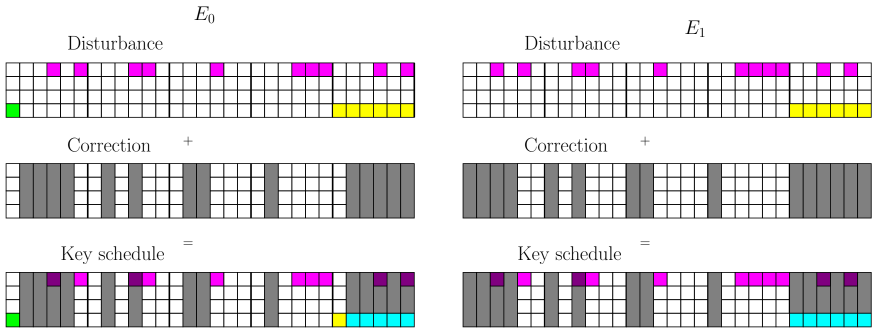
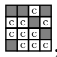


Fig. 5. AES-192 key schedule codeword.

Related keys We define the relation between four keys similarly to the attack on AES-256. Assume we are given a key K_A, which the attacker tries to find. We compute its subkey K_A^1 and apply the difference ΔK^1 to get the subkey K_B^1, from which the key K_B is computed. Then we compute the subkeys K_A^4 and K_B^4 and apply the difference ∇K^4 to them. We get subkeys K_C^4 and K_D^4, from which the keys K_C and K_D are computed.

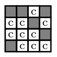
Now we prove that keys K_A, K_B, K_C, and K_D form a quartet, i.e. the subkeys of K_C and K_D satisfy the equations $K_C^r \oplus K_D^r = \Delta K^r$, $r = 1, 2, 3$. The only active S-box is positioned between K^3 and K^4, whose input is $k_{0,5}^3$. However, this S-box gets the same pair of inputs in both key pairs (see the *"Feistel switch"* in Sec. 3.1). Indeed, if we compute $\nabla k_{0,5}^3$ from ΔK^4, then it is equal to $\Delta k_{0,5}^3 = 0x01$. Therefore, if the active S-box gets as input α and $\alpha \oplus 1$ in K_A and K_B, respectively, then it gets $a \oplus 1$ and a in K_C and K_D, respectively. As a result, $K_C^3 \oplus K_D^3 = \Delta K^3$, the further propagation is linear, so the four keys form a quartet.

Due to a slow diffusion in the backward direction, we can compute some values in ∇K^i even for small i (Table 5). Hence given $k_{i,j}^r$ for K_A we can partly compute K_B, K_C and K_D, which provides additional filtration in the attack.

Internal state The plaintext difference is specified in 10 bytes , the difference in the other six bytes not restricted. The three active S-boxes in rounds 2–4 are passed with probability 2^{-6} each. In round 6 (the switching round) we ask for the fixed difference only in $a^6_{0,2}$, the other two S-boxes can output any difference such that it is the same as in the second related-key pair. Therefore, the amplified probability of round 6 equals to $2^{-6-2\cdot 3.5} = 2^{-13}$. We switch between the two trails before the key addition in round 6 in all bytes except $b^6_{0,2}$, where we switch after the S-box application in round 7 (the *Ladder switch*). This trick allows us not to take into account the only active S-box in the lower trail in round 7. The overall probability of the rounds 3–6 is $2^{-3\cdot 6-13} = 2^{-31}$.

The lower trail has 8 active S-boxes in rounds 8–12. Only the first four active S-boxes are restricted in the output difference, which gives us probability 2^{-24} for the lower trail. The ciphertext difference is fully specified in the middle two rows, and has 35 bits of entropy in the other bytes. More precisely, each $\nabla c_{0,*}$ is taken from a set of size 2^7, and all the $\nabla c_{3,*}$ should be the same on both sides of the boomerang and again should belong to a set of size 2^7. Therefore, the ciphertext difference gives us a 93-bit filter.

