Improved Related-Key Attacks on AES-192

Florent Mazelet, María Naya-Plasencia

Inria, Paris, France
{florent.mazelet,maria.naya_plasencia}@inria.fr

Abstract. The Advanced Encryption Standard (AES) is the most well-known and studied block cipher. Out of its three standardized versions, related-key full round attacks were proposed in Asiacrypt 2009 by Biryukov and Khovratovich on the 192 and 256-bit key variants. These attacks were highly impractical. No meaningful improvement over them was known until Asiacrypt 2022, where a better attack on the 192-bit version was published by Derbez *et al.*: the data was increased by a factor of 2, but the time complexity was reduced by a factor of 2^{52}. Also in 2022, Guo, Song and Wang proposed a new attack against the full version of AES-256 with reduced data and time complexity at the cost of considering a bigger set of related keys. The authors were unable to apply their technique to the AES-192 version.
As AES is a widely used basic brick, often considered in hash functions, related-key attacks might have some applications and need to be considered seriously. In 2012, low data attacks on reduced variants of AES were proposed by Bouillaguet *et al.*, arguing the importance of reducing the data needs of attacks.
With this in mind, we studied the previous best attack on AES-192, and by applying a different key-recovery step to the previous distinguisher, we managed to considerately reduce the data complexity by a factor of $2^{8.5}$. This is the first attack improving the data needs from 2009, and can (arguably) be considered the best known attack on full-round AES-192. In addition, we use the key-structure technique from Guo *et al.* to propose for the first time other trade offs with a bigger set of related keys on AES-192 attacks.

Keywords: AES · (Amplified) Boomerang Attack · Key-Recovery· Related-Key · Full-Round Attack · Best Attacks

1 Introduction

The Advanced Encryption Standard [DR02a] is the most used, popular and studied block cipher. The standard was chosen in 2000 after a public competition organized by the NIST and includes three variants, all having block size of 128 bits, and with a key of size 128, 192 and 256 bits for 10, 12 and 14 rounds respectively. Besides being included in many protocols and applications, it has often been used as building block for other constructions, as for instance in hash functions (*e.g.* Grostl [GKM+09], LANE [IAC+09] Echo [BBG+08],...).

Since 2000, a huge number of cryptanalysis and security analysis papers have been published, trying to better understand the security of these AES constructions, and to reduce their security margin. In 2009, a very surprising result appeared: in [BK09] the full variants of the 192 and the 256 versions with 12 and 14 rounds, respectively, were fully cryptanalyzed in the related-key setting. This setting, where the attacker can ask for the encryption of messages with a secret master key, and a related key with a fixed difference, is obviously much less strong than the normal single-key setting. Nevertheless, this setting has a certain importance when applied to the AES, as in many cases, when used in a hash function construction, the message takes the role of the key and the related-key setting becomes much more easy to implement. In their paper, they proposed a boomerang attack on the full 12-round AES-192 with complexities of 2^{176} in time, 2^{123} in data and 2^{152} in memory; and a boomerang attack on the full 14-round AES-256 with complexities of $2^{99.5}$ in time, $2^{99.5}$ in data and 2^{77} in memory.

These attacks showed that the larger versions security margin were smaller than that of the 128-bit version. They remained the best known attacks until 2022 [1], when [DEFN22] proposed a new boomerang attack on AES-192, that reduced the time to 2^{124} and the memory to $2^{79.8}$, while the data was slightly increased of a factor 2. In [YSZ+24], the previous distinguisher was considered with their automatic key-recovery tool, finding an amplified boomerang attack with worse time and memory complexity but slightly better data complexity. Also in 2022, another trade-off for a full-round AES-256 attack was proposed in [GSW22], where more related keys are used. All these results can be seen in Table 1.

In this paper we have studied in detail previous attacks, with the main aim of improving the key-recovery part in order to provide better overall attacks, and in particular, attacks reducing the data needs, that has been often seen as the most problematic part of the attacks (see for instance [BDD+12,BDK+10] on low-data attacks on AES).

We have managed to considerately improve for the first time the data complexity from [BK09], while also improving the original time and memory. We started by considering the core distinguisher from [DEFN22] where we have relaxed some conditions, and in addition we consider an amplified boomerang setting. In Table 1 we summarize the main proposed known attacks on 12 round AES-192. In the corresponding section we also point out some small errors in the original paper, that do not invalidate their attack.

[1] In [BKR11] the authors claim to present full round biclique attacks on all the three variants, but these attacks are just an improved exhaustive search, where around 2 bits are gained and a loop over the whole key is still needed. In [BN10] an improved attack is also mentioned, but with the considered distinguisher, we do not see how they could reach the claimed data of 2^{116} (the naif one would be 2^{124}), and no details are given in the paper.

We have also considered the setting from [GSW22] and managed to apply it for the first time to the 192 bit version, proposing new trade offs on the number of related-keys and data and time complexities, that can also be seen in Table1.

Key Size	#Keys	Time	Data	Memory	Type	Reference
192	4	2^{176}	2^{123}	2^{152}	Boomerang	[BK09]
	4	2^{124}	2^{124}	$2^{79.8}$	Boomerang	[DEFN22]
	4	$2^{135.5}$	$2^{120.5}$	$2^{127.5}$	Amplified Boomerang	[YSZ+24]
	4	$2^{133.5}$	$2^{115.5}$	2^{131}	Amplified Boomerang	Section 3
	2^{17}	2^{117}	2^{117}	2^{78}	Boomerang	Section 4
256	4	$2^{99.5}$	$2^{99.5}$	2^{77}	Boomerang	[BK09]
	2^{19-s}	2^{92+s}	2^{91+s}	2^{89-s}	Boomerang	[GSW22]

Table 1: Summary of the best known full-round attacks on Related-key AES.

The paper is organized as follows: Section 2 describes the variant of AES we are considering in this paper, AES-192, and briefly presents the previously mentioned and relevant attacks. Section 3 describes our new amplified boomerang attack on AES-192 and Section 4 our new trade-off attacks considering key structures in AES-192 for the first time. The paper ends with a conclusion in Section 5.

2 Preliminaries

In this section we will start by briefly presenting boomerang and amplified boomerang attacks in the related key setting. We will also provide the main specification of AES, and present the two previous results on the best related-key attack on AES-192, and on the new key-structure trade-off on AES-256.

2.1 Boomerang and Amplified Boomerang Attacks.

First introduced in 1999 by Wagner [Wag99] the boomerang attack is a chosen plaintext and chosen ciphertext attack, where the cipher E is decomposed in two parts E_0, E_1 such that $E = E_1 \circ E_0$, and where there exists one good differential characteristic $\alpha \to \beta$ covering E_0 and one differential characteristic $\gamma \to \phi$ covering E_1 with probabilities p and q respectively. A simple boomerang attack proceeds as follows :

1. Encrypt P_1 and $P_2 = P_1 \oplus \alpha$ to C_1 and C_2.
2. Decrypt $C_3 = C_1 \oplus \gamma$ and $C_4 = C_2 \oplus \gamma$ to P_3 and P_4.
3. Check whether $P_3 \oplus P_4 = \alpha$.

If the differential characteristics are independent, then the probability that $P_3 \oplus P_4 = \alpha$ is p^2q^2. Another version of this attack is the amplified boomerang: instead of choosing the ciphertexts, we can turn the boomerang attack into an only chosen plaintexts attack, but the probability of the attack would be decreased by a factor 2^{-n}, where n is the size of the state. The attack is the same except that we only encrypt plaintexts P_1, P_2, P_3, P_4 such that $P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha$. For more details about the probabilities, we refer to [Wag99] for the original attack, and [KKS01] for the amplified boomerang attack. Over the past years some observations have been made on the success of boomerang distinguishers even when the differentials were independent, and it could happen that the boomerang "never comes back" [Mur11]. To overcome the issue, some tools have been developed such as the Boomerang Connectivity Table [CHP+18] which studies the compatibility of some boomerang distinguishers.

