Subspace Trail Cryptanalysis and its Applications to AES

Lorenzo Grassi[1], Christian Rechberger[1,3] and Sondre Rønjom[2]

[1] IAIK, Graz University of Technology, Austria
[2] Nasjonal sikkerhetsmyndighet, Norway
[3] DTU Compute, DTU, Denmark
{firstname.lastname}@iaik.tugraz.at, sondrer@gmail.com

Abstract. We introduce subspace trail cryptanalysis, a generalization of invariant subspace cryptanalysis. With this more generic treatment of subspaces we do no longer rely on specific choices of round constants or subkeys, and the resulting method is as such a potentially more powerful attack vector.
We provide a general framework for subspace trail cryptanalysis of AES-like Substitution-Permutation Network (SPN) constructions. Interestingly, subspace trail cryptanalysis in fact includes earlier techniques based on impossible or truncated differential cryptanalysis and the integral property as special cases.
Choosing AES-128 as the perhaps most studied cipher, we describe distinguishers up to 5 round-reduced AES with a single unknown key. Using subspace trails as distinguishers, we report (and practically verify) competitive key-recovery attacks with very low data-complexity on 2, 3 and 4 rounds of AES.
As the most interesting concrete result, we are able to describe (and practically verify on small-scale AES) the first 5-round distinguisher for (all versions of) AES that does not require any knowledge about subkeys, any assumption on the MDS matrices or/and on the SubBytes operation, and it needs only $2^{59.7}$ chosen plaintexts or ciphertexts. This thereby significantly improves over very recent results of Sun, Liu, Guo, Qu and Rijmen from CRYPTO 2016.

Keywords: Block cipher - AES - Invariant Subspace Attack - Subspace Trail Cryptanalysis - Secret-Key Distinguisher - Low-Data Attacks - Truncated Differential Cryptanalysis - Zero-Sum - Impossible Differential Cryptanalysis

1 Introduction

In this paper we present a new cryptanalysis technique that adds to the toolbox of techniques at the disposal for cryptanalysts to evaluate the security of designs in symmetric cryptography.

1.1 Background and motivation

If a cryptographic primitive succumbs to particular non-random behavior, it might be possible to distinguish it from what one would expect from sufficiently generic behavior. Invariant subspace cryptanalysis is a cryptanalytic technique that is powerful for certain block ciphers. If there exists invariant subspace for the round function and for the key schedule, then this technique can be used to mount fast distinguishers and key recovery. This technique was introduced in [25] at CRYPTO 2011 for the cryptanalysis of PRINTcipher. Its efficiency has also been demonstrated on on the CAESAR candidate iSCREAM, on the LS-design Robin and on the lightweight cipher Zorro in [26], and on the block cipher Midori64 [23]. However, if such symmetries do not exist or are not found, invariant subspace cryptanalysis is not applicable. This leads to the natural question: *Can subspace properties still be used, even if no special symmetries or constants allow for invariant subspaces?* This paper will answer this question in the affirmative.

1.2 High-level overview of subspace-trail technique

Our main contribution is the analysis of subspaces in SPNs (substitution-permutation networks) constructions, which can be seen as a generalization of the invariant subspace attack [25,26]. While invariant subspace cryptanalysis relies on iterative subspace structures, our analysis focuses on *trails* of different subspaces. To clarify the presentation, we focus on the well-known block cipher AES-128. In particular, we study the propagation of subspaces trough various building blocks like S-Box and linear layers. In that sense it has similarities with SASAS cryptanalysis [8], but also with Evertse's linear structures [22].

In this paper we investigate the behavior of subspaces in keyed permutations. At a high level, we fix subspaces of the plaintext that maintain predictable properties after repeated applications of a key-dependent round function. First we identify what we call *subspace trails* which is essentially a coset of a plaintext subspace that encrypts to proper subspaces of the state space over several rounds. The trails are formed by the affine hulls of the intermediate ciphertexts. Subspace trails typically consist of subspaces that increase in dimension for each round, meaning that if the plaintext subspace has low dimension in comparison to the block length, the subsequent subspaces dimension increases for each round. For byte-based ciphers (like AES), a quick and dirty test for subspaces is to compute the affine hulls of a n-round encryption (for a certain $n \geq 1$) of all values for each byte and then identify these subspaces. For bit-based ciphers, it is more important to determine what was coined a *nucleon* in [26], that is candidate plaintext subspaces that seem to fit symmetries in the round function. Trails of affine hulls of the intermediate ciphertexts that grow slowly in dimension for each round, typically reflect slow diffusion in the round function. This is often the case for ciphers that iterate simple round functions many times. In this paper we will focus on what we call *constant dimensional subspace trails*, which are trails of cosets that preserve dimension over several rounds. We show how to connect two or more trails and form longer trails that preserve predictable structure. In particular, when we connect two trails we typically seek to describe an output coset of a first trail in terms of cosets of the input coset for the second trail.

1.3 Contributions

There are three types of contributions in this paper. *Firstly* the definition and description of the subspace trails techniques and as examples of first applications of it the generalization of the secret-key distinguishers for AES-128 from up to 4 rounds. The approach to the generalization from invariant subspace cryptanalysis to subspace trail is outlined in Sect. 2. In Sect. 3 we give technical preliminaries with respect to AES-like permutations, and in Sect. 4 we state central theorems related to subspace trails and their intersections. When concretely applying it to AES, we describe in Sect. 5 distinguishers of round-reduced AES with a single unknown key up to 4 rounds, which correspond to truncated differential, impossible differential, and integral distinguishers. From this it will become clear that well known techniques such as impossible or truncated differentials as well as integral properties can be seen as special cases of subspace trails.

Secondly, we describe new low data-complexity key-recovery attacks on AES up to 4 rounds, based on a combination of a truncated differential property (i.e. a relation among pairs of texts) and of properties of individual texts, which follows naturally from the proposed subspace trail approach, see Sect. 6 to 9. The differences and the relationships between these attacks based on the subspace trail and the others present in literature are discussed in Sect. 7.

Finally, in Sect. 10, building up on the 4-round distinguisher and some techniques from the key recovery sections, we are able to present *the first 5-rounds secret key distinguisher for all version of AES which needs much less than the full codebooks (it has a data complexity of* $2^{59.7}$ *chosen plaintexts/ciphertexts) and which doesn't require any assumption on the MDS matrices or/and on the SubBytes operation.*

Before to start with these sections, we discuss our concrete results: first the distinguishers in the unknown (secret)-key model, and then key-recovery attacks, and in both cases we compare them with earlier works.

1.4 Secret-Key Distinguishers for AES

The Rijndael block cipher [14] has been designed by Daemen and Rijmen in 1997 and accepted as the AES (Advanced Encryption Standard) standard since October 2000 by the NIST. Nowadays, it is probably the most used block cipher. No real mathematical improvements on the first analysis performed during the AES competition has been made, and the current standard AES is almost as secure as it was 15 years ago.

Since practical attacks on block ciphers became extremely rare in the last two decades, the approaches of the cryptanalysis community has been to concentrate on attacking reduced-round variants of block ciphers and/or to allow the adversary more degrees of freedom in its control. In the first approach, the usual goal of the adversary is to maximize the number of rounds that can be broken, using less data than the entire codebook and less time than exhaustive key search. Attacks following each of these approaches are of importance, both because they ensure that the block ciphers are strong enough and since they help to establish the security margins offered by the cipher. However this drives to aim for the highest number of rounds often leads cryptanalysts to consider effects that are just slightly deviating from a perfect behavior leading to attacks very close to brute force attacks or requiring completely impractical amounts of chosen or known inputs up to the full codebook. Practical attacks, especially those focusing on low data complexity, rightfully gained more attention recently, and this is also the the focus of the key-recovery part in this paper.

In the usual security model, the adversary is given a *black box* (oracle) access to an instance of the encryption function associated with a random secret key and its inverse. The goal is to find the key or more generally to efficiently distinguish the encryption function from a random permutation.

In Table 1.4 we summarize the secret-key distinguishers for 1 up to 5 rounds. Such results often serve as a basis for key recovery attacks in the most relevant single-key setting. The subspace trail cryptanalysis includes as special cases of differential cryptanalysis techniques (like truncated or impossible differentials) and integral cryptanalysis, hence the complexities for distinguishers up to 4 rounds is the same.

The first distinguisher for five rounds of AES-128 has been proposed recently in CRYPTO 2016 [30]. However, it requires the *whole* input-output space to work, or 2^{96} texts if some knowledge of subkey bytes is assumed. Thus, our proposed secret key distinguisher for all versions of AES is the first one that requires (much) less than the whole input-output space without any knowledge about subkeys.

Table 1. *AES secret-key distinguishers, independent of key schedule.* Data Complexity is measured in minimum number of chosen plaintexts CP or/and chosen ciphertexts CC (which is equal for the random and the subspace case) which are needed to distinguish the two cases with high probability (usually higher than 95%). The case in which the MixColumns operation is omitted in the last round is denoted by "r.5 rounds", that is r full rounds and the final round.

Rounds	**Data**	**CP**	**CC**	**Property**	**Reference**
1 - 1.5 - 2	2	×	×	Subspace Trail	Sect. 5.1
1 - 1.5 - 2	2	×	×	Truncated Differential	[15]
2.5 - 3	$20 \simeq 2^{4.3}$	×	×	Subspace Trail	Sect. 5.2
2.5 - 3	$20 \simeq 2^{4.3}$	×	×	Truncated Differential	[7]
2.5 - 3	2^{8}	×	×	Integral	[13]
3.5 - 4	$2^{16.25}$	×	×	Impossible Differential	[5]
3.5 - 4	$2^{16.25}$	×	×	Subspace Trail	Sect. 5.3
3.5 - 4	2^{32}	×	×	Integral	[13] - Sect. 5.3
4.5 - 5	$2^{59.7}$	×	×	Subspace Trail	Sect. 10
4.5 - 5	$2^{98.2}$	×		Subspace Trail	App. B
5	2^{128}		×	Integral	[30]

Relation to Differential and Integral Distinguishers. The 1-, 2- and 3-round distinguishers exploit the same well-known structural properties that also truncated differentials exhibit. Using a different notation (namely the AES "Super S-Box"), 2-rounds subspace trails were already discovered and investigated in [15] and [16], with the objective to understand how the components of the AES interact. In these papers, authors study the probability of differentials and characteristics over 2 rounds of AES, giving bound on the maximum differential probability (which can be used to derive bounds on the expected differential probability of four-round differentials). Starting from such a 2-rounds subspace trail, in the paper we present competitive key-recovery attacks on 2-, 3- and 4-rounds of AES.

The first key-recovery attacks on round-reduced AES were obtained by introducing an attack vector that uses a 3-round distinguisher to attack up to 6 rounds of the cipher that goes back to the block cipher Square [13] and later became known as integral attacks.

In the meanwhile the most recent attacks achieve 7 rounds (using either impossible differentials or meet-in-the-middle techniques) with complexity significantly faster than brute-force search [28,19]. The impossible differential distinguishers used for those attacks are up to 4 rounds. Our 4-round subspace trail distinguisher uses the same structural properties exploited by impossible differential distinguishers.

In [30], authors present the first 5-rounds secret key distinguisher for AES-128. First they construct several types of 5-rounds zero-correlation linear hulls for AES-like ciphers, and then, using the link between integrals and zero correlation linear hulls [31], they are able to construct an integral distinguisher on 5 rounds. However, this distinguisher requires all the input-output space to work, that is the data complexity is of 2^{128} texts, or alternatively some knowledge about subkey bits where it then requires 2^{120} texts. Moreover, this distinguisher is constructed in the chosen-ciphertext mode, and only in the case in which MixColumns in the last round is not omitted. For this reason, authors claim that "*since the 5-round distinguisher for AES can only be constructed in the chosen-ciphertexts mode, the security margin for the round-reduced AES under the chosen-plaintext attack may be different from that under the chosen-ciphertext attack*".

Finally, the distinguisher presented in [30] works only for "*AES-like ciphers that adopt identical S-boxes to construct the round function and that have two identical elements in a column of the inverse of their MDS matrices*". For this reason, authors claim that "*when design an AES-like cipher, it is better to choose those MDS matrices M_{MC} such that both M_{MC} and M_{MC}^{-1} do not have identical elements in the same columns*".

Our 5-rounds secret key distinguisher presented in Sect. 10 is constructed in the chosen-plaintexts setting, extending the impossible 4-rounds distinguisher presented in Sect. 5.3 at the beginning. Our distinguisher works independent of the presence of the last MixColumns operation, and it has a data complexity of only $2^{59.7}$ chosen plaintexts instead of 2^{128}. Moreover, we show that it is possible to construct an equivalent distinguisher also in the chosen-ciphertexts setting, independently on the presence of the final MixColumns operation and with the same data complexity. Hence it provides a counterexample to the conjecture made in [30], i.e. it seems there is no clear evidence that chosen-ciphertext security is less than chosen-plaintext security in AES. Moreover, our distinguisher doesn't require any assumptions both on the SubBytes operations and on the MixColumns operations. Thus, it provides a counterexample to the advice given in [30] about the choice of the MDS matrices and of the S-Boxes for AES-like cipher design. In particular, *for each choice of invertible MDS matrices and of SubBytes operation, it is always possible to set up our secret-key distinguisher on 5-rounds AES, both in chosen-ciphertexts and in the chosen-plaintexts setting and independently of the presence of the final MixColumns operation.*

The subspace trail approach is mostly providing an alternative description of known properties under the umbrella of a single framework. However, there are other recent techniques that this approach does *not* seem to include. Recently integral distinguishers have been generalized by Todo [34] and in there also applied to AES-like primitives. Distinguishers for AES itself were not improved, but clear progress e.g. with MISTY cryptanalysis was demonstrated [33]. Todo's generalization can take S-Box properties into account, on the other hand the property exploited is still a type of zero-sum. Thus it complements our approach which is independent of the S-Box, but exploits properties more subtle than zero-sums.

Polytopic cryptanalysis, introduced by Tiessen in [32], is a generalization of differential cryptanalysis, and provides another type of distinguisher. While standard differential cryptanalysis uses statistical dependencies between the difference of two plaintexts and the difference of the respective two ciphertexts to attack a cipher, polytopic cryptanalysis considers interdependencies between larger sets of texts as they traverse through the cipher. Subspace trails do not seem to capture this type of distinguisher.

1.5 Low-Data Complexity Attacks on AES Using Subspace Trails

State of the Art of Attacks on AES. AES with its wide-trail strategy was designed to withstand *Differential* and *Linear cryptanalyses* [14], so pure versions of these techniques have limited applications in attacks. Hence, it is widely believed that no regular differential attack can be mounted on more than 5 rounds of AES (for example it was proved in [29] that any 4-round differential of AES has probability of at most 2^{-110}). With respect to AES, probably some of the most powerful single-key recovery methods are *Impossible differential cryptanalysis* [28] and *Square attacks* [13].

Impossible differential cryptanalysis yielded the first attack on the 7-round AES-128 with non-marginal data complexity. The Square attack and its variations such as integral attack and multiset attack resulted in the cryptanalysis of round-reduced AES variants with lowest computational complexity to date.

Another attack that initially has obtained less attention than the previous ones (due to the requirement for large parts of the cipher to be independent of particular key bits) is the *Meet-in-the-middle attack* [17]. This attack has great potential if enhanced by other techniques/attacks, as the differential attack [21,19,18] or as the *Bicliques technique* [9]. The biclique cryptanalysis applies to all full versions of AES (at the price of using an exhaustive loop on all the key bits), and compared to brute force provides a computational advantage of about a factor 3 to 5, depending on the version.

In works like [10] authors consider *Low-Data Complexity* attacks on reduced-rounds of AES, that is they apply the previous attacks in the case in which the attacker has limited resources, e.g. few plaintext/ciphertext pairs (which is much more relevant for practice). The results of this work have then been improved in [11]. In this paper, authors set up tools which try to find attacks automatically by searching some classes of Guess-and-Determine and Meet-in-the-Middle attacks. These tools take as input a system of equations that describes the cryptographic primitive and some constraints on the plaintext and ciphertext variables. Then, they first run a search for an "ad hoc" solver for the equations to solve, build it, and then run it to obtain the actual solutions.

Another work in the low-data complexity scenario is the *Polytopic Cryptanalysis* presented in [32], which is a generalization of differential cryptanalysis (as we have already discussed). In particular, the impossible polytopic cryptanalysis variant (that is, polytopic cryptanalysis that makes use of differentials with probability zero) allows competitive low-data attacks on round-reduced AES.

Our Key-Recovery Results. In this paper, we also present key-recovery attacks on reduced-round variants of AES-128 based on *subspace trail cryptanalysis*. A comparison of all known state of art of attacks on AES and our attacks presented in this paper is given in Table 1.5. To better understand this Table, we'd like to highlight some aspects. Without going into the details, AES is a key-iterated block cipher that consists of the repeated application of a round transformation on the state (called intermediate result). Each round transformation is a sequence of four steps. All the rounds are equal, expect for the last one which is a slightly different. Indeed, in order to define a decryption algorithm equivalent to the encryption one, one of the step that composed each round (precisely, the MixColumns operation) is omitted in the last round. The effect of the omission of the last round's MixColumns has been studied in detail e.g. in [20]. Often this omission doesn't affect the security of AES. On the other hand, attacks that exploit Meet-in-the-Middle techniques are not immune to this omission, and in particular they work in a (much) better way when all the rounds are equal (i.e. when last round's MixColumns is not omitted). Since the main technique exploited in [10] and by the tool described in [11] is the Meet-in-the-Middle one, the complexity of these attacks is in general higher if the MixColumns operation is omitted in the last round than if it is not omitted. Moreover, for these attacks no result is reported in the case in which the final MixColumns operation is omitted for 4 or more rounds. With this in mind, we'd like to highlight that our attack works exactly in the same way both if the MixColumns operation is omitted or not in the last round.

The idea of our attack on 3-round AES-128 is very simple. Suppose to fix a coset of a particular subspace $\mathcal{D}$ of the plaintexts space. As we'll show in the first part of the paper, after 2 rounds each elements of a (fixed) coset of $\mathcal{D}$ belongs to a coset of another particular subspace $\mathcal{M}$, that is a coset of $\mathcal{D}$ is mapped into a coset of $\mathcal{M}$ after

Table 2. *Comparison table of low-data attacks on round-reduced AES.* Data complexity is measured in number of required known/chosen plaintexts (KP/CP). Time complexity is measured in round-reduced AES encryption equivalents (E) and in memory accesses (M). Memory complexity is measured in plaintexts (16 bytes). The case in which the MixColumns operation is omitted in the last round is denoted by "r.5 rounds", that is r full rounds and the final round. The attacks of this paper are in bold.

Attack	Rounds	Data	Computation (E)	Memory	Reference
G&D	1.5	1 KP	2^{56}	1	[11]
G&D	2	1 KP	2^{64}	2^{48}	[11]
G&D	1.5	2 KP	2^{24}	2^{16}	[11]
G&D	2	2 KP	2^{32}	2^{24}	[11]
G&D	2	2 CP	2^{8}	2^{8}	[11]
ST	**1.5 - 2**	**3 CP**	$\mathbf{2^{11.8}}$	**1**	**Sect. 6.2**
ST	**1.5 - 2**	**3 CP**	$\mathbf{2^{10}}$ **M** $+\mathbf{2^{6.3}}$ **E** $\approx \mathbf{2^{6.6}}$	$\mathbf{2^{12}}$	**Sect. 6.2**
D	1.5 - 2	3 KP	2^{32}	1	[10]
G&D	2.5	1 KP	2^{88}	2^{88}	[11]
G&D	3	1 KP	2^{96}	2^{96}	[11]
G&D	2.5	2 KP	2^{80}	2^{80}	[11]
G&D	2.5	2 CP	2^{24}	2^{16}	[11]
D-MitM	3	2 CP	2^{32}	2^{1}	[10]
G&D	3	2 CP	2^{16}	2^{8}	[11]
ST	**2.5 - 3**	**2 CP**	$\mathbf{2^{32}}$ **M** $+\mathbf{2^{31.55}}$ **E** $\approx \mathbf{2^{31.6}}$	$\mathbf{2^{8}}$	**Sect. 6**
ST	**2.5 - 3**	**3 CP**	$\mathbf{2^{11.2}}$	**1**	**Sect. 6**
ST	**2.5 - 3**	**3 CP**	$\mathbf{2^{10}}$ **M** $+\mathbf{2^{5.1}}$ **E** $\approx \mathbf{2^{5.7}}$	$\mathbf{2^{12}}$	**Sect. 6**
D-MitM	3	9 KP	2^{40}	2^{35}	[10]
D-MitM	4	2 CP	2^{104}	1	[10]
G&D	4	2 CP	2^{80}	2^{80}	[11]
ST (EE)	**3.5 - 4**	**2 CP**	$\mathbf{2^{96}}$	**1**	**Sect. 8**
ST (EE)	**3.5 - 4**	**3 CP**	$\mathbf{2^{74.7}}$	**1**	**Sect. 8**
ST (EE)	**3.5 - 4**	**3 CP**	$\mathbf{2^{76}}$ **M** $+\mathbf{2^{64}}$ **E** $\approx \mathbf{2^{69.7}}$	$\mathbf{2^{12}}$	**Sect. 8**
G&D	4	4 CP	2^{32}	2^{24}	[11]
D-MitM	4	5 CP	2^{64}	2^{68}	[10]
I-Pol	3.5 - 4	8 CP	2^{38}	2^{15}	[32]
D-MitM	4	10 CP	2^{40}	2^{43}	[10]
ST (EB)	**3.5 - 4**	**24 CP**	$\mathbf{2^{40.6}}$ **M** $+\mathbf{2^{33.9}}$ **E** $\approx \mathbf{2^{35.1}}$	$\mathbf{2^{17}}$	**Sect. 9**
S	3.5 - 4	2^{9} CP	2^{14}	small	[13]

G&D: Guess & Det., ST: Subspace Trail, D: Diff., D-MitM: Diff. Meet-in-the-Middle, S: Square, I-Pol: Imp. polytopic, EE: Extension at End, EB: Extension at Beginning.

two rounds. Equivalently, this means that if two elements belong to the same coset of $\mathcal{D}$, then after two rounds they belong to the same coset of $\mathcal{M}$ independently by the secret key. Nevertheless, the particular coset of $\mathcal{M}$ in which $\mathcal{D}$ is mapped depends on the secret key.

Our attack on 3 rounds as described in Sect. 6 is based on this property. Consider 3 rounds of AES. Given two ciphertexts (which plaintexts belong to the same coset of $\mathcal{D}$), the right key is one of those such that these two ciphertexts belong to the same coset of $\mathcal{M}$ one round before. Moreover, taking advantage of the particular shape of the subspace $\mathcal{M}$, the computational cost of this attack turns out to be very low.

We highlight the relationship and the major difference among our attack and others present in literature (with particular attention to the differential attack) in Sect. 7. In the next sections, we show how to extend this approach in order to attack 4 rounds. In particular, in Sect. 8 we show how to extend our attack at the end, while in Sect. 9 we show how to extend it at the beginning. The attack on 4 rounds with the extension at the end needs only 2 or 3 chosen plaintexts, and the computational cost is the lowest among the (low-data scenario) attacks currently present in literature in the case in which MixColumns operation is omitted in the last round.

1.6 Practical Verification

Secret-Key Distinguishers. We practically verified the secret-key distinguishers using a C implementation [2] for up to 5 rounds, and we have found that the practical results are consistent with our theory.

Regarding the 5-rounds secret-key distinguishers, we have verified it on a *small scale variant of AES* as described in [12], since the complexity of this distinguisher is too high for a practical verification. We give all the details for this case in Sect. 10.3.

Key-Recovery Attacks. We practically verified the low-data complexity attacks on 1, 2, 3 and 4 rounds using a C implementation [1].

For the 3 rounds attack, one or two pairs of plaintexts (that is two or three different plaintexts) are sufficient to discover the key of the final round, as predicted. Since the attack on 4 rounds described in Sect. 8 has a very high computational cost, we have tested it in a different way, which is explained in detail with the presentation of the attack.

2 Subspace Trails and Distinguishers

In this section, we recall the invariant subspace cryptanalysis of [25,26] (depicted in Fig. 1), and then we introduce the concept of subspace trails (Fig. 2).

Invariant subspace cryptanalysis can be a powerful cryptanalytic tool. Let F denote a round function in an iterative block cipher and assume there exists a coset[1] $V \oplus a$ such that $F(V \oplus a) = V \oplus a'$. Then if the round key K resides in $V \oplus (a \oplus a')$, it follows that $F(V \oplus a) \oplus K = V \oplus a$ and we get an iterative invariant subspace.

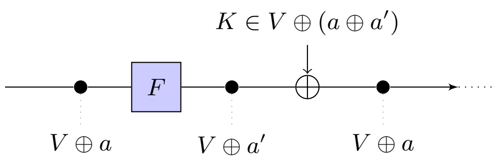

Fig. 1. Invariant subspaces.

A slightly more powerful property can occur if for each a, there exists unique b such that $F_K(V \oplus a) := F(V \oplus a) \oplus K = V \oplus b$ meaning that the subspace property is invariant, but not the initial coset. That is, for each initial coset $V \oplus a$, its image under the application of F_K is another coset of V, in general different from the initial one. Equivalently, the initial coset $V \oplus a$ is mapped into another coset $V \oplus b$, where b depends on a and on the round key. In this paper, we generalize this concept and search

[1] For completeness, we recall the definition of coset, largely used in the paper. Let W a vector space and V a subspace of W. A *coset* of V in W is a subset of the form $V \oplus a = \{v \oplus a \,|\, \forall v \in V\}$.

for trails of subspaces. In the simplest case we look for pairs of subspaces V_1 and V_2 such that

$$F(V_1 \oplus a) \oplus K = V_2 \oplus b$$

holds for any constant a, that is for each a there exists unique b for which the previous equivalence is satisfied.

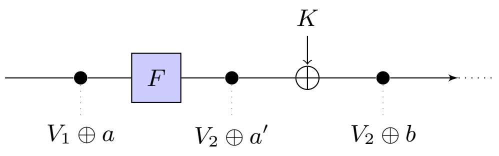

Fig. 2. Trail of subspaces.

A *subspace trail* of length r is then simply a set of $r+1$ subspaces $(V_1, V_2, \ldots, V_{r+1})$ that satisfy

$$F(V_i \oplus a_i) \oplus K \subseteq V_{i+1} \oplus a_{i+1}.$$

When the relation holds with equality, the trail is called a *constant-dimensional* subspace trail. In this case, if we let F_K^t denote the application of t rounds with fixed keys, it means that

$$F_K^t(V_1 \oplus a_1) = V_{t+1} \oplus a_{t+1}.$$

Definition 1. *Let* $(V_1, V_2, ..., V_{r+1})$ *denote a set of* $r+1$ *subspaces with* $\dim(V_i) \leq \dim(V_{i+1})$. *If for each* $i = 1, ..., r$ *and for each* $a_i \in V_i^{\perp}$, *there exist (unique)* $a_{i+1} \in V_{i+1}^{\perp}$ *such that*

$$F_K(V_i \oplus a_i) \subseteq V_{i+1} \oplus a_{i+1},$$

then $(V_1, V_2, ..., V_{r+1})$ *is subspace trail of length* r *for the function* F_K. *If all the previous relations hold with equality, the trail is called a constant-dimensional subspace trail.*

Note that a_{i+1} depends on a_i and on the secret round key. With the aim to simplify the notation, we use simply a_{i+1} instead of $a_{i+1}(a_i, k)$.