6.2 The attack

We compose 2^{73} structures of type  with 2^{48} texts each. Then we encrypt all the texts with the keys K_A and K_C, and their complements w.r.t. $\varDelta P$ on K_B and K_D. We keep all the data in memory and analyze it with the following procedure:

1. Compose all candidate plaintext pairs for the key pairs (K_A, K_B) and (K_C, K_D).
2. Compose and store all the candidate quartets of the ciphertexts.
3. For each guess of the subkey bytes: $k^0_{0,3}$, $k^0_{2,3}$, and $k^0_{0,5}$ in K_A; $k^7_{0,5}$ in K_A and K_B (see also Figure 6):
   (a) Derive values for these bytes in all the keys from the differential trail. Derive yet unknown key differences in $\varDelta K^0$ and ∇K^8.
   (b) Filter out candidate quartets that contradict ∇K^8.
   (c) Prepare counters for yet unknown subkey bytes that correspond to active S-boxes in the first two rounds and in the last round: $k^0_{0,0}$, $k^0_{0,1}$, $k^0_{1,2}$, $k^0_{3,0}$ — in keys K_A and K_C, $k^8_{0,0}$, $k^8_{0,1}$, $k^8_{0,2}$, $k^8_{0,3}$ — in keys K_A and K_B, i.e. 16 bytes in total.
   (d) For each candidate quartet derive possible values for these unknown bytes and increase the counters.
   (e) Pick the group of 16 subkey bytes with the maximal number of votes.
   (f) Try all possible values of the yet unknown 9 key bytes in K^0 and check whether it is the right key. If not then go to the first step.

Right quartets. Let us first count the number of right quartets in the data. Evidently, there exist 2^{128} pairs of internal states with the difference $\varDelta A^2$. The inverse application of 1.5 rounds maps these pairs into structures that we have defined, with 2^{48} pairs per structure. Therefore, each structure has 2^{48} pairs that pass 1.5 rounds, and 2^{73} structures have 2^{121} pairs. Of these pairs $2^{(121-31)\cdot 2-128} = 2^{52}$ right quartets can be composed after the switch in the middle. Of these quartets $2^{52-2\cdot 24} = 16$ right quartets come out of the last round.

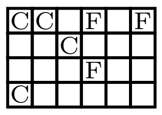


Fig. 6. Processing the subkey K^0. "F" stands for bytes fixed by a guess, "C" — for those that we count in the attack.

Steps 1–2. The first two steps are the most time-consuming part of the attack. We propose the following approach based on the ideas of [3] and the fact that pairs of plaintexts in a right quartet should belong to the same structure:

– Having encrypted the data, group all the ciphertexts into buckets according to the 88-bit ciphertext filter: fixed differences in the middle rows, equal differences in the last row.
– Prepare a two-dimensional table of plaintexts indexed by the indices of structures and a key.
– For every pair (C_A, C_C) of ciphertexts in a same bucket, that were encrypted under K_A and K_C, respectively:
  - Check if the pair satisfies the additional 5-bit filter in the differences corresponding to the active S-boxes, where there are only 2^7 possibilities per byte.
  - If yes, detect structures S_A and S_C, to which the corresponding plaintexts belong, and insert the pair in a table into a cell indexed by these structures.
– Repeat the previous step for the keys K_B and K_D.
– For every pair of structures compose all the possible quartets of plaintexts.
– Put all the quartets into a hash table indexed by the two differences $\nabla c_{3,0}$.

Every bucket contains $2^{(121-88)\cdot 2-5} = 2^{61}$ pairs. The overall number of pairs is $2^{88+61} = 2^{149}$ pairs, or 2^3 pairs for a pair of structures. Therefore, we compose $2^{3\cdot 2+73\cdot 2} = 2^{152}$ candidate quartets and then rank them according to $\nabla c_{3,0}$. We thus get 2^{14} groups of quartets each having 2^{138} candidate quartets.