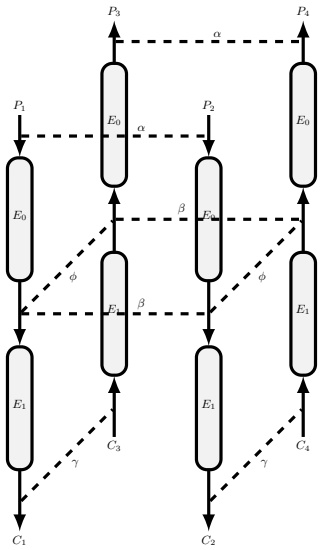


Fig. 1: The boomerang attack.

2.2 Boomerang Attacks in the Related-Key Setting

In the related key setting, the attacker is allowed to encrypt and decrypt data using multiple keys which are related by some properties. Boomerang attacks in the related key settings have already been formulated in [KKS01], [KHP+10]. We have related-key differential characteristics $\alpha \to \beta$, $\gamma \to \phi$ under ΔK and ∇K respectively. The attacker can consider 4 related keys (K_A, K_B, K_C, K_D) that are unknown, but that verify that $K_B = K_A \oplus \Delta$, $K_C = K_A \oplus \nabla$ and $K_D = K_B \oplus \nabla$. The difference between the keys can allow to have a more sparse

characteristic, and therefore a better probability. Next, the attack is the same as the previous boomerang attack, but we encrypt and decrypt under multiple keys:

1. Encrypt P_1 under K_A and $P_2 = P_1 \oplus \alpha$ under K_B to C_1 and C_2.
2. Decrypt $C_3 = C_1 \oplus \gamma$ under K_C and $C_4 = C_2 \oplus \gamma$ under K_D to P_3 and P_4.
3. Check wether $P_3 \oplus P_4 = \alpha$.

The success probability of the attack is still p^2q^2. Again, we can adapt this attack to the chosen plaintext setting by encrypting P_1, P_2, P_3, P_4 under K_A, K_B, K_C, K_D. The probability of having a good quartet also remains $2^{-n}p^2q^2$.

2.3 AES

The AES (Advanced Encryption Standard) [DR02b] is a Substitution-Permutation Network that takes a 128-bit plaintext as input and outputs a 128-bit ciphertext, encrypted with secret keys of different lengths. The number of rounds depends on the key size and can be 10, 12 or 14 for key sizes 128, 192 and 256 bits respectively.
The round function consists of four operations :

- **AddRoundKey** (AK) : The 128-bit round key is xored to the state
- **SubBytes** (SB) : The same S-box is applied to every byte of the state
- **ShiftRows** (SR) : The i-th row of the state is shifted by i positions to the left.
- **MixColumns** (MC) : Each column is multiplicated by an MDS matrix.

In the last round there is no **MixColumns** application, and a last round key is xored to the final state.

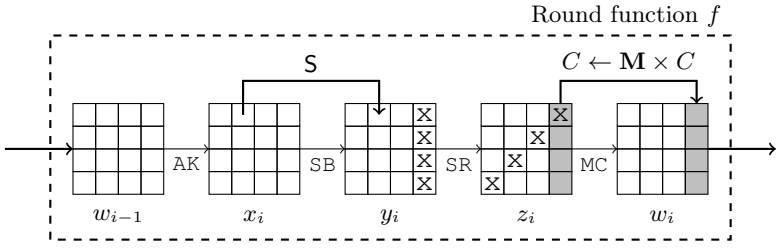
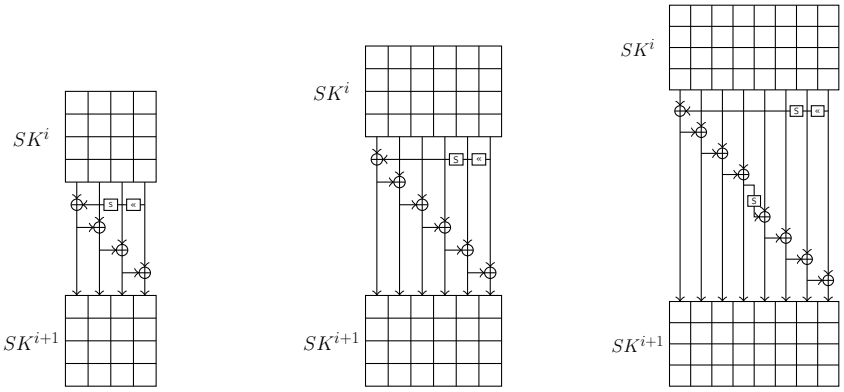


Fig. 2: An AES round ([Jea16])

The round keys are obtained by applying the key schedule depicted in Fig. 4 to the master key. In this paper, a round key will be denoted k and the bytes will be indexed as in 4. The subkeys will be denoted K and byte on line i and column j of the subkey will be denoted $K[i, j]$.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Fig. 3: AES byte ordering.



(a) AES-128 **(b)** AES-192 **(c)** AES-256

Fig. 4: The key schedules of the AES

Property of the AES S-box We recall a property of the AES S-box that will be used later for key-recovery.

Let Δ_{in} be an input difference and Δ_{out} an output difference. Three cases are possible:

- The differential transition is impossible for $129/256$ pairs of differences $(\Delta_{in}, \Delta_{out})$;
- There exist two ordered pairs of values: $(x, x \oplus \Delta_{in}), (x \oplus \Delta_{in}, x)$ that satisfy the transition for $126/256$ pairs of differences;
- There exist four ordered pairs of values that satisfy the transition for $1/256$ pair of differences.

This implies that for a given input/output difference, we either get 2, 4 or 0 input/output possible values for a total average of 1 solution.

2.4 On Previous Known Attacks on Full-Round AES-192

For a long time, the best attacks covering the longest number of rounds on AES-192 and AES-256 were the ones mentioned in table 1 from [BK09], that proposed boomerang related-key attacks on both versions, which was a very important and surprising result. Recently, another related-key boomerang attack, improving the

memory and time while slightly having a worst data, was proposed in [DEFN22]. A different trade-off in the amplified boomerang setting was proposed using the same distinguisher while applying the automatic key-recovery tool in [YSZ+24], with slightly better data and worse time and memory. We will describe in the next subsection the attack from [DEFN22], which is the one with the best known time complexity.

Related keys and local collisions. In the previous attacks on the AES, in the related-key setting, the main idea was to create local collisions by adding some disturbances in the state and correcting them a few steps after. This notion comes from attacks on hash functions in [CJ98] and an example is given in Fig. 5. One issue with local collisions is that correcting can be hard because of the key schedule or the round function itself, and some disturbances could get propagated and cause more disturbances and so on.
In the case of the AES-192, it is possible to maintain local collisions for a few rounds before the subkeys start to inject unknown disturbances due to the non linearity of the key schedule, but we can see that if the last column of a subkey is inactive, the propagation of the key difference to the next subkey will be linear so most of the good trails we can find in the literature will have empty last columns for most of the subkeys.
Moreover, there are some differences that one may choose rather than other, because of the DDT of the AES S-box. Given an input difference and an output difference, the highest probability that one can have for the transition to be occur is 2^{-6}. For example the probability of the differential depicted in Fig. 5 is 2^{-6} with the blue value set to $\Delta_{in} = 0x01$ and the purple columns set to $MC(\Delta_{out}) = MC(0x1f)$.

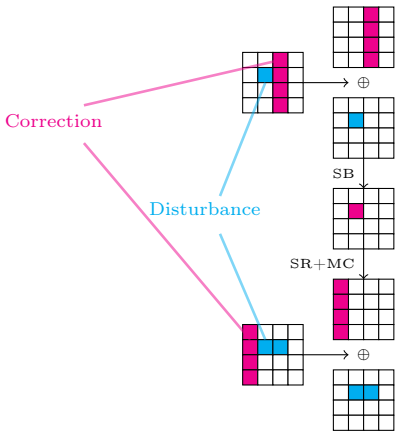


Fig. 5: Disturbances and corrections in 1 round of AES

2.5 Previous Best Attack on AES-192.

In 2022, Derbez *et al* [DEFN22] used MILP models to find a new boomerang trail with probability 2^{-108} covering all the rounds except the first and the last one, and this trail allowed to improve the time and memory complexity of the best attack.