With subspace structures at hand, we might ask questions about the probability that ciphertexts or sums of ciphertexts reside in certain subspaces, given that the plaintexts obey certain subspace structure (e.g. their sum is also in a fixed subspace). If the sum is over two texts this approaches resembles (truncated) differential cryptanalysis, if the sum is over more it can resemble integral cryptanalysis.

For AES-type block ciphers, we are typically not able to construct very long trails. In this case we can connect trails together and depending on the intersection properties of the endpoints of the trails, get predictable subspace properties for longer trails. However, in general these are not necessarily simple constant dimensional trails. In the following we describe subspace trail cryptanalysis and later on distinguishers based on it. For sake of concreteness and better exposition, we focus on the case of AES. We'd like to emphasize that the properties described here extend almost immediately to any AES-like cipher with little modifications.

Before we continue, we give the following definition of *equivalence cosets* of a generic subspace X:

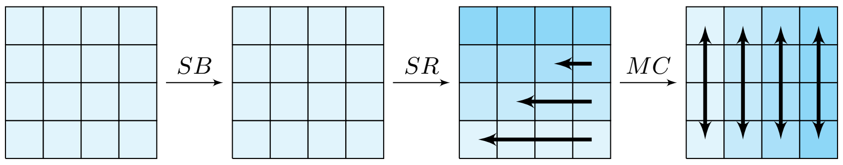

Fig. 3. The essential structure of an AES round.

Definition 2. *Let X a be generic subspace, and let $X \oplus a$ and $X \oplus b$ be two different cosets of X (that is $a \neq b$). We say that they are equivalent under an "equivalence relationship" (that is $X \oplus a \sim X \oplus b$) if and only if $a \oplus b \in X$:*

$$X \oplus a \sim X \oplus b \qquad \textit{if and only if} \qquad a \oplus b \in X.$$

3 Preliminaries - Description of AES

The Advanced Encryption Standard [14] is a *Substitution-Permutation network* that supports key size of 128, 192 and 256 bits. The 128-bit plaintext initializes the internal state as a 4×4 matrix of bytes as values in the finite fields $\mathbb{F}_{256}$, defined using the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. Depending on the version of AES, N_r round are applied to the state: $N_r = 10$ for AES-128, $N_r = 12$ for AES-192 and $N_r = 14$ for AES-256. An AES round applies four operations to the state matrix:

- *SubBytes* (S-Box) - applying the same 8-bit to 8-bit invertible S-Box 16 times in parallel on each byte of the state (it provides the non-linearity in the cipher);
- *ShiftRows* (SR) - cyclic shift of each row (i-th row is shifted by i bytes to the left);
- *MixColumns* (MC) - multiplication of each column by a constant 4×4 invertible matrix over the field $GF(2^8)$ (it and ShiftRows provide diffusion in the cipher[2]);
- *AddRoundKey* (ARK) - XORing the state with a 128-bit subkey.

One round of AES can be described as $R(x) = K \oplus MC \circ SR \circ$ S-Box(x). In the first round an additional AddRoundKey operation (using a whitening key) is applied, and in the last round the MixColumns operation is omitted.

As we consider only AES with 128-bit key, we shall describe only its key schedule algorithm. The key schedule of AES-128 takes the user key and transforms it into 11 subkeys of 128 bits each. The subkey array is denoted by $W[0, ..., 43]$, where each word of $W[\cdot]$ consists of 32 bits and where the first 4 words of $W[\cdot]$ are loaded with the user secret key. The remaining words of $W[\cdot]$ are updated according to the following rule:

- if $i \equiv 0 \bmod 4$, then $W[i] = W[i-4] \oplus RotByte(\text{S-Box}(W[i-1])) \oplus RCON[i/4]$,
- otherwise, $W[i] = W[i-1] \oplus W[i-4]$,

where $i = 4, ..., 43$, *RotByte* rotates the word by 8 bits to the left and $RCON[\cdot]$ is an array of predetermined constant.

The Notation Used in the Paper Let x denote a plaintext, a ciphertext, an intermediate state or a key. Then $x_{i,j}$ with $i, j \in \{0, ..., 3\}$ denotes the byte in the row i and in the column j. We denote by k^r the key of the r-th round, where k^0 is the secret key. If only the key of the final round is used, then we denote it by k to simplify the notation.

[2] ShiftRows makes sure column values are spread and MixColumns makes sure each column is mixed.

Finally, we denote by R one round of AES[3], while we denote i rounds of AES by $R^{(i)}$. If the MixColumns operation is omitted in the last round, then we denote it by R_f. As last thing, in the paper we often use the term "*collision*" when two texts belong to the same coset of a given subspace X.

3.1 Subspaces through 1-Round of AES

For a vector space V and a function F on $\mathbb{F}_{2^8}^{4\times 4}$, let $F(V) = \{F(v) \,|\, v \in V\}$ (as usual). For a subset $I \subseteq \{1, 2, \ldots, n\}$ and a subset of vector spaces $\{G_1, G_2, \ldots, G_n\}$, we define G_I as $G_I := \bigoplus_{i\in I} G_i$.

In the following we define three families of subspaces essential to AES; the diagonal spaces $\mathcal{D}_I$, the column spaces $\mathcal{C}_I$ and the mixed spaces $\mathcal{M}_I$. Since AES operates on 4×4 matrices over $\mathbb{F}_{2^8}$, then we work with vectors and vector spaces over $\mathbb{F}_{2^8}^{4\times 4}$ (that is, all the subspaces considered in the paper are subspace over $\mathbb{F}_{2^8}^{4\times 4}$). Moreover, we denote with $E = \{e_{0,0}, ..., e_{3,3}\}$ the unit vectors of $\mathbb{F}_{2^8}^{4\times 4}$ (e.g. $e_{i,j}$ has a single 1 in row i and column j).

Definition 3. ***(Diagonal spaces)*** *The diagonal spaces $\mathcal{D}_i$ are defined as*

$$\mathcal{D}_i = \langle e_{0,i}, e_{1,i+1}, e_{2,i+2}, e_{3,i+3}\rangle$$

where the index $i + j$ is computed modulo 4. For instance, the diagonal space $\mathcal{D}_0$ corresponds to the symbolic matrix

$$\mathcal{D}_0 = \left\{ \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & x_4 \end{bmatrix} \middle| \forall x_1, x_2, x_3, x_4 \in \mathbb{F}_{2^8} \right\}.$$

Definition 4. ***(Column spaces)*** *The column spaces $\mathcal{C}_i$ are defined as*

$$\mathcal{C}_i = \langle e_{0,i}, e_{1,i}, e_{2,i}, e_{3,i}\rangle.$$

For instance, the column space $\mathcal{C}_0$ corresponds to the symbolic matrix

$$\mathcal{C}_0 = \left\{ \begin{bmatrix} x_1 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 \end{bmatrix} \middle| \forall x_1, x_2, x_3, x_4 \in \mathbb{F}_{2^8} \right\}.$$

The last type of subspaces we define are called mixed subspaces.

Definition 5. ***(Mixed spaces)*** *The i-th mixed subspace $\mathcal{M}_i$ is defined as*

$$\mathcal{M}_i = MC \circ SR(\mathcal{C}_i).$$

These subspaces are formed by applying ShiftRows and then MixColumns to a column space. For instance, $\mathcal{M}_0$ corresponds to symbolic matrix

$$\mathcal{M}_0 = \left\{ \begin{bmatrix} \alpha \cdot x_1 & x_4 & x_3 & (\alpha+1)\cdot x_2 \\ x_1 & x_4 & (\alpha+1)\cdot x_3 & \alpha \cdot x_2 \\ x_1 & (\alpha+1)\cdot x_4 & \alpha\cdot x_3 & x_2 \\ (\alpha+1)\cdot x_1 & \alpha\cdot x_4 & x_3 & x_2 \end{bmatrix} \middle| \forall x_1, x_2, x_3, x_4 \in \mathbb{F}_{2^8} \right\}.$$

where $0x02 \equiv \alpha$ and $0x03 \equiv \alpha + 1$.

[3] Sometimes we use the notation R_K instead of R to highlight that the round key is K.

Definition 6. *Given $I \subseteq \{0,1,2,3\}$ where $0 < |I| \leq 3$, we define:*

$$\mathcal{C}_I = \bigoplus_{i \in I} \mathcal{C}_i, \qquad \mathcal{D}_I = \bigoplus_{i \in I} \mathcal{D}_i, \qquad \mathcal{M}_I = \bigoplus_{i \in I} \mathcal{M}_i.$$

The dimension of any of the spaces $\mathcal{D}_I$,$\mathcal{C}_I$ and $\mathcal{M}_I$ is $4 \cdot |I|$. The essential subspaces in AES are built from diagonal spaces $\mathcal{D}_i$, column spaces $\mathcal{C}_j$ and mixed spaces $\mathcal{M}_k$. There are four of each of these spaces, and direct sums of subsets of these result in higher-dimensional diagonal, column and mixed spaces.

It is easy to see that SubBytes maps cosets of diagonal and column spaces to cosets of diagonal and column spaces. Since SubBytes operates on each byte individually and it is bijective, and since the bytes of column and diagonal spaces are independent, its only effect is to change the coset. It is also easy to see that ShiftRows maps a coset of a diagonal space to a coset of a column space, since diagonals are mapped to columns. The effect of MixColumns to a columns space $\mathcal{C}_I \oplus a$ is simply to change the coset, since applying the MixColumns matrix to a column space $\mathcal{C}_i$ has no effect.

Lemma 1. *Let $I \subseteq \{0,1,2,3\}$ where $0 < |I| \leq 3$ and $a \in \mathcal{D}_I^{\perp}$. There exists unique $b \in \mathcal{C}_I^{\perp}$ such that*

$$R_K(\mathcal{D}_I \oplus a) = \mathcal{C}_I \oplus b.$$

Note that b is unique with respect to the equivalence relationship defined before (analogous in the following).

Proof. As we have just seen, since SubBytes is bijective and operates on each byte independently, it simply changes the coset $\mathcal{D}_I \oplus a$ to $\mathcal{D}_I \oplus a'$, where $a'_{i,j} = \text{S-Box}(a_{i,j})$ for each $i, j = 0, ..., 3$. ShiftRows simply moves the bytes of $\mathcal{D}_I \oplus a'$ to a column space $\mathcal{C}_I \oplus b'$, where $b' = SR(a')$. MixColumns affects only the constant columns, thus $MC(\mathcal{C}_I \oplus b') = \mathcal{C}_I \oplus MC(b') = \mathcal{C}_I \oplus b''$. Key addition then changes the coset to $\mathcal{C}_I \oplus b$. □

Lemma 2. *Let $I \subseteq \{0,1,2,3\}$ where $0 < |I| \leq 3$ and $a \in \mathcal{C}_I^{\perp}$. There exists unique $b \in \mathcal{M}_I^{\perp}$ such that*

$$R_K(\mathcal{C}_I \oplus a) = \mathcal{M}_I \oplus b.$$

Proof. By definition 5, the mixed spaces $\mathcal{M}_I$ are defined as the application of the linear layer in AES to column spaces $\mathcal{C}_I$. Since the SubBytes layer only moves a coset $\mathcal{C}_I \oplus a$ to a coset $\mathcal{C}_I \oplus a'$, it follows that for any fixed coset $\mathcal{C}_I \oplus a$, there exists $b \in \mathcal{M}_I^{\perp}$ such that $MC \circ SR \circ SB(\mathcal{C}_I \oplus a) \oplus K = \mathcal{M}_I \oplus b$, where $b = MC \circ SR(a') \oplus K$ and $a'_{i,j} = \text{S-Box}(a_{i,j})$ for each $i, j = 0, ..., 3$. □

This simply states that a coset of a sum of diagonal spaces $\mathcal{D}_I$ encrypt to a coset of a corresponding sum of column spaces $\mathcal{C}_I$ through one round. Similarly, a coset of a sum of column spaces $\mathcal{C}_I$ encrypts to a coset of the corresponding sum of mixed spaces $\mathcal{M}_I$ over one round.

4 Intersecting AES Subspaces

We continue with useful properties of AES subspaces. In this section we show the following: diagonal spaces and column spaces have non-trivial intersection, column spaces and mixed spaces have non-trivial intersection, but diagonal spaces and mixed spaces have only trivial intersection. This will be useful for creating subspace trails covering a higher number of rounds.

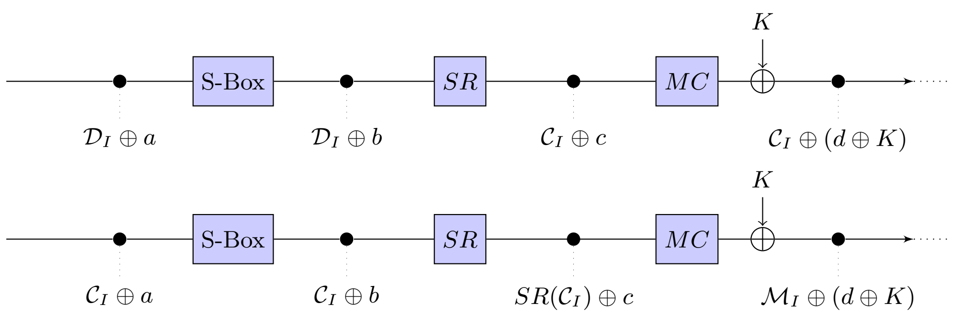

Fig. 4. The essential subspaces in the AES round.

Lemma 3. $\mathcal{D}_i \cap \mathcal{C}_j = \langle e_{j+i,j} \rangle$.

Proof. $\mathcal{D}_i$ space corresponds to a symbolic matrix with variables along the i-th diagonal, while $\mathcal{C}_j$ has variables in the j-th column. Any diagonal and column meets in exactly one byte, precisely in row $j+i$ and column j. □

It follows that $\mathcal{D}_I \cap \mathcal{C}_J = \langle e_{j+i,j} \,|\, i \in I, j \in J \rangle$ where $j+i$ is taken modulo 4. In particular, the intersection has dimension $|I| \cdot |J|$.

Lemma 4. $\mathcal{C}_i \cap \mathcal{M}_j = MC \circ SR(\mathcal{D}_i \cap \mathcal{C}_j) = \langle MC(e_{j+i,i}) \rangle$.

Proof. We have that $MC \circ SR(\mathcal{D}_i) = \mathcal{C}_i$ and by definition 5, $\mathcal{M}_i = MC \circ SR(\mathcal{C}_i)$. By Lemma 3, $\mathcal{D}_i \cap \mathcal{C}_j = \langle e_{j+i,j} \rangle$. Thus it follows that $\langle MC(e_{j+i,j}) \rangle = MC \circ SR(\mathcal{D}_i) \cap MC \circ SR(\mathcal{C}_j) = \mathcal{D}_i \cap \mathcal{M}_j$. Finally, since $SR(e_{r,c}) = e_{r,c-r}$, we obtain that $\langle MC \circ SR(e_{j+i,j}) \rangle = \langle MC(e_{j+i,i}) \rangle$. □

Thus, for two subspaces $\mathcal{C}_I$ and $\mathcal{M}_J$ for non-empty subsets I and J of $\{0,1,2,3\}$, it follows that $\mathcal{C}_I \cap \mathcal{M}_J = \langle MC(e_{j+i,i}) \,|\, i \in I, j \in J \rangle$ (where $i+j$ is taken modulo 4) which has dimension $|I| \cdot |J|$. While the spaces $\mathcal{D}_I$ and $\mathcal{C}_J$, and $\mathcal{C}_I$ and $\mathcal{M}_J$ intersect non-trivially, the spaces $\mathcal{D}_I$ and $\mathcal{M}_J$ intersect trivially.

Lemma 5. $\mathcal{D}_i \cap \mathcal{M}_j = \{0\}$ *for all* i *and* j.

Proof. A basis for $\mathcal{M}_j$ is given by:

$$\mathcal{M}_j = \langle MC(e_{0,j}), MC(e_{1,j-1}), MC(e_{2,j-2}), MC(e_{3,j-3}) \rangle,$$

while a basis for $\mathcal{D}_i$ is given by $\mathcal{D}_i = \langle \langle e_{0,i}, e_{1,i+1}, e_{2,i+2}, e_{3,i+3} \rangle$, where in both cases the indexes are taken modulo 4.

Suppose by contradiction that $\mathcal{D}_i$ and $\mathcal{M}_j$ has a nonzero intersection. This implies that there exist x_k and y_k for $k = 0, ..., 3$ such that

$$\bigoplus_{k=0}^{3} x_k \cdot \langle e_{k,i+k} \rangle \oplus \bigoplus_{k=0}^{3} y_k \cdot \langle MC(e_{k,j-k}) \rangle =$$
$$= \bigoplus_{k=0}^{3} \Big[x_{k-i} \cdot \langle e_{k-i,k} \rangle \oplus y_{k+j} \cdot \langle MC(e_{k+j,k}) \rangle \Big] = 0. \tag{1}$$

has a nontrivial solution. This is clearly impossible since $\langle e_{k-i,k} \rangle$ and $\langle MC(e_{k+j,k}) \rangle$ are linearly independent for each $k = 0, ..., 3$. Thus, $\mathcal{D}_i$ and $\mathcal{M}_j$ intersect only in zero. □

As long as $|I| + |J| \leq 4$, we have that any combinations of subspaces $\mathcal{D}_I$ and $\mathcal{M}_J$ only intersect in the zero vector. Indeed, consider the sum over k defined in eq. (1). If $|I| + |J| \leq 4$, then for each k (i.e. for each column) there are at most four terms. Among them, there is at least one term of the form $\langle e_{\cdot,k} \rangle$ and at least one of the form $\langle MC(e_{\cdot,k}) \rangle$. Thus, equation (1) has only trivial solutions. Instead, note that this is not true if $|I| + |J| > 4$. Indeed, in this case for each k (i.e. for each column), the equation (1) has at least 5 terms. Since there are only 4 rows, it is always possible to find non trivial solutions[4].

Lemma 6. *$\mathcal{D}_I \cap \mathcal{M}_J = \{0\}$ for all I and J such that $|I| + |J| \leq 4$.*

5 Subspace Distinguishers for AES with Secret Round-Keys

In this section we describe a series of subspace trails for AES. Additionally we also describe how these trails can be used to formulate ways to detect non-randomness, often colloquially referred to a distinguishers. All distinguishers in this section, ranging from two up to four rounds, are independent of the round keys and are formulated without the knowledge of the key. From now on, we assume that any subspaces $\mathcal{D}_I$, $\mathcal{C}_I$ or $\mathcal{M}_I$ has nonzero dimension (that is, $I \subseteq \{0, 1, 2, 3\}$ is not empty). Moreover, when we intersect two subspaces $\mathcal{D}_I$ and $\mathcal{M}_J$, where both I and J are assumed non-empty, we always assume that the sum of their dimensions is not larger than 16. Typically, the sum of their dimensions will be exactly 16.

5.1 2-Rounds Subspace Distinguisher for AES

It follows directly from Section 3.1 that plaintexts from diagonal spaces are encrypted over two rounds to ciphertexts in mixed subspaces. Let $R^{(2)}$ denote two AES rounds with fixed random round keys $K = K_1, K_2$. Let $I \subseteq \{1, 2, 3, 4\}$ nonzero and fixed. By Lemma 1, a coset $\mathcal{D}_I \oplus a$ of dimension $4 \cdot |I|$ encrypts to a coset $R_{K_1}(\mathcal{D}_I \oplus a) = \mathcal{C}_I \oplus a'$ over one round. By Lemma 2, there exists unique b (relative to the round keys and the constant a') such that $R_{K_2}(\mathcal{C}_I \oplus a') = \mathcal{M}_I \oplus b$. By combining the two rounds, we get that for each $a \in \mathcal{D}_I^\perp$, there exists unique $b \in \mathcal{M}_I^\perp$ such that $R^{(2)}(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b$.

Consequently, we get the following properties. If two plaintexts belong to the same coset of a diagonal space $\mathcal{D}_I$, then their encryption belongs to the same coset of a mixed space $\mathcal{M}_I$. In particular, for a two round encryption R^2 with fixed keys, we have that

$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_I \,|\, u \oplus v \in \mathcal{D}_I) = 1 \qquad (2)$$

for nonzero set I of $\{0, 1, 2, 3\}$ (i.e. $|I| \neq 0$) and where $u \neq v$. The opposite follows directly; if two plaintexts belong to different cosets of a diagonal space $\mathcal{D}_I$, then their encryption belongs to different cosets of a mixed space $\mathcal{W}_I$. In other words,

$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_I \,|\, u \oplus v \notin \mathcal{D}_I) = 0.$$

[4] For example, the first column (i.e. $k = 0$) of the intersection $\mathcal{D}_{0,1,2} \cap \mathcal{M}_{0,1}$ is equal to:

$$(\mathcal{D}_{0,1,2} \cap \mathcal{M}_{0,1})_{col(0)} \equiv MC\left(\begin{bmatrix} x \\ (\alpha+1) \cdot x \\ 0 \\ 0 \end{bmatrix} \right) \equiv \begin{bmatrix} (\alpha^2+\alpha+1) \cdot x \\ (\alpha^2+\alpha+1) \cdot x \\ \alpha \cdot x \\ 0 \end{bmatrix} \qquad \forall x \in \mathbb{F}_{2^8}.$$

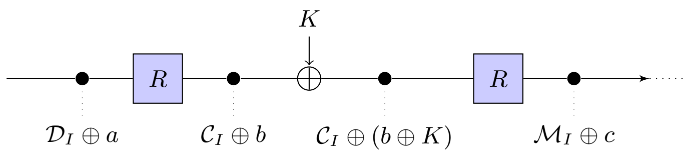

Fig. 5. Subspaces over two rounds.

These properties are used to set up the distinguisher for two rounds. However, other interesting properties hold when one considers two rounds of encryption. In particular, by Lemma 6, the intersection between a mixed space $\mathcal{M}_I$ space and a diagonal space $\mathcal{D}_J$ space contains only zero, if $|I| + |J|$ is less than 4. Thus, if two plaintexts are in the same coset of $\mathcal{M}_I$, they must belong to different cosets of $\mathcal{D}_J$. In other words, for $\mathcal{D}_I$ and $\mathcal{D}_J$ such that $\dim(\mathcal{D}_I) + \dim(\mathcal{D}_J) \leq 16$ (and $|I|, |J| \neq 0$)

$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{D}_J \,|\, u \oplus v \in \mathcal{D}_I) = 0 \tag{3}$$

where $u \neq v$, since $R^{(2)}(u)$ and $R^{(2)}(v)$ are both in the same coset of $\mathcal{M}_I$ and thus are always in different cosets of $\mathcal{D}_J$. We can get similar results for the mixed spaces $\mathcal{M}_I$. In particular, if two plaintexts belong to the same coset of a mixed space $\mathcal{M}_I$, then their two round encryptions belong to different cosets of any mixed space $\mathcal{M}_J$. Indeed, two (different) elements of $\mathcal{M}_I$ belong to different cosets of $\mathcal{D}_J$ (since $\mathcal{M}_I \cap \mathcal{D}_J = \{0\}$). Since $R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_J$ if and only if $u \oplus v \in \mathcal{D}_J$, we obtain the desired result. Thus, for $\mathcal{M}_I$ and $\mathcal{M}_J$ such that $0 < \dim(\mathcal{M}_I) + \dim(\mathcal{M}_J) \leq 16$, we have that

$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_J \,|\, u \oplus v \in \mathcal{M}_I) = 0 \tag{4}$$

if $u \neq v$. We'll use these probabilities to set up an efficient 4 rounds distinguisher.

A Concrete Distinguisher for 2 Rounds. As we have seen, if two plaintexts belong to the same coset of $\mathcal{D}_I$, then they belong to the same coset of $\mathcal{M}_I$ with probability 1 after two rounds - for each I. Consider instead two random texts. By simple computation, the probability that there exists I such that they belong to the same cosets of $\mathcal{M}_I$ is $\binom{4}{|I|} \cdot (2^8)^{-16+4\cdot|I|}$ (note that there are $\binom{4}{|I|}$ different subspaces $\mathcal{M}_I$). Setting $|I| = 1$, this probability is equal to 2^{-94}.

Thus, one pair of plaintexts (that is 2 texts) is sufficient to distinguish the random case from the other one. Indeed, on average in the random case we expect $2^{-94} \cdot 2 = 2^{-93} \simeq 0$ *collisions* (when two elements belong to the same coset of $\mathcal{M}_I$, we say that there is a "collision"), while this number is equal to 1 (with probability 1) in the other case. The cost of this distinguisher is hence 2 texts. An equivalent distinguisher over 2 rounds was already introduced in [16], where authors investigated how the components of the AES interact over 2 rounds.

Finally, note that a similar distinguisher can be used for the 1 round case. Indeed, note that if two plaintexts belong to the same coset of $\mathcal{D}_I$ (equivalently $\mathcal{C}_I$), then they belong to the same coset of $\mathcal{C}_I$ (equivalently $\mathcal{M}_I$) with probability 1 for each I after 1 round. Moreover, observe that it also is possible to set up a 2 rounds distinguisher using the impossible differential properties defined in (3) or (4).

5.2 3-Round Subspace Distinguisher for AES

To form a three round distinguisher, we extend a two round distinguisher to three rounds. The following theorem describes the essential step for the extension.

```
Data: Pair of texts $c^1$ and $c^2$.
Result: $i$ such that $c^1 \oplus c^2 \in \mathcal{M}_i$, $-1$ otherwise.
$c \leftarrow MC^{-1}(c^1 \oplus c^2)$;
for $i$ from 0 to 3 do
    if $c_{(i+1)\%4,0} = 0$ AND $c_{(i+2)\%4,0} = 0$ AND $c_{(i+3)\%4,0} = 0$
      AND $c_{i,1} = 0$ AND $c_{(i+1)\%4,1} = 0$ AND $c_{(i+2)\%4,1} = 0$
      AND $c_{i,2} = 0$ AND $c_{(i+1)\%4,2} = 0$ AND $c_{(i+3)\%4,2} = 0$
      AND $c_{i,3} = 0$ AND $c_{(i+2)\%4,3} = 0$ AND $c_{(i+3)\%4,3} = 0$ then
        return $i$;
    end
end
return $-1$.
```

Algorithm 1: Distinguisher for 2-rounds of AES - Pseudo-code.

Theorem 1. *For any $\mathcal{M}_I$ and $\mathcal{M}_J$, we have that*

$$Pr(R(u) \oplus R(v) \in \mathcal{M}_J \,|\, u \oplus v \in \mathcal{M}_I) = (2^8)^{-4|I|+|I|\cdot|J|}.$$

Proof. In the previous section, we have seen that $R(x) \oplus R(y) \in \mathcal{M}_J$ if and only if $x \oplus y \in \mathcal{C}_J$. This implies that the probability given in the Theorem is equivalent to the following:

$$Pr(R(x) \oplus R(y) \in \mathcal{M}_J \,|\, x \oplus y \in \mathcal{M}_I) = Pr(x \oplus y \in \mathcal{C}_J \,|\, x \oplus y \in \mathcal{M}_I) = \\ = Pr(z \in \mathcal{C}_J \,|\, z \in \mathcal{M}_I).$$

Let $\mathcal{Z} = \mathcal{M}_I \cap \mathcal{C}_J$. In Section 4, it is shown that $\dim(\mathcal{Z}) = \dim(\mathcal{M}_I \cap \mathcal{C}_J) = |I| \cdot |J|$. Let $\mathcal{Y}$ the subspace of dimension $4 \cdot |I| - |I| \cdot |J|$ such that $\mathcal{M}_I = \mathcal{Y} \oplus \mathcal{Z}$, and let $\pi_\mathcal{Y}$ and $\pi_\mathcal{Z}$ the projection of $\mathcal{M}_I$ on $\mathcal{Y}$ and $\mathcal{Z}$ respectively:

$$\pi_\mathcal{Y} : \mathcal{M}_I \to \mathcal{Y}, \qquad \pi_\mathcal{Y}(x) = x_y,$$
$$\pi_\mathcal{Z} : \mathcal{M}_I \to \mathcal{Z}, \qquad \pi_\mathcal{Z}(x) = x_z.$$

That is, $\forall x \in \mathcal{M}_I$, there exists unique $x_y \in \mathcal{Y}$ and $x_z \in \mathcal{Z}$ such that $x = x_z \oplus x_y$.

