Designs, Codes and Cryptography (2020) 88:1401–1445
https://doi.org/10.1007/s10623-020-00756-5

Revisiting Gilbert's known-key distinguisher

Lorenzo Grassi[1,2,3] · **Christian Rechberger**[1]

Received: 8 October 2018 / Revised: 30 March 2020 / Accepted: 31 March 2020 / Published online: 4 May 2020

Abstract
Known-key distinguishers have been introduced by Knudsen and Rijmen in 2007 to better understand the security of block ciphers in situations where the key can not be considered to be secret, i.e. the "thing between secret-key model and hash function use-cases". Trying to find a rigorous model to fit this intuition is still ongoing. The most recent advance by Gilbert (Asiacrypt 2014) describes a new model that—even if it is well justified—seemingly does not match this intuition. AES is often considered as a target of such analyses, simply because AES or its building blocks are used in many settings that go beyond classical encryption. Consider AES-128. Results in the secret-key model cover up to 6 rounds, while results in the chosen-key model reach up to 9 rounds. Gilbert however showed a result in the known-key model that goes even further, covering 10 rounds. *Does it mean that the use cases corresponding to the cryptanalysis of hash-function use-cases are inherently less efficient, or is it rather an artifact of the new model? In this paper we give strong evidence for the latter*. In Gilbert's work, two types of arguments or rather conjectures are put forward suggesting that the new model is meaningful. Firstly that the number of "extension rounds" due to the new model is limited to two. And secondly that only a distinguisher that exploits the uniform distribution property can be extended in such way. We disprove both conjectures and arrive at the following results: First, we are also able to show that more than two extension rounds are possible. As a result of this, we describe the first known-key distinguishers on 12 rounds of AES that fit into Gilbert's model. The second conjecture is disproven by showing that the technique proposed by Gilbert can also be used to extend a known-key distinguisher based on another property: truncated differentials. A potential conclusion of this work would be that the counter-intuitive gap between Gilbert's known-key model and the chosen-key model is wider than initially thought. We however conclude that results in Gilbert's model are due to an artifact in the model. To remedy this situation, we propose a refinement of the known-key model which restores its original intent to fit the original intuition.

Communicated by T. Iwata.

✉ Lorenzo Grassi
lgrassi@science.ru.nl

[1] IAIK, Graz University of Technology, Graz, Austria

[2] Know-Center, Graz, Austria

[3] Digital Security Group, Radboud University, Nijmegen, The Netherlands

Keywords Block cipher · Permutation · AES · Known-Key Distinguisher

Mathematics Subject Classification 68P25 · 94A60

1 Introduction

Block ciphers play an important role in symmetric cryptography, providing a basic tool for encryption. They are (probably) the most scrutinized cryptographic tools, and they are often used as the underlying tool to construct other cryptographic algorithms, whose proofs of security are performed under the assumption that the underlying block cipher is ideal.

The concept of known-key distinguishers was introduced by Knudsen and Rijmen in [20]. In the classical single secret-key setting, the attacker does not know the randomly generated key and aims to recover it or to build a (secret-key) distinguisher that allows to distinguish the cipher from a random permutation. The security model in known-key attacks is quite different though: the attacker knows the randomly drawn key the block cipher operates with and aims to find a structural property for the cipher under the known key—a property which an ideal cipher (roughly speaking, a permutation drawn at random) would not have. For completeness, we mention that a more relaxed version – called chosen-key distinguisher—can be considered, where the adversary is assumed to have a full control over the key. This model was introduced in [5], and has been extended to a related-key attack on the full-round AES-256, while the best chosen-key distinguisher for AES-128 [14] currently present in the literature covers 9 rounds out of 10. In this paper however we focus on the known-key model and do not allow/assume related keys.