Step 3(a-b) (filtering). We apply the ∇K^8 filter and analyze 2^{138} candidate quartets. We know $k^0_{0,3}$ from the guess and $\Delta a^1_{0,3}$ from the trail, which gives us an 8-bit filter on $p_{0,3}$, and a 16-bit filter on the quartets. We also know $k^0_{2,3}$ so for both pairs of plaintexts in each of remaining 2^{122} quartets we compute $\Delta a^1_{2,3}$. Since $\Delta a^1_{2,3}$ is a value in the column that should collapse to one non-zero byte

$\Delta b^1_{0,1}$ by the MixColumns, we derive all the values on its diagonal and $\Delta b^1_{0,1}$. Actually, the value of $b^1_{0,1}$ can be restricted to two options, since we know $k^0_{0,1}$, $\Delta k^0_{0,1}$, and $\Delta a^2_{0,1}$. For a given difference in the plaintext and provided with ΔA^1 there exist 8 possible combinations of $k^0_{0,1}$, $k^0_{1,2}$, and $k^0_{3,0}$, and the probability that any of them matches the two options for $a^1_{0,1}$ is 1/16. Therefore, the value of $\Delta p_{2,3}$ restricts the other three differences on its diagonal by $3+4=7$ bits. One more bit comes from the fact that only half of $\Delta a^1_{0,1}$ can be converted into $\Delta a^2_{0,1}$. Therefore, we consider only $2^{122-2\cdot(7+1)} = 2^{106}$ quartets.

Step 3(c-e) (counting). Both pairs of plaintexts in a quartet propose key candidates: the first pair for K_A and the second pair for K_B. Each pair proposes one candidate for $(k^0_{0,1}, k^0_{1,2}, k^0_{3,0})$ and the two candidates for $k^0_{0,0}$, so we have 2^{10} candidates for 16 key bytes. The probability that 16 false quartets propose the same key candidate can be upper bounded by $2^{116\cdot 16-128\cdot 15} = 2^{-64}$. The exhaustive search for the remaining 9 key bytes can be done with the complexity 2^{72}.

The overall time complexity is the number of quartets analyzed at Step 3 times the number of the key guesses. Now we point out that we do not have to guess all 8 bits of $k^7_{0,5}$ since we need only the output S-box difference. Therefore, we try $2^{3\cdot 8+2\cdot 7} = 2^{38}$ key guesses, so the time complexity of the attack is 2^{138+38} $= 2^{176}$, and the data complexity is 2^{123}.

7 Conclusions

We presented related-key boomerang attacks on the full AES-192 and the full AES-256. The differential trails for the attacks are based on the idea of finding local collisions in the block cipher. We showed that optimal key-schedule trails should be based on low-weight codewords in the key schedule. We also exploit various boomerang-switching techniques, which help us to gain free rounds in the middle of the cipher. However, both our attacks are still mainly of theoretical interest and do not present a threat to practical applications using AES.

References

1. *FIPS-197: Advanced Encryption Standard*, November 2001, available at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.
2. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the Serpent. In *EUROCRYPT'01*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001.
3. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *FSE'02*, volume 2365 of *LNCS*, pages 1–16. Springer, 2002.
4. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 507–525. Springer, 2005.
5. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *CRYPTO'09*, LNCS. Springer, 2009. to appear.

6. Florent Chabaud and Antoine Joux. Differential collisions in SHA-0. In *CRYPTO'98*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
7. Joan Daemen and Vincent Rijmen. *The Design of Rijndael. AES — the Advanced Encryption Standard*. Springer, 2002.
8. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In *FSE'00*, volume 1978 of *LNCS*, pages 213–230. Springer, 2000.
9. Henri Gilbert and Marine Minier. A collision attack on 7 rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.
10. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In *FSE'00*, volume 1978 of *LNCS*, pages 75–93. Springer, 2000.
11. Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In *FSE'07*, volume 4593 of *LNCS*, pages 225–241. Springer, 2007.
12. David Wagner. The boomerang attack. In *FSE'99*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