It follows that:

$$Pr(x \in \mathcal{C}_J \,|\, x \in \mathcal{M}_I) = Pr(\pi_\mathcal{Y}(z) = 0 \,|\, z \in \mathcal{M}_I).$$

Since $\mathcal{Y}$ has dimension $4 \cdot |I| - |I| \cdot |J|$, we obtain:

$$Pr(R(x) \oplus R(y) \in \mathcal{M}_J \,|\, x \oplus y \in \mathcal{M}_I) = Pr(\pi_\mathcal{Y}(z) = 0 \,|\, z \in \mathcal{M}_I) = (2^8)^{-4\cdot|I|+|I|\cdot|J|}.$$

□

Note that if $|J| = 4$ (i.e. if $\mathcal{M}_J$ is all the space), then the probability is equal to 1.

Let $c \in \mathcal{W}_I^\perp$. Given $\mathcal{Z} := \mathcal{M}_I \cap \mathcal{C}_J$ and $\mathcal{Y} := \mathcal{M}_I \setminus \mathcal{Z}$, then

$$\mathcal{M}_I = \mathcal{Z} \oplus \mathcal{Y} = \bigcup_{a \in \mathcal{Y}} \mathcal{Z} \oplus a \subseteq \bigcup_{a \in \mathcal{Y}} \mathcal{C}_J \oplus a = \mathcal{C}_J \oplus \mathcal{Y}, \qquad \text{and}$$

$$\mathcal{M}_I \oplus c = \bigcup_{a'_i \in \mathcal{C}_J \setminus \mathcal{Z}} \mathcal{Z} \oplus (a'_i \oplus c) = \bigcup_{i=1}^{(2^8)^{4\cdot|I|-|I|\cdot|J|}} \mathcal{Z} \oplus a_i,$$

where $\mathcal{Z} \oplus a_i = (\mathcal{M}_I \cap \mathcal{C}_J) \oplus a_i$ are cosets of dimension $|I| \cdot |J|$.

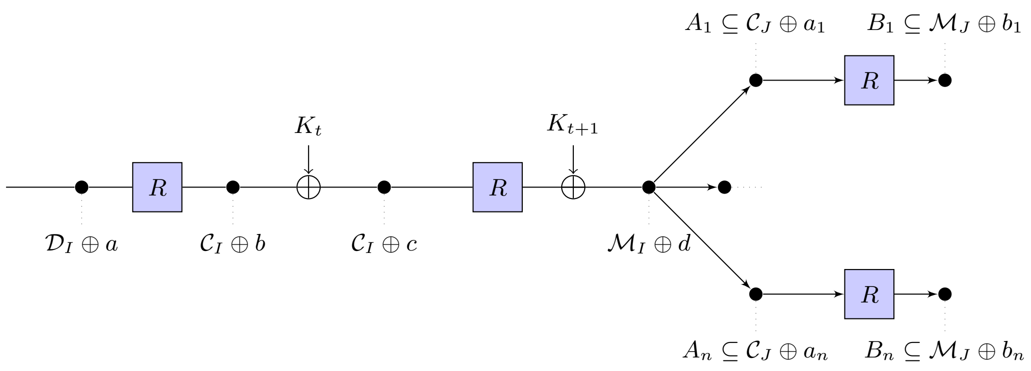
$A_1 \subseteq \mathcal{C}_J \oplus a_1$
$B_1 \subseteq \mathcal{M}_J \oplus b_1$
R
K_t
K_{t+1}
R
R
$\mathcal{D}_I \oplus a$
$\mathcal{C}_I \oplus b$
$\mathcal{C}_I \oplus c$
$\mathcal{M}_I \oplus d$
R
$A_n \subseteq \mathcal{C}_J \oplus a_n$
$B_n \subseteq \mathcal{M}_J \oplus b_n$

Fig. 6. 3-round distinguishers for AES (the index n is defined as $n := (2^8)^{4\cdot|I| - |I|\cdot|J|}$).

Let $A_i := \mathcal{Z} \oplus a_i$. Since cosets of $\mathcal{C}_J$ spaces encrypts to cosets of $\mathcal{M}_J$ spaces, we get that

$$B_i := R(A_i) \subseteq R(\mathcal{C}_J \oplus a_i) = \mathcal{M}_J \oplus b_i,$$

since $A_i = \mathcal{Z} \oplus a_i \subseteq \mathcal{C}_J \oplus a_i$ by definition of $\mathcal{Z}$. As a consequence:

$$R^{(3)}(\mathcal{D}_I \oplus a) = R(\mathcal{M}_I \oplus c) = R\left(\bigcup_{i=1}^{n} A_i\right) = \bigcup_{i=1}^{n} R(A_i) = \bigcup_{i=1}^{n} B_i \subseteq \bigcup_{i=1}^{n} \mathcal{M}_J \oplus b_i,$$

where $n := (2^8)^{4\cdot|I| - |I|\cdot|J|}$, R is the application of one round with a fixed key and $R^{(3)}$ the application of three rounds.

For instance, consider a coset of $\mathcal{D}_I$. After two rounds, each element belongs to a coset of $\mathcal{M}_I$. Equivalently, for a given J, after two rounds the texts are *uniform* distributed in $(2^8)^{4\cdot|I| - |I|\cdot|J|}$ cosets of $\mathcal{C}_J$. That is, after two rounds, there exist $(2^8)^{4\cdot|I| - |I|\cdot|J|}$ cosets of $\mathcal{C}_J$ such that each one of these cosets contains exactly $(2^8)^{|I|\cdot|J|}$ elements. Since each coset of $\mathcal{C}_J$ encrypts to a unique coset of $\mathcal{M}_J$ (that is, two elements that belong to different coset of $\mathcal{C}_J$ can not belong to the same coset of $\mathcal{M}_J$), if we start with a coset of $\mathcal{D}_I$, after three rounds the texts are *uniform* distributed in $(2^8)^{4\cdot|I| - |I|\cdot|J|}$ cosets of $\mathcal{M}_J$.

In order to better understand it, we give an example for the particular case in which $\mathcal{D}_I = \mathcal{D}_i$ is of dimension 1 and $\mathcal{M}_J = \mathcal{M}_{i_1} \oplus \mathcal{M}_{i_2} \oplus \mathcal{M}_{i_3}$ of dimension 12. So, after 2 rounds, the texts are uniform distributed in 2^8 cosets of $\mathcal{C}_J$, i.e. there exist 2^8 cosets of $\mathcal{C}_J$ such that each one of them contains exactly 2^{24} texts. Note that the remaining $2^{32} - 2^8$ cosets of $\mathcal{C}_J$ don't contain any texts. Thus, after 3 rounds the texts are uniform distributed in 2^8 cosets of $\mathcal{M}_J$, i.e. there exist 2^8 cosets of $\mathcal{M}_J$ such that each one of them contains exactly 2^{24} texts, since (as we have seen) each coset of $\mathcal{C}_J$ is mapped in exactly one coset of $\mathcal{M}_J$, while the remaining $2^{32} - 2^8$ cosets of $\mathcal{C}_J$ don't contain any texts.

A Concrete Distinguisher for 3 Rounds. In order to set up the distinguisher, we exploit the difference of probability to have a collision in the ciphertexts set between the case in which two plaintexts are taken in a random way and the case in which two plaintexts belong to the same coset of $\mathcal{D}_I$.

The probabilities that two elements drawn randomly from $\mathbb{F}_{2^8}^{4\times 4}$ (denoted by p_1) and that two plaintexts drawn from a coset of $\mathcal{D}_I$ (denoted by p_2) belong to the same coset of $\mathcal{M}_J$ are respectively:

$$p_1 = \binom{4}{|J|} \cdot (2^8)^{-16+4|J|}, \qquad p_2 = \binom{4}{|J|} \cdot (2^8)^{-4|I|+|I||J|}.$$

It is very easy to observe that the probability to have a collision in the second case is higher than in the random case. In particular, for $|J| = 3$ and $|I| = 1$, we obtain that $p_2 = 2^{-6}$ while $p_1 = 2^{-30}$. Thus, the idea is to look for the minimum number of texts m in order to guarantee at least one collision in the "subspace case" and zero in the random case (with high probability).

To do this, we recall the *birthday paradox*. Given d (equally likely) values and n variables, the probability that at least two of them have the same value is given by:

$$p = 1 - \frac{n!}{(n-d)! \cdot n^d} = 1 - \frac{(d)!}{n^d} \cdot \binom{n}{d} \simeq 1 - e^{\frac{-d(d-1)}{2n}},$$

where the last one is an useful approximation.

Since if we encrypt two plaintexts from a coset of $\mathcal{D}_I$, each of them can only belong to one of the 2^8 cosets of $\mathcal{M}_J$ defined as before, the probability that there is at least one collision in a coset is equal to the probability that two elements belong to the same cosets of $\mathcal{M}_J$, that is $p = 1 - e^{-m(m-1)/(2\cdot 2^8)}$. However, this property holds if we choose any of the four 12-dimensional space $\mathcal{M}_J$ as a target distinguisher space, each yielding an independent experiment. Since this experiments are independent, we have that the probability to have at least one collision in the subspace case given m texts is:

$$p = 1 - \left(\frac{2^8!}{(2^8 - m)! \cdot (2^8)^d}\right)^4 \simeq 1 - \left(e^{\frac{-m(m-1)}{2\cdot 2^8}}\right)^4 = 1 - e^{\frac{-m(m-1)}{2\cdot 2^6}}.$$

Thus, if we set $m = 20$, we get that the probability to have at least one collision in one of the four different $\mathcal{M}_J$ spaces (with $|J| = 1$) is 95.251% (14 texts are sufficient to have at least one collision with probability greater than 75%). In order to distinguish the two sets (that is, the random one and the "subspace" one), the verifier has to construct all the possible pairs of texts and to count the number of collisions, for each of them. In particular, given 20 texts (that is, 190 different pairs), we expect $190 \cdot 2^{-6} \simeq 3$ collisions in the subspace case and $190 \cdot 2^{-30} = 2^{-22.4} \simeq 0$ in the random case.

Observe that the distinguisher works in similar way in the decryption direction, with the same complexity.

```
Data: 20 texts $c^i$ (for $i = 1, ..., 20$).
Result: number of collisions.
$n \leftarrow 0$;
for each pair $(c^i, c^j)$ with $i \neq j$ do
    $c \leftarrow MC^{-1}(c^i \oplus c^j)$;
    for $k$ from 0 to 3 do
        if $c_{k,0} = 0$ AND $c_{(3+k)\%4,1} = 0$ AND $c_{(2+k)\%4,2} = 0$ AND $c_{(1+k)\%4,3} = 0$ then
            $n \leftarrow n + 1$;
            next pair
        end
    end
end
return $n$.
```

Algorithm 2: Distinguisher for 3-rounds of AES - Pseudo-code.

Finally, in a very different scenario, an analogous distinguisher (based on truncated differential) was introduced in [7], and showed in Fig. 7. Consider a pair of plaintexts that belong to the same coset of $\mathcal{D}_0$. With probability $2^{-8} \cdot 4 = 2^{-6}$, after one round they

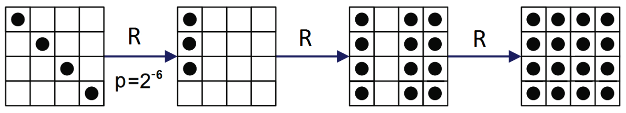

Fig. 7. Truncated differential characteristic over 3-rounds AES. White box denotes a byte with a zero difference, while black box denotes a byte with a non-zero difference.

belong to the same coset of $\mathcal{C}_0 \cap \mathcal{D}_I$, for a certain I with $|I| = 3$. That is, with probability 2^{-6}, after one round only three bytes are active instead of four. Thus, since $\mathcal{C}_0 \cap \mathcal{D}_I \subseteq \mathcal{D}_I$ and since for each $a \in \mathcal{D}_I^{\perp}$ there exists unique $b \in \mathcal{M}_I^{\perp}$ such that $R^{(2)}(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b$, if two texts belong to the same coset of $\mathcal{D}_i$, then they belong to the same coset of $\mathcal{M}_I$ with $|I| = 3$ after three rounds with probability 2^{-6}.

5.3 4-Rounds Subspace Distinguisher for AES

From now on, we assume that I and J satisfy the condition $0 < |I| + |J| \leq 4$ (in order to use Lemma 6). To set up the 4-rounds distinguisher, we start from the 2-rounds one. Fix $\mathcal{D}_I$ and $\mathcal{D}_J$ such that $0 < \dim(\mathcal{D}_I) + \dim(\mathcal{D}_J) \leq 16$. We can construct a four round trail by simply combining two-round subspaces properties. Indeed, we have seen that

$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_I \,|\, u \oplus v \in \mathcal{D}_I) = 1$$
$$Pr(R^{(2)}(u) \oplus R^{(2)}(v) \in \mathcal{M}_J \,|\, u \oplus v \in \mathcal{M}_I) = 0$$

if $u \neq v$. Combining these two probabilities for two-rounds yields a four round probability

$$Pr(R^{(4)}(u) \oplus R^{(4)}(v) \in \mathcal{M}_J \,|\, u \oplus v \in \mathcal{D}_I) = 0 \tag{5}$$

where $u \neq v$. This means that the adversary can pick any coset of a non-zero plaintext space $\mathcal{D}_I$ and a non-zero ciphertext space $\mathcal{M}_J$, as long as $0 < \dim(\mathcal{D}_I) + \dim(\mathcal{M}_J) \leq 16$, and distinguish on the fact that the probability that two plaintexts encrypt to the same coset of the ciphertext space is zero over four rounds.

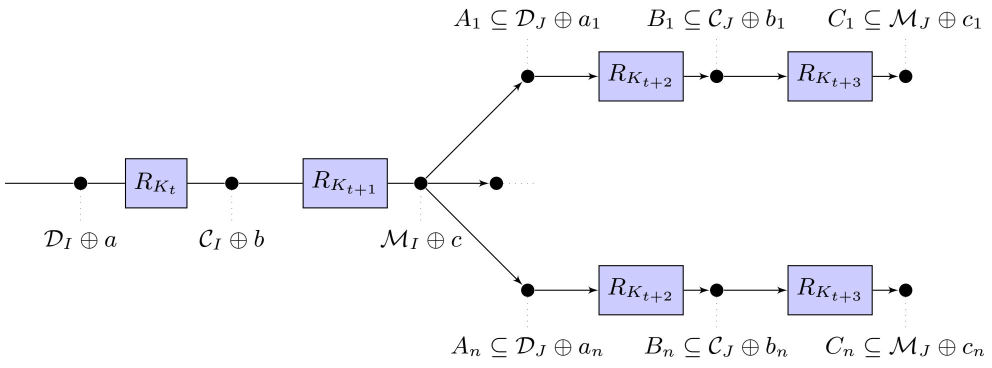
$A_1 \subseteq \mathcal{D}_J \oplus a_1$
$B_1 \subseteq \mathcal{C}_J \oplus b_1$
$C_1 \subseteq \mathcal{M}_J \oplus c_1$
$R_{K_{t+2}}$
$R_{K_{t+3}}$
R_{K_t}
$R_{K_{t+1}}$
$\mathcal{D}_I \oplus a$
$\mathcal{C}_I \oplus b$
$\mathcal{M}_I \oplus c$
$R_{K_{t+2}}$
$R_{K_{t+3}}$
$A_n \subseteq \mathcal{D}_J \oplus a_n$
$B_n \subseteq \mathcal{C}_J \oplus b_n$
$C_n \subseteq \mathcal{M}_J \oplus c_n$

Fig. 8. 4-round distinguishers for AES (where the index n is defined as $n := (2^8)^{4|I|}$ and the indexes I and J satisfy the condition $0 < |I| + |J| \leq 4$).

A Concrete Distinguisher for 4 Rounds. The idea is pick parameters that maximize probability in the random case. The best minimal data complexity is found if we choose $|J| = 3$. This implies that $|I| = 1$, since we have the condition that $|I|+|J| \leq 4$. In this case, the probability that two random elements belong to the same coset of $\mathcal{M}_J$ for a certain J with $|J| = 3$ is 2^{-30} (as we have already seen). Instead, the probability that two elements, that belong to the same coset of $\mathcal{D}_I$, belong to the same coset of $\mathcal{M}_J$ after four rounds is 0.

Exactly as before, the idea is to look for the minimum number of texts m in order to guarantee at least one collision in the random case with high probability. Since there are four 12-dimensional space $\mathcal{M}_J$ and using the birthday paradox, the probability to have at least one collision in the random case given m texts is well approximated by $p = 1 - e^{-m(m-1)/(2\cdot 2^{30})}$. Thus, $m \simeq 2^{16.25}$ texts are sufficient to set up a 4-Rounds distinguisher (in this case, the probability to have a collision in the random case is approximately 95% - note that $2^{15.75}$ texts are sufficient to have at least one collision with probability of 75%). Indeed, given $2^{16.25}$ texts (that is about $2^{31.5}$ pairs), the number of collision in the random case is on average $2^{31.5} \cdot 2^{-30} = 2^{1.5} \approx 3$, while the number of collision in the other case is $2^{31.5} \cdot 0 = 0$. That is, $2^{16.25}$ chosen plaintexts are sufficient for this distinguisher.

```
Data: $2^{16.25}$ texts $c^i$ (for $i = 1, ..., 2^{16.25}$).
Result: 1 if there is at least one collision, 0 otherwise.
for each pair $(c^i, c^j)$ with $i \neq j$ do
    $c \leftarrow MC^{-1}(c^i \oplus c^j)$;
    for $k$ from 0 to 3 do
        if $c_{k,0} = 0$ AND $c_{(3+k)\%4,1} = 0$ AND $c_{(2+k)\%4,2} = 0$ AND $c_{(1+k)\%4,3} = 0$ then
            return 1;
        end
    end
end
return 0.
```
Algorithm 3: Distinguisher for 4-rounds of AES - Pseudo-code.

Note that this distinguisher exploits the Impossible Differential property presented in [5]. Thus, it is not a surprise that the computational complexity of these two distinguishers is the same. Only for completeness, note that it is possible to set up a 0-probability distinguishers also for the 3-rounds case:

$$\begin{aligned} &Pr(R^{(3)}(x) \oplus R^{(3)}(y) \in \mathcal{M}_I \,|\, x \oplus y \in \mathcal{C}_J) = \\ =&Pr(R^{(3)}(x) \oplus R^{(3)}(y) \in \mathcal{C}_I \,|\, x \oplus y \in \mathcal{D}_J) = 0 \end{aligned}$$

where $0 < |I| + |J| \leq 4$. Since in the random case, the probability that two elements belong to the same coset of $\mathcal{C}_I$ or $\mathcal{M}_I$ is upper bounded by 2^{-30} for each I and J, one needs at least $2^{15.75}$ chosen plaintexts to set up this distinguisher. That is, in the case of 3-rounds of AES, the 0-probability distinguisher is worse than the one described in the previous section

Finally, note that also the 4-rounds distinguisher (as the 3-rounds one) works also in the decryption direction. In this case, using the same argumentation as before, if we two texts belong to the same coset of $\mathcal{M}_I$, then they belong to two different cosets of $\mathcal{D}_J$ four rounds before for $|I| + |J| \leq 4$.

Relationship between 4-Rounds Subspace Trail and Impossible Differential Cryptanalysis. We would like to highlight the relationship between the 4-rounds subspace trails found in Sect. 5.3 and the impossible differential cryptanalysis. As we have seen, if $0 < \dim(\mathcal{D}_I) + \dim(\mathcal{M}_J) \le 16$ then $Pr(R^{(4)}(x) \oplus R^{(4)}(y) \in \mathcal{M}_J \,|\, x \oplus y \in \mathcal{D}_I) = 0$. We define this subspace trail as a "*0-Probability Subspace Trail*" or "*Impossible subspace trail*". In the following, we'd like to show the relationship between (5) and *Impossible Differential Analysis* [5], [4], which is a generalization of Differential Analysis [6]. Differential cryptanalysis traditionally considers characteristics or differentials with relatively high probabilities and uses them to distinguish the correct unknown keys from the wrong keys. The idea is that the difference predicted by the differential appears frequently only when the correct key is used to decrypt the last few rounds of many pairs of ciphertexts. Impossible differential analysis exploits instead the differences which should not occur (i.e., that have probability exactly zero). In this case, a key that decrypts a pair of ciphertexts to that difference is certainly wrong.

Definition 7. *(**Inverse-diagonal spaces**) The inverse-diagonal spaces $\mathcal{ID}_i$ are defined as*

$$\mathcal{ID}_i = \langle e_{0,i}, e_{1,i-1}, e_{2,i-2}, e_{3,i-3} \rangle.$$

If $I \subseteq \{0, 1, 2, 3\}$, the subspace $\mathcal{ID}_I$ is defined as $\mathcal{ID}_I = \bigoplus_{i \in I} \mathcal{ID}_i$.

For instance, $\mathcal{ID}_0 = SR(\mathcal{C}_0)$ corresponds to the symbolic matrix

$$\mathcal{ID}_0 = \left\{ \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 \\ 0 & 0 & x_3 & 0 \\ 0 & x_4 & 0 & 0 \end{bmatrix} \middle| \forall x_1, x_2, x_3, x_4 \in \mathbb{F}_{2^8} \right\}.$$

Using similar argumentations as before, if $|I| + |J| \le 4$ and if the final MixColumns operation is omitted, then $Pr(R_f \circ R^{(3)}(x) \oplus R_f \circ R^{(3)}(y) \in \mathcal{ID}_J \,|\, x \oplus y \in \mathcal{D}_I) = 0$. Thus, consider 5 rounds of AES:

$$p^h \xrightarrow{R(\cdot)} s^h \xrightarrow{R_f \circ R^{(3)}(\cdot)} c^h$$

for $h = 1, 2$. If there exists a pair of ciphertexts c^1 and c^2 that belong to the same coset of $\mathcal{ID}_J$ (that is $c^1 \oplus c^2 \in \mathcal{ID}_J$), then all the keys of the first round such that $s^1 \oplus s^2 = R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for $0 < \dim(\mathcal{D}_I) + \dim(\mathcal{ID}_J) \le 16$ are certainly wrong.

To exploit this fact in order to discover the key, the idea is to choose plaintexts with a particular shape. For simplicity, let $I = \{0\}$ fixed. Suppose to considers pair of plaintexts p^1, p^2 such that $p^1_{i,j} = p^2_{i,j}$ for each $i, j = 0, ...3$ with $(i, j) \neq \{(0, 0), (1, 3), (2, 2), (3, 1)\}$ (that is $SR^{-1}(p^1)_{col(i)} = SR^{-1}(p^2)_{col(i)}$ for $i = 1, 2, 3$). This choice implies that for each key K:

$$R(p^1)_{col(i)} = R(p^2)_{col(i)} \qquad \forall i = 1, 2, 3,$$

that is the second, the third and the fourth columns of the two texts are equal after one round[5]. Given c^1 and c^2 such that $c^1 \oplus c^2 \in \mathcal{ID}_J$ (with $\dim(Y_J) \ge 12$), in order to guarantee that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$, the attacker has to work only on the first column

[5] For completeness, to show this fact we compute the i-th column of $SR^{-1}(p^1)$ and $SR^{-1}(p^2)$ after one round for $i = 1, 2, 3$. By simple computation, we have that for each $j = 1, 2$:

$$R(SR^{-1}(p^j)_{col(i)} = [k^1 \oplus MC \circ \text{S-Box}(p^j \oplus k^0)]_{col(i)},$$

where we use the fact that she ShiftRows, the SubBytes and the AddRoundKey operations can be switched positions. Thus, since the MixColumns operation works on each column independently by the others and since $SR \circ SR^{-1}(p^1)_{col(i)} = p^1_{col(i)} = p^2_{col(i)} = SR \circ SR^{-1}(p^1)_{col(i)}$ for each $i = 1, 2, 3$, it follows that the second, the third and the fourth columns of the two texts are equal after one round.

of $R(p^1)$ and $R(p^2)$, that is only on the first column of $SR^{-1}(k)$ (for the other columns, all the values are fine). Thus, all the keys such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ are certainly wrong.

There are three possibilities that can be exploited for an impossible differential attack, which are $\dim(\mathcal{D}_I) = 4$ and $\dim(\mathcal{ID}_J) = 12$, $\dim(\mathcal{D}_I) = 12$ and $\dim(\mathcal{ID}_J) = 4$, and finally $\dim(\mathcal{D}_I) = \dim(\mathcal{ID}_J) = 8$. For each of these combinations, using the definitions of $\mathcal{D}_I$ and $\mathcal{ID}_J$ it is possible to obtain and to list all the impossible input/output combinations of difference that can be exploited to set up the attack. In particular, the first combination is exploited for example in [27] and in [3], while the second one is exploited in [28]. Interestingly, in literature there isn't any attack that exploits the last (impossible) input/output combination of differences. A possible reason of this fact is that using this combination it is not possible to attack 7 rounds of AES-128 as for the other combinations. Moreover, even if it is possible to attack 7 rounds of AES-192 and 8 rounds of AES-256 using it, our results (omitted due to page limit) show that in this case the data and the computational complexity is not better than the other attacks already present in literature that exploit the first and the second impossible combinations.

Relationship between 3- and 4-Rounds Subspace Trail and Integral Attack. For comparison, another four round integral distinguisher for AES uses the fact that summing over all 2^{32} ciphertexts (formed by encrypting a coset of a diagonal space $\mathcal{D}_i$ four rounds) is zero. In terms of subspaces, this has a different interpretation.

First of all, note that the entire space $\mathbb{F}_{2^8}^{4\times 4}$ can be decomposed as $\mathbb{F}_{2^8}^{4\times 4} = \mathcal{M}_0 \oplus \mathcal{M}_1 \oplus \mathcal{M}_2 \oplus \mathcal{M}_3$, where $\mathcal{M}_j$ is the j-th mixed space defined above. Let $\mathcal{M}_I = \mathcal{M}_0 \oplus \mathcal{M}_1 \oplus \mathcal{M}_2$. If we encrypt the 2^{32} plaintexts of a coset of $\mathcal{D}_i$ (for four round with the final MixColumns), we get a set of 2^{32} ciphertexts $C = \{c_1, c_2, \ldots, c_{2^{32}}\}$, where each c_i belongs to a different coset of $\mathcal{M}_I$. If we decompose these vectors with respect to the subspaces $\mathcal{M}_i$, each c_i can be written as $c_i = c_{i,0} \oplus c_{i,1} \oplus c_{i,2} \oplus c_{i,3}$ where $c_{i,j} \in \mathcal{M}_j$. Since each c_i belongs to a different coset of $\mathcal{M}_I$, it means that the components $c_{i,3}$ are all different; thus their sum must be zero since it amounts to summing over all vectors in $\mathcal{M}_3$. Since this property holds for all four choices of $\mathcal{M}_I$, it means that all of the components $c_{i,j}$ must be different with respect to the same subspace $\mathcal{M}_j$, thus the sum over all the vectors in C is zero. In comparison to integrals we have more structure that allows for distinguisher with lower data-complexity. Note that an analogous argumentation can be used for the three-rounds case. Moreover, in the case in which the final MixColumns operation is omitted, it is sufficient modify the previous argumentation using the subspaces $\mathcal{ID}_j$ instead of the subspaces $\mathcal{M}_j$.