The attack they proposed is a boomerang attack starting from the ciphertexts, as they claimed that it was better to consider this direction because of the difference configurations in the plaintext and ciphertext. The related keys are generated from the second subkey for the upper trail and the eighth subkey for the lower trail. The related key trails are given in Fig. 7, and the trail on full-round AES-192 is given in Fig. 6. The probability of the distinguisher is $2^{-2(4\times6+5\times6)} = 2^{-108}$. The steps of the attack are the following, and will be repeated $2^{108-40} = 2^{68}$ in order to obtain one good quartet, and the size of the structures is 2^{40}:

1. Decrypt under K_A a structure of 2^{40} ciphertexts where bytes 1, 2, 5, 6, 9, 10, 13, 14, $c[3] \oplus c[7], c[3] \oplus c[11], c[3] \oplus c[15]$ are constant and the others can take all the possible values. Decrypt a similar structure under K_C.
2. When looking at the distinguisher, two bytes of the plaintexts difference are unknown, so we need to try 2^{14} possible differences (as Δ_{out} is fixed, only 2^7 input differences are possible per byte) for each of the 2^{40} P_A and P_C to obtain P_B and P_D. Then we encrypt P_B and P_D under K_B and K_D respectively.
3. Look for collisions on the 11 bytes of $C_B \oplus C_D$. On average $2^{2(40+14)-11\times8} = 2^{20}$ quartets remain.
4. Apply filters on the quartets due to some known S-box outputs in the trail : each of the bytes 0, 3, 4, 8, 12 of the ciphertexts pairs apply a 1-bit filter due to the difference transitions of the S-boxes.
5. Recover the key bytes by observing S-box transitions and using 2.3 for bytes $K^0[1,5], K^0[1,3], K^8[0,0], K^8[0,1], K^8[0,2], K^8[0,3], K^7[0,5]$.
6. For each key value, update a counter. Once the counter reaches two, return the key bytes configurations that appeared twice.

The authors then state that they recover $(5+2)\times4$ bytes of key (7 for each key), and so they have 2^{14} values for 28 bytes. We realized that there is a small mistake in the original attack, as actually we can see that for $K^0[1,5]$ and $K^0[1,3]$, the bytes are related for every key because of the known difference in both ΔK and ∇K, so we only have 2^8 possibilities for each quartet of bytes, hence we have 2^{12} values for 28 bytes. Note that the knowledge of ΔK and ∇K can be used to do more filtering before updating counters on the key bytes.
This also has an impact on the probability to have noisy quartets that could give the same wrong key, indeed the number of different sequences of bytes of key is not 2^{28*8-1} but 2^{12*8-1}, which is much less. This could have an impact on the number of structures to use to be sure that the key bytes recovered are the right ones (or the impact could be on the time complexity of the attack).
However we believe that the complexities are not that much affected by these

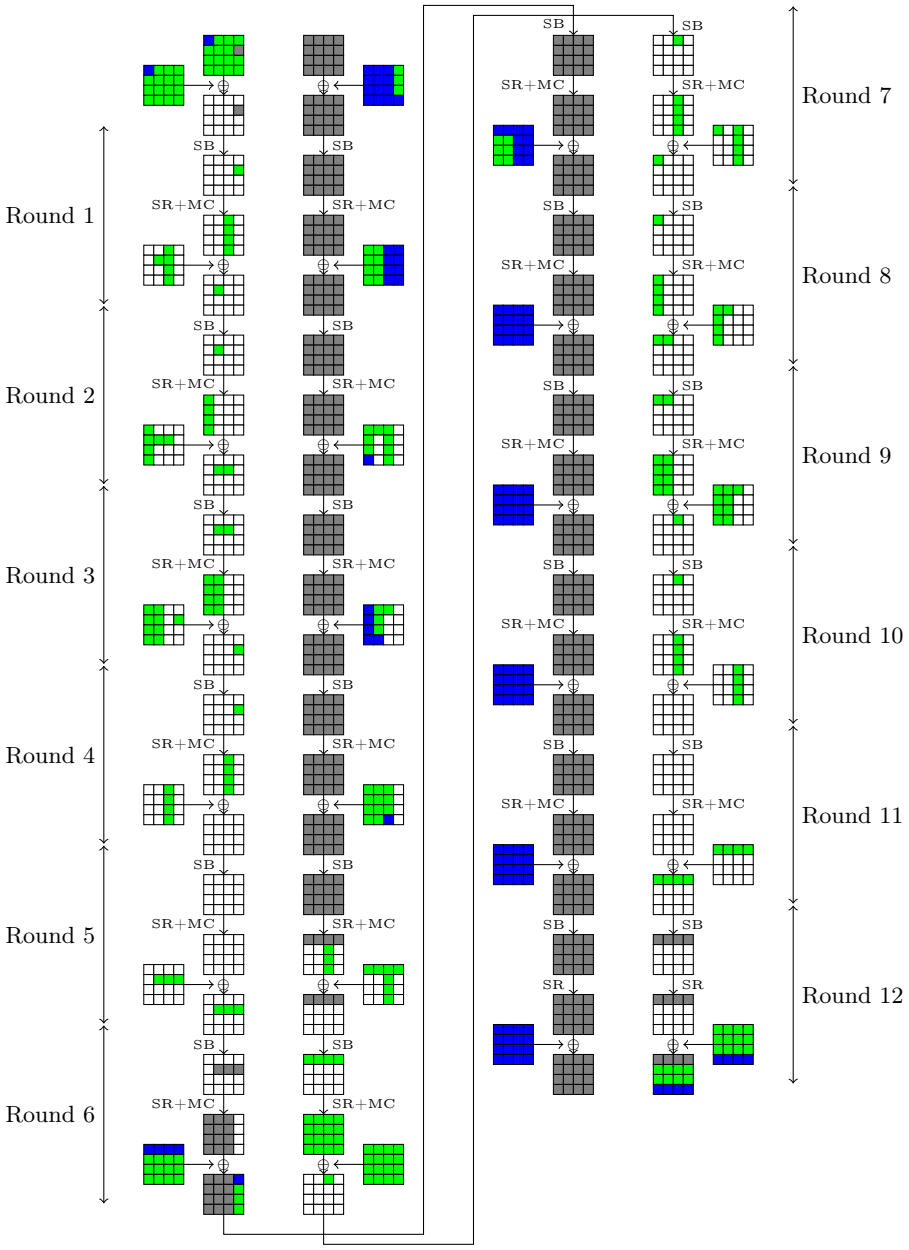


Fig. 6: The trail found in [DEFN22]. Green stands for known difference, blue for unknown differences that depends on the key values, gray for unknown differences.

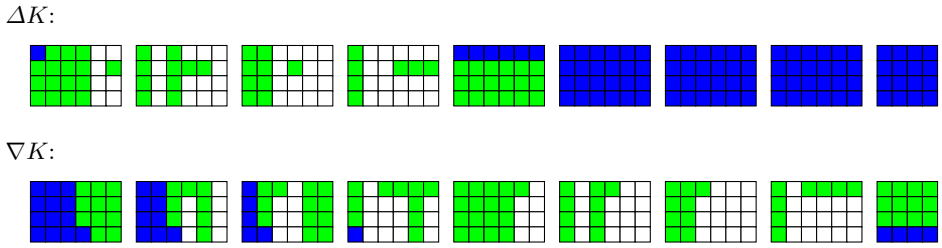


Fig. 7: Related keys trails from [DEFN22].

observations, and the authors state to have a data and time complexity of 2^{124}, and a memory complexity of $2^{79.8}$.

2.6 Attack on AES-256 using Key Structures

Also in 2022, a new attack by Jian Guo *et al* [GSW22] using the idea of building key structures in addition to data structures to generate more pairs to verify the probabilistic part for a smaller overall cost. They use the same distinguisher as in [BK09] (see Fig. 8), which covers all the rounds except the first one and 2^{19-s} or 2^{17-s} related-keys, and managed to reduce the data complexity, time and memory complexity to $2^{92.5+s}$, 2^{92-s}, 2^{89-s} respectively, which is an improvement in time and data complexity compared to the original attack. To make things easier, we will state the case where the number of related keys is 2^{17-s} and $s = 0$.