ΔK^i					
0	? 01 01 01 3*e* 3*e* 3*e* 3*e* ? 00 00 00 1*f* 1*f* 1*f* 1*f* ? 00 00 00 1*f* 1*f* 1*f* 1*f* ? 00 00 00 21 21 21 21	1	01 00 01 00 3*e* 00 3*e* 00 00 00 00 00 1*f* 00 1*f* 00 00 00 00 00 1*f* 00 1*f* 00 00 00 00 00 21 00 21 00	2	01 01 00 00 3*e* 3*e* 00 00 00 00 00 00 1*f* 1*f* 00 00 00 00 00 00 1*f* 1*f* 00 00 00 00 00 00 21 21 00 00
3	01 00 00 00 3*e* 00 00 00 00 00 00 00 1*f* 00 00 00 00 00 00 00 1*f* 00 00 00 00 00 00 00 21 00 00 00	4	01 01 01 01 ? ? ? ? 00 00 00 00 1*f* 1*f* 1*f* 1*f* 00 00 00 00 1*f* 1*f* 1*f* 1*f* 00 00 00 00 21 21 21 21		

∇K^i					
0	? ? ? ? ? ? ? 00 ? ? ? ? 1*f* 1*f* 1*f* 00 ? ? ? ? 1*f* 1*f* 1*f* 00 ? ? ? ? 21 21 21 00	1	? 01 ? 00 ? ? 00 00 ? 00 ? 00 1*f* 1*f* 00 00 ? 00 ? 00 1*f* 1*f* 00 00 ? 00 ? 00 21 21 00 00	2	? ? 00 00 ? 00 00 00 ? ? 00 00 1*f* 00 00 00 ? ? 00 00 1*f* 00 00 00 ? ? 00 00 21 00 00 00
3	? 01 01 01 3*e* 3*e* 3*e* 3*e* ? 00 00 00 1*f* 1*f* 1*f* 1*f* ? 00 00 00 1*f* 1*f* 1*f* 1*f* ? 00 00 00 21 21 21 21	4	01 00 01 00 3*e* 00 3*e* 00 00 00 00 00 1*f* 00 1*f* 00 00 00 00 00 1*f* 00 1*f* 00 00 00 00 00 21 00 21 00	5	01 01 00 00 3*e* 3*e* 00 00 00 00 00 00 1*f* 1*f* 00 00 00 00 00 00 1*f* 1*f* 00 00 00 00 00 00 21 21 00 00
6	01 00 00 00 3*e* 00 00 00 00 00 00 00 1*f* 00 00 00 00 00 00 00 1*f* 00 00 00 00 00 00 00 21 00 00 00	7	01 01 01 01 ? ? ? ? 00 00 00 00 1*f* 1*f* 1*f* 1*f* 00 00 00 00 1*f* 1*f* 1*f* 1*f* 00 00 00 00 21 21 21 21		

Table 2. Subkey difference in the AES-256 trail.

Disclaimer on colors. We intensively use colors in our figures in order to provide better understanding on the trail construction. In figures, different colors refer to different values, which is hard to depict in black and white. However, we also list all the trail differences in the tables, so all the color information is actually dubbed. Therefore, a reader may choose the view which is the best for him.

ΔP	? ? ? ? ? 00 00 00 ? 00 00 00 ? 00 00 00	ΔA^1 ∇A^7	$1f$ $1f$ $1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^2 ∇A^8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^3 ∇A^9	$1f$ 00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00
ΔA^4 ∇A^{10}	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^1 ∇A^7	$1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^1 ∇A^7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^1 ∇A^7	$1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ΔA^1 ∇A^7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				

Table 3. Internal state difference in the AES-256 trail.

ΔP	? ? $3e$? $1f$ $1f$? $1f$ $1f$ $1f$ $1f$? ? 21 21 21	ΔA^1	$1f$? 00 $1f$ 00 00 ? 00 00 00 00 ? ? 00 00 00	ΔA^2	00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^3	00 $1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00
ΔA^4	00 00 00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^6	00 $1f$ $1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^7	00 00 00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00
∇A^6	$1f$ $1f$ $1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00	∇A^7	00 00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^8	$1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^9	$1f$ $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00 00
∇A^{10}	00 00 $1f$ 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^{11}	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^{12}	? ? ? ? 00 00 00 00 00 00 00 00 00 00 00 00	ΔC	? ? ? ? $1f$ $1f$ $1f$ $1f$ $1f$ $1f$ $1f$ $1f$? ? ? ?

Table 4. Internal state difference in the AES-192 trail.