For the four-rounds case, another explanation exploits the subspace trail and the three-rounds zero sum property. As we have seen, a coset of a diagonal space $\mathcal{D}_i \oplus a$ is mapped after one round into a coset of a column space $\mathcal{C}_i \oplus b$. Let $i = 0$ for simplicity. By definition, $\mathcal{C}_0 \oplus b = (\mathcal{C}_0 \cap \mathcal{D}_0) \oplus (\mathcal{C}_0 \cap \mathcal{D}_{1,2,3}) \oplus b$, that is:

$$\mathcal{C}_0 \oplus b = \bigcup_{x \in (\mathcal{C}_0 \cap \mathcal{D}_{1,2,3}) \oplus b} (\mathcal{C}_0 \cap \mathcal{D}_0) \oplus x,$$

where $|\mathcal{C}_0 \cap \mathcal{D}_{1,2,3}| = 2^{24}$. Since each coset $(\mathcal{C}_0 \cap \mathcal{D}_0) \oplus x$ corresponds to a set with only one active bytes, the sum of corresponding ciphertexts after three rounds is equal to zero (due to the three-rounds zero sum property of AES). Note that this property holds for each coset $(\mathcal{C}_0 \cap \mathcal{D}_0) \oplus x$, that is for each $x \in (\mathcal{C}_0 \cap \mathcal{D}_{1,2,3}) \oplus b$. Thus, the sum over all 2^{32} ciphertexts formed by encrypting a coset of a diagonal space $\mathcal{D}_i$ four rounds is equal to zero.

Finally, note that the integral attack on 3- and 4-rounds works exactly in the same way also in the decryption direction (independently of the presence of the final MixColumns operation). For example, for the 4-rounds case and if the final MixColumns operation is not omitted, given 2^{32} ciphertexts that belong to the same coset of $\mathcal{M}_i$ with $|i| = 1$, then the sum of the corresponding plaintexts is equal to zero. Instead, if the final MixColumns operation is omitted, given 2^{32} ciphertexts in the same coset of $\mathcal{ID}_i$ with $|i| = 1$, then the sum of the corresponding plaintexts is zero.

6 Attack 3 Rounds of AES - $\dim(\mathcal{D}_I) = 4$

Starting from the subspace trails of AES found in the previous section, in the following we show how to exploit them to set up low data complexity attacks. In particular, we focus on the subspace trail distinguisher on 2 rounds presented in Sect. 5.1 and on AES-128.

Consider $\mathcal{D}_I$ with $\dim(\mathcal{D}_I) = 4$ (that is $|I| = 1$). For simplicity, we show our attack only for the case $I = \{0\}$. In this case, $\mathcal{D}_I$ and $\mathcal{M}_I$ are the subsets of dimension 4 corresponding to the following symbolic matrix:

$$\mathcal{D}_I \equiv \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & x_4 \end{bmatrix} \qquad \mathcal{M}_I \equiv \begin{bmatrix} a_1(x_1) & a_2(x_4) & a_2(x_3) & a_3(x_2) \\ a_2(x_1) & a_2(x_4) & a_3(x_3) & a_1(x_2) \\ a_2(x_1) & a_3(x_4) & a_1(x_3) & a_2(x_2) \\ a_3(x_1) & a_1(x_4) & a_2(x_3) & a_2(x_2) \end{bmatrix}, \tag{6}$$

where $a_i(\cdot)$ are linear functions defined as follow:

$$a_1(x) = \alpha x, \qquad a_2(x) = x, \qquad a_3(x) = (\alpha + 1)x. \tag{7}$$

As we have seen, for all $a \in \mathcal{D}_I^\perp$ there exists one and only one $b \in \mathcal{M}_I^\perp$ such that $R^{(2)}(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b$. The attack that we are going to present is based on this property and on the probability from Eq. (2).

6.1 The Attack on 3 Rounds

Consider 3 rounds of AES:

$$p \xrightarrow{R^{(2)}(\cdot)} s \xrightarrow{R_f(\cdot)} c,$$

where $p \in \mathcal{D}_I \oplus a$ (for a fixed $a \in \mathcal{D}_I^\perp$), and where MixColumns operation is omitted in the final round for simplicity. However, adding the MixColumns operation in the last round would not increase the resistance.

Let p^1 and p^2 two different plaintexts that belong to the same coset of $\mathcal{D}_I$, and let c^1 and c^2 the corresponding ciphertexts. The idea of the attack is to find all the keys of the final round such that

$$R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) = s^1 \oplus s^2 \in \mathcal{M}_I, \tag{8}$$

that is such that s^1 and s^2 belong to the same coset of $\mathcal{M}_I$.

Theorem 2. *Let p^1 and p^2 be two plaintexts of the same coset of $\mathcal{D}_I$, and let c^1 and c^2 the respective ciphertexts. Let k be the secret round-key of the final round. If there exists a pair of ciphertexts (c^1, c^2) such that k doesn't satisfy (8), then k is certainly wrong.*

Proof. Suppose by contradiction that k is the right key.

If there exists a pair (c^1, c^2) such that k doesn't satisfy (8), then $R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \notin \mathcal{M}_I$ (i.e. $R_f^{-1}(c^1)$ and $R_f^{-1}(c^2)$ belong to two different cosets of $\mathcal{M}_I$), that is

$$Pr(R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I \,|\, p^1 \oplus p^2 \in \mathcal{D}_I) \neq 1.$$

Since k is the right key, then

$$Pr(R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I \,|\, p^1 \oplus p^2 \in \mathcal{D}_I) = 1,$$

(see (2)) which is a contradiction. $\square$

By the previous Theorem, it follows that the secret key is certainly one of those that satisfy the equivalence (8). In order to find all the keys that satisfy (8), the idea is to take advantage of the particular form of $\mathcal{M}_I$. In particular, observe that the columns of the subspace $\mathcal{M}_I$ depend on different and independent variables, and that each element of a fixed column depends on a single variable in a very particular form.

Theorem 3. *Let $a, b \in \mathcal{M}_I^{\perp}$, and let $s^1 \in \mathcal{M}_I \oplus a$ and $s^2 \in \mathcal{M}_I \oplus b$. Denote s as the sum of s^1 and s^2, i.e. $s = s^1 \oplus s^2$.*

Then, $a = b$ if and only if all the following equivalences are satisfied:

$$\begin{array}{lll}
s_{0,0} = \alpha s_{1,0}, & s_{2,0} = s_{1,0}, & s_{3,0} = (\alpha+1) s_{1,0}; \\
s_{0,1} = s_{1,1}, & s_{2,1} = (\alpha+1) s_{1,1}, & s_{3,1} = \alpha s_{1,1}; \\
s_{1,2} = (\alpha+1) s_{0,2}, & s_{2,2} = \alpha s_{0,2}, & s_{3,2} = s_{0,2}; \\
s_{0,3} = (\alpha+1) s_{2,3}, & s_{1,3} = \alpha s_{2,3}, & s_{3,3} = s_{2,3}.
\end{array}$$

Proof. If $a = b$, it is straightforward to prove that all the previous equivalences are satisfied. Suppose instead that all the previous equivalences are satisfied. Then, working on the first column (analogous for the others), it follows that:

$$a_{0,0} \oplus b_{0,0} = \alpha[a_{1,0} \oplus b_{1,0}]; \quad a_{2,0} \oplus b_{2,0} = a_{1,0} \oplus b_{1,0}; \quad a_{3,0} \oplus b_{3,0} = (\alpha+1)[a_{1,0} \oplus b_{1,0}].$$

That is, there exist x_1 and y_1 such that:

$$\begin{array}{ll}
s^1_{0,0} = \alpha x_1 \oplus a_{0,0}, & s^2_{0,0} = \alpha y_1 \oplus a_{0,0} \oplus \alpha[a_{1,0} \oplus b_{1,0}]; \\
s^1_{1,0} = x_1 \oplus a_{1,0}, & s^2_{1,0} = y_1 \oplus b_{1,0}; \\
s^1_{2,0} = x_1 \oplus a_{2,0}, & s^2_{2,0} = y_1 \oplus a_{2,0} \oplus a_{1,0} \oplus b_{1,0}; \\
s^1_{3,0} = (\alpha+1) x_1 \oplus a_{3,0}, & s^2_{3,0} = (\alpha+1) y_1 \oplus a_{3,0} \oplus (\alpha+1)[a_{1,0} \oplus b_{1,0}].
\end{array}$$

Let $y_1' = y_1 \oplus a_{1,0} \oplus b_{1,0}$ and observe that $b_{1,0} = b_{1,0} \oplus a_{1,0} \oplus a_{1,0}$. Rewriting $s^2_{\cdot,0}$ using y_1' instead of y_1 and working in a similar way on all the other columns, it is easy to prove that s^1 and s^2 belong to the same coset of $\mathcal{M}_I$, that is $a = b$. $\square$

Using the previous Theorem and the fact that the columns of $\mathcal{M}_I$ depend on different and independent variables, the attacker can work independently on each column of $\mathcal{M}_I$ and so on each column of $SR^{-1}(k)$. Thus, we show our attack only on the first column (it is completely equivalent for the others).

Given c^1 and c^2, the attacker guesses (for example) the bytes $k_{1,3}$ and finds all the values of bytes $k_{0,0}, k_{2,2}$ and $k_{3,1}$ of the key of the final round such that s^1 and s^2 belong to the same coset of $\mathcal{M}_I$. Using the previous Theorem and given $k_{1,3}$, the other bytes of the first column of $SR^{-1}(k)$ have to satisfy the following equalities:

$$\begin{aligned}
&\text{S-Box}^{-1}(c^1_{0,0} \oplus k_{0,0}) \oplus \text{S-Box}^{-1}(c^2_{0,0} \oplus k_{0,0}) = \\
&=\alpha[\text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3}) \oplus \text{S-Box}^{-1}(c^2_{1,3} \oplus k_{1,3})]; \\
\\
&\text{S-Box}^{-1}(c^1_{2,2} \oplus k_{2,2}) \oplus \text{S-Box}^{-1}(c^2_{2,2} \oplus k_{2,2}) = \\
&=\text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3}) \oplus \text{S-Box}^{-1}(c^2_{1,3} \oplus k_{1,3})]; \\
\\
&\text{S-Box}^{-1}(c^1_{3,1} \oplus k_{3,1}) \oplus \text{S-Box}^{-1}(c^2_{3,1} \oplus k_{3,1}) = \\
&=(\alpha+1)[\text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3}) \oplus \text{S-Box}^{-1}(c^2_{1,3} \oplus k_{1,3})].
\end{aligned} \tag{9}$$

Suppose to guess one value of $k_{1,3}$. Then, it is very easy to observe that each equality of (9) can be rewritten in the following way

$$\text{S-Box}^{-1}(\alpha \oplus x) \oplus \text{S-Box}^{-1}(x) = \beta(k_{1,3}), \tag{10}$$

where $\alpha = c^1_{i,j} \oplus c^2_{i,j}$, β *depends on* $k_{1,3}$ and on two ciphertexts bytes, and $x = c^2_{i,j} \oplus k_{i,j}$ is the unknown variable. First of all, note that if $\alpha = 0$ then this equality is impossible if $\beta \neq 0$, while it is always satisfied if $\beta = 0$.

The solutions of equation (10) are related to the differential uniformity of the S-Box. Since the inverse function S-Box^{-1} is differential 4-uniform, there are at most four different solutions[6]. In particular, there are two solutions with probability 126/256, four solutions with probability 1/256 and zero solutions with probability 129/256. That is, on average there exist about 2.016 different values that satisfy equation (10) with probability 49.6%.

Even if Eq. (10) is very similar to that used in Differential Cryptanalysis, there are some important differences. First of all, β doesn't simply assume some values with certain probability as in Differential Cryptanalysis, but it depends on the secret key $k_{1,3}$ (i.e. $\beta = \beta(k_{1,3})$). Indeed, in our case, for each values of $k_{1,3}$ we are looking for an x that satisfies a certain relationship with the guessed secret key $k_{1,3}$ (and so with β), that is we are looking for which combinations of bytes of the secret key the relationships (9) hold. Remember that these relationships guarantee that the two elements belong to the same coset of $\mathcal{M}_I$, which is a necessary condition that the right key has to satisfy. Thus, we are exploring the relationship between the bytes of $s := s^1 \oplus s^2 = R^{-1}(c^1) \oplus R^{-1}(c^2)$ in order to recover information on the secret key, instead of working on each byte of s independently by the others as in the Differential Cryptanalysis. We discuss this topic in detail in Sect. 7.

Suppose for the moment that c^1 and c^2 are two ciphertexts such that $c^1_{i,j} \neq c^2_{i,j}$ for each $i, j \in \{0, ..., 3\}$. Guessed the byte key $k_{1,3}$, then the attacker finds $2.016^3 \simeq 8.194$ possible combinations $(k_{0,0}, k_{2,2}, k_{3,1})$ that satisfy the equivalence (9) with probability $(49.6\%)^3 \simeq 12.2\%$. Since there are 256 values of $k_{1,3}$, we have in total $0.122 \times 256 \times (2.016)^3 \simeq 255.95$ combinations $(k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ for the first column (analogous for the others).

The computational cost to find these 256 combinations $(k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ for the first column (and analogous for the other ones) can be estimated at 3 (conditions)$\times 2^8$

[6] Observe that if x satisfies (10), then also $x \oplus \alpha$ satisfies it, so the number of solutions can not be odd.

(values of $k_{1,3}$)$\times$[$2 + 2 \times 2^8$ (values of $k_{0,0}, k_{2,2}, k_{3,1}$ for each condition)]$= 2^{18.59}$ S-Box look-ups and $2^{17.59}$ XOR operations.

Actually, it is possible to improve this result. Indeed, observe that if the value $k_{0,0}$ satisfies (or not) equation (9) for a given value $k_{1,3}$, then this equation is also satisfied (or not) by the value $k_{0,0} \oplus c^1_{0,0} \oplus c^2_{0,0}$. That is, for each $k_{1,3}$ the attacker has to test only 128 values of $k_{0,0}$ and not 256. The same consideration holds for the other bytes of the key. Thus, the computational cost to find the 256 combinations $(k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ for the first column (and analogous for other columns) can be estimated at $2^{16.59}$ S-Box look-ups.

Recover the Secret Key using only 2 Chosen Plaintexts. Using the previous procedure, the attacker is able to find 2^8 combinations for each column, that is $(2^8)^4 = 2^{32}$ candidates of the keys in total. A first possibility is simple to store them in the memory and to do a brute force attack on these 2^{32} possible keys, that is to check for which of these keys the condition $c^1 = R^{(3)}(p^1)$ (or equivalently $c^2 = R^{(3)}(p^2)$) is satisfied. In this case, the total cost of the attack is approximately 2^{32} executions of the three-round cipher (observe that this second step is more expensive than the first one), 2^{32} memory access and the memory cost is approximately 256×4 (byte)$\times 4$ (columns) $= 2^{12}$ byte.

This result can be improved if the plaintexts belong to the same coset of $\mathcal{D}_0 \cap \mathcal{C}_0$ (where $\mathcal{D}_0 \cap \mathcal{C}_0$ has dimension 4 since $\mathcal{D}_0 \cap \mathcal{C}_0 = \langle e_{0,0} \rangle$ - Lemma 6). By definition, $\mathcal{D}_0 \cap \mathcal{C}_0$ and $\mathcal{C}_0 \cap \mathcal{M}_0$ correspond to symbolic matrix:

$$\mathcal{D}_0 \cap \mathcal{C}_0 \equiv \begin{bmatrix} x\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix} \qquad \mathcal{C}_0 \cap \mathcal{M}_0 \equiv \begin{bmatrix} a_1(x)\,0\,0\,0 \\ a_2(x)\,0\,0\,0 \\ a_2(x)\,0\,0\,0 \\ a_3(x)\,0\,0\,0 \end{bmatrix},$$

where $a_i(\cdot)$ are defined as in (7). Using similar argumentations as before, for each $a \in (\mathcal{D}_0 \cap \mathcal{C}_0)^\perp$ there exist unique $b \in (\mathcal{C}_0 \cap \mathcal{M}_0)^\perp$ and unique $c \in \mathcal{M}_0^\perp$ such that:

$$R((\mathcal{D}_0 \cap \mathcal{C}_0) \oplus a) = (\mathcal{C}_0 \cap \mathcal{M}_0) \oplus b \qquad \text{and} \qquad R^{(2)}((\mathcal{D}_0 \cap \mathcal{C}_0) \oplus a) \subseteq \mathcal{M}_0 \oplus c.$$

Thus, suppose that the attacker has found 2^8 possible combinations for each column of the secret key. Instead to attack by brute force all the 2^{32} possible keys, the idea is the following. Given c^1 and c^2 and a (possible) key k^3 of the final round, the idea is to compute $s^1 = R^{-1}(c^1)$ and $s^2 = R^{-1}(c^2)$, that is to decrypt one round c^1 and c^2, with a cost of $16 \times 2 = 32$ S-Box look-ups. Then, the attacker can easily compute k^2 using the inverse key schedule (cost of 4 S-Box look-ups). Finally, she checks if $R^{-1}(s^1)$ and $R^{-1}(s^2)$ belong to the same coset of $\mathcal{C}_0 \cap \mathcal{M}_0$, that is if the first columns of $R^{-1}(s^1)$ and $R^{-1}(s^2)$ satisfied conditions similar to (9). The cost of this last operation is 8 S-Box look-ups. Since these conditions are verified with probability 2^{-24}, only 2^8 possible keys of k^3 satisfy these conditions. The cost of this step is $2^{32} \times (32 + 4 + 8) = 2^{37.46}$ S-Box look-ups, that is $2^{31.55}$ executions of the three-round cipher. Using a brute force attack, the attacker can find the right key among the last 2^8 candidates.

For this attack, the attacker needs about 2 chosen plaintexts in the same coset of $\mathcal{D}_0 \cap \mathcal{C}_0$, the total computational cost is about $2^{10.7}$ (first step) $+2^{31.55}$ (check) $+2^8$ (brute force) $= 2^{31.55}$ executions of the three-round cipher and 2^{32} memory access, and a memory cost of about 2^{12} bytes of memory (to store the combinations of the columns of the key found in the first step). In a 32-bit implementation, each round of AES can be implemented by 20 memory accesses[7], that is 16 look-ups table for S-Box $+SR+MC$ and 4 look-ups table for the key schedule. Thus, one can declare that the total complexity of the attack is approximately $(20\cdot 3)^{-1}\times 2^{32}+2^{31.55}=2^{31.58}$ executions of the three-round encryption.

Remember that these results are based on the hypothesis that $c^1_{i,j}\neq c^2_{i,j}$ for each $i,j\in\{0,...,3\}$. By simple computation, the probability that this condition is satisfied is $(255/256)^{16}=0.9393$, so everything works with high probability.

Recover the Secret Key using 3 Chosen Plaintexts. Another possibility is to use a third plaintext p^3 that belongs to the same coset of $\mathcal{D}_I$ of p^1 and p^2 (i.e. $p^1\oplus p^3\in\mathcal{D}_I$), and such that the corresponding ciphertexts satisfy the following conditions:

$$c^1_{i,j}\neq c^2_{i,j},\quad c^1_{i,j}\neq c^3_{i,j},\quad c^2_{i,j}\neq c^3_{i,j}\qquad \forall i,j\in\{0,...,3\}. \tag{11}$$

The idea is simple to repeat the first step of the attack, on the combinations of key columns that satisfy condition (9) for the first pair of texts, using a second pair of texts composed of p^3 and one of the first two plaintexts. That is, the idea is to eliminate the wrong keys checking the keys found previously (at the first step) with this second pair. As before, the idea is to work again on each column independently. Remember that, for each column, the right combination is the only one that satisfies the condition (9) for each pair of ciphertexts such that the corresponding plaintexts belong to the same coset of $\mathcal{D}_I$.

Using 3 chosen plaintexts and to improve the total computational cost, the attack should be a little modified. Working independently for each column, the best idea is to work only on one of the equations of (9), e.g. the first one. Given the first pair of plaintexts, the attacker is able to recover 256 possible combinations for the pair $(k_{0,0},k_{1,3})$. For each one of them, the attacker checks if it verifies the condition (9) also for the second pair of plaintexts. Since on average each equation of the condition (9) is satisfied with probability 2^{-8}, the attacker finds the correct combination of $(k_{0,0},k_{1,3})$. In this way, the attacker knows $k_{1,3}$ and so she knows the right part of equations (9). Thus, in the same way of before, she can easily discover $k_{2,2}$ and $k_{3,1}$. However, since the attacker knows the right part of the equations, only two texts pass the first step. Indeed, observe that there are 2^8 possible values and that each equation is satisfied only with probability 2^{-8}, thus only one text passes the test. Anyway, if x satisfies (10), then also $x\oplus\alpha$ satisfies it. Using the second pair of texts, the attacker finds the right bytes of the key.

Working independently on each column, a good idea is to perform these two steps at the same time, that is to check the combination found with the first pair of plaintexts immediately using the second one: in this way, the attacker doesn't need to store anything (except the right combination for each column).

Remember that this result is based on the hypothesis that condition (11) is satisfied. By simple computation, this happens with probability

$$\left(\frac{255\cdot 254}{256^2}\right)^{16}=82.85\%.$$

Actually, the condition (11) can be (a little) relaxed. For example, three chosen plaintext such that $c^1_{0,0}=c^2_{0,0}\neq c^3_{0,0}$ and $c^1_{1,3}\neq c^2_{1,3}\neq c^3_{1,3}$ can also be used to find the key. Thus,

[7] This approximation has been proposed and used in [28].

using 3 chosen plaintexts the probability of success is:

$$\left[\sum_{i=0}^{3} \binom{4}{i} \left(\frac{255 \cdot 254}{256^2}\right)^{4-i} \left(\frac{255}{256^2}\right)^{i}\right]^4 = 88.83\%.$$

The computational cost for the first step can be approximated by $2^{-1} \cdot 2^8$ (values of $k_{1,3}$) $\times (2 + 2 \times 2^{-1} \cdot 2^8$ (values of $k_{0,0}$) $) + 4 \times 256$ (check with 2-nd pair) $= 2^{15.055}$ S-Box look-ups, while the cost for the second step is of $2 \times 2^{-1} \cdot 2^8$ (values of $k_{2,2}$ and $k_{3,1}$) $+ 2 \times 2$ (check with 2-nd pair) $= 2^8$ S-Box look-ups. In conclusion, our attack needs 3 chosen plaintexts (one plaintext is in common for the two pairs) and the total computational cost is approximately at 4 (columns) $\times (2^{15.055} + 2 \times 2^8) = 2^{17.08}$ S-Box look-ups, that is about $2^{11.2}$ executions of the three-round cipher (the memory cost is negligible).

```
Data: 2 ciphertexts pairs $(c^1, c^2)$ and $(c^1, c^3)$, whose corresponding plaintexts belong in the
      same coset of $\mathcal{D}_0$.
Result: First diagonal of the secret key $k^3$ (i.e. $k^3_{i,i}$ for each $i = 0, ..., 3$).
for all values of $k^3_{1,3}$ do
    for all values of $k^3_{0,0}$ do
        check if 1-st equivalence of (9) is satisfied for both pairs of ciphertexts
        If satisfied, then identify candidates for $k^3_{1,3}$ and $k^3_{0,0}$
    end
end
for all candidates $k^3_{1,3}$ do                         // on average only 1 candidate for $k^3_{1,3}$
    for all values of $k^3_{2,2}$ do
        check if 2-nd equivalence of (9) is satisfied for both pairs of ciphertexts
        If satisfied, then identify candidate for $k^3_{2,2}$
        else reject $k^3_{1,3}$ as candidate
    end
    for all values of $k^3_{3,1}$ do
        check if 3-rd equivalence of (9) is satisfied for both pairs of ciphertexts
        If satisfied, then identify candidate for $k^3_{3,1}$
        else reject $k^3_{1,3}$ as candidate
    end
end
return $k^3_{i,i}$ for each $i = 0, ..., 3$.
```

Algorithm 4: *Attack on 3 rounds of AES-128 - Pseudo Code.* For simplicity, in this pseudo-code, we show how to find only the first diagonal of the secret key of the last round, and we don't use the "trick" that if x satisfy (10) then also $x \oplus \alpha$ satisfy it. To recover the entire key, it is sufficient to repeat exactly the same attack for the other diagonals.

Perform the Attack using Table Look-ups. It is also possible to perform the previous attack in a slight different way, that is using look-ups table. First of all, for each values of α and β, the idea is to precompute the values of x that satisfy condition (10) and to store them in a table. The cost of this precomputation is approximately 2^{16} S-Box look-ups[8], that is about $2^{10.1}$ executions of three-round AES. The memory cost to store this table is approximately $(2^8)^2 = 2^{16}$ bytes.

[8] For each of the 2^8 values of β and of the 2^7 values of x (remember that if x satisfies (10), then also $x \oplus \alpha$ satisfies it), the attacker finds the value of α that satisfies the equality (10), and stores this combination in the memory. For the remaining pairs (α, β), the equality (10) has no solutions.

Then, for each column, the attacker is able to discover the 256 key combinations with $3 \cdot 2^8$ look-ups table (that is, 3 look-ups table for each values of $k_{1,3}$). In order to find the right key among these combinations, she can use the same techniques described previously, or she can use another pair of plaintexts and again the look-ups table. On average, using 3 chosen plaintexts and working in the same way described previously, for each column the attacker needs 2^8 look-ups table and $2^8 \cdot 4 = 2^{10}$ S-Box look-ups to discover $k_{0,0}$ and $k_{1,3}$ (i.e. the attacker finds the 256 possible values of $(k_{0,0}, k_{1,3})$ using look-ups table, and then tests them with the second pair of texts), and 2 look-ups table to discover the other 2 bytes. Thus, the total computational cost is of $4 \cdot (2^8 + 2) = 2^{10.02}$ look-ups table (with a memory cost of 2^{16} bytes) and $2^{10} \cdot 2 = 2^{11}$ S-Box look-ups (since she needs 2^{10} S-Box look-ups to compute the value of β for each value of $k_{1,3}$), that is $2^{5.1}$ executions of three-round AES (the precomputation cost is of $2^{10.1}$ executions of three-round AES). Using a previous observation (1 round of AES $\approx$ 20 memory accesses), one can declare that the total complexity of the attack is approximately $(20 \cdot 3)^{-1} \times 2^{10} + 2^{5.1} = 2^{5.68}$ executions of the three-round encryption.

Final Round with MixColumns Operation. For sake of completeness, these attacks on 3 rounds works the same in case the MixColumns operation is *not* omitted in the last round. In this case, the idea (very common in the literature) is simply to change the position of the final MixColumns operation with the final AddRoundKey operation (remember that these operations are linear). In this case, the major difference is that the attacker has to work with $\tilde{k}$ defined as:

$$\tilde{k} := MC^{-1}(k), \tag{12}$$

instead of k (the secret key of the final round). Moreover, in this case $s^h = R^{-1}(c^h)$ (for $h = 1, 2$) are defined as $s^h_{i,j} = \text{S-Box}^{-1}(\tilde{c}^h_{i,j+i} \oplus \tilde{k}_{i,j+i})$ where $i, j = 0, ..., 3$ (and $i + j$ is taken modulo 4) and where $\tilde{c}^h := MC^{-1}(c^h)$. Note that when all bytes of $\tilde{k}$ have been determined, the secret key k can be recovered by $k = MC(\tilde{k})$.