Since their introductions, known-key attacks have been a major research topic in the symmetric-key community. Only to provide some examples besides AES, known-key distinguishers have been proposed for full PRESENT [9] (one of the most studied lightweight block cipher proposed at CHES 2007) and for Feistel networks [29]. This is justified by the fact that if known-key distinguishers could be considered less relevant than secret-key ones, they anyway allow to learn something about the security margin of a cipher. For example, if it is not possible to find distinguishers for a block cipher when the key is given, then one cannot find a distinguisher when the key is secret. Secondly and more importantly, hash functions can be built from block ciphers, and vice versa. For example, given a hash function, it is always possible to set up a block cipher using the Feistel construction. Vice versa, e.g. the Davies-Meyer construction or the Miyaguchi-Preneel construction can transform a secure block cipher into a secure compression function. In a hash setting, block cipher security models such as the known-key model (or the chosen-key model) make sense since in practice the attacker has full access and control over the internal computations. Finally, an attack in these models depicts a structural flaw of the cipher, while it should be desired to work with a primitive that does not have any flaw, even in the most generous security model for the attacker. A classical example is the devastating effect on the compression function security of weak keys for a block cipher [34], which are usually considered as a minor flaw for a block cipher if the set of these weak-keys is small. Therefore, the security notions to consider for a block cipher will vary depending on whether this block cipher is used in a hash function setting or not.

Despite this cumulative impact in the symmetric-key community over the last years, known-key attacks have been known to be difficult to formalize since [1] proposed the notion of known-key indifferentiability to capture the security of block ciphers under a known key.

In particular, they focus on known-key distinguishers for block ciphers based on *idealized primitives* such as randomly drawn functions or permutations, that is block ciphers for which the round function looks like an ideal primitive and where the adversary can have access to this underlying ideal primitive. Later on, in [27] the impact of attacks in the known-key model on hash functions is studied.

Citing Knudsen and Rijmen [20], "*imagine a block cipher*" for which a known-key distinguisher exists, "*but where no efficient attacks are known in the traditional black-box model. Should we recommend the use of such a cipher? We do not think so!*"

1.1 Known-key distinguishers for AES: the state of the art

The known-key model. In the known-key model, a full access to an instance of the encryption function associated with a *known* random key and its inverse is given. The purpose is to simultaneously control the inputs and the outputs of the primitive, i.e. to achieve input-output correlations that one could not efficiently achieve with inputs and outputs of a perfect random permutation to which one would have an oracle access. A formal definition of a known-key distinguisher is provided in Sect. 3, where we propose and describe in detail a generic scenario for known-key distinguishers. We emphasize that *all known-key distinguishers currently present in the literature—including the ones presented in this paper – implicitly exploit (and can be described in) the scenario proposed in Sect. 3.2.*

Known-key distinguishers for AES. AES and related constructions served as a benchmark for cryptanalytic techniques since the very introduction of this model by Knudsen and Rijmen [20] with a 7-round result. Subsequently, 8-round results were obtained using truncated differentials [17], which were later on improved in [19]. Currently, this last one—which exploits the rebound technique [22] and the so called "multiple limited-birthday problem"—is the best 8-round known-key distinguisher in the literature. At Asiacrypt 2014 Gilbert [16] found a way to extend an 8-round known-key distinguisher (using a novel representation of AES) into a more intricate 10-round distinguisher and hence presented for the first time a known-key distinguisher for full AES-128.

All the known-key distinguishers on AES currently present in the literature are briefly recalled in Sect. 3 using the "subspace trail notation"[1], recently introduced at FSE/ToSC 2017. In Table 1 we list the known-key distinguishers for AES, including our main results.

On Gilbert's Approach. As we will describe in more detail in Sect. 3.1, the approach of [16] makes use of a freedom in the known-key model that was actually always there but never spelled out explicitly. In more detail, there is always the role of a "verifier", in addition to a "shortcut-player" and a "generic player". In [16] the verifier has perhaps for the first time some non-negligible computations to do. The details of Gilbert's approach are such that it is still not possible to simply "peel-off" an arbitrary number of rounds, on the contrary it seems that only the detection of a very specific property (the so called "uniform distribution property") could take advantage of computations of the verifier.

[1] Our choice to use the subspace trail notation is due to the fact that it allows in some cases an easier and more formal description than the original notation.