ΔK^0 $\begin{matrix} 00 & 3e & 3e & 3f & 3e & 01 \\ 00 & 1f & 1f & 1f & 1f & 00 \\ 00 & 1f & 1f & 1f & 1f & 00 \\ ? & 21 & 21 & 21 & 21 & 00 \end{matrix}$	ΔK^1 $\begin{matrix} 00 & 3e & 00 & 3f & 01 & 00 \\ 00 & 1f & 00 & 1f & 00 & 00 \\ 00 & 1f & 00 & 1f & 00 & 00 \\ 00 & 21 & 00 & 21 & 00 & 00 \end{matrix}$	ΔK^2 $\begin{matrix} 00 & 3e & 3e & 01 & 00 & 00 \\ 00 & 1f & 1f & 00 & 00 & 00 \\ 00 & 1f & 1f & 00 & 00 & 00 \\ 00 & 21 & 21 & 00 & 00 & 00 \end{matrix}$
ΔK^3 $\begin{matrix} 00 & 3e & 00 & 01 & 01 & 01 \\ 00 & 1f & 00 & 00 & 00 & 00 \\ 00 & 1f & 00 & 00 & 00 & 00 \\ 00 & 21 & 00 & 00 & 00 & 00 \end{matrix}$	ΔK^4 $\begin{matrix} 00 & 3e & 3e & 3f & 3e & 3f \\ 00 & 1f & 1f & 1f & 1f & 1f \\ 00 & 1f & 1f & 1f & 1f & 1f \\ ? & ? & ? & ? & ? & ? \end{matrix}$	
∇K^0 $\begin{matrix} ? & ? & ? & 3e & 3f & 3e \\ ? & ? & ? & 1f & 1f & 1f \\ ? & ? & ? & 1f & 1f & 1f \\ ? & ? & ? & ? & 21 & 21 \end{matrix}$	∇K^1 $\begin{matrix} ? & ? & 3f & 01 & 3e & 00 \\ ? & ? & 1f & 00 & 1f & 00 \\ ? & ? & 1f & 00 & 1f & 00 \\ ? & ? & ? & 00 & 21 & 00 \end{matrix}$	∇K^2 $\begin{matrix} ? & 3e & 01 & 00 & 3e & 3e \\ ? & 1f & 00 & 00 & 1f & 1f \\ ? & 1f & 00 & 00 & 1f & 1f \\ ? & ? & 00 & 00 & 21 & 21 \end{matrix}$
∇K^3 $\begin{matrix} 3e & 00 & 01 & 01 & 3f & 01 \\ 1f & 00 & 00 & 00 & 1f & 00 \\ 1f & 00 & 00 & 00 & 1f & 00 \\ ? & 00 & 00 & 00 & 21 & 00 \end{matrix}$	∇K^4 $\begin{matrix} 3e & 3e & 3f & 3e & 01 & 00 \\ 1f & 1f & 1f & 1f & 00 & 00 \\ 1f & 1f & 1f & 1f & 00 & 00 \\ 21 & 21 & 21 & 21 & 00 & 00 \end{matrix}$	∇K^5 $\begin{matrix} 3e & 00 & 3f & 01 & 00 & 00 \\ 1f & 00 & 1f & 00 & 00 & 00 \\ 1f & 00 & 1f & 00 & 00 & 00 \\ 21 & 00 & 21 & 00 & 00 & 00 \end{matrix}$
∇K^6 $\begin{matrix} 3e & 3e & 01 & 00 & 00 & 00 \\ 1f & 1f & 00 & 00 & 00 & 00 \\ 1f & 1f & 00 & 00 & 00 & 00 \\ 21 & 21 & 00 & 00 & 00 & 00 \end{matrix}$	∇K^7 $\begin{matrix} 3e & 00 & 01 & 01 & 01 & 01 \\ 1f & 00 & 00 & 00 & 00 & 00 \\ 1f & 00 & 00 & 00 & 00 & 00 \\ 21 & 00 & 00 & 00 & 00 & 00 \end{matrix}$	∇K^8 $\begin{matrix} 3e & 3e & 3f & 3e & 3f & 3e \\ 1f & 1f & 1f & 1f & 1f & 1f \\ 1f & 1f & 1f & 1f & 1f & 1f \\ ? & ? & ? & ? & ? & ? \end{matrix}$

Table 5. Subkey difference in the AES-192 trail.