6.2 The Attack on 1 and 2 Rounds of AES

Finally, note that the same attack can be used to attack 1 and 2 rounds of AES. In the case of 2 rounds of AES, consider the following situation:

$$p \xrightarrow{R(\cdot)} s \xrightarrow{R_f(\cdot)} c,$$

where $p \in \mathcal{C}_I \oplus a$ (for a fixed $a \in \mathcal{C}_I^{\perp}$) and MixColumns operation is omitted in the final round (for simplicity). The attack is completely equivalent to the previous one using the following probability[9]:

$$Pr(R(x) \oplus R(y) \in \mathcal{M}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1.$$

In this case, our attack needs 3 chosen plaintexts (in the same coset of $\mathcal{C}_I$) and the total computational cost is approximately at $2^{17.08}$ S-Box look-ups, that is about $2^{11.8}$ executions of the two-round cipher. As before, it is possible to perform the attack using memory access instead of S-Box look-ups. By simple computation and with the approximation of 1 round of AES with 20 memory accesses, the complexity of the attack is

[9] Note that it is not possible to use the probability $Pr(R(x) \oplus R(y) \in \mathcal{C}_I \,|\, x \oplus y \in \mathcal{D}_I) = 1$, since $Pr(R_f(x) \oplus R_f(y) \in \mathcal{ID}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1$ for *each possible candidate of the secret key.*

approximately $(20 \cdot 2)^{-1} \times 2^{10} + 2^{6.32} = 2^{6.6}$ executions of the two-round encryption, the cost of the precomputation is of $2^{10.7}$ executions of two-round AES, and a memory cost of 2^{16} bytes.

The attack on 1 round (where MixColumns operation is not omitted) is completely equivalent to the previous one. In this case, the idea is to simply choose plaintexts that belong to the same coset of $\mathcal{M}_I$.

7 Relationship with other Attacks

In this section, we discuss the relationship between our proposed attack and others present in literature, with particular attention to differential cryptanalysis and integral attacks. We remind the reader that the relationship between subspace trail and impossible differential attacks has already been discussed in Sect. 5.3.

Differential Cryptanalysis. Differential cryptanalysis [6] is a general form of cryptanalysis applicable to block ciphers. It studies how differences in information input can affect the resulting difference at the output. Differential attacks exploit the fact that pairs of plaintexts with certain differences yield other certain differences in the corresponding ciphertexts with a non-uniformity probability distribution. Statistical key information is deduced from ciphertext blocks obtained by encrypting pairs of plaintext block with a specific (bitwise) difference under the target key. In particular, for a pair of plaintexts related by a constant (bitwise) difference, one tries for all values of the round key in the last round, if the expected difference in the ciphertexts occurs. This is repeated several times and the most suggested values are assumed to be the value of the secret key of the last round. Attacks following this basic attack vector can not be described without considering the details of the S-Boxes of the cipher as it affects the probability of events. Subspace trail cryptanalysis is largely independent of concrete choices of an S-box. An important variant of this attack is the *truncated differential attack* [24], in which the attacker considers only part of the difference between pairs of texts, i.e. it is a differential attack where only a part of the difference in the ciphertexts can be predicted. As truncated differential cryptanalysis can be described without considering details of S-Boxes it is hence much closer in nature to subspace trail cryptanalysis.

The most important difference between our attack and the previous ones is the following. In (truncated) differential cryptanalysis, the attacker looks for *differences* between bytes of pairs of texts which are in the *same* position (that is, in the same row and column). Indeed, given a pair of plaintext with a certain difference, the attacker exploits the probability of the difference between the bytes of corresponding ciphertexts to deduce statistical information of the key. That is, she considers only the differences between bytes which are in the same row and column. Instead, in our attack the attacker exploits the *relationship* between bytes of the *same* text which are in *different* positions (that is, which are *not* in the same row and column).

In order to highlight this fact, we reconsider our 3-round attack from Section 6.1 using a 2-round subspace trail from the last section in more detail. Consider two plaintext p^1 and p^2 such that $p^1 \oplus p^2 \in \mathcal{D}_0$. As we have seen, after two rounds there exists $a \in \mathcal{M}_0^{\perp}$ (which depends on the secret key) such that $s^1 = R^{(2)}(p^1)$ and $s^2 = R^{(2)}(p^2)$ belong in $\mathcal{M}_0 \oplus a$. That is, there exist $x, y \in GF(2^8)$ such that the first columns of s^1, s^2 and

$s := s^1 \oplus s^2$ (analogous for the others) are:

$$s^1 \equiv \begin{bmatrix} \alpha \cdot x \oplus a_0 \\ x \\ x \oplus a_1 \\ (\alpha+1) \cdot x \oplus a_2 \end{bmatrix}, \quad s^2 \equiv \begin{bmatrix} \alpha \cdot y \oplus a_0 \\ y \\ y \oplus a_1 \\ (\alpha+1) \cdot y \oplus a_2 \end{bmatrix}, \quad s \equiv \begin{bmatrix} \alpha \cdot (x \oplus y) \\ x \oplus y \\ x \oplus y \\ (\alpha+1) \cdot (x \oplus y) \end{bmatrix}. \tag{13}$$

Suppose for a moment that we can work with subspaces instead of cosets, that is that $a_0 = a_1 = a_2 = 0$ (or equivalently suppose to know a_0, a_1, a_2). In this case, the attacker can perform our attack using only one plaintext, instead of a pair of plaintexts. Indeed, if the attacker knows c^1 (such that p^1 is in $\mathcal{D}_0$), the she can exploits the relationship among the bytes of s^1 to recover the key:

$$\begin{aligned} &\text{S-Box}^{-1}(c^1_{0,0} \oplus k_{0,0}) \oplus a_0 = \alpha[\text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3})]; \\ &\text{S-Box}^{-1}(c^1_{2,2} \oplus k_{2,2}) \oplus a_1 = \text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3}); \\ &\text{S-Box}^{-1}(c^1_{3,1} \oplus k_{3,1}) \oplus a_2 = (\alpha+1)[\text{S-Box}^{-1}(c^1_{1,3} \oplus k_{1,3})]. \end{aligned}$$

However, since the values a_i for $i = 0, 1, 2$ are unknown, the attacker is not able to recover any information using a single plaintext/ciphertext. One possibility is to consider a pair of plaintexts, and to work as follows. Working independently on each column, the attacker guesses one column of the final key $SR^{-1}(k)$ and she partially decrypts c^1 and c^2 in order to recover one column (e.g. the first one) of s^1 and s^2. Then she can easily discover x, a_0, a_1, a_2 that satisfy the conditions (13) for s^1 and the value of y that satisfies $s^2[1, 0] = y$. Finally, she checks if the values of a_0, a_1, a_2 found for s^1 satisfy the conditions (13) for s^2. Observe that each guessed column of the key satisfies these conditions only with probability $(2^8)^{-3} = 2^{-24}$. Thus, only $(2^8)^4 \times 2^{-24} = 256$ combinations of $SR^{-1}(k)$ for the each column survived this first check.

This method is completely analogous to that used in our attack: the only difference is that our attack is more efficient. Indeed, instead of guessing all the column of the key (that is 2^{32} possible values), we guess only one byte (that is 2^8 possible values) and we find the others in order to satisfy condition (13) rewritten in a "better way". To do this, the idea is simply to consider pairs of texts and to eliminate the unknown values a_i simply XORing them, in order to work with something that belongs to a subspace and not to a coset. Thus, given $s := s^1 \oplus s^2$ in our attack we exploit the relationships among the bytes of s which are in different positions. Moreover, note that these relationships hold *only* among the part of bytes which are of the same original state, since there isn't any relationship between x and y. That is, if we consider the equality $s^1_{0,0} \oplus s^2_{0,0} = \alpha s^1_{1,0} \oplus \alpha s^2_{1,0}$, then there isn't any relationship between $s^1_{0,0}$ and $s^2_{0,0}$, since x and y are completely independent (the only relationship holds between $s^j_{0,0}$ and $s^j_{1,0}$ for the same $j = 1, 2$).

In conclusion, given a pair of plaintext with a certain difference, consider the difference of the texts in the last round, that is $s := s^1 \oplus s^2$. In (truncated) differential cryptanalysis, the attacker works on each bit/byte of s independently by the others, i.e. the attacker is not interested to the relationships among the bits/bytes of s. Instead, in our attack, we exploit the relationships between the different bytes of s to recover information about the secret key.

Integral Attack. *Integral cryptanalysis* [13,?] (or *Square attack*) is a cryptanalytic attack that is particularly applicable to block ciphers based on substitution-permutation networks, like AES. Unlike differential cryptanalysis, which uses pairs of chosen plaintexts with a fixed XOR difference, integral cryptanalysis uses sets (or multi-sets) of chosen plaintexts in which one part is held constant and another part varies through *all* possibilities. For example, an attack might use 256 chosen plaintexts that have all but 1 of its byte the same, and all differ in that 1 byte. Such a set necessarily has a XOR sum of 0. Using the fact that the XOR sum of the corresponding sets after 2.5 rounds is equal to 0, the attacker can attack 3 rounds of AES and deduce information on the secret key. In our attack, we use a more detailed study of the subspace properties in order to discover the secret key, that is the relationship among the bytes of pairs of ciphertexts obtained by encrypting pairs of plaintexts that belong to the same coset of a given subspace.

8 Attack 4 Rounds of AES - Extension at the End

Starting from the attack on 3 rounds, we show how to extend it at the end in order to attack 4 rounds of AES. Consider the following situation:

$$p \xrightarrow{R^{(2)}(\cdot)} s \xrightarrow{R(\cdot)} z \xrightarrow{R_f(\cdot)} c,$$

where $p \in \mathcal{D}_I \oplus a$ (for a fixed $a \in \mathcal{D}_I^{\perp}$) and MixColumns operation is omitted (only for simplicity) in the final round.

As we have seen, if $p^1 \oplus p^2 \in \mathcal{D}_I$, then $s^1 \oplus s^2 \in \mathcal{M}_I$. The idea of the attack is simply to guess part of the key of the final round, in order to partially decrypt c and obtain (part of) z. Then the attacker can repeat the attack on 3 rounds, working on z and exploiting the relationships that hold between the bytes of $s^1 \oplus s^2$. In this case, the attacker founds on average one values of k^3 for each guessed value of k^4. Thus, she can not say anything about k^3 a priori, since it depends on the guessed value of k^4. That is, she has to check that the key of the third round k^3 and of the final round k^4 satisfy the key schedule. If they satisfy the key schedule, then the attacker has found the right key, otherwise she has to repeat the previous procedure for the other values of k^4.

In the following, we give all the details of the attack. In the first part, the attacker guesses two columns of the last key k^4, partial decrypts the pairs of ciphertexts and discovers part of the key of the third round k^3, using the attack on 3 rounds described previously. Then she checks that the found key satisfies the key schedule with the guessed key k^4. In the second part of the attack, for the key candidates that satisfy the key schedule, the attacker is able to recover other bytes of k^4 and k^3 (using again the key schedule). In order to find the complete secret key, the attacker has to guess the remaining unknown bytes of the secret key and to do a brute force attack. The attack can be performed using three or only two chosen plaintexts.

Details of the First Part of the Attack. In order to show our attack, suppose that the attacker guesses the following eight bytes of the key of the final round k^4:

$$SR(k^4_{i,3}) = k^4_{i,3-i} \qquad \text{and} \qquad SR(k^4_{i,0}) = k^4_{i,-i} \qquad \forall i \in \{0,1,2,3\}, \tag{14}$$

that is the first and the fourth columns of k^4 after the ShiftRows operation (the index $-i$ is taken module 4). Observe that there are $(2^8)^8 = 2^{64}$ possibilities in total. Note that since the attacker can not impose any restriction/condition on the secret key, she has to repeat the following steps for each possible values of these eight bytes of k^4.

Using these guessed key bytes of k^4 and three given ciphertexts c^1, c^2, c^3 (which plaintexts p^1, p^2, p^3 belong to the same coset of $\mathcal{D}_I$), the attacker is able to compute:

$$z^h_{i,j} = \text{S-Box}^{-1}(c^h_{i,j+i \bmod 4} \oplus k^4_{i,j+i \bmod 4}) \qquad \forall h \in \{1,2,3\},\ \forall j \in \{0,3\},\ \forall i.$$

The cost of this step is $2^{64} \cdot 8 \cdot 3 = 2^{68.6}$ S-Box look-ups. Note that the positions of the guessed bytes of the key k^4 can not be chosen in an arbitrary way, since in the following step the attacker has to apply the InverseMixColumns operation on two columns of z.

Using z^1, z^2 and z^3 (instead of c^1, c^2, c^3), the attacker can repeat the previous attack on 3 rounds and finds eight bytes (i.e. first and fourth columns) of the key k^3, in order to guarantee that $s^i \oplus s^j \in \mathcal{M}_I$ for each $i, j = 1, 2, 3$. In this case, note that for each column the attacker can impose only one relationship that involves two bytes, and not three relationships as in the previous attack on 3 round. For example, the only relationship that holds for the first column is the following (analogous for the others):

$$\text{S-Box}^{-1}(\hat{z}^1_{0,0} \oplus \tilde{k}^3_{0,0}) \oplus \text{S-Box}^{-1}(\hat{z}^2_{0,0} \oplus \tilde{k}^3_{0,0}) = \alpha[\text{S-Box}^{-1}(\hat{z}^1_{1,0} \oplus \tilde{k}^3_{1,0}) \oplus \text{S-Box}^{-1}(\hat{z}^2_{1,0} \oplus \tilde{k}^3_{1,0})],$$

where $\tilde{k}^3 = SR^{-1}(MC^{-1}(k^3))$ and $\hat{z} = SR^{-1}(MC^{-1}(z))$.

Thus, given eight byte of k^4, the computational cost to find 8 bytes of k^3 is approximately $4 \times 2^7 \times (2 + 2 \cdot 2^7) = 2^{17}$ S-Box look-ups and 2^5 byte of memory. That is, the computational cost to find the 2^{64} combinations of all the eight bytes of k^3 (equivalently $\tilde{k}^3$ - note that we are working with columns of $SR(k^3)$) and k^4 is $2^{64} \times 2^{17} = 2^{81}$ S-Box look-ups.

When the attacker has found eight bytes of k^3, she has to check if they satisfy the key schedule. In particular, the following three conditions between the eight bytes of k^3 and the eight bytes of k^4 hold:

$$k^4_{0,0} = \text{S-Box}(k^3_{1,3}) \oplus k^3_{0,0} \oplus \text{0x08}, \quad k^4_{3,0} = \text{S-Box}(k^3_{0,3}) \oplus k^3_{3,0}, \quad k^4_{1,3} = k^3_{1,3} \oplus k^4_{1,2}. \tag{15}$$

Since these conditions are satisfied with probability 2^{-24}, only $2^{64} \times 2^{-24} = 2^{40}$ possible combinations of the 8 bytes of k^3 (equivalently $\tilde{k}^3$) and k^4 satisfied them. The cost of this step is $2 \times 2^{64} \times 2^{-8} = 2^{57}$ S-Box look-ups (since one condition doesn't involve any S-Box, and the probability that it is satisfied is 2^{-8}).

Details of the Second Part of the Attack. In order to find the right key, the idea is to test the 2^{40} found combinations using a brute force attack. Observe that for each of these combinations (composed of eight bytes of k^4 and 8 bytes of k^3), the attacker can compute other four bytes of k^4 using the key schedule, that is:

$$\begin{aligned} k^4_{1,0} &= \text{S-Box}(k^3_{2,3}) \oplus k^3_{1,0}, & k^4_{2,3} &= k^3_{2,3} \oplus k^4_{2,2}, \\ k^4_{2,0} &= \text{S-Box}(k^3_{3,3}) \oplus k^3_{2,0}, & k^4_{0,2} &= k^3_{0,3} \oplus k^4_{0,3}. \end{aligned} \tag{16}$$

and (in an analogous way) three bytes of k^3, which are $k^3_{2,1}, k^3_{2,2}$ and $k^3_{3,1}$. Thus, four bytes of the key of the final round k^4 are still unknown, which are:

$$k^4_{0,1} \qquad k^4_{1,1} \qquad k^4_{3,2} \qquad k^4_{3,3}, \tag{17}$$

where

$$k^4_{3,2} \oplus k^4_{3,3} = k^3_{3,0} \tag{18}$$

and $k^3_{3,0}$ is known. To do the brute force attack, the idea is simple to guess those three bytes. That is, the attacker has to test $2^{40} \cdot (2^8)^3 = 2^{64}$ possible keys by brute force.

The cost of this step (that is, of the brute force attack of 2^{64} values) is 2^{64} four-rounds AES. Only for completeness, note that another possibility is to consider plaintexts that belong to the same coset of $\mathcal{D}_0 \cap \mathcal{C}_0$, and to use the fact that after one round they belong to the same coset of $\mathcal{C}_0 \cap \mathcal{M}_0$.

As for the attack on 3 rounds, a good idea is to perform these two steps at the same time, that is to test the keys found in the first step by the brute force attack. In this way, the attacker doesn't need to store anything.

In conclusion, for this attack the attacker needs 3 different chosen plaintexts and the computational cost is approximately $2^{68.6} + 2^{81} \simeq 2^{81}$ S-Box look-ups for the first part of the attack, that is about $2^{74.7}$ four-rounds AES, and 2^{64} four-rounds AES for the brute force attack of the second part, that is in total $2^{74.7} + 2^{64} \simeq 2^{74.7}$ four-rounds AES (the memory cost is negligible).

```
Data: 2 ciphertexts pairs $(c^1, c^2)$ and $(c^1, c^3)$, whose corresponding plaintexts belong in the
      same coset of $\mathcal{D}_0$.
Result: Secret key $k^4$.
for all $2^{64}$ values of two columns of $k^4$ defined as in (14) do
    partial decrypt
    3-Rounds Attack (see Algorithm 4): identify candidates for eight bytes of $k^3$ (two per
      column)                                          // on average only 1 candidate
    check key schedule conditions given in (15)
    if key schedule satisfied then                     // probability equal to $2^{-24}$
        find other four bytes of $k^4$ using (16)
        for all $2^{24}$ values of the four remaining bytes of $k^4$ (see (17)-(18)) do
            Brute Force attack on all possible candidates
            if key $k^4$ found then
                return $k^4$.
            end
        end
    end
end
```

Algorithm 5: *Attack on 4-rounds (EE) of AES-128 - Pseudo-code.*

Observe that the first step of this attack can be performed using only table look-ups, in the same way of the attack described in Sect. 6. In this case, the cost of the first step of the attack becomes $2^{64} \cdot 2^{12} = 2^{76}$ memory access, since the cost of the attack on 3 rounds is 2^{12} memory access and the attacker has to repeat this step for all the 2^{64} possible values of the 8 bytes of the key of the final rounds. Thus, in this case the attacker needs 3 different chosen plaintexts, the computational cost is approximately 2^{76} memory access and 2^{64} four-rounds AES encryptions, and the memory cost is about 2^{16} bytes. Using a previous observation (1 round of AES $\approx$ 20 memory accesses), one can declare that the total complexity of the attack is approximately $(20 \cdot 4)^{-1} \times 2^{76} + 2^{64} = 2^{69.71}$ executions of the four-round encryption.

Attack with *only* 2 Chosen Plaintexts (or more than 3). It is also possible to mount this attack using only 1 pair of chosen plaintexts, that is 2 chosen plaintexts are sufficient to discover the secret key. At the first step, using a single pair of chosen plaintexts, the attacker is able to discover 2^{32} combinations for the eight bytes of k^3 for each combination of the eight bytes of k^4. Instead to use a third chosen plaintext to find the right combination, the idea is simply to do a brute force attack.

In particular, using the conditions (15), the attacker is able to eliminate 2^{24} wrong combinations. Then, for each of the 2^8 survived combinations, she can easily find other four bytes of k^4 using the conditions (16), and (in the same way as before) she guesses the remaining four bytes of k^4 (remember that it is sufficient to guess only three of them). Thus, for each combination of the eight bytes of k^4, the attacker has to test by brute force 2^{32} values. This means that in total she has to test by brute force $2^{32} \times (2^8)^8 = 2^{96}$ possible values. In conclusion, for this attack, the attacker needs 2 different chosen plaintexts and the computational cost is 2^{96} four-rounds AES (the memory cost is negligible).

We also would like to show that it is not possible to improve the computational cost of the attack using more chosen plaintexts. Indeed, in the second step of the attack with three chosen plaintexts, suppose to check the 2^{40} survived combinations with other two pairs of plaintexts that belong to the same coset $\mathcal{D}_I \oplus a$. That is, given p^4 and p^5, the attacker check for which combinations of the 8 bytes of the keys k^3 and k^4, the conditions $s^1 \oplus s^4 \in \mathcal{M}_I$ and $s^1 \oplus s^5 \in \mathcal{M}_I$ are satisfied. Observe that the probability that both these conditions are satisfied is 2^{-64}, thus only one key (the right one) survived. Anyway, the total computational cost of this variant is again approximately $2^{74.7}$ four-rounds AES, since the most expensive step of the attack is the first one, which doesn't change in any way if the attacker uses more than three chosen plaintexts.

These two versions of this attack work in a similar way if MixColumns operation is not omitted in the last round. In this case, the idea is simply to change the position of MixColumns operation with the final AddRoundKey operation. As these operations are linear they can be interchanged, by first XORing the data with an equivalent version of the key (that is $\tilde{k}^4 := MC^{-1}(k^4)$) and only then applying the MixColumns operation.

Practical Result. Since this attack on 4 rounds with the extension at the end has a very high computational cost, we have tested it in a different way. As we have seen, the attacker has to guess eight bytes of the final key, for a total of 2^{64} possibilities. In our experiments, the attacker guesses only two bytes of the final key instead of eight, where the remaining six bytes are fixed and equal to those of the secret key. In this way, the total complexity of the attack becomes more feasible for a real test and allows us to have a practical verification of the attack. As with over practical verifications, it is available from [1]

9 Attack 4 Rounds of AES - Extension at the Beginning

In the previous section, we have seen how to extend at the end the attack on 3 rounds. In this section, we show how to attack 4 rounds extending at the beginning the attack on 3 rounds. As we'll show, both from the computational point of view and from the data complexity point of view, the attack on 4 rounds is better if $\dim(\mathcal{D}_I) = 12$ than $\dim(\mathcal{D}_I) = 4$. Since the attack on 3 rounds given in Sect. 6 works the same when $\dim(\mathcal{D}_I) = 12$, we present it in detail in Appendix A. We limit ourselves to report the data and the computational complexity of this attack. In the case in which $\dim(\mathcal{D}_I) = 12$, the attack on 3 rounds needs 4 pairs of plaintexts, that is 5 chosen plaintexts, and the total computational cost is approximately at $2^{31.09}$ S-Box look-ups, that is about $2^{25.18}$ executions of the three-round cipher (the memory consumption is negligible), or $2^{24.6}$ memory access and $2^{24.18}$ S-Box look-ups (with a memory cost of approximately 2^{16} bytes).

In order to attack 4 rounds of AES, the idea is to extend the attack on 3 rounds (described in Appendix A) adding an initial round. Consider two plaintexts p^1 and p^2:

$$p^h \xrightarrow{R(\cdot)} R(p^h) \xrightarrow{R^{(2)}(\cdot)} s^h \xrightarrow{R_f(\cdot)} c^h.$$

where $h = 1, 2$. If the attacker is able to guarantee that after one round they belong to the same coset of $\mathcal{D}_I$, then she can repeat the attack on 3 rounds, using $R(p^h)$ instead of p^h. Observe that if $p^1 \oplus p^2 \in \mathcal{C}_J$, then $R(p^1) \oplus R(p^2) \in \mathcal{M}_J$ and (Lemma 6) $R(p^1) \oplus R(p^2) \notin \mathcal{D}_I$, for each I and J such that $\dim(\mathcal{M}_J)+\dim(\mathcal{D}_I) \leq 16$. Thus, p^1 and p^2 have to be chosen such that $p^1 \oplus p^2$ doesn't belong to $\mathcal{C}_J$ for each J such that $|J| + |I| \leq 4$, in order to guarantee that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$.

We present our attack in the case in which $\dim(\mathcal{D}_I) = 12$, and only for simplicity, we suppose that MixColumns operation is omitted in the last round (however, our attack works in the same way in the case in which it is not omitted).

Given pairs of plaintexts p^1 and p^2, our main goal is to minimize the number of bytes of k^0 that the attacker has to guess in order to guarantee the condition

$$R(p^1) \oplus R(p^2) \in \mathcal{D}_I \tag{19}$$

for a certain I with $|I| = 3$. For the following, it is important to note that we don't fix a particular I. A possible choice for the pair of plaintexts p^1 and p^2 can be the following:

$$p^1_{i,j} = p^2_{i,j} \qquad \text{for all} \quad (i,j) \neq \{(0,3),(2,1)\}. \tag{20}$$

As we show in the following, this choice allows the attacker to guess only 2 bytes of k^0.

Due to the previous choice of p^1 and p^2, it follows that

$$R(p^1)_{i,j} = R(p^2)_{i,j} \qquad \forall i, \forall j \neq 3.$$

independent of the secret key. Thus, to guarantee that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for a certain I with $|I| = 3$, it is sufficient for the attacker to guess only two bytes of the secret key (that is, $k^0_{0,3}$ and $k^0_{2,1}$), since it is sufficient that one byte of the first column of $R(p^1) \oplus R(p^2)$ is equal to zero to guarantee (19). For example, for the case $I = \{0, 1, 2\}$ (studied in App. A), the condition $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ is satisfied if and only if $R(p^1)_{3,0} \oplus R(p^2)_{3,0} = 0$, that is if the following equivalence is satisfied

$$\alpha \cdot [\text{S-Box}(p^1_{0,3} \oplus k^0_{0,3}) \oplus \text{S-Box}(p^2_{0,3} \oplus k^0_{0,3})] = \text{S-Box}(p^1_{2,1} \oplus k^0_{2,1}) \oplus \text{S-Box}(p^2_{2,1} \oplus k^0_{2,1}).$$

Thus, for each bytes $k^0_{0,3}$ and $k^0_{2,1}$ of the secret key, the attacker has to find pairs of plaintexts p^1 and p^2 that satisfy Eq.(20) and such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for that key and for a certain I with $|I| = 3$, in order to repeat the attack on 3 rounds.

On average, for a fixed I (with $|I| = 3$), there exist 2^{24} combinations $(p^1_{0,3}, p^1_{2,1}, p^2_{0,3}, p^2_{2,1})$ that satisfy Eq. (19) and Eq. (20). On the other hand, given a particular combination $(p^1_{0,3}, p^1_{2,1}, p^2_{0,3}, p^2_{2,1})$ and for a fixed I, on average there are 2^8 different pair of key bytes that satisfy (20). Since there are four possible I with $|I| = 3$, for each combination $(p^1_{0,3}, p^1_{2,1}, p^2_{0,3}, p^2_{2,1})$ on average there are $2^8 \times 2^2 = 2^{10}$ different pair of key bytes such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for a certain I with $|I| = 3$.