Key structures. In the original attack [BK09], the related keys K_A, K_B are created by applying the right differences to the subkey K^1 and propagating them forward and backwards to the other subkeys. When using key structures, we want the differences in K^1 to take all the possible values, so we build a key structure $S = \{K_i | K \oplus K_i = \Delta_i\}$ where Δ_i is zero on the right bytes and takes all different values on the other bytes, and K is the master key. The advantage of having key structures is that if we encrypt 1 plaintext structure of size m with 1 key structure of size n, we would get $2^{2m+2n-1}$ pairs instead of having 2^{2n+m-1} pairs for the same data cost obtained by encrypting n plaintext structures of size m.
For instance, in the attack of [GSW22] a key structure of size 2^{16} is created from the second subkey, see Fig. 8. The probability of the distinguisher can be found below, and the attack proceeds as follows:

1. Encrypt under each of the keys K belonging to the key structure a structure of 2^{72} plaintexts where bytes 0, 1, 2, 3, 4, 6, 9, 10, 13, 15 are constant and the other can take all the possible values.
2. For each ciphertext C, compute $C' = C \oplus \Delta$ where Δ is a well chosen difference for the boomerang to work.

3. Decrypt C' under K', where K' is computed from the corresponding K.
4. Store (P, P') in an hash table indexed by the 7 bytes of P' that correspond to the constant bytes of the plaintext structure and bytes 2, 3 of $P \oplus P'$.
5. Filter and recover the key bytes as depicted in [GSW22].

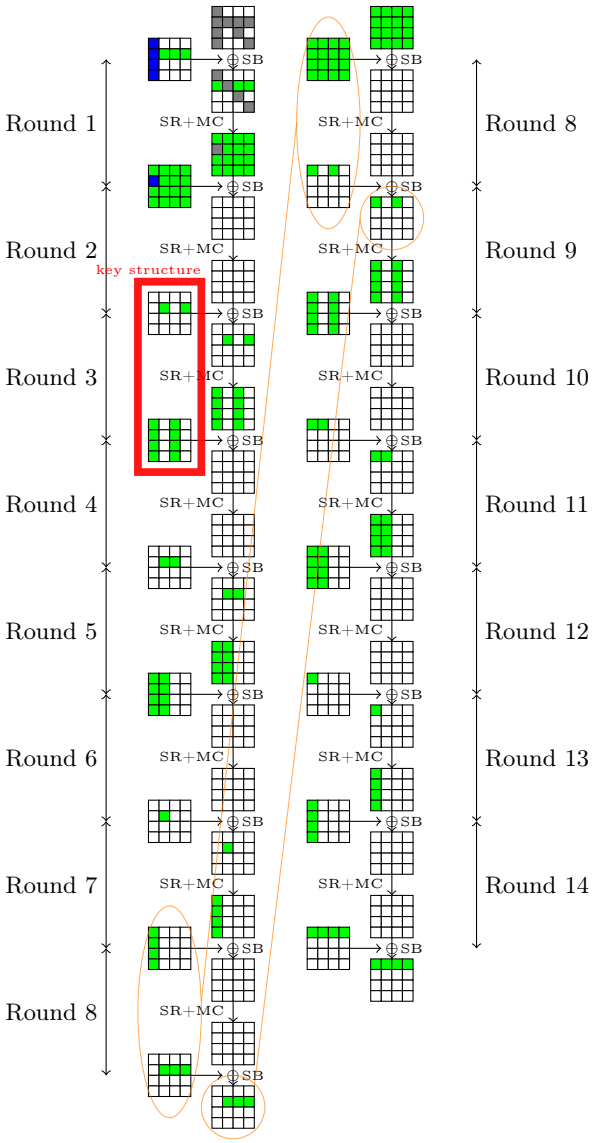


Fig. 8: Upper and lower trail for AES-256.

We will now discuss the probability of the distinguisher. In the original attack on AES-256 there were 2 keys per trail, so the differences in input and output of the S-boxes could be chosen in order to have the best probability to succeed. When one uses a key structure, the difference between each pair of key is not fixed and for $1/2^8$ of the keys the transition probabilities will be 2^{-6} per S-box, but for the $\frac{2^7-2}{2^8}$, it will be 2^{-7}.
Note that in their case this happens in the upper trail only, because the ciphertexts are decrypted only with two keys: the key that encrypted the plaintext, and its related key, which is computed with a chosen difference.
Thus, the probability of the whole distinguisher can either be $2^{2(-7\times5-6\times3)} = 2^{-106}$ or $2^{2(-6\times5-6\times3)} = 2^{-96}$.
For one plaintext structure, the number of plaintexts encrypted is $2^{72+16} = 2^{88}$ and the number of pairs that pass the first round is $2^{72+31} = 2^{103}$ for all the 2^{31} key pairs. Hence depending on the key pair, the average number of good quartets we get is $2^{103} \times \frac{2^7-2}{2^8} \times 2^{-106} \approx 2^{-4}$ or $2^{103} \times \frac{1}{2^8} \times 2^{-96} = 2^{-1}$. They adopt the second distinguisher because the number of average quartets is better than the other one. Thus in order to have 4 right quartets, 2^4 plaintext structures will be needed.
The complexities in data, time and memory are 2^{92}, $2^{92.5}$ and 2^{98}. To get to the complexities of Table-1, several tradeoffs are proposed, we refer to [GSW22] for more details.

Inapplicability to AES-192 The authors point out that, though the idea of key structures could potentially apply to the attack on AES-192, the particular configuration does not make it possible, and leave as an open problem to find a new application of key structures for AES-192.

3 New Improved Attack on AES-192

In this section we describe our attack on AES-192, which uses a modified version of the distinguisher from [DEFN22]. We relax the conditions of the distinguisher, consider the amplified boomerang instead of the boomerang framework, and take advantage of relations between bytes of keys in the key schedule that had not been noticed before. We start by describing a more detailed propagation of the related-keys in 3.1. Next, in 3.2 we present our new attack in detail.

3.1 Propagation of the key differences

In the previous attacks, the differences in the keys were applied on a subkey, and then propagated to the other subkeys with the key schedule. Due to the application of S-boxes in the AES key schedule, some differences were unpredictable. However, the key schedule is mostly linear and these unknown differences may cancel each other or be duplicated over different bytes of the subkeys after a few propagation steps, see Fig. 9, 10. This will be useful for filtering and recovering the key bytes. The differences of our attack can be found in Table 2.

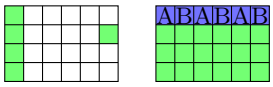


Fig. 9: Unknown difference propagation.

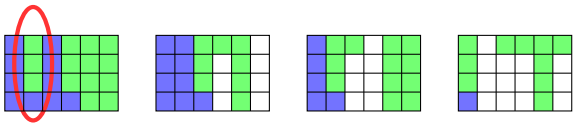


Fig. 10: Unknown difference cancellation of bytes of ∇K_0 in the differential key schedule of [DEFN22] when propagating backwards.

3.2 Improved Key-Recovery Extension

We consider the distinguisher from [DEFN22], where we are not enforcing the transition of the second round, and therefore the overall probability of the distinguisher we consider (represented in Figure 11 from Round 3 to Round 11) becomes $2^{-(2(3\times6)+2(5\times6))} = 2^{-96}$, which is 2^{12} smaller than the previous attack. We can see the internal state differences in Table 3. In the original attack, the authors claim it was better to start the attack in the decryption direction, but with the new difference configuration we can perform the attack in the encryption direction.

Data complexity. The size of the unknown differences in the plaintexts is now 32 bits higher than before. When using the boomerang attack, this would have increased the overall complexity of the attack, but we aim at applying the amplified boomerang framework, where all the data is queried at the encryption oracle. The number of quartets needed in the second round is therefore $2^{128+96} = 2^{224}$.

We build input structures of plaintexts S_A (or S_C) of size 2^{48}, with all the input bytes without a fixed known difference taking all possible values, and a fixed value in the rest, and another structure S_B (or S_D) equal to the states from S_A (or S_C) plus the fixed difference on the green bytes. We compute the ciphertext of each one of them with their corresponding key K_A, K_B, K_C or K_D. For each pair of elements from (S_A, S_B) (or (S_C, S_D)), we have a probability of 2^{-48} of verifying the transitions of the two first rounds. We therefore can produce, with a pair of these structures, $2^{48+48-48} = 2^{48}$ different pairs verifying the input difference of the distinguisher. If we combine them to form quartets, we could build $2^{48+48} = 2^{96}$ different quartets, verifying each corresponding pair the input differences of the distinguisher.