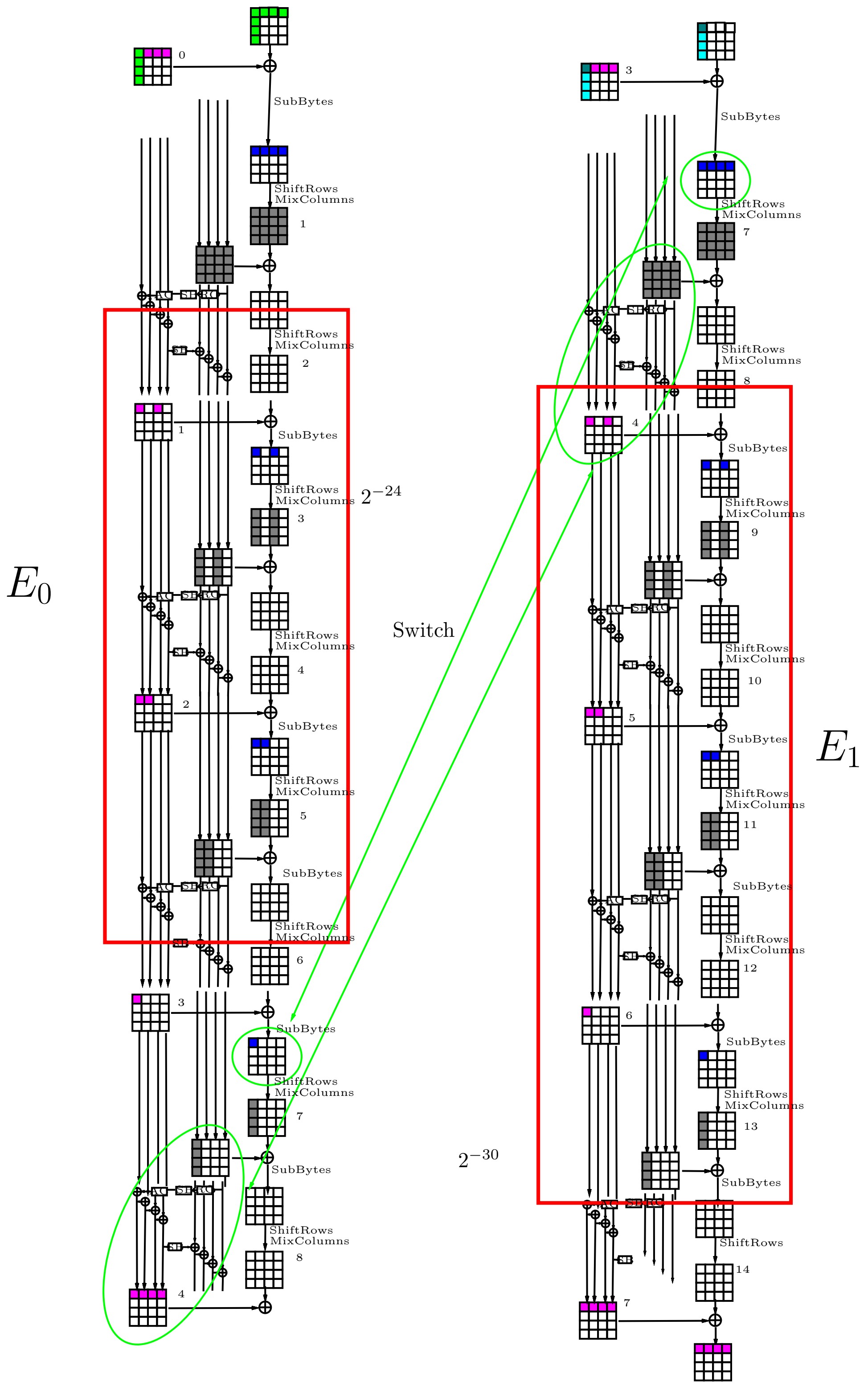


Fig. 7. AES-256 E_0 and E_1 trails. Green ovals show an overlap between the two trails where the switch happens.