Note that since the attacker can not impose any restriction on the secret key, she has to repeat this (and the next steps) for all the possible pairs $(k^0_{0,3}, k^0_{2,1})$.

Proposition 1. *Let $(k_{0,3}, k_{2,1})$ and $(p^1_{0,3}, p^1_{2,1}, p^2_{0,3}, p^2_{2,1})$ be a pair of key bytes and a combination of plaintexts bytes that satisfy (20). If $(\hat{k}_{0,3}, \hat{k}_{2,1})$ denote another pair of key bytes, then the combination $(q^1_{0,3}, q^1_{2,1}, q^2_{0,3}, q^2_{2,1})$ of plaintext bytes defined as $q^h_{i,j} := p^h_{i,j} \oplus \hat{k}_{i,j} \oplus k_{i,j}$ where $h = 1, 2$ and $(i, j) \in \{(0, 3), (2, 1)\}$ satisfies (19) for that key.*

Finally, observe that if $\dim(\mathcal{D}_I) = 4$, the condition such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ becomes more complicated, since the attacker has to guess 4 bytes of the initial key instead of 2. This justifies the initial choice of $\dim(\mathcal{D}_I) = 12$.

Suppose that for each pair of key bytes $(k^0_{0,3}, k^0_{2,1})$ the attacker knows a combination $(p^1_{0,3}, p^1_{2,1}, p^2_{0,3}, p^2_{2,1})$ such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for that key and for a certain I with $|I| = 3$. The general idea of the attack is simply to repeat the previous attack on 3 rounds described in Appendix A, that is to use 4 pairs of plaintexts (that satisfy (19) and (20)) in order to discover the key k^4 such that $R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I$. As for the attack on 4 rounds with the extension at the end, when the attacker has found k^4, she has to check if it is compatible with k^0 (i.e. that they satisfy the key schedule), in order to verify that it is the right key. If they are compatible, then she has discovered the secret key, otherwise she has to repeat this procedure for another pair of key bytes $(k^0_{0,3}, k^0_{2,1})$.

In order to check if k^4 is compatible with k^0, we recall the following useful theorem (see [10] for more details):

Theorem 4. *For each round r and for each $i = 0, ..., 3$:*

$$k^r_{i,1} = k^{r+2}_{i,1} \oplus v^{r+1}_i, \qquad k^r_{i,2} = k^{r+2}_{i,3} \oplus k^{r+2}_{i,0},$$
$$k^r_{i,3} = k^{r+2}_{i,3} \oplus k^{r+2}_{i,1} = k^{r+4}_{i,3} \oplus v^{r+3}_i,$$

where $v^r_i = \text{S-Box}\,(k^r_{(i+1) \bmod 4,3}) \oplus RCON[r+1]$.

By simple computation:

$$k^0_{0,3} = k^4_{0,3} \oplus \text{S-Box}(k^4_{1,2} \oplus k^4_{1,3}) \oplus 0\text{x}08,$$
$$k^0_{2,1} = k^4_{2,1} \oplus \text{S-Box}(k^4_{3,0} \oplus k^4_{3,1} \oplus k^4_{3,2} \oplus k^4_{3,3}) \oplus \text{S-Box}(k^4_{3,2} \oplus k^4_{3,3}).$$

Thus, only 3 S-Box look-ups are sufficient to check if k^4 is compatible with k^0 (the choice of (20) is also due to the simplicity of this check operation). Note that since the probability that the key schedule is satisfied is 2^{-16} and since there are only 2^{16} possible combinations of $(k^0_{0,3}, k^0_{2,1})$, on average only one key satisfied the key schedule (which is the right one), that is the attacker certainly finds the right key.

In the following, we present how to implement the attack in order to minimize the data complexity cost.

Minimal Data Complexity. As we have seen, the attacker needs on average 4 pairs of plaintexts (that satisfy Eq.(20)) for each pair of key bytes $(k_{0,3}, k_{2,1})$.

Observe that given n plaintexts p^i for $i = 0, ..., n-1$ that satisfy condition (20), then it is possible to construct $n \cdot (n-1)/2$ pairs of plaintexts that satisfy the condition (20). On average for each of these pairs of plaintexts (p^1, p^2), there are 2^{10} pairs of key bytes that satisfy $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ for $|I| = 3$. Since the number of pair of keys bytes

$(k^0_{0,3}, k^0_{2,1})$ is 2^{16}, then the attacker needs on average $2^{16} \times 4 \times 2^{-10} = 2^8$ pairs of chosen plaintexts for the attack. That is[10], she needs about $2^{4.55} \simeq 24$ chosen plaintexts.

These chosen plaintexts can be precomputed in advance. For each of the 2^{16} keys bytes $(k^0_{0,3}, k^0_{2,1})$, the idea is to store the four pairs of chosen plaintexts and for each pair (p^1, p^2) the corresponding I such that $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$ (in order to implement the attack), using a predetermined order[11]. By simple computation, the attacker needs 4 (pairs of CP) $\times 2 \times 4$ (bytes to store) $\times 2^{16}$ (number of keys) $= 2^{21}$ bytes to store the plaintexts and 2 (bits of I) $\times$ 2^{16} $=$ 2^{17} bits $=$ 2^{14} bytes to store the corresponding I, that is approximately $2^{21} + 2^{14} = 2^{21.01}$ bytes of memory. Thus, our attack needs $24 \simeq 2^{4.55}$ chosen plaintexts and the total computational cost is approximately at $(2^8)^2$ (2 guessed bytes of k^0) $\times 2^{31.09}$ (cost of the attack on 3 rounds) $+3 \times 2^{16}$ (check the key schedule) $= 2^{47.09}$ S-Box look-ups, that is about $2^{40.7}$ executions of the four-round cipher and 2^{16} (sequential) memory access.

```
Data: 24 ciphertexts such that for each keys bytes $(k^0_{0,3}, k^0_{2,1})$ there exist 4 different pairs of
      chosen plaintexts that satisfy (20) and such that there exists $I$ with $|I| = 3$ such that
      $R(p^1) \oplus R(p^2) \in \mathcal{D}_I$.
Result: Secret key $k^4$.
for all $2^{16}$ values of two bytes $(k^0_{0,3}, k^0_{2,1})$ do
    for the guessed key, pick up the 4 different pairs of chosen plaintexts stored in memory (and
      corresponding $I$)
    3-Rounds Attack (see App. A): identify candidates for $k^4$ // on average only 1 candidate
    check key schedule conditions given in (9)
    if key schedule satisfied then                          // probability equal to $2^{-16}$
        return $k^4$.
    end
end
```

Algorithm 6: *Attack on 4 rounds (EB) of AES-128 - Pseudo Code.* For simplicity, we assume that the attacker has already found 24 texts defined as in the text.

In the same way as before, it is also possible to perform this attack using memory access. In this case, the total computational cost is approximately at $2^{16} \times 2^{24.6}$ (cost of the attack on 3 rounds)$= 2^{40.6}$ memory access and $2^{16} \times 2^{24.18}$ (cost of the attack on 3 rounds)$+3 \times 2^{16}$ (check the key schedule) $= 2^{40.18}$ S-Box look-ups, that is $2^{33.86}$ executions of the four-round AES. With the approximation of 1 round of AES with 20 memory accesses, one can declare that the total complexity of the attack is approximately $(20 \cdot 4)^{-1} \times 2^{40.6} + 2^{33.86} = 2^{35.08}$ executions of the four-round encryption.

10 Secret-Key Distinguishers for 5-Round of AES

In this section we describe our new 5-round distinguishers in a model where the adversary is not assumed to know anything about the key, i.e. the secret-key model.

[10] Given 2^n elements, the number of different pairs are $2^{n-1} \cdot (2^n - 1)$. Viceversa, in order to have 2^n different pairs, we need 2^m elements, where

$$m = \log_2 \left((2^{n+3} + 1)^{\frac{1}{2}} + 1 \right) - 1 \simeq (n+1)/2. \tag{21}$$

[11] For example, let $f : \mathbb{Z}_{2^8} \times \mathbb{Z}_{2^8} \to \mathbb{Z}_{2^{16}}$ the (bijective) function defined as $f(x_1, x_2) = 4 \cdot (x_1 + 256 \cdot x_2)$. The combination of plaintexts for the key (k_1, k_2) are in positions $f(k_1, k_2), f(k_1, k_2) + 1, f(k_1, k_2) + 2$ and $f(k_1, k_2) + 3$.

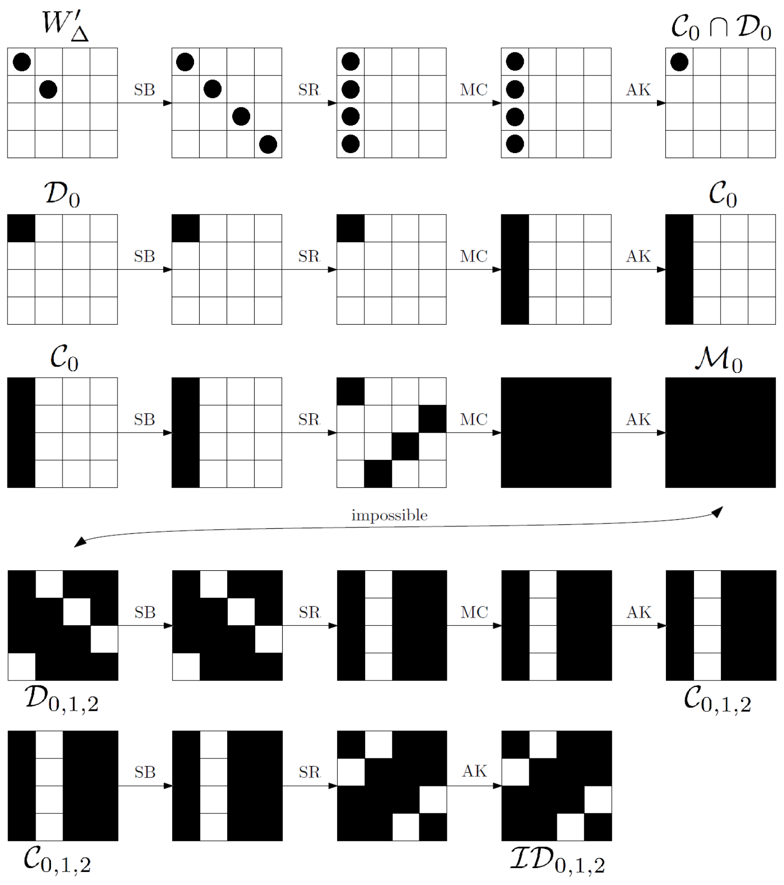

Fig. 9. 5-Rounds Secret Key Distinguisher with data complexity $2^{59.7}$ based on the Impossible Subspace Trail on 4-Rounds (from Sect. 5.3). The choice of the plaintexts (defined by the set W'_Δ in (10.4)-(23)) guarantees that if $\delta_i = k_{i,i}$ for each $i = 0, ..., 3$ after one round only one byte is active (i.e. the difference between the two texts on that byte is non-zero) instead of four. That is, after one round, the plaintexts belong to the same coset of $\mathcal{C}_0 \cap \mathcal{D}_0$. The probability that two ciphertexts belong to the same coset of $\mathcal{M}_J$ for $|J| = 3$ is zero. White box denotes denotes a byte with a zero-difference, while a black box denotes a byte with non-zero difference.

In order to construct this distinguisher, our idea is basically to extend the impossible subspace trail distinguisher on 4-rounds presented in Sect. 5.3 at the beginning, using a similar technique presented to extend the 3-rounds attack key-recovery attack on AES at the beginning in Sect 9. First of all, we assume that four bytes of the secret key are known, and using this assumption we set up the basis for our distinguisher. Then, we show how to extend this model to the case where no information of the secret key is given. As a result, this 5-rounds secret key distinguisher for AES has a data complexity of only $2^{59.7}$ chosen plaintexts, independent of the presence of the final MixColumns. We will also discuss a practical verification. Finally, we will also how that the distinguishers works independent of the presence of the last MixColumn operations, and also in decryption direction, all with the same complexity.

10.1 A Distinguisher assuming that Four Bytes of the Secret Key are Known

Suppose only for the moment to know four bytes of the secret key, that is $k_{i,i}$ for each $i = 0, ..., 3$. As we've already said, the idea is to extend the Impossible Subspace Trail on 4-Rounds described in Sect. 5.3, using a similar technique presented in Sect. 9. To do this, the idea is to choose plaintexts that belong to the same coset of $\mathcal{D}_i$ for a certain $|i| = 3$ after one round.

Consider a set of plaintext-ciphertext W_Δ defined as follows

$$W_\Delta = \left\{ (p^i, c^i) \text{ for } i = 0, ..., 2^8 - 1 \;\middle|\; \begin{bmatrix} p^i_{0,0} \\ p^i_{1,1} \\ p^i_{2,2} \\ p^i_{3,3} \end{bmatrix} := \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \oplus \text{S-Box}^{-1} \circ MC^{-1} \cdot \begin{bmatrix} x \\ 0 \\ 0 \\ 0 \end{bmatrix}, \right. \tag{22}$$
$$\left. \text{for } x \in \mathbb{F}_{2^8} \text{ and } p^i_{k,l} = p^j_{k,l} \, \forall i \neq j \text{ and } \forall k, l \text{ s.t. } k \neq l \right\}.$$

where $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ can assume 2^{32} different values. Note that W_Δ contains 2^8 different pairs (p, c) (that is, approximately 2^{15} different pairs) and that the 256 vectors defined as $\text{S-Box}^{-1} \circ M^{-1} \cdot \left[x, 0, 0, 0\right]^T$ can be simply precomputed in advance and stored in a table (computational cost of 2^{10} S-Box look-ups and memory cost of 2^{10} bytes).

The choice of these plaintexts is similar to the one done in Sect. 9. In particular, using similar argumentation given in the previous section, the choice of the plaintexts guarantees that they belong to the same coset of $\mathcal{D}_0$ (in particular of $\mathcal{D}_0 \cap \mathcal{C}_0$) after one round if $\delta_i = k_i$ for $i = 0, ..., 3$.

Proposition 2. *Let W_Δ defined as before. If $\delta_i = k_{i,i}$ for each $i = 0, ..., 3$, then there exists (unique) $a \in (\mathcal{D}_0 \cap \mathcal{C}_0)^\perp$ such that $R(W_\Delta) \subseteq (\mathcal{D}_0 \cap \mathcal{C}_0) \oplus a$.*

The proof follows immediately by the definition of W_Δ. In particular, note that since $p^i_{k,l} = p^j_{k,l}$ for each $i \neq j$ and for each $k \neq l$, it follows that

$$R(p^i)_{col(k)} = R(p^j)_{col(k)}$$

for each $k = 1, 2, 3$, exactly in the same way as in Sect. 9.

Thus, suppose initially to know all the sub-key bytes $k_{i,i}$ for each $i = 0, ..., 3$, and suppose that $\delta_i = k_{i,i}$ for each $i = 0, ..., 3$. In this case, if two different plaintexts p and q belong to W_Δ, then after one round they belong to the same coset of $\mathcal{D}_0 \cap \mathcal{C}_0 \subseteq \mathcal{D}_0$, that is $R(p) \oplus R(q) \in \mathcal{D}_0$. Thus, after five rounds they belong to different cosets of $\mathcal{M}_I$ for each $|I| = 3$ with probability 1, that is $R^{(4)} \circ R(p) \oplus R^{(4)} \circ R(q) \notin \mathcal{M}_I$ with $|I| = 3$, due to the impossible subspace trail described in details in Sect. 5.3). That is, if $\delta_i = k_{i,i}$ for each $i = 0, ..., 3$, then:

$$Pr(R^{(5)}(x) \oplus R^{(5)}(y) \in \mathcal{M}_J \,|\, x \oplus y \in W_\Delta) = 0,$$

where $x \neq y$ and for each J with $|J| = 3$.

In the case of a random permutation instead, given two plaintexts in W_Δ, the probability that two ciphertexts belong to the same coset of $\mathcal{M}_I$ for a certain I with $|I| = 3$ is 2^{-30}. Thus, for a random permutation, the probability that among the ciphertexts there is at least one collision in same coset of $\mathcal{M}_I$ for a certain I with $|I| = 3$ given n pairs of texts is:

$$p = 1 - \left(e^{-n/2^{32}} \right)^4 = 1 - e^{-n/2^{30}}.$$

If the number of pairs n is approximately $2^{31.6}$, then p is greater than 95%.

As a consequence, if $k_{i,i}$ are known for each $i = 0, ..., 3$ and if $\delta_i = k_{i,i}$ for each $i = 0, ..., 3$, to distinguish a random permutation from an AES one, we can proceeded as follows. For an AES permutation, the number of collision in $\mathcal{M}_I$ for $|I| = 3$ among the ciphertexts of the corresponding plaintexts that belong to the same set of W_δ is always zero. That is, given two plaintexts in W_Δ, the probability that the corresponding ciphertexts belong to the same coset of $\mathcal{M}_I$ is zero. Instead, for a random permutation, it is possible to have a collision among the ciphertexts which plaintexts belong to the same W_Δ. Thus, the idea is to consider a sufficient number of different sets W_Δ in order to guarantee that in the random case there is at least one set W_Δ for which there is at least one collision among the ciphertexts with probability 95%. In this way, we can distinguish the two cases.

Given a single set W_Δ, it is possible to construct $2^7 \cdot (2^8 - 1) \simeq 2^{15}$ different pairs. Thus, for the distinguisher we need approximately $2^{31.6} \cdot 2^{-15} = 2^{16.6}$ different sets W_Δ. Since each of these sets contains 2^8 texts, the data complexity of the distinguisher (in the case in which the four bytes $k_{i,i}$ are known) is of $2^{16.6} \cdot 2^8 = 2^{24.6}$ texts.

10.2 The 5-Rounds Secret Key Distinguisher

Starting from the previous distinguisher, we show how to extend it in the case in which all the sub-key bytes $k_{i,i}$ are unknown for each i.

Since we assume to not know Δ, the idea is basically to construct 2^{32} collections (each one with a certain number of sets W_Δ), each one for each possible combination of the 2^{32} values of $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$. For the AES permutation, we expect that there is one Δ such that for each corresponding set W_Δ the number of collisions among the ciphertexts is zero. Note that this value $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ corresponds to the values of the four-bytes of the secret key, that is $\delta_i = k_{i,i}$. For the other values of Δ, we expect a behavior similar to the one of a random permutation. In order to distinguish between the random permutation and the AES one, our goal is to choose the number of sets W_Δ such that for each possible combination of values of $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ there is at least one set W_Δ in which there is at least one collision among the ciphertexts. Thus, it follows that the AES permutation is the one in which there is one Δ for which the corresponding number of collision among the ciphertexts is zero, while for the random permutation we expect that for each one of the 2^{32} collections there is at least one set W_Δ in which there is at least one collision among the ciphertexts.

To compute the number of W_Δ, our goal is to guarantee that in the random case, for each possible value of Δ there is at least one set W_Δ for which there is at least one collision among the ciphertexts with *total probability* greater than 95%. If we use the same number of sets of before (that is $2^{16.6}$), since the 2^{32} collections are independent, the total probability that there is at least one set W_Δ with one collision for each one of the 2^{32} collections is $0.95^{2^{32}} \simeq 0$. Thus, for each one of the 2^{32} collections (i.e. for each combination of values of $(\delta_0, \delta_1, \delta_2, \delta_3)$), we need at least one collision with probability higher than $0.95^{1/2^{32}} \simeq 1 - 1.1 \cdot 10^{-11}$. To have at least one collision with this probability and using the same computation as before, each one of the 2^{32} collections has to be composed of $2^{34.7}$ pairs of texts (instead of $2^{31.6}$ pairs). Equivalently, this means that each one of the 2^{32} collections has to be composed of about $2^{34.7} \cdot 2^{-15} = 2^{19.7}$ different sets W_Δ. Since each one of these sets contains 2^8 texts (equivalently 2^{15} pairs) and since there are 2^{32} possible Δ, the total number of texts is 2^{32} (values of $\delta_0, ..., \delta_3$) $\times 2^{19.7}$ (number of sets W_Δ) $\times 2^8$ (texts for each set) $= 2^{59.7}$.

To summarize, suppose to have 2^{32} collections (one for each $\varDelta$), each one with $2^{19.7}$ different sets $W_\varDelta$, where each of these sets contains 2^8 texts, for a total of $2^{59.7}$ texts. In the random case and with probability 95%, we expect that in each one of these 2^{32} collections there is at least one set $W_\varDelta$ with one collision. Note that the average number of collisions for each collection (i.e. for each $\varDelta$) is about $2^{-30} \cdot 2^{34.7} = 2^{4.7} \simeq 26$. For the AES permutation, we expect that there exists one combination of $\varDelta$ for which there is no collision with probability 1 in the corresponding collection of sets. For all the other collections (i.e. for each other values of $\varDelta$), the behavior is similar to the random case, that is we expect to have at least one collision with probability 95%. Thus, we are able to distinguish a random permutation from an AES one.

How to Divide $2^{59.7}$ Texts into 2^{32} Collections? We have seen that $2^{59.7}$ chosen plaintexts are sufficient to set up the distinguisher for 5-rounds of AES. However, a problem arises. In order to distinguish the two cases, as first thing one has to be able to divide the $2^{59.7}$ texts in 2^{32} collection (one for each combination of values of $\varDelta = (\delta_0, ..., \delta_3)$), and for each one of these collections one has to divide the texts in the respective $2^{19.7}$ sets $W_\varDelta$. In particular, given two plaintexts p and q, it is very easy to check if $p_{i,j} = q_{i,j}$ for each $i \neq j$. However, given p and q with $p_{i,j} = q_{i,j}$ for each $i \neq j$, it is not possible to determine if they belong to the same set of $W_\varDelta$ (that is, if the combination of $\delta_0, ..., \delta_3$ of the set $W_{\varDelta^1}$ in which the first plaintext belongs is equal to the ones of the set $W_{\varDelta^2}$ in which the second plaintext belongs). Indeed, note that for each $x \in \mathbb{F}_{2^8}$ there exists $\varDelta$ such that $\left[\delta_0, \delta_1, \delta_2, \delta_3\right]^T = \left[p_{0,0}, p_{1,1}, p_{2,2}, p_{3,3}\right]^T \oplus$ S-Box$^{-1} \circ M^{-1} \cdot \left[x, 0, 0, 0\right]^T$, and similar for q.

Thus, we present a possible (simple) way to overcome this problem. First of all, observe that there are 2^{32} different values of $\delta_0, ..., \delta_3$. By definition, given two plaintexts p and q in $W_\varDelta$, note that their bytes that don't lie on the first diagonal have to satisfy the condition $p_{i,j} = q_{i,j}$ for $i \neq j$, but nothing more. Our idea is to impose a further condition on these bytes. Note that for each combination of $\delta_0, ..., \delta_3$, there are $(2^8)^{12} = 2^{96}$ different sets $W_\varDelta$[12]. Let $n \geq 1$ an integer, and let $f : (\mathbb{F}_{2^8})^n \to \mathbb{Z}^+$ defined as follows:

$$f(x_0, x_1, ..., x_{n-1}) := \sum_{i=0}^{n-1} 2^{8i} \times x_i,$$

where $0 \leq x_i < 256$ for each x_i. We use this function f to (slightly) modify the definition of $W_\varDelta$. In particular, we introduce the set of plaintexts-ciphertexts $W'_\varDelta$ defined as $W_\varDelta$ but with a further condition on the plaintexts:

$$\begin{split} W'_\varDelta = \big\{ (p^i, c^i) \text{ for } i = 0, ..., 2^8 - 1 \,\big|\, \text{ for each } i : \quad (p^i, c^i) \in W_\varDelta \text{ and } \\ 2^{64} \cdot \delta \leq f(p^i_{1,0}, p^i_{2,0}, p^i_{3,0}, p^i_{0,1}, p^i_{2,1}, ..., p^i_{2,3}) < 2^{64} \cdot (1 + \delta) \big\}, \end{split} \tag{23}$$

where $W_\varDelta$ is defined as in (10.4), the input of the function f are all the bytes of p^i that don't lie on the first diagonal (that is $p^i_{j,k}$ for $j \neq k$) and where δ is defined as follows:

$$\delta := f(\delta_0, \delta_1, \delta_2, \delta_3). \tag{24}$$

[12] Note that each set $W_\varDelta$ contains 2^8 elements, for a total of $2^{96} \cdot 2^8 = 2^{104}$ plaintexts instead of 2^{128}. This is due to the definition of $W_\varDelta$. To fix this problem, in the definition of $W_\varDelta$ (10.4), one can substitute $\left[x, 0, 0, 0\right]^T$ with $\left[x, \hat{c}_1, \hat{c}_2, \hat{c}_3\right]^T$ where $\hat{c}_i$ are constants in $\mathbb{F}_{2^8}$ to have all the 2^{128} plaintexts. However, this is not necessary for our scope. Thus, we consider for simplicity $\hat{c}_1 = \hat{c}_2 = \hat{c}_3 = 0$.

Thus, the distinguisher is set up using W'_Δ instead of W_Δ. Note that for each collection (that is, for each combination of values of $\Delta = (\delta_0, ..., \delta_3)$), we need about $2^{19.7}$ different sets W'_Δ. If the previous restriction (23) on the plaintext holds, then there are 2^{64} different sets W'_Δ for each combination of values of $\delta_0, ..., \delta_3$. Since we need only $2^{19.7}$ of them, everything works. Moreover, it is simply to observe that using the previous restriction and given two plaintexts p and q with $p_{i,j} = q_{i,j}$ for each $i \neq j$, it is very easy to establish if the values of $\delta_0, ..., \delta_3$ for the first plaintext are the same of those of the second one. In particular, it is sufficient to compute the corresponding values of $f(p_{1,0}, ..., p_{2,3})$ and $f(q_{1,0}, ..., q_{2,3})$ and to check if they belong to the same interval defined as $(2^{64} \cdot \delta, 2^{64} \cdot (\delta + 1))$. If they belong to the same interval, then the two plaintexts belong to the same set W'_Δ (that is, the values of $\delta_0, ..., \delta_3$ are equal), otherwise they belong to different sets W'_{Δ^1} and W'_{Δ^2}.

Thus, $2^{59.7}$ chosen plaintexts (that is 2^{32} collections - one for each possible combination of values of Δ - each one of $2^{19.7}$ different sets W'_Δ) are sufficient to set up the distinguisher. The cost to construct them is of $2^{59.7}$ encryptions or oracle queries and the verification cost is well-approximated by $2^{32} \cdot 2^{19.7} \cdot 2^7 \cdot (2^8 - 1) = 2^{66.7}$ table look-ups (the cost to divide the texts in the corresponding collection and sets W'_Δ is well-approximated by $2^{59.7}$ table look-ups). Thus, this distinguisher on 5-rounds AES with secret key is (much) better than the previous one which exploits a different 0-probability subspace trail.

```
Data: $2^{32}$ collections (one for each possible value of $\Delta$. Each collection contains $2^{19.7}$ different
      sets $W'_\Delta$ defined as in (23).
Result: $\Delta$ if the permutation is an AES permutation (where $\delta_i = k_{i,i}$ for $i = 0, ..., 3$); $-1$ if the
        permutation is a Random one.
for $\Delta$ from 0 to $2^{32} - 1$ do
    flag = 0;
    for each one of the $2^{19.7}$ different sets $W'_\Delta$ do
        for each pair $(c^i, c^j) \in W'_\Delta$ do                    // about $2^{15}$ different pairs
            if $c^i \oplus c^j \in \mathcal{M}_J$ for $|J| = 3$ then        // e.g. see Algorithm 3
                flag = 1;
                next collection (i.e. next $\Delta$);
            end
        end
    end
    if flag = 0 then                                              // AES permutation
        return $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$;
    end
end
return $-1$.                                                      // Random permutation
```

Algorithm 7: Distinguisher for 5-rounds of AES with data-complexity of $2^{59.7}$ - Pseudo-code. The $2^{59.7}$ input texts are already divided in 2^{32} collections (one for each Δ), and for each collection the texts are already divided in the sets W'_Δ. Given a pair (c^i, c^j), use for example Algorithm 3 to check if $c^i \oplus c^j$ belongs (or not) in $\mathcal{M}_J$ for $|J| = 3$.