As in total we need 2^{224} such quartets, we will repeat this for 2^s structures such that $2^{2s+2\times48} = 2^{224}$, so we need $s = 64$. The data complexity will become $4 \times 2^{48+64} = 2^{114}$. We will explain now the attack procedure for efficiently building and merging these structures, in order to recover the key information.

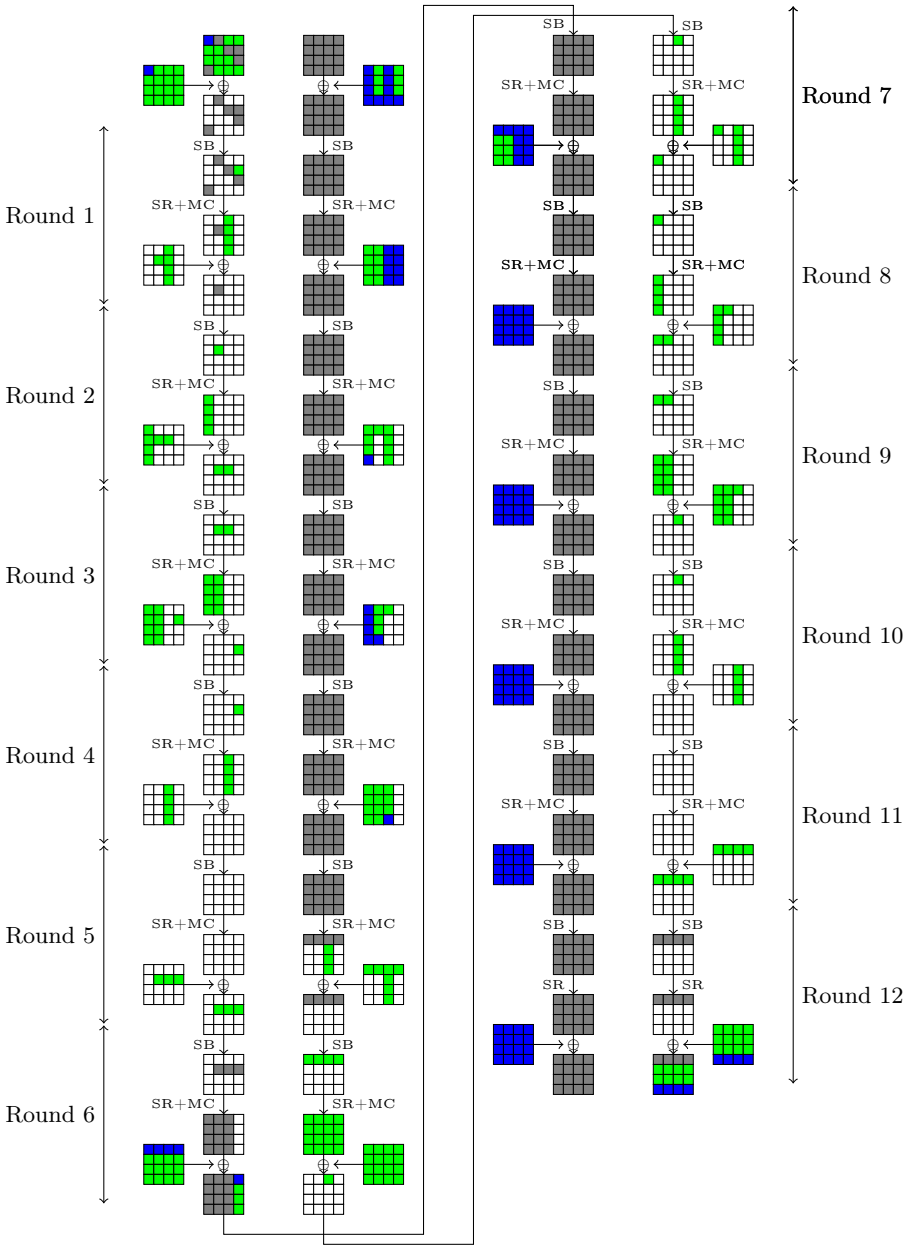


Fig. 11: The trail for AES-192. Green stands for known difference, blue for unknown differences that depends on the key values, gray for unknown differences.

ΔK^0	?	21	21	21	00	00	ΔK^1	21	00	21	00	00	00	ΔK^2	21	21	00	00	00	00
	3e	3e	3e	3f	00	01		3e	00	3e	01	01	00		3e	3e	00	01	00	00
	1f	1f	1f	1f	00	00		1f	00	1f	00	00	00		1f	1f	00	00	00	00
	1f	1f	1f	1f	00	00		1f	00	1f	00	00	00		1f	1f	00	00	00	00
ΔK^3	21	00	00	00	00	00	ΔK^4	A	A	A	A	A	A	ΔK^5	B	C	B	C	B	C
	3e	00	00	01	01	01		3e	3e	3e	3f	3e	3f		?	?	?	?	?	?
	1f	00	00	00	00	00		1f	1f	1f	1f	1f	1f		?	?	?	?	?	?
	1f	00	00	00	00	00		1f	1f	1f	1f	1f	1f		?	?	?	?	?	?
ΔK^6	D	E	F	G	D	E	ΔK^7	J	K	L	M	J	K	ΔK^8	?	?	?	?	?	?
	?	?	?	?	?	?		?	?	?	?	?	?		?	?	?	?	?	?
	?	?	?	?	?	?		?	?	?	?	?	?		?	?	?	?	?	?
	?	?	?	?	?	?		?	?	?	?	?	?		?	?	?	?	?	?
∇K^0	?	f8	?	f8	33	f8	∇K^1	?	?	33	cb	f8	00	∇K^2	?	f8	cb	00	f8	f8
	?	7c	?	7c	7c	7c		?	?	7c	00	7c	00		?	7c	00	00	7c	7c
	?	7c	?	7c	7c	7c		?	?	7c	00	7c	00		?	7c	00	00	7c	7c
	?	?	?	?	84	84		?	?	?	00	84	00		?	?	00	00	84	84
∇K^3	f8	00	cb	cb	33	cb	∇K^4	f8	f8	33	f8	cb	00	∇K^5	f8	00	33	cb	00	00
	7c	00	00	00	7c	00		7c	7c	7c	7c	00	00		7c	00	7c	00	00	00
	7c	00	00	00	7c	00		7c	7c	7c	7c	00	00		7c	00	7c	00	00	00
	?	00	00	00	84	00		84	84	84	84	00	00		84	00	84	00	00	00
∇K^6	f8	f8	cb	00	00	00	∇K^7	f8	00	cb	cb	cb	cb	∇K^8	f8	f8	33	f8	33	f8
	7c	7c	00	00	00	00		7c	00	00	00	00	00		7c	7c	7c	7c	7c	7c
	7c	7c	00	00	00	00		7c	00	00	00	00	00		7c	7c	7c	7c	7c	7c
	84	84	00	00	00	00		84	00	00	00	00	00		?	?	?	?	?	?

Table 2: Key difference in the AES-192 trail, determined by the key schedule. Capital letters stand for useful unknown differences in the key recovery phase.

3.3 The attack

The attack proceeds in two phases: collecting the data, and filtering it to recover key candidates.

Collecting quartets

1. For 2^{64} different values of i, ask for the encryption of a structure of 2^{48} plaintexts P_A under K_A where bytes 0, 3, 4, 9, 13, 14 take all the possible values and the other bytes are a constant i, and store all of them in a hashtable S_A of size $2^{64+48} = 2^{112}$, indexed by the values of the bytes 1, 2, 5, 6, 7, 9, 10, 11, 13, 14, 15 in the ciphertexts. Ask for the encryption of a similar structure under K_B and choose the constant bytes such that the right differences are applied between P_A and P_B and store all of them in a hashtable S_B of size 2^{112}, also indexed by the values of the bytes 1, 2, 5, 6, 7, 9, 10, 11, 13, 14, 15 in the ciphertexts.
2. Do the same for 2^{64} different values of j of P_C and P_D (encrypted with keys K_C and K_D), and store them respectively in S_C and S_D.