Table 1 AES known-key distinguishers

Rounds	Computations	Memory	Property	KK	Gil-KK	Reference
7	2^{56}	2^{56}	Zero-Sum	✓		[20]
7	2^{24}	2^{16}	Differential Trail	✓		[25]
7	$\mathbf{2^{20}}$	$\mathbf{2^{16}}$	**Multiple Diff. Trail**	✓		**Appendix** E.1
8	2^{64}	2^{64}	Uniform Distribution	✓		[16] - Appendix C
8	2^{48}	2^{32}	Differential Trail	✓		[17]
8	2^{44}	2^{32}	Multiple Diff. Trail	✓		[19]
8	$2^{42.6}$	2^{13}	Statistical Integral	✓		[10]
8	$\mathbf{2^{23}}$	$\mathbf{2^{16}}$	***extended* 7-Round MultDT**		✓	**Appendix** E.2
10	2^{64}	2^{64}	*extended* 8-Round Unif. Dist.		✓	[16]
10	$2^{59.6}$	2^{59}	*extended* 8-Round Stat. Integral		✓	[10]
10	$\mathbf{2^{50}}$	$\mathbf{2^{32}}$	***extended* 8-Round MultDT**		✓	**Sect.** 5
12	$\mathbf{2^{82}}$	$\mathbf{2^{32}}$	***extended* 8-Round MultDT**		✓	**Sect.** 6
12	$\mathbf{2^{66}}$	$\mathbf{2^{64}}$	***extended* 8-Round Unif. Dist.**		✓	**Sect.** 7

The computation cost is the sum of the computational cost to generate N-tuples of plaintexts/ciphertexts and the verification cost. Here we show which known-key distinguishers are defined in Gilbert's model proposed in [16]—denoted by *Gil-KK*—and which are defined in the "classical" known-key model—denoted by *KK*. We recall that the latter ones exploit a property directly on the plaintexts/cipheretxts which is independent of the details of the cipher $E(\cdot)$ and of the secret key. For distinguishers which exploit the technique introduced by Gilbert [16], we highlight the distinguisher used as starting point and extended with the technique (initially) proposed in [16]. As remarked in the main text, such distinguishers are meaningful *only* under the assumption of validity of Gilbert's model. *MultDT* multiple differential trail

1.2 Our contributions

Systematization of known-key distinguisher

First of all, we begin with recapitulating the literature about the concept of known-key distinguisher in Sect. 3. By doing so, we propose and describe a generic scenario for a known-key distinguisher by interpreting it as a game between two players—a "shortcut-player" and a "generic player"—that face the same cipher. The idea is that the player who knows the key—namely, the "shortcut-player"—must be able to generate a set of plaintexts/ciphertexts (that satisfies a required property) faster than the other player who does not know the key (or equivalently, that faces an ideal cipher)—namely, the "generic-player". Then we recall the known-key distinguishers present in the literature in the above scenario. In particular, we focus on Gilbert's distinguisher proposed in [16], and we show that it can be easier explained using the "subspace trail notation" proposed at ToSC/FSE 2017 than using the "twisted representation".

Analysis of Gilbert's known-key distinguisher: refuting both conjectures via progress in cryptanalysis

As we already recalled, at Asiacrypt 2014 Gilbert proposed a new known-key distinguisher on full AES-128 [16], by extending an 8-round known-key distinguisher based on the so-called uniform distribution property into a more intricate 10-round distinguisher. In the conclusion

of his paper, Gilbert claims that it seems technically difficult to use a stronger property than the uniform distribution one to extend an 8-round known-key distinguisher to a 10-round one:

1st *Conjecture:* "*while we do not preclude that the use of the stronger property that several pairs satisfying the differential relation of* [17] [i.e. truncated diff. relations exploited by the rebound distinguisher] *can be derived might potentially result in a 10-round distinguisher that outperforms the 10-round distinguisher presented above, giving a rigorous proof seems technically difficult.*"