10.3 Practical Verification

Since the complexity of the 5 rounds distinguisher is very high, we have practical verified it on a small scale variant of AES presented in [12]. In the actual AES, each word of AES is composed of 8 bits. In our variant, each word of AES is composed of 4 bits. We refer to the above mentioned paper for a complete description of this small-scale AES, and we limit ourselves to describe the theoretical result of our 5-rounds distinguisher in this case. Our implementation can be found in [2].

First of all, since the words are composed of 4 bits instead of 8, the probability p that two texts belong to the same coset of $\mathcal{W}_I$ for $|I| = 3$ is $p = 4 \times (2^4)^{-4} = 2^{-14}$. Moreover, each set W'_Δ contains 2^4 different texts (that is, approximately $2^3 \cdot (2^4 - 1) = 2^7$ different pairs), and finally there are $(2^4)^4 = 2^{16}$ different combinations of Δ instead of 2^{32}.

As before, we have to guarantee that in the random case and for each one of the 2^{16} collections (one for each combinations of Δ), there is at least one collision with a total probability of 95%. Using similar computations as before, we need approximately $2^{18.7}$ pairs for each Δ, and in the random case we expect for each Δ on average $2^{18.7} \cdot (4 \cdot 2^{-16}) = 2^{4.7} = 26$ collisions. Since each set W'_Δ contains 2^7 different pairs, $2^{18.7} \cdot 2^{-7} = 2^{11.7}$ different sets W'_Δ are sufficient. Thus, the total number of different texts required for this distinguisher on the 5-rounds small-scale AES is $2^{16} \cdot 2^{11.7} \cdot 2^4 = 2^{31.7}$, and the computational cost of the verification can be approximated at $2^{34.7}$ table look-ups (which is the cost to construct all the pairs).

The above computational cost just given is actually only an upper bound. Indeed, observe that for our goal it is sufficient to find one collision, that is when the first collision is found then it is not necessary to look for others. Thus, for each Δ the upper bound for table look-ups is $2^{18.7}$, but in our experiments the average number of table look-ups in order to find the first collision is $2^{13.5}$. Note that this number is consistent with the average number of collisions that we expect for each Δ (indeed, since we expect 26 collisions, on average $2^{18.7}/26 \simeq 2^{14}$ pairs of texts are sufficient to find the first collision). Thus, in our experiments, the average number of table look-ups is much lower that $2^{34.7}$ and is well approximated by 2^{30}.

The practical results are consistent with our theory. In particular, given an AES permutation, our distinguisher is always able to identify it (i.e. with probability 1, there is one value of Δ for which there is no collision and which corresponds to the four byte of the secret key). Instead, given a random permutation, the practical probability that for each $\Delta = (\delta_0, ..., \delta_3)$ there is at least one collision is close to 100% (in 993 out of 1000 of experiments).

10.4 Some Variants

We show that this distinguisher works in the same way also in the case in which the final MixColumns is omitted, and also in the decryption mode (i.e. with chosen ciphertexts instead of chosen plaintexts) independent of the presence of the final MixColumns operation. For both cases, the distinguisher works as before, with the same data complexity. Thus, we limit ourselves to highlight the major differences.

A Distinguisher on 4.5-Rounds of AES with Data Complexity of $2^{59.7}$ Chosen-Plaintexts. If the final MixColumns operations is omitted, then the distinguisher works in the same way as before. The only difference is that one has to consider the space $\mathcal{ID}_J$

instead of final space $\mathcal{M}_J$ for $|J| = 3$. That is:

$$Pr(R^{(5)}(x) \oplus R^{(5)}(y) \in \mathcal{ID}_J \,|\, x \oplus y \in W'_\Delta) = 0,$$

if $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ with $\delta_i = k_{i,i}$ for each i, and where $|J| = 3$ and W'_Δ is defined as in (23).

A Distinguisher on 4.5-Rounds of AES with Data Complexity of $2^{59.7}$ Chosen-Ciphertext. For the decryption case, the idea is to define a set of plaintexts-ciphertexts Z'_Δ (analogous of W'_Δ) such that the ciphertexts belong to the same coset of $\mathcal{M}_i$ one rounds before for $|i| = 1$. Due to the impossible subspace trail, the probability that a pair of corresponding plaintexts belong to the same coset of $\mathcal{D}_J$ for $|J| = 3$ is 0. Thus, the distinguisher works exactly as before and it has the same complexity of the one just described. For this reason, we limit ourselves only to define the set of plaintext-ciphertext Z_Δ (analogous of W_Δ).

The definition of Z_Δ (analogous of W_Δ) is easy if the final MixColumns operation is omitted. In particular, in the case in which the final MixColumns operation is omitted, the set Z_Δ is defined as:

$$Z_\Delta = \left\{ (p^i, c^i) \text{ for } i = 0, ..., 2^8 - 1 \,\middle|\, \begin{bmatrix} c^i_{0,0} \\ c^i_{1,3} \\ c^i_{2,2} \\ c^i_{3,1} \end{bmatrix} := \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \oplus \begin{bmatrix} \text{S-Box}(\alpha \cdot x) \\ \text{S-Box}(x) \\ \text{S-Box}(x) \\ \text{S-Box}((\alpha + 1) \cdot x) \end{bmatrix}, \right.$$
$$\left. \text{for } x \in \mathbb{F}_{2^8} \text{ and } c^i_{k,l} = c^j_{k,l} \; \forall k, l \text{ s.t. } k + l \neq 0 \pmod 4 \text{ and } \forall i \neq j \right\},$$

where $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ can assume 2^{32} different values, Z_Δ contains 2^8 different pairs (p, c) (that is, approximately 2^{15} different pairs) and the 256 vectors defined as $\left[\text{S-Box}(\alpha \cdot x), \text{S-Box}(x), \text{S-Box}(x), \text{S-Box}((\alpha + 1) \cdot x)\right]^T$ can be simply precomputed in advance and stored in a table.

Proposition 3. *Let Z_Δ defined as before. If $\delta_0 = k^5_{0,0}$, $\delta_1 = k^5_{1,3}$, $\delta_2 = k^5_{2,2}$ and $\delta_3 = k^5_{3,1}$ (where k^5 denote the key of the final round), then there exists (unique) $a \in (\mathcal{M}_0 \cap \mathcal{C}_0)^\perp$ such that $R_f^{-1}(Z_\Delta) \subseteq (\mathcal{M}_0 \cap \mathcal{C}_0) \oplus a$.*

The proof follows immediately by the definition of Z_Δ. Given Z_Δ, Z'_Δ is defined as:

$$Z'_\Delta = \big\{ (p^i, c^i) \text{ for } i = 0, ..., 2^8 - 1 \,\big|\, \text{for each } i: \quad (p^i, c^i) \in Z_\Delta \text{ and}$$
$$2^{64} \cdot \delta \leq f(c^i_{1,0}, c^i_{2,0}, c^i_{3,0}, p^i_{0,1}, c^i_{1,1}, c^i_{3,1}, ..., c^i_{3,3}) < 2^{64} \cdot (1 + \delta) \big\}, \tag{25}$$

where Z_Δ is defined as before, the input of the function f are all the bytes of p^i that don't lie on the first inverse diagonal (that is $c^i_{j,k}$ for $j + k \neq 0 \pmod 4$) and where δ is equal to $\delta := f(\delta_0, \delta_1, \delta_2, \delta_3)$.

A Distinguisher on 5-Rounds of AES with Data Complexity of $2^{59.7}$ Chosen-Ciphertext. We consider now the case in which the final MixColumns operation is not omitted. The distinguisher works exactly as before, but the definition of the Z_Δ is a little more complicated. The idea is to swamp the position of the final MixColumns operations and of the final AddRoundKeys operation. Thus, in order to define the set

Z_Δ in the case in which in the final MixColumns is not omitted, the idea is to apply the MixColumns operation to the set Z_Δ found in the previous subsection. Moreover, also the values of δ_i for which there are no collisions with probability 1 in the corresponding sets are defined in a different way.

First of all, we define some matrices which are useful for the following definition of Z_Δ. For each $x \in \mathbb{F}_{2^8}$, let $\Gamma(x)$ defined as:

$$\Gamma(x) := MC \cdot \begin{bmatrix} \text{S-Box}(\alpha \cdot x) & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{S-Box}(x) \\ 0 & 0 & \text{S-Box}(x) & 0 \\ 0 & \text{S-Box}((\alpha+1)\cdot x) & 0 & 0 \end{bmatrix}$$

where MC is the MixColumns operation. Note that these 256 matrices can be simply precomputed in advance and stored in a table. Moreover, let $\Psi \in \mathbb{F}_{2^8}^{4\times 4}$ defined as:

$$\Psi := MC \cdot \begin{bmatrix} 0 & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & 0 \\ a_{2,0} & a_{2,1} & 0 & a_{2,3} \\ a_{3,0} & 0 & y_{3,2} & a_{3,3} \end{bmatrix}$$

or equivalently:

$$\Psi \equiv \begin{bmatrix} \psi(a_{1,0}, a_{2,0}, a_{3,0}) & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & \psi(a_{2,3}, a_{3,3}, a_{0,3}) \\ a_{2,0} & a_{2,1} & \psi(a_{1,2}, a_{3,2}, a_{0,2}) & a_{2,3} \\ a_{3,0} & \psi(a_{0,1}, a_{1,1}, a_{2,1}) & a_{3,2} & a_{3,3} \end{bmatrix}$$

where $a_{i,j}$ are in $\mathbb{F}_{2^8}$ and where the function $\psi(\cdot,\cdot,\cdot)$ is defined as:

$$\psi(x,y,z) = (\alpha^6+\alpha^4+\alpha^3+\alpha^2+1)\cdot x \oplus (\alpha^5+\alpha^3+\alpha^2+\alpha+1)\cdot y \oplus (\alpha^7+\alpha^3+\alpha^2)\cdot z.$$

Given $\Gamma(x)$ and Ψ as before, in the case in which the final MixColumns operation is not omitted and for a fixed Ψ, the set Z_Δ is defined as

$$Z_\Delta = \left\{ (p^i, c^i) \text{ for } i = 0, ..., 2^8-1 \,\middle|\, c^i = \Gamma(x) \oplus MC \cdot \begin{bmatrix} \delta_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \delta_1 \\ 0 & 0 & \delta_2 & 0 \\ 0 & \delta_3 & 0 & 0 \end{bmatrix} \oplus \Psi \right.$$
$$\left. \text{for } x \in \mathbb{F}_{2^8} \text{ and for a } \textit{fixed } \Psi \right\},$$

where $\Delta = (\delta_0, \delta_1, \delta_2, \delta_3)$ can assume 2^{32} different values and Z_Δ contains 2^8 different pairs (p, c) (that is, approximately 2^{15} different pairs). Note that the condition that Ψ if fixed for a given set Z_Δ is equivalent to the condition $c^i_{k,l} = c^j_{k,l}$ for $k + l \neq 0 \pmod 4$ given in the previous definition of Z_Δ where the final MixColumns operation is not omitted.

Proposition 4. *Let Z_Δ defined as before. If*

$$\begin{aligned} \delta_0 &= 0x0e \cdot k^5_{0,0} \oplus 0x0b \cdot k^5_{1,0} \oplus 0x0d \cdot k^5_{2,0} \oplus 0x09 \cdot k^5_{3,0}, \\ \delta_1 &= 0x09 \cdot k^5_{0,3} \oplus 0x0e \cdot k^5_{1,3} \oplus 0x0b \cdot k^5_{2,3} \oplus 0x0d \cdot k^5_{3,3}, \\ \delta_2 &= 0x0d \cdot k^5_{0,2} \oplus 0x09 \cdot k^5_{1,2} \oplus 0x0e \cdot k^5_{2,2} \oplus 0x0b \cdot k^5_{3,2}, \\ \delta_3 &= 0x0b \cdot k^5_{0,1} \oplus 0x0d \cdot k^5_{1,1} \oplus 0x09 \cdot k^5_{2,1} \oplus 0x0e \cdot k^5_{3,1}, \end{aligned}$$

where k^5 is the secret of the final round, then there exists (unique) $a \in (\mathcal{M}_0 \cap \mathcal{C}_0)^\perp$ such that $R^{-1}(Z_\Delta) \subseteq (\mathcal{M}_0 \cap \mathcal{C}_0) \oplus a$.

For completeness, $0x09 \equiv \alpha^3 + 1$, $0x0b \equiv \alpha^3 + \alpha + 1$, $0x0d \equiv \alpha^3 + \alpha^2 + 1$ and $0x0e \equiv \alpha^3 + \alpha^2 + \alpha$. The proof follows immediately by the definition of Z_Δ.

Proof. To prove the proposition, we simply compute $R^{-1}(Z_\Delta)$. By simple computation, after the AddRoundkey and the inverse MixColumns operations, the first column is equal to:

$$\begin{bmatrix} \text{S-Box}(\alpha \cdot x) \oplus \delta_0 \oplus 0x0e \cdot k^5_{0,0} \oplus 0x0b \cdot k^5_{1,0} \oplus 0x0d \cdot k^5_{2,0} \oplus 0x09 \cdot k^5_{3,0} \\ a_{1,0} \oplus 0x09 \cdot k^5_{0,0} \oplus 0x0e \cdot k^5_{1,0} \oplus 0x0b \cdot k^5_{2,0} \oplus 0x0d \cdot k^5_{3,0} \\ a_{2,0} \oplus 0x0d \cdot k^5_{0,0} \oplus 0x09 \cdot k^5_{1,0} \oplus 0x0e \cdot k^5_{2,0} \oplus 0x0b \cdot k^5_{3,0} \\ a_{3,0} \oplus 0x0b \cdot k^5_{0,0} \oplus 0x0d \cdot k^5_{1,0} \oplus 0x09 \cdot k^5_{2,0} \oplus 0x0e \cdot k^5_{3,0} \end{bmatrix}$$

for $x, a_{i,0} \in \mathbb{F}_{2^8}$. Note that $a_{i,0}$ for $i = 1, 2, 3$ are equal for each texts in Z_Δ. Since $\delta_0 = 0x0e \cdot k^5_{0,0} \oplus 0x0b \cdot k^5_{1,0} \oplus 0x0d \cdot k^5_{2,0} \oplus 0x09 \cdot k^5_{3,0}$, then the first element of the first row is equal to S-Box$(\alpha \cdot x)$. The other three columns are analogous.

After the inverse ShiftRows operation, the second, the third and the fourth columns are constant and equal for each texts in Z_Δ (only for completeness, note that they are unknown). Finally, after the inverse SubBytes operation, the first column is given by:

$$\left[\alpha \cdot x, x, x, (\alpha + 1) \cdot x\right]^T,$$

that is the thesis. □

Finally, the definition of Z'_Δ is equal to the previous one (25).

As for the encryption case, we expect that there is a combination of values of $(\delta_0, ..., \delta_3)$ for which for each set Z'_Δ there is no collision among the plaintexts in the same coset of $\mathcal{D}_I$ for $|I| = 3$ with probability 1. For the random permutation, we expect to have at least one collision among the plaintexts in at least one set Z'_Δ for each one of the 2^{32} values of Δ. Thus, given $2^{59.7}$ texts (that is, 2^{32} collections - one for each Δ, each one with $2^{19.7}$ different sets Z'_Δ), it is possible to distinguish the two cases.

10.5 Comparison with 5-Rounds Distinguisher proposed by Sun, Liu, Guo, Qu and Rijmen, and Possible Generalizations

At CRYPTO 2016, the first 5-rounds secret key distinguishers of AES-128 have been presented [30]. In that paper, authors construct a 5-rounds *zero-correlation linear hulls* for AES, and then use it to construct 5-rounds integral distinguisher for AES. First, they present them in the case in which the difference of two sub-key bytes is known (in this case, the distinguisher requires 2^{120} texts). Then they extend it to the general case, i.e. they prove that it is always possible to distinguish 5 rounds of AES from random permutations even when the difference of the sub-keys is unknown (in this case, the distinguisher requires 2^{128} texts, i.e. the entire input-output space). We refer to [30] for more details.

In some sense, the procedure that we have used to set up our distinguisher is similar to the one used in [30]. In particular, in our cases we have extended the impossible subspace trail distinguisher on 4-rounds presented in Sect. 5.3 at the beginning (i.e. an impossible differential distinguisher), instead of a zero-sum distinguisher.

The first major difference between our distinguisher and the one presented in [30] is that our distinguisher works both in the encryption mode (i.e. chosen plaintexts) and in the decryption mode (i.e. chosen ciphertexts), independent of the presence of the final MixColumns operation and with the same data complexity. Instead, the distinguisher presented in [30] works only in the decryption mode and only if the final MixColumns operation is not omitted.

The second major difference between our distinguisher and the one presented in [30] is the data complexity and the verification cost. For our distinguisher, the cost to construct all the plaintext-ciphertext pairs is of $2^{59.7}$ encryptions or oracle queries. For comparison, in [30], since the entire input-output space is required, the cost is of 2^{128} encryptions or oracle queries. Consider instead the cost to check that the presence of at least one collision. To do this check, one has to construct all the possible pairs and to check that there is at least one pair that collides in the same coset of $\mathcal{M}_J$ for $|J| = 3$ (note that when the first collision is found, one can consider the next collection). First of all, given a pair of texts, the cost to verify that they collide in the same coset of $\mathcal{M}_J$ is approximate the cost of one XOR operation and the cost of one inverse MixColumns operation. Since the cost of these two operations is negligible compared to a table look-ups, the total cost can be approximated by the cost to construct all the pairs. Note that one has to construct only the pairs of texts that belong to the same coset of W'_Δ. Thus, the cost for this step can be approximated by 2^{32} (number of Δ) $\cdot 2^{19.7}$ (number of sets)$\cdot 2^7 \cdot (2^8 - 1)$ (number of pairs) $\simeq 2^{64.7}$ table look-ups and inverse MixColumns operations. For the distinguisher presented in [30], the cost to do the verification operation can be approximated to 2^{128} XOR operations.

Besides the previous ones, some other more important differences can be highlighted among these two distinguishers. The distinguisher presented in [30], besides needing the entire input-output space, requires some important assumptions on the components of the MixColumns operation and on the SubBytes one, that is:

- the encryption scheme has to adopt *identical S-Boxes*;
- *at least one column of the MixColumns matrix MC* (or its inverse) *has to contain* (at least) *two identical elements.*

Our distinguisher doesn't require such assumptions. Indeed, the sets W'_Δ and Z'_Δ can be constructed in a similar way even in the case the used S-Boxes are not identical and for each possible kind of invertible MixColumns matrix MC.

Finally, a more detailed comparison between the distinguisher presented in [30] and (a modified version of) our distinguisher on 5 rounds is presented in App. B.

11 Conclusion

We have proposed a generalization of invariant subspace cryptanalysis: *Subspace-Trail Cryptanalysis*. In terms of secret-key distinguishers, compared to other attack vectors and the state-of-the-art so far, it's application to 1-4 rounds of AES leads to similar or identical distinguishers. However for 5-round of AES we reported the by far best distinguisher known. We've also described new low data-complexity key-recovery attacks on AES up to 4 rounds, based on a combination of a truncated differential property (i.e. a relation among pairs of texts) and of properties of individual texts that follow naturally from the proposed subspace trail approach. It is conceivable that such attacks are also found without the subspace trail approach (truncated differential + ad-hoc optimizations of key-recovery method that go beyond looking at the differences only), but the combination of properties of individual texts and sums of text follows more naturally from the subspace trail approach.

Future work includes trying to exploit the subspace properties in other ways to get more efficient or longer distinguishers, perhaps by considering also S-Box properties, to use this approach to devise more key-recovery attacks and to apply the approach to other schemes.

Acknowledgments The work in this paper has been partially supported by the Austrian Science Fund (project P26494-N15). We thank Bing Sun for helpful discussions on [30].

References

1. "Low data-complexity attacks on up to 4-round AES," 2016, https://github.com/Krypto-iaik/LowDataAttacks_AES.
2. "Verification of all distinguishers up to 5-rounds AES," 2016, https://github.com/Krypto-iaik/Distinguishers_AES.
3. B. Bahrak and M. R. Aref, "Impossible differential attack on seven-round AES-128." *IET Information Security*, vol. 2, no. 2, pp. 28–32, 2008.
4. E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," in *Advances in Cryptology — EUROCRYPT 1999: International Conference on the Theory and Application of Cryptographic Techniques, Czech Republic. Proceedings*, J. Stern, Ed., 1999, pp. 12–23.
5. E. Biham and N. Keller, "Cryptanalysis of Reduced Variants of Rijndael," unpublished, 2001, http://csrc.nist.gov/archive/aes/round2/conf3/papers/35-ebiham.pdf.
6. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.
7. A. Biryukov and D. Khovratovich, "Two New Techniques of Side-Channel Cryptanalysis," in *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop, Austria. Proceedings*, 2007, pp. 195–208.
8. A. Biryukov and A. Shamir, "Structural Cryptanalysis of SASAS," *Journal of Cryptology*, vol. 23, no. 4, pp. 505–518, 2010.
9. A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full AES," in *Advances in Cryptology – ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, South Korea. Proceedings*, 2011, pp. 344–371.
10. C. Bouillaguet, P. Derbez, O. Dunkelman, P. Fouque, N. Keller, and V. Rijmen, "Low-Data Complexity Attacks on AES," *IEEE Trans. Information Theory*, vol. 58, no. 11, pp. 7002–7017, 2012.
11. C. Bouillaguet, P. Derbez, and P. Fouque, "Automatic search of attacks on round-reduced AES and applications," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA. Proceedings*, 2011, pp. 169–187.
12. C. Cid, S. Murphy, and M. J. B. Robshaw, "Small Scale Variants of the AES," in *Fast Software Encryption - FSE 2005: 12th International Workshop, France: Revised Selected Papers*, vol. 3557, 2005, pp. 145–162.
13. J. Daemen, L. R. Knudsen, and V. Rijmen, "The block cipher square," in *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 1267. Springer, 1997, pp. 149–165. [Online]. Available: http://dx.doi.org/10.1007/BFb0052343
14. J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, ser. Information Security and Cryptography. Springer, 2002.
15. ——, "Two-Round AES Differentials," Cryptology ePrint Archive, Report 2006/039, 2006.
16. ——, "Understanding Two-Round Differentials in AES," in *Security and Cryptography for Networks 2006*, vol. 4116, 2006, pp. 78 – 94.
17. H. Demirci and A. A. Selçuk, "A meet-in-the-middle attack on 8-round AES," in *Fast Software Encryption - FSE 2008: 15th International Workshop, Switzerland. Revised Selected Papers*, 2008, pp. 116–126.
18. P. Derbez and P. Fouque, "Exhausting Demirci-Selçuk Meet-in-the-Middle Attacks Against Reduced-Round AES," in *Fast Software Encryption - FSE 2013: 20th International Workshop, Singapore. Revised Selected Papers*, 2013, pp. 541–560.

19. P. Derbez, P. Fouque, and J. Jean, "Improved key recovery attacks on reduced-round AES in the single-key setting," in *Advances in Cryptology - EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Greece. Proceedings*, 2013, pp. 371–387.
20. O. Dunkelman and N. Keller, "The Effects of the Omission of Last Round's MixColumns on AES," *Inf. Process. Lett.*, vol. 110, no. 8-9, pp. 304–308, 2010.
21. O. Dunkelman, N. Keller, and A. Shamir, "Improved single-key attacks on 8-round AES-192 and AES-256," in *Advances in Cryptology - ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore. Proceedings*, 2010, pp. 158–176.
22. J. Evertse, "Linear Structures in Blockciphers," in *Advances in Cryptology - EUROCRYPT 1987: Workshop on the Theory and Application of of Cryptographic Techniques, Netherlands. Proceedings*, 1987, pp. 249–266.
23. J. Guo, J. Jean, I. Nikolic, K. Qiao, Y. Sasaki, and S. M. Sim, "Invariant Subspace Attack Against Full Midori64," Cryptology ePrint Archive, Report 2015/1189, 2015.
24. L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption - FSE 1994: Second International Workshop Leuven, Belgium. Proceedings*, 1995, pp. 196–211.
25. G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner, "A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack," in *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, 2011. Proceedings*, 2011, pp. 206–221.
26. G. Leander, B. Minaud, and S. Rønjom, "A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro," in *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Bulgaria. Proceedings, Part I*, 2015, pp. 254–283.
27. J. Lu, O. Dunkelman, N. Keller, and J. Kim, *Progress in Cryptology - INDOCRYPT 2008: 9th International Conference on Cryptology in India, India. Proceedings*, 2008, ch. New Impossible Differential Attacks on AES, pp. 279–293.
28. H. Mala, M. Dakhilalian, V. Rijmen, and M. Modarres-Hashemi, "Improved impossible differential cryptanalysis of 7-round AES-128," in *Progress in Cryptology - INDOCRYPT 2010: 11th International Conference on Cryptology in India, India. Proceedings*, 2010, pp. 282–291.
29. S. Park, S. H. Sung, S. Chee, E. Yoon, and J. Lim, "On the Security of Rijndael-Like Structures against Differential and Linear Cryptanalysis," in *Advances in Cryptology - ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, New Zealand. Proceedings*, 2002, pp. 176–191.
30. B. Sun, M. Liu, J. Guo, L. Qu, and V. Rijmen, "New insights on AES-like SPN ciphers," To appear at CRYPTO 2016, Cryptology ePrint Archive, Report 2016/533, 2016.
31. B. Sun, Z. Liu, V. Rijmen, R. Li, L. Cheng, Q. Wang, H. Alkhzaimi, and C. Li, "Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis," in *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA. Proceedings*, 2015, pp. 95–115.
32. T. Tiessen, "Polytopic Cryptanalysis," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Austria. Proceedings, Part I*, 2016, pp. 214–239.
33. Y. Todo, "Integral cryptanalysis on full MISTY1," in *Advances in Cryptology - CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA. Proceedings, Part I*, 2015, pp. 413–432.
34. ——, "Structural evaluation by generalized integral property," in *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Bulgaria. Proceedings, Part I*, 2015, pp. 287–314.

A Attacks on 3 Rounds of AES - dim($\mathcal{D}_I$) = 12

In this section, we present the attack on 3 rounds of AES in the case in which $\dim(\mathcal{D}_I) = 12$. The computational cost of the attack on 3 rounds for this case is higher than for the case in which $\dim(\mathcal{D}_I) = 4$. However, the 4-round attack presented in Sect. 9 is obtained extending the attack on 3 rounds of this section at the beginning. Indeed, it is possible to prove that the situation is completely different (in particular, it is the opposite) when we consider the extension to 4 rounds, adding an initial round.