ΔP	? ? 00 00 00 00 ? ? 00 00 00 ? ? 00 00 00	Δy^1	00 ? 00 00 00 00 ? 1f 00 00 00 ? ? 00 00 00	Δy^2	00 00 00 00 00 1f 00 00 00 00 00 00 00 00 00 00	Δy^3	00 00 00 00 00 1f 1f 00 00 00 00 00 00 00 00 00
Δy^4	00 00 00 00 00 00 00 1f 00 00 00 00 00 00 00 00	Δy^5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Δy^6	00 00 00 00 00 ? ? ? 00 00 00 00 00 00 00 00	Δy^7	? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
∇y^6	7c 7c 7c 7c 00 00 00 00 00 00 00 00 00 00 00 00	∇y^7	00 00 7c 00 00 00 00 00 00 00 00 00 00 00 00 00	∇y^8	7c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇y^9	7c 7c 00 00 00 00 00 00 00 00 00 00 00 00 00 00
∇y^{10}	00 00 7c 00 00 00 00 00 00 00 00 00 00 00 00 00	∇y^{11}	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇y^{12}	? ? ? ? 00 00 00 00 00 00 00 00 00 00 00 00	ΔC	? ? ? ? 7c 7c 7c 7c 7c 7c 7c 7c ? ? ? ?

Table 3: Internal state difference in the AES-192 trail.

3. Look for collisions between S_A and S_C and between S_B and S_D, implying that bytes 1, 2, 5, 6, 7, 9, 10, 11, 13, 14, 15 have the right difference. This applies a 88-bit filter. We also know the input of 5 S-boxes (one from the key schedule in $\nabla K^7[0,5]$ and four from the last S-box layer), this leads to an additional 5-bit filter.

   We will recover $2^{2\times 112} \times 2^{-88-5} = 2^{131}$ collisions in (C_A, C_C) stored in hashtable $S_{A,C}$, and the same amount between (C_B, C_D), stored in $S_{B,D}$ (though we only need to store one of them). This will be the bottleneck in time and memory.
4. There are potentially 2^{262} possible quartets (C_A, C_B, C_C, C_D), but we do not need to check all of them: we can index hashtables $S_{A,C}$ (or $S_{B,D}$) with respect to the values of their plaintexts on bytes 1, 2, 5, 6, 7, 8, 10, 11, 12 and 15, as in order to form a right quartet, $P_A \oplus P_B$ and $P_C \oplus P_D$ need to have the right values. This can be seen as a filtering of $2^{-2\times 64} = 2^{-128}$ filter, as these is the number of different i (or j) we considered in the beginning. We also know two S-box outputs per pair of plaintext (one in $\Delta K^0[1,5]$, and one in the first S-box layer) so we also have a 2×2 bit filter.
   Thus we get $2^{262-128-4} = 2^{130}$ good quartets.

Recovering key bytes

We will see here which words w (using the numbering from Fig. 3) of each round keys $k^i[w]$ will be determined for each quartet, and which filtering we can apply to reduce the number of candidates from the starting 2^{130}. We recall that bytes of round keys will be denoted $k[i]$ and bytes of subkeys (of size 192 bits) will be denoted $K[line, column]$.

1. Similarly to the attack in [DEFN22] one quartet gives us candidates for bytes $k^0[13]$ for every related key, because we know the input and the output difference of the first S-box layer for the state byte 13. These candidates can be filtered by looking at ∇K^0 : for $k^0[13]$, the difference between $k^0_A[13]$ and $k^0_C[13]$ is already given by $\nabla K^0[1,3]$ so we need the candidates to satisfy this fixed difference. This provides an 8-bit filter.
2. We can do the same for $k^1[5]$. In the trail $\Delta K^0[0,0]$ is the result of the output difference of $\Delta K^0[1,5]$ through an S-box, and is also equal to $\Delta P[0]$ so we can deduce $k^1[5]$ for all the keys. We also have that $\nabla K^0[1,5]$ is determined by the key schedule so our candidates should satisfy this difference. Again, this provides an 8-bit filter.
3. We get 4 bytes for $k^{12}[0,4,8,12]$.This time the difference is not directly known in $\Delta K^8[0,i]$ but we can still apply some filters. Once we have candidates for $k^{12}[0,4,8,12]$ (or $K^8[0,i]$) for each related key, we can replace the unknown differences in $\Delta K^8[0,i]$ by known differences and we can propagate these differences backwards.
   The propagation gives us $\Delta K^7[0,1], \Delta K^7[0,2], \Delta K^7[0,3], \Delta K^6[0,2]$, $\Delta K^6[0,3], \Delta K^5[0,3]$. Then when looking at the unknown difference propagation (see Fig. 12) we can observe that $\Delta K^5[0,3] = \Delta K^5[0,1]$, and that $\Delta K^5[0,1]$ is equal to the output of the known difference $\Delta K^4[1,5] = 0x3f$ through an S-box. We can check if the difference transition is possible to get a 1-bit filter. If the transition is possible we get 8 bytes of information on the key (4 for each pair: $k^{12}[0,4,8,12]$).
4. We can also retrieve $k^{11}[12]$ from the ciphertexts difference. We can deduce key difference in $\Delta K^7[0,5]$, by observing that it is equal to $\Delta K^7[0,1]$ (see Appendix A for relations determined by the key schedule), which is fully determined by key bytes we recovered before. Thus we get an additional 8-bit filter, and determine 2 bytes of key (1 for each pair of keys: $k^{11}[12]$).
5. We now recover $k^0[3], k^0[4], k^0[9], k^0[14]$: There are 2^7 differences that can satisfy the S-box transition of the second round, so there are on average only 2^7 values for these four key bytes. So given (P_A, P_B) and (P_C, P_D), we have 2^7 candidates for these bytes of from K_A and K_B and 2^7 candidates for the bytes of from K_C and K_D, so 2^{14} possible values for the 4×4 bytes from K^0_A, K^0_B, K^0_C, K^0_D. We also have that $\nabla K^0[0,1]$ and $\nabla K^0[2,3]$ are known, which leads to a 16-bit filter, which considering the 2^{14} candidates, leaves a 2^{-2} filter. We already have determined $k^1[5] = K^0[1,5]$, so with the remaining key bytes that passed the filter we can compute the state byte before the S-box transition of the second round for each of the four related keys, and check if this S-box transition is possible, which adds a 7-bit filter per transition (not 8-bit, as we already considered that the input difference at round 2 is a valid one), so an additional 14-bit filter, so 16-bit in total. Finally, we can replace $\nabla K^0[1,2]$ by a known difference coming from the key candidates and propagate it to $\nabla K^2[1,2]$, in order to check if the S-box transition between the two known differences $\nabla K^2[1,2]$ and $\nabla K^2[2,5]$ is possible, this gives us a 1-bit filter. Overall we have a $16 + 1 = 17$ bit filter, and recover 6 bytes of the keys (4 for each pair of keys but two are related in both trails).

When we combine together all these filters we finally get a $8+8+1+8+17 =$ 42-bit filter and the number of good quartets is $2^{130-42} = 2^{88}$.

Each quartet gives us 18 key bytes (144 bits) that can take 2^{18} values (each determined key byte can take two possible values per fixed differential transition). As we have 2^{88} quartets, this gives $2^{88+18} = 2^{106}$ candidates for $(18 \times 8) = 144$ bits of information on the keys (11 bytes from the master key, and 7 bytes from other not directly related). We can repeat the procedure a total of three times, which will gives us another two sets of 2^{106} candidates. The good candidate needs to be in all the three sets, which happens with a probability of $(2^{106}/2^{144})^2 = 2^{-76}$, so the number of candidates in both sets will be $2^{106-76} = 2^{30}$. If we consider only the master key, we recover 88 bits from it in here. We have $(192 - 88) = 104$ remaining bits to guess to have the whole key. We could just guess them for each of the 2^{30} candidates, having a complexity of 2^{134}, but we can actually do better: we can easily chose which guesses to do first in order to be able to filter regarding the 7 bytes of the related keys known, such that this part won't be the bottleneck of the attack.