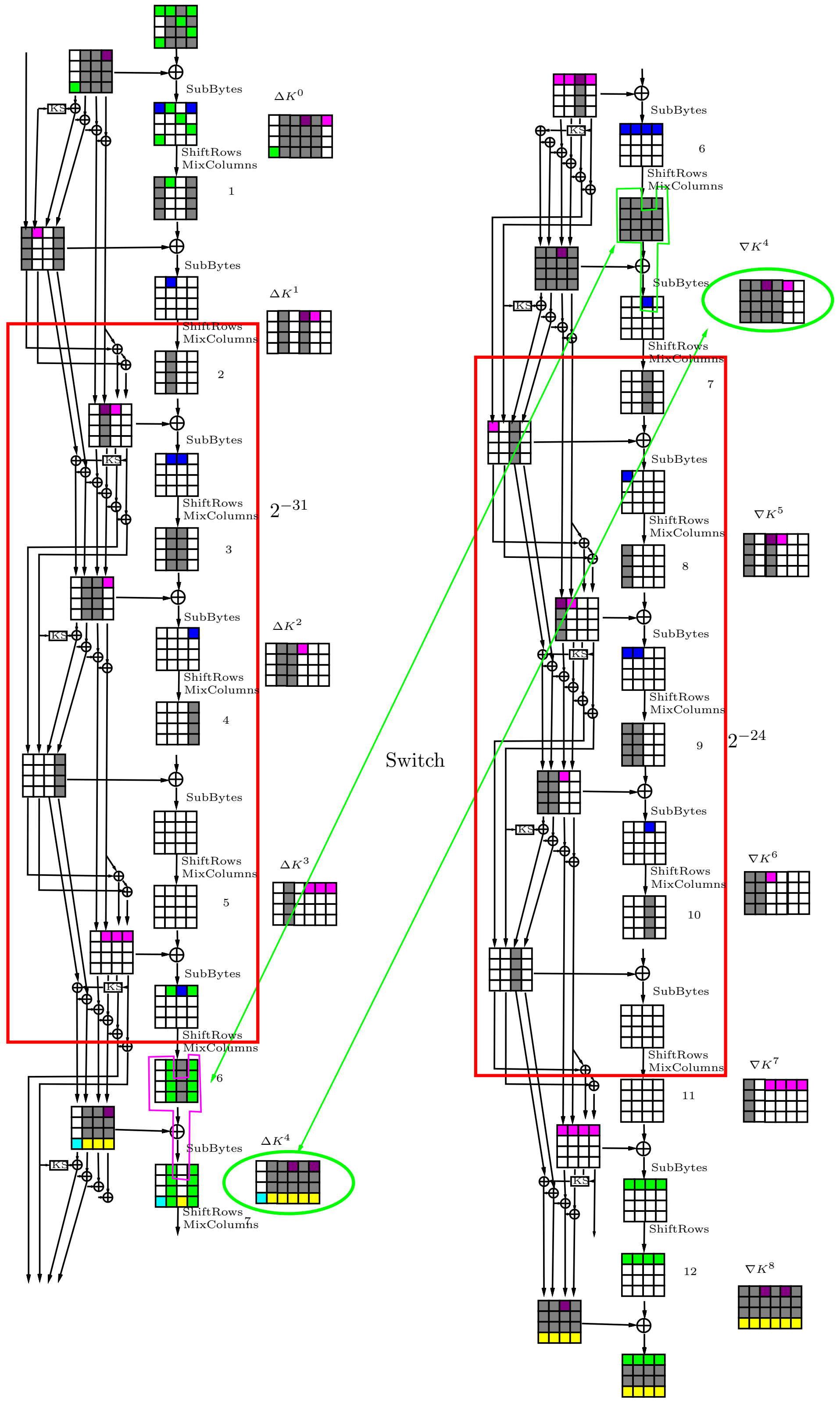


Fig. 8. AES-192 trail.