For simplicity, we consider only the case $I = \{0, 1, 2\}$. By definition, $\mathcal{D}_I$ is the subspace with zero-elements on the fourth diagonal. For all $a \in \mathcal{D}_I^{\perp}$ there exists unique $b \in \mathcal{M}_I^{\perp}$ such that $R^{(2)}(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b$, where

$$\mathcal{M}_I \equiv \begin{bmatrix} a_1(x_1, x_6, x_{11}) & a_2(x_4, x_5, x_{10}) & a_3(x_3, x_8, x_9) & a_4(x_2, x_7, x_{12}) \\ a_2(x_1, x_6, x_{11}) & a_3(x_4, x_5, x_{10}) & a_4(x_3, x_8, x_9) & a_1(x_2, x_7, x_{12}) \\ a_3(x_1, x_6, x_{11}) & a_4(x_4, x_5, x_{10}) & a_1(x_3, x_8, x_9) & a_2(x_2, x_7, x_{12}) \\ a_4(x_1, x_6, x_{11}) & a_1(x_4, x_5, x_{10}) & a_2(x_3, x_8, x_9) & a_3(x_2, x_7, x_{12}) \end{bmatrix},$$

and where $a_i(\cdot, \cdot, \cdot)$ are defined in the following way ($\forall i = 1, 2, 3, 4$):

$$a_1(x, y, z) = \alpha x \oplus (\alpha + 1)y \oplus z, \qquad a_2(x, y, z) = x \oplus \alpha y \oplus (\alpha + 1)z,$$
$$a_3(x, y, z) = x \oplus y \oplus \alpha z, \qquad a_4(x, y, z) = (\alpha + 1)x \oplus y \oplus z.$$

As before, given two ciphertexts c^1 and c^2, the idea is to find all the keys of the final round such that $R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I$. As we've seen, the right key is the only one that satisfies this previous condition for each pair p^1 and p^2 such that $p^1 \oplus p^2 \in \mathcal{D}_I$. The idea is to work again independently on each column, but in this case the attacker has to guess 3 bytes for each column. Thus, the number of possible keys found at each step is higher than before: this explains why the total computational cost and the number of requested chosen plaintexts is higher.

Since the attack is equivalent to the previous one, we only show which conditions the key bytes have to satisfy in order to guarantee that $R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I$ for the first column. Suppose the attacker guesses (for example) the bytes $k_{0,0}, k_{1,3}$ and $k_{2,2}$. By simple computation, $R_f^{-1}(c^1) \oplus R_f^{-1}(c^2) \in \mathcal{M}_I$ if $k_{3,1}$ satisfies the following equivalence:

$$\begin{aligned} &\text{S-Box}^{(-1)}(c^1_{3,1} \oplus k_{3,1}) \oplus \text{S-Box}^{(-1)}(c^2_{3,1} \oplus k_{3,1}) = \\ &= (\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + 1) \cdot [\text{S-Box}^{(-1)}(c^1_{0,0} \oplus k_{0,0}) \oplus \text{S-Box}^{(-1)}(c^2_{0,0} \oplus k_{0,0})] \oplus \\ &\oplus (\alpha^5 + \alpha^4 + \alpha^2 + 1) \cdot [\text{S-Box}^{(-1)}(c^1_{1,3} \oplus k_{1,3}) \oplus \text{S-Box}^{(-1)}(c^2_{1,3} \oplus k_{1,3})] \oplus \\ &\oplus (\alpha^7 + \alpha^3 + \alpha^2) \cdot [\text{S-Box}^{(-1)}(c^1_{2,2} \oplus k_{2,2}) \oplus \text{S-Box}^{(-1)}(c^2_{2,2} \oplus k_{2,2})]. \end{aligned} \tag{26}$$

If ciphertexts satisfy conditions similar to (11), the attacker is able to reduce the number of possible keys to $2^{128} \times (2^8)^{-4} = 2^{96}$, with a total computational cost of about 2^{31} S-Box look-ups.

As before, the idea is to eliminate some of the keys found in the previous step using other pairs of ciphertexts, and the attacker can take advantage of the independence of the columns to perform this step with a low computational cost. In particular, the probability that each column of the key found previously satisfies (26) for another pair of ciphertexts is on average 2^{-8}. Thus, using a second pair of plaintexts, the attacker reduces the keys to $2^{96} \times (2^8)^{-4} = 2^{64}$ with a computational cost of 2^{27} S-Box look-ups. Thus, the attacker needs other two pairs of plaintexts to discover the secret key (with a total computational cost of about $2^{19} + 2^{11}$ S-Box look-ups). Performing these steps at the same time, that is checking the combinations found with the first pair of plaintexts immediately with the other ones, allows to save memory.

Note that using 5 chosen plaintexts the attacker can construct 10 different pairs, but only 4 of them are useful for the attack. For example, suppose that the attacker uses a (first) pair formed by the first and the by second ciphertext, and a (second) that formed by the first and by the third ciphertext. Then the keys that satisfy (26) for these two pairs, automatically satisfy (26) for the pair formed by the second and by the third ciphertext. However, using four pairs of plaintexts with one plaintext in common (that is, 5 different chosen plaintexts), the probability of success is greater than 99.9%.

In conclusion, this attack needs 4 pairs of plaintexts, that is 5 chosen plaintexts, the total computational cost is approximately at $2^{31.09}$ S-Box look-ups, that is about $2^{25.18}$ executions of the three-round cipher (the memory cost is negligible). Performing the attack using memory access, the total computational cost is approximately at $2^{24.6}$ memory access, $2^{24.18}$ S-Box look-ups (that is about $2^{18.27}$ executions of the three-round cipher) and the memory cost is approximately 2^{16} bytes, and the cost of the precomputation is of $2^{10.1}$ executions of three-round AES.

B A 5-Rounds Secret Key Distinguisher for AES with Data Complexity of $2^{98.2}$

Even though we present in the main part of the paper a distinguisher that is strictly better with respect to all properties than the one in this appendix, it may well be instructive to the reader as it simpler.

The technique used to build it is similar to the one used in Sect. 10. That is, first we suppose to known the difference of two sub-key bytes (that is $\varDelta := k_{0,0} \oplus k_{1,1}$), and then we generalize the distinguisher in the case in which no information on the secret key is known. In this case, only 8 bits of the secret key are guessing instead of 32, and the idea is to look for plaintexts that belong to the same coset of $\mathcal{D}_i$ after one round with $|i| = 1$.

As we show in detail in the following, this distinguisher is worse than the previous one, due to the following reasons:

- it requires $2^{98.2}$ chosen plaintexts instead of $2^{59.7}$ (also the computational cost for the verification cost is higher);
- it doesn't work in the decryption mode (that is, with chosen ciphertexts instead of chosen plaintexts);
- some assumptions on the MixColumns matrix MC are necessary.

However, we have decided to present it since it provides an interesting comparison with the first 5-rounds secret key distinguisher presented in [30], and it highlights one more times the quality of the distinguisher presented in Sect. 10 with respect to the one presented in [30].

B.1 The Difference of Two Sub-Key Bytes is Known

Suppose for the moment to know the difference of two sub-key bytes, that is $\varDelta := k_{0,0} \oplus k_{1,1}$. As we've already said, the idea is to extend at the beginning the 4-rounds distinguisher based on impossible differential presented in Sect. 5.3. To do this, the idea is to choose plaintexts that belong to the same coset of $\mathcal{D}_I$ for a certain I after one round.

Consider a set of plaintexts-ciphertexts $V_\varDelta$ of the form[13]:

$$\begin{aligned} V_\varDelta = &\{(p^i, c^i) \text{ for } i = 0, ..., 2^8 - 1 \,|\, p^i_{0,0} \oplus p^i_{1,1} = \varDelta \quad \forall i \quad \text{and} \\ &\text{and} \quad p^i_{k,l} = p^j_{k,l} \, \forall (k,l) \neq \{(0,0),(1,1)\} \text{ and } i \neq j\}, \end{aligned} \tag{27}$$

[13] In [30], authors consider a set of plaintexts-ciphertexts $\tilde{V}_{\tilde{\varDelta}}$ of the form $\tilde{V}_{\tilde{\varDelta}} = \{(p,c) \,|\, c_{0,0} \oplus c_{1,3} = \tilde{\varDelta}\}$ where $\tilde{\varDelta} = k_{0,0} \oplus k_{1,3}$ and with anyone assumptions on the other bytes. Note that $|\tilde{V}_{\tilde{\varDelta}}| = 2^{120}$.

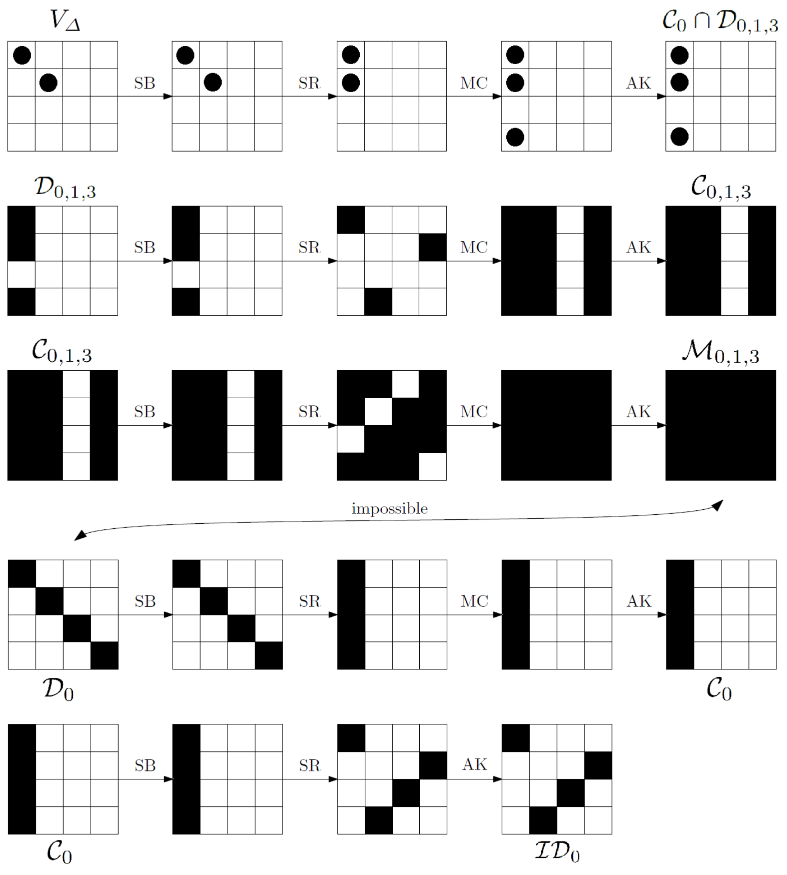

Fig. 10. 5-Rounds Secret Key Distinguisher with data complexity $2^{98.2}$ based on the Impossible Subspace Trail on 4-Rounds (from Sect. 5.3). The choice of the plaintexts (i.e. $p_{0,0} \oplus p_{1,1} = k_{0,0} \oplus k_{1,1}$) guarantees that after one round there are only three bytes with non-zero difference instead of four, that is the plaintexts belong to the same coset of $\mathcal{C}_0 \cap \mathcal{D}_{0,1,3}$. The probability the two ciphertexts belong to the same coset of $\mathcal{M}_k$ for $|k| = 1$ is zero. White box denotes denotes a byte with a zero-difference, while a black box denotes a byte with non-zero difference.

that is plaintexts with 14 constants bytes and where $\Delta := k_{0,0} \oplus k_{1,1}$. Note that $|V_\Delta| = 2^8$ (each pairs of plaintexts-ciphertexts are different). It is easy to prove that this choice of plaintexts guarantees that after one round they belong to the same coset of $\mathcal{D}_I$ where $I = \{0,1,3\}$ (see for example Footnote 6). That is, there exists unique (unknown) $\tilde{a} \in \mathcal{D}_I^\perp$ such that for each $p \in V_\Delta$, then $R(p) \in \mathcal{D}_I \oplus \tilde{a}$ for $I = \{0,1,3\}$. Equivalently, if $p, q \in \mathcal{V}_\Delta$, then $R(p) \oplus R(q) \in \mathcal{D}_I$. In more detail, there exists unique (unknown) $a \in (\mathcal{D}_I \cap \mathcal{C}_0)^\perp$ such that $R(V_\Delta) \subseteq (\mathcal{C}_0 \cap \mathcal{D}_{0,1,3}) \oplus a$ (note that $|(\mathcal{C}_0 \cap \mathcal{D}_{0,1,3}) \oplus a| = 2^{24}$).

Proposition 5. *Let V_Δ defined as in (27) and let $I = \{0,1,3\}$. There exists (unique) $a \in (\mathcal{D}_I \cap \mathcal{C}_0)^\perp$ such that $R(V_\Delta) \subseteq (\mathcal{C}_0 \cap \mathcal{D}_{0,1,3}) \oplus a$.*

Proof. First of all, note that given two arbitrary elements p and q in V_Δ, then after one round their second, third and fourth columns are equal, that is $R(p)_{i,j} = R(q)_{i,j}$ $\forall i = 0, ..., 3$ and $\forall j \neq 0$ as for the attack on 4-rounds of AES with the extension at the beginning - Sect.9. Thus, in order to prove that $R(V_\Delta) \subseteq (\mathcal{C}_0 \cap \mathcal{D}_{0,1,3}) \oplus a$, it is sufficient to prove that given two arbitrary elements p and q in V_Δ, then $R(p)_{2,0} \oplus R(q)_{2,0} = 0$.

By simple computation:

$$R(p)_{2,0} = \text{S-Box}(p_{0,0} \oplus k^0_{0,0}) \oplus \text{ S-Box}(p_{1,1} \oplus k^0_{1,1}) \oplus \\ \oplus \alpha \cdot \text{ S-Box}(p_{2,2} \oplus k_{2,2}) \oplus (\alpha + 1) \cdot \text{ S-Box}(p_{3,3} \oplus k_{3,3}).$$

First of all observe that S-Box$(p_{0,0} \oplus k^0_{0,0}) \oplus$ S-Box$(p_{1,1} \oplus k^0_{1,1}) = 0$. Indeed, since $p_{0,0} \oplus p_{1,1} = k_{0,0} \oplus k_{1,1}$ by definition, then $p_{0,0} \oplus k^0_{0,0} = p_{1,1} \oplus k^0_{1,1}$, that is S-Box$(p_{0,0} \oplus k^0_{0,0}) =$ S-Box$(p_{1,1} \oplus k^0_{1,1})$ and so S-Box$(p_{0,0} \oplus k^0_{0,0}) \oplus$ S-Box$(p_{1,1} \oplus k^0_{1,1}) = 0$. Thus:

$$R(p)_{2,0} = \alpha \cdot \text{ S-Box}(p_{2,2} \oplus k_{2,2}) \oplus (\alpha + 1) \cdot \text{ S-Box}(p_{3,3} \oplus k_{3,3})$$

and in a similar way:

$$R(q)_{2,0} = \alpha \cdot \text{ S-Box}(q_{2,2} \oplus k_{2,2}) \oplus (\alpha + 1) \cdot \text{ S-Box}(q_{3,3} \oplus k_{3,3}).$$

Since $p_{2,2} = q_{2,2}$ and $p_{3,3} = q_{3,3}$ by definition, it follows that $R(p)_{2,0} = R(q)_{2,0}$, and so the thesis. □

Note that the assumption $p^i_{k,l} = p^j_{k,l}$ for each $(k, l) \neq \{(0, 0), (1, 1)\}$ and $i \neq j$ is necessary. Indeed, without this assumption, it is not true that all the plaintexts belong to the same coset of $\mathcal{D}_I$ for $I = \{0, 1, 3\}$ after one round.

Since $R(p) \oplus R(q) \in \mathcal{D}_{\{0,1,3\}}$ for each pair of plaintexts p and q in V_Δ, then $R^{(4)} \circ R(p) \oplus R^{(4)} \circ R(q) = R^{(5)}(p) \oplus R^{(5)}(q) \notin \mathcal{M}_J$ for $|I| + |J| \leq 4$ with probability 1 due to the 4-rounds impossible differential distinguisher of Sect. 5.3. That is:

$$Pr(R^{(5)}(x) \oplus R^{(5)}(y) \in \mathcal{M}_J \,|\, x_{0,0} \oplus x_{1,1} = y_{0,0} \oplus y_{1,1} = \Delta \text{ and} \\ \text{and } x_{i,j} = y_{i,j} \quad \forall (i, j) \neq \{(0, 0), (1, 1)\}) = 0,$$

for each J with $|J| = 1$ and where $\Delta := k_{0,0} \oplus k_{1,1}$ is known. Thus, if the difference of two sub-key bytes ($\Delta := k_{0,0} \oplus k_{1,1}$) is known, it is possible to construct an impossible differential distinguisher over 5 rounds. Only for completeness, in the case in which the final MixColumns operation is omitted, the previous probability becomes

$$Pr(R^{(5)}_f(x) \oplus R^{(5)}_f(y) \in \mathcal{ID}_J \,|\, x_{0,0} \oplus x_{1,1} = y_{0,0} \oplus y_{1,1} = \Delta \text{ and} \\ \text{and } x_{i,j} = y_{i,j} \quad \forall (i, j) \neq \{(0, 0), (1, 1)\}) = 0,$$

for each J with $|J| = 1$, where $R^{(5)}_f(\cdot) := R_f \circ R^{(4)}(\cdot)$ and $\mathcal{ID}_j$ is the inverse-diagonal space (defined as $\mathcal{ID}_j = SR(\mathcal{C}_j)$).

In order to set up the distinguisher, we look for the minimum number of texts necessary to have a collision in the random case with high probability. Since $|J| = 1$ and since there are four different J such that $|J| = 1$, the probability that two texts belong to the same coset of $\mathcal{M}_J$ is $4 \cdot (2^8)^{-16+4} = 2^{-94}$ (analogous for $\mathcal{ID}_J$). Thus, given n pairs, the probability to have at least one collision in the same coset of $\mathcal{M}_J$ for $|J| = 1$ is given by

$$p = 1 - \left(e^{-n/2^{96}}\right)^4 = 1 - e^{-n/2^{94}}.$$

If the number of pairs n is approximately $2^{95.6}$, then p is greater than 95%. Given a single set V_Δ, it is possible to construct $2^7 \cdot (2^8 - 1) \simeq 2^{15}$ different pairs. Thus, for the distinguisher we need approximately $2^{95.6} \cdot 2^{-15} = 2^{80.6}$ different sets V_Δ. Since each of this set contains 2^8 texts, the data complexity of the distinguisher is of $2^{80.6} \cdot 2^8 = 2^{88.6}$ text. We'd like to emphasize that for each difference Δ fixed, there are $2^{128} \cdot 2^{-8} = 2^{120}$ different sets of V_Δ.

B.2 The 5-Round Secret Key Distinguisher for AES

Next we choose how to extend the previous distinguisher in the case in which the difference $\varDelta := k_{0,0} \oplus k_{1,3}$ is not known.

First of all, note that $\varDelta$ can only assume 2^8 values. The idea is simply to "repeat" the previous distinguisher for each possible values of $\varDelta$, i.e. the idea is to construct a sufficient number of different sets V_Δ for each possible values of $\varDelta$ [14]. That is, the idea is to construct 2^8 collections of sets, one for each possible value of $\varDelta$. For each one of these 2^8 collections, one has to count the number of collisions, i.e. the number of pairs of texts that belong to the same coset of $\mathcal{M}_J$ for $|J| = 1$. For a random permutation, the goal is to have at least one collision for each one of the 2^8 collections, i.e. for each value of $\varDelta$. Instead, for the AES permutation, note that there exists one collection in which there is no collisions with probability 1. This collection corresponds to the one for which $\varDelta := k_{0,0} \oplus k_{1,1}$. For the other values of $\varDelta$, the behavior is similar to that of the random case. Thus, it is not difficult to distinguish the two cases: the random case is the one for which all the collections have at least one collision, while the AES case is the one for which there is one collection with no collisions.

To set up the distinguisher, we are interested to compute the number of sets of the form V_Δ for each one of the 2^8 collection. If each collection has $2^{80.6}$ sets (as before), then for each fixed collection the probability to have one collision is 95%. Since all the 2^8 collections are independent, the probability that there is at least one collision for each one of the 2^8 collections is $0.95^{256} \simeq 2 \cdot 10^{-6}$. In order to have a *total* probability of about 95%, the probability to have at least one collision in each fixed collection has to be approximately $(0.95)^{1/2^8} = 0.9998$. In this way, the total probability is given by $0.9998^{256} = 0.95$. Thus, for each one of the 2^8 collections (i.e. for each $\varDelta$), we need at least $2^{97.2}$ pairs to have at least one collision with probability 0.9998 (analogous computation as before). Since each set V_Δ has about 2^{15} different pairs, then we need about $2^{97.2} \cdot 2^{-15} = 2^{82.2}$ different sets for each $\varDelta$ (instead of $2^{80.6}$ as before), that is $2^{90.2}$ texts for each $\varDelta$. Since each set V_Δ has 2^8 texts, the total number of texts required for this distinguisher is of $2^8 \cdot 2^{90.2} = 2^{98.2}$ texts, which is lower than the total input-output space.

To summarize, suppose to have 2^8 collections (one for each $\varDelta$), each one with $2^{82.2}$ different sets V_Δ, where each of this set contains 2^8 texts, for a total of $2^{98.2}$ texts. In the random case and with probability 95%, we expect that in each one of these 2^8 collections there is at least one collision. Note that the average number of collisions for each collection (i.e. for each $\varDelta$) is about $2^{-94} \cdot 2^{97.2} = 2^{3.2} \simeq 9$. For the AES permutation, we expect that there exists one $\varDelta$ for which there is no collision with probability 1 in the corresponding collection of sets. For all the other collections, we expect to have at least one collision with probability 95%. We'd like to highlight that given the $2^{98.2}$ texts defined as before, it is always possible to divide them in 2^8 collections (one for each $\varDelta$), and that each collection can be divided in a very simple way in $2^{82.2}$ different sets V_Δ (simply using the definition of V_Δ). For example, given a fixed $\varDelta$, the corresponding collection is composed of all the texts p such that $p_{0,0} \oplus p_{1,1} = \varDelta$.

In order to compare this distinguisher with the one presented in [30], we analyze the data and the computational cost of our distinguisher. First of all, to construct all

[14] In [30], in order to construct the secret key distinguisher, authors simply consider all the input-output space, and divide it in the 2^8 subsets defined by $\tilde{V}_{\tilde{\Delta}}$. Then they argue that there exists $\tilde{\varDelta}$ such that after 5 rounds the zero-sum property holds (and which corresponds to $\tilde{\varDelta} := k_{0,0} \oplus k_{1,3}$). For random permutation, this happens with probability 2^{-120}.

```
Data: $2^8$ collections (one for each possible value of $\Delta$. Each collection contains $2^{82.2}$ different
      sets $V_\Delta$ defined as in (27).
Result: $\Delta$ if the permutation is an AES permutation (where $\Delta = k_{0,0} \oplus k_{1,1}$); $-1$ if the
        permutation is a Random one.
for $\Delta$ from 0 to $2^8 - 1$ do
    flag = 0;
    for each one of the $2^{82.2}$ different sets $V_\Delta$ do
        for each pair $(c^i, c^j) \in V_\Delta$ do                    // about $2^{15}$ different pairs
            if $c^i \oplus c^j \in \mathcal{M}_k$ for $|k| = 1$ then     // e.g. see Algorithm 1
                flag = 1;
                next collection (i.e. next $\Delta$);
            end
        end
    end
    if flag = 0 then                                              // AES permutation
        return $\Delta$;
    end
end
return $-1$.                                                      // Random permutation
```

Algorithm 8: Distinguisher for 5-rounds of AES - Pseudo-code. The $2^{98.2}$ input texts are already divided in 2^8 collections (one for each Δ), and for each collection the texts are already divided in the sets V_Δ. Given a pair (c^i, c^j), use for example Algorithm 1 to check if $c^i \oplus c^j$ belongs (or not) in $\mathcal{M}_k$ for a $|k| = 1$.

the plaintext-ciphertext pairs, the cost is of $2^{98.2}$ encryptions or oracle queries. For comparison, in [30], since the entire input-output space is required, the cost is of 2^{128} Encryptions or Oracle Queries. Consider instead the cost to check that there exists at least one collision. To do this check, one has to construct all the possible pairs and to check that there is at least one pair that collides in the same coset of $\mathcal{M}_J$ for $|J| = 1$ (note that when the first collision is found, one can consider the next collection). First of all, given a pair of texts, the cost to verify that they collide in the same coset of $\mathcal{M}_J$ is approximately the cost of 1 XOR operation and the cost of an inverse MixColumns operation. Since the cost of these two operations is negligible compared to a table look-ups, the total cost can be approximated by the cost to construct all the pairs. Note that one has to construct only the pairs of texts that belong to the same set V_Δ (that is, a pair composed of texts that belong to different set V_Δ is not required). Thus, the cost for this step can be approximated by 2^8 (number of Δ) $\cdot 2^{82.2}$ (number of sets)$\cdot 2^7 \cdot (2^8 - 1)$ (number of pairs) $\simeq 2^{105.2}$ table look-ups. For the distinguisher presented in [30], the cost to do the verification operation can be approximated by 2^{128} XOR operations. In conclusion, our distinguisher of 5 rounds of AES with a secret key requires $2^{98.2}$ texts, the cost to construct them is of $2^{98.2}$ encryptions and the verification cost is of $2^{105.2}$ table look-ups. The distinguisher presented in [30] requires 2^{128} texts, the cost to construct them is of 2^{128} encryptions or oracle queries and the verification cost is of 2^{128} XOR operations.

Finally, note that besides the possibility to distinguish between a random permutation and an AES one using less than the entire input-output space, we can also recover some information on the secret key (that is, the difference $k_{0,0} \oplus k_{1,1}$), with a computational cost that is lower than a brute force attack.

Observations. We'd like to conclude with some observations regarding this distinguisher with data complexity of $2^{98.2}$. First of all, we present it only for the AES case.

However, it is simply to generalize it for other encryption scheme design *if* some important assumptions hold. These assumptions (equivalent to the ones for the distinguisher proposed in [30]) are:

- the encryption scheme has to adopt *identical S-Boxes*;
- *at least one column of the MixColumns matrix MC* (or its inverse) *has to contain* (at least) *two identical elements.*

If one of these two assumptions is missing, the above distinguishers don't work. Actually, the first one can be relaxed. Indeed, it is sufficient that only the two S-Boxes that are in the positions in which the MixColumns matrix has identical elements are equal. Note that both the assumptions are necessary to construct V_{Δ} (for example, the second one is necessary to prove Prop. 5). Thus, the distinguisher describes in this section can not work in the decryption mode (that is, with chosen ciphertexts instead of chosen plaintexts), since no one of the columns of the inverse MixColumns MC^{-1} has two equal elements. However, note that *the above distinguishers don't depend on the particular choice of which S-Box and MixColumns matrix* MC *are used in the cipher*. That is, it works for each S-Box and for each MC matrix for which the previous assumptions hold.

Finally, we'd like to emphasize that these assumptions are quite common for the construction of AES-like ciphers (or more in general, for Substitution-Permutation Network (SPN) ciphers). Indeed, symmetric encryption schemes are usually a trade-off between the security and computational efficiency. Thus, to enhance the performance of an encryption scheme (especially for lightweight cryptography), designers usually use identical S-Box and a diffusion layer which maximize the number of 1's (or elements with relatively low hamming weights).

However, we conclude remembering that, as we have shown in Sect. 10, it is possible to construct more efficient secret-key distinguisher on 5 rounds of AES such that no assumption on the MixColumns matrix MC or on the SubBytes operations are required.