Actually just repeating the procedure two times and wisely choosing the guessing should already be enough.

Data complexity. We encrypt 2^{112} plaintexts for each related key, and next we repeat the procedure three times, so the data complexity will be $2^{115.58}$.

Time complexity For reaching the 2^{106} candidates for 144 information key bits:

$$\mathcal{T} = 3 \times (4 \times 2^{112} + 2 \times 2^{131} + 2^{130} + 2^{106}) = 2^{133.58}$$

As discussed, the last step is negligible compared to this one.

Memory complexity The memory complexity bottleneck is storing one of the hashtables $S_{A,C}$ or $S_{B,D}$, that is 2^{131}.

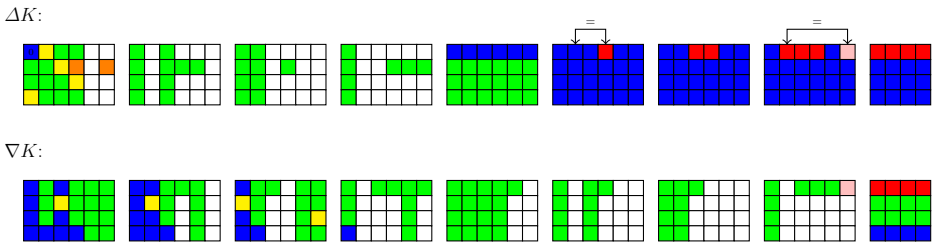


Fig. 12: Difference and value deductions in the key recovery. Green stands for known difference, blue for unknown, orange for steps 1 and 2, red for step 3, pink for step 4, yellow for step 5.

4 Using Related-Key Structures for new trade-offs

In order to reduce the data of the original boomerang attack from [DEFN22] and provide a new interesting trade-off, we use key structures as proposed in [GSW22]. In their paper, they could not apply the technique to the best existing attacks on AES-192. Taking into account the original distinguisher from [DEFN22] represented in Fig. 6, we have managed to provide new trade-offs with more related keys and reduced data and time complexities.

We recall that the main idea is that, instead of having two related keys (K_A, K_C) from the decryption side, we consider a structure $S = \{K_A + \Delta_i\}$ for a certain secret key K_A, where in our case, the zero bytes of Δ_i are the zero bytes of $K_A \oplus K_C$, and the other bytes take all the possible values that keep the right propagation of differences in the key-schedule trail. For instance, we can see that $\nabla K^5[0,3]$ and $\nabla K^6[0,2]$ must be equal to have that $\nabla K^6[0,3] = 0$, and also that the first and second columns of ∇K^6 must be equal and can take 2^8 values because these columns are produced by a MC operation on a column with one active byte. Therefore a key structure from the last three columns of K^5 and the three first columns of K^6 would be of size 2^{16} and the number of possible key pairs would be 2^{31}.

Using key structures in the lower trail instead of two keys, changes the probability of the distinguisher. Indeed the transitions through S-boxes will now have probability 2^{-7} for $(2^7 - 2)/2^8$ pairs, and probability 2^{-6} for a proportion of $1/2^8$ pairs. Because we use undetermined differences only in the lower trail, the probabilities in the upper trail are unchanged. Thus the probability of the distinguisher will be either $2^{-2(4\times6)+2(5\times7)} = 2^{-118}$ or 2^{-108}, depending on the key-schedule differences defined by a quartet of keys.

The attack. We use the key structure from the last three columns of K^5 and the three first columns of K^6 We decrypt 2^{40+16} ciphertexts (2^{40} from the ciphertext structure, and 2^{16} from the key structure) under $K_A + \Delta_i$ and $K_C = K_A + \Delta_j$ belonging both to the key structure, to have $2^{40+40+16+16-1-40} = 2^{71}$ ciphertext pairs that pass the last round, and we encrypt the 2^{56} obtained plaintexts after adding to each one all the possible 2^{14} input differences, recovering the encryption of $2^{14+56} = 2^{70}$ plaintexts. The average number of quartets we should get is $2^{71} \times \frac{2^7-2}{2^8} \times 2^{-118} = 2^{-48}$ or $2^{71} \times \frac{1}{2^8} \times 2^{-108} = 2^{-45}$, depending on the key pair used.
Therefore in order to recover 4 good quartets, we will need 2^{47} structures using the distinguisher with probability 2^{-108}, and the total data complexity is $2^{47+40+16+14} = 2^{117}$ (note that we added a 2^{14} factor for the plaintext encryption). The key recovery works the same way as in [DEFN22] but we need to make sure that we can recover the master key from the key pairs in the structure, because we could deduce some relations between two keys bytes in the key structure without knowing the relation between the master key and the keys for this byte.
Here are the steps of the attack:

1. Decrypt under each of the keys K belonging to the key structure a structure of 2^{40} ciphertexts where bytes 1, 2, 5, 6, 9, 10, 13, 14, $c[3] \oplus c[7], c[3] \oplus c[11], c[3] \oplus c[15]$ are constant and the others can take all the possible values.
2. For each plaintext P, compute the 2^{14} P' such that $P \oplus P'$ takes all the 2^{14} possible values in bytes 0 and 13, and the other bytes have the right fixed difference, and encrypt P' under the key K' related to the corresponding K.
3. Store the (C, C') in a hash table indexed by the bytes 1, 2, 5, 6, 9, 10, 13, 14, $C'[3] \oplus C'[7], C'[3] \oplus C'[11], C'[3] \oplus C'[15]$.
4. Look for good pairs (C_1, C_1', C_2, C_2') in the hash table where $C_1' \oplus C_2'$ have the right difference on the same bytes as before, this applies a 8×11-bit filter and $2^{2(40+16+14)-8\times 11} = 2^{52}$ quartets remain.
5. Apply filters on the quartets due to some known S-box outputs in the trail : each of the bytes 0, 3, 4, 8, 12 of the ciphertexts pairs apply a 1-bit filter. This applies a 10-bit filter and the number of quartets becomes 2^{42}
6. Apply the same filters as in step 1, 2, 3 and 4 of 3.3, to reduce the number of quartets to $2^{42-8-8-1-8} = 2^{17}$. Each quartet proposes 12 bytes of keys which are the same as those from 2.5.

In our case, for K_A, K_C in the key structure and K_B, K_D their related keys, we recover bytes $K^0[1,3], K^0[1,5]$ which are related for all the keys, and $K^7[0,5]$, $K^8[0,0]K^8[0,1], K^8[0,2], K^8[0,3]$ which are independent between each pair (K_A, K_C) and (K_B, K_D), hence we get $2 + 2 \times 5 = 12$ independent key bytes. Luckily we know for each key its difference from the master key in the specified positions, so we can deduce these bytes in the master key and an efficient key recovery is possible.
We recover the right key using a counter as for previous attacks.

Data complexity. We decrypt $2^{40+16+47} = 2^{103}$ ciphertexts and encrypt $2^{103+14} = 2^{117}$ plaintexts so the data complexity is 2^{117}.

Memory complexity. We need to store 2×2^{70} ciphertexts in a hash table and $2^{17+47+14} \times 2^{12} = 2^{78}$ key bytes. Thus the memory complexity is 2^{78}.

Time complexity We fill the hash table $2^{70+47} = 2^{117}$ times and we recover $12 \times 8 = 96$ bits of key. Then we can use the tool developed by Bouillaguet *et al* [BDF12] already used in [DEFN22] to recover the missing key bytes, and the cost of this phase should be negligible as pointed out in [DEFN22] . Thus the time complexity will be 2^{117}.

Though the data complexity of this attack is higher than that of our previous attack and the number of related keys needed is bigger, the time complexity is better and provides an interesting trade-off.

5 Conclusion

We have proposed two new attacks on AES-192. The first one, using only 4 related keys as the previous ones, considerably reduces the data complexity of the

best attack increasing the time complexity by a smaller factor than the reduction. It is the first time the full-round attack from [BK09] has been significantly improved with respect to data complexity. To achieve this, we consider the core distinguisher given in [DEFN22] and relax some conditions on the distinguisher, leading to a more complex key recovery part, with a higher probability distinguisher. This allows us to efficiently transform the previous attack into an amplified boomerang attack. In addition we provide for the first time new trade-offs, also improving the data and time, when using the key structure technique from [GSW22] and a bigger set of related-keys.

Discussion tools for Boomerang attacks on AES In [BN10] a tool for finding related-key differential characteristics is presented, and with it an interesting distinguisher on 9 rounds and 2^{-96} probability, though the authors did not bother to describe the key-recovery part of the attack, that seems to need more data than originally claimed, so not to produce a real improvement when compared to the attacks from 2022. In [DEFN22] an improved attack is found thanks to an automated tool, though the authors claim it is too slow for exhaustive search. In [YSZ+24], the authors propose a tool for automatizing the key recovery step in rectangle attacks. They are able to recover the previous attack from [DEFN22], and they propose a new attack in the amplified boomerang setting with complexities of $2^{135.5}$ in time, $2^{120.5}$ in data and $2^{127.5}$ in memory, which strcitly worse than the new attack we proposed. In addition, we have launched their tool with our modified distinguisher, and recovered a complexity of $2^{114.79}$ data (versus $2^{115.58}$ for our attack), $2^{148.79}$ memory (versus $2^{133.5}$) and $2^{135.9}$ time (versus 2^{132}). We do not have the details about how the tool would perform the key recovery attack, but we found it surprising that, though the authors claim the tool provides optimal time, our procedure has better time complexity by a factor of around 2^6.

We believe it is important to point out that, even though automatic tools for finding efficient distinguishers are very important and have benefited from great advances lately, a very important next step would be to try to include possible key-recovery extensions in these automatic tools to find the most interesting trade-offs. While this is not achieved, we need to keep on carefully performing these key-recovery extensions by hand, trying different configurations for finding new improved attacks, as we have done in this work. This will also help us understand what ideas might be missing from the tools and tell us how to improve them.

Acknowledgments This research is partially funded by the European Union (ERC2023-COG, SoBaSyC, 101125450). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This work has also been partially supported by the France 2030 program under grant agreement No. ANR-22-PECY-0010 CRYPTANALYSE.

References

BBG+08. Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. SHA-3 proposal: ECHO. In *Submission to NIST*, 2008.

BDD+12. Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Pierre-Alain Fouque, Nathan Keller, and Vincent Rijmen. Low-data complexity attacks on AES. *IEEE Trans. Inf. Theory*, 58(11):7002–7017, 2012.

BDF12. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. Cryptology ePrint Archive, Paper 2012/069, 2012.

BDK+10. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 299–319. Springer, 2010.

BK09. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

BKR11. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011.

BN10. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 322–344. Springer, 2010.

CHP+18. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. Cryptology ePrint Archive, Paper 2018/161, 2018.

CJ98. Florent Chabaud and Antoine Joux. Differential collisions in SHA-0. pages 56–71, 1998.

DEFN22. Patrick Derbez, Marie Euler, Pierre-Alain Fouque, and Phuong Hoa Nguyen. Revisiting related-key boomerang attacks on AES using computer-aided tool. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 68–88. Springer, 2022.

DR02a. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

DR02b. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography)*. Springer, 1 edition, 2002.

GKM+09. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. In Helena Handschuh, Stefan Lucks, Bart Preneel, and Phillip Rogaway, editors, *Symmetric Cryptography, 11.01. - 16.01.2009*, volume 09031 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009.

GSW22. Jian Guo, Ling Song, and Haoyang Wang. Key structures: Improved related-key boomerang attack against the full aes-256. In Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo, editors, *Information Security and Privacy*, pages 3–23, Cham, 2022. Springer International Publishing.

IAC+09. Sebastiaan Indesteege, Elena Andreeva, Christophe De Cannière, Orr Dunkelman, Emilia Käsper, Svetla Nikova, Bart Preneel, and Elmar Tischhauser. The lane hash function. In Helena Handschuh, Stefan Lucks, Bart Preneel, and Phillip Rogaway, editors, *Symmetric Cryptography, 11.01. - 16.01.2009*, volume 09031 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009.

Jea16. Jérémy Jean. TikZ for Cryptographers. `https://www.iacr.org/authors/tikz/`, 2016.

KHP+10. Jongsung Kim, Seokhie Hong, Bart Preneel, Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. Cryptology ePrint Archive, Paper 2010/019, 2010.

KKS01. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round mars and serpent. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 75–93, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

Mur11. Sean Murphy. The return of the cryptographic boomerang. *Information Theory, IEEE Transactions on*, 57:2517 – 2521, 05 2011.

Wag99. David Wagner. The boomerang attack. pages 156–170, 1999.

YSZ+24. Qianqian Yang, Ling Song, Nana Zhang, Danping Shi, Libo Wang, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. *J. Cryptol.*, 37(2):19, 2024.

A Unknown differences propagation in the key schedule

$\Delta K^4[0,i]$	$S(\Delta 0x91) \oplus 21$
$\Delta K^5[0,0]$	$S(\Delta 0x3f) \oplus \Delta K^4[0,0] = S(\Delta 0x3f) \oplus S(\Delta 0x91) \oplus 21$
$\Delta K^5[0,1]$	$\Delta K^5[0,0] \oplus \Delta K^4[0,1] = (\Delta 0x3f) \oplus S(\Delta 0x91) \oplus 21 = S(\Delta 0x3f)$
$\Delta K^5[0,2i]$	$\Delta K^5[0,0]$
$\Delta K^5[0,2i+1]$	$\Delta K^5[0,1]$
$\Delta K^6[0,0]$	$S(\Delta_{unk1}) \oplus \Delta K^5[0,0] = S(\Delta_{unk1}) \oplus S(\Delta 0x3f) \oplus S(\Delta 0x91) \oplus 21$
$\Delta K^6[0,1]$	$\Delta K^5[0,1] \oplus \Delta K^6[0,0] = S(\Delta_{unk1}) + S(\Delta 0x91) \oplus 21$
$\Delta K^6[0,2]$	$\Delta K^5[0,2] \oplus \Delta K^6[0,1] = S(\Delta 0x3f) \oplus S(\Delta_{unk1})$
$\Delta K^6[0,3]$	$\Delta K^5[0,3] \oplus \Delta K^6[0,2] = S(\Delta 0x3f) \oplus S(\Delta 0x3f) \oplus S(\Delta_{unk1}) = S(\Delta_{unk1})$
$\Delta K^6[0,4]$	$\Delta K^5[0,4] \oplus \Delta K^6[0,3] = S(\Delta_{unk1}) \oplus S(\Delta 0x3f) \oplus S(\Delta 0x91) \oplus 21 = \Delta K^6[0,0]$
$\Delta K^6[0,5]$	$\Delta K^5[0,5] \oplus \Delta K^6[0,4] = \Delta K^6[0,0] \oplus S(\Delta 0x3f) = \Delta K^6[0,1]$
$\Delta K^7[0,0]$	$\Delta K^6[0,0] \oplus S(\Delta_{unk2}) = S(\Delta_{unk1}) \oplus S(\Delta 0x3f) \oplus S(\Delta 0x91) \oplus 21 \oplus S(\Delta_{unk2})$
$\Delta K^7[0,1]$	$\Delta K^6[0,1] \oplus \Delta K^7[0,0] = S(\Delta_{unk2}) \oplus S(\Delta 0x3f)$
$\Delta K^7[0,2]$	$\Delta K^6[0,2] \oplus \Delta K^7[0,1] = S(\Delta_{unk2}) \oplus S(\Delta_{unk1})$
$\Delta K^7[0,3]$	$\Delta K^6[0,3] \oplus \Delta K^7[0,2] = S(\Delta_{unk2})$
$\Delta K^7[0,4]$	$\Delta K^6[0,4] \oplus \Delta K^7[0,3] = S(\Delta_{unk2}) \oplus \Delta K^6[0,0] = \Delta K^7[0,0]$
$\Delta K^7[0,5]$	$\Delta K^6[0,5] \oplus \Delta K^7[0,4] = \Delta K^7[0,0] \oplus \Delta K^6[0,1] = \Delta K^7[0,1]$

Table 4: Explanation on the propagation of the differences in the AES-192 key schedule. Δ_{unk1} and Δ_{unk2} stand for some unknown differences, and $S(\Delta x)$ stands for the output difference from the S-box (which is unknown)