In particular, he left "*the investigation of improved 10-round known-key distinguishers and associated proofs—or even plausible heuristic arguments if rigorous proofs turn out to be too difficult to obtain—as an open issue.*"

In this paper, we pick up this challenge, and using a strategy similar to the one proposed by Gilbert in [16], *we show how to construct a more efficient 10-round distinguisher, by exploiting known-key distinguishers based on truncated differential trails.* In particular, we use as a starting point the 8-round known-key distinguisher presented in [19], and we extend it at the end and at the beginning using the strategy proposed by Gilbert. This allows to set up a 10-round known-key distinguisher for AES (see Sect. 5) with a time complexity of approximately 2^{50}.

As one of the main (cryptanalytic) results, in Sect. 6 we show that it is possible to extend our 10-round distinguisher up to 12 rounds. Moreover, exploiting a similar strategy, in Sect. 7 we extend Gilbert's 10-round distinguisher based on the uniform distribution property up to 12 rounds. These *12-round AES known-key distinguishers* provide counter-examples of the claim made in [16] about the (im)possibility to use Gilbert's technique to extend an 8-round distinguisher by more than 2 rounds:

2nd *Conjecture* "*The reader might wonder whether the technique we used to derive a known-key distinguisher for the 10-round AES from a known-key distinguisher for the 8-round AES does not allow to extend this 8-round known distinguisher by an arbitrary number of rounds. It is easy however to see that the argument showing that 10-round relation $\mathcal{R}$ is efficiently checkable does not transpose for showing that the relations over $r > 10$ rounds one could derive from the 8-round relation by expressing that the r-round inputs and outputs are related by $r - 8 > 2$ outer rounds to intermediate blocks that satisfy the 8-round relation are efficiently checkable.*"

Our results are summarized in Table 2.

Table 2 1st/2nd Conjectures and AES Gilbert's known-key distinguishers

Rounds	Property	1st Conjecture	2nd Conjecture	References
10	*Extended* 8-Round Unif. Dist.			[16]
10	*Extended* 8-Round Stat. Integral	✓		[10]
10	***Extended* 8-Round MultDT**	✓		**Sect. 5**
12	***Extended* 8-Round MultDT**	✓	✓	**Sect. 6**
12	***Extended* 8-Round Unif. Dist.**		✓	**Sect. 7**

Referring to the 1*st* and the 2*nd* conjectures given in the main text, in this table we emphasize which ones of our results disprove them. *MultDT* multiple differential trail

A proposal of a "New" model for known-key distinguishers

The second main contribution is the high level insight that the details of the known-key model need to be changed if we aim to restore the original intent of the known-key model. The reason is that with our new result the difference between the chosen-key model and what is currently thought of as the known-key model is counter-intuitive: As we show it is now possible to have cryptanalytic results on more rounds of AES in the known-key model than in the chosen-key model and this is true for more than a single property. Hence we propose a simple restriction of the verifier in the known-key model to remedy the situation.

Firstly, we remark and emphasize that *the goal of this paper is to discuss the validity of Gilbert's model independently of its (possible) practical applications*. In particular, even if Gilbert's known-key distinguisher leads to statements on more rounds of AES than ever before (without related keys) that seem meaningful, then it is not clear if such statements can become useful in the sense of e.g. having an impact on the case where a block cipher is used to construct a hash function. This has also been noticed in [16], where it is pointed out that even if the strategy proposed by Gilbert allows to set up efficient known-key distinguishers, its "*impact on the security of [...] AES when used as a known key primitive, e.g. in a hash function construction, is questionable*" (see abstract of [16]).

To achieve our goal, *under the assumption of the validity of such model*, we set up distinguishers based on the truncated differential property instead of the uniform distribution one in Gilbert's framework, and we show that it is also possible to extend them for up to 12-round AES, that is two rounds beyond the claim given by Gilbert in [16]. Using these results as a starting point, we propose—*with more confidence than would be possible without our results*—a (new) definition of known-key distinguisher model that rules out Gilbert's and our attacks proposed in this paper. As our results show, this seems necessary for better capturing the original idea of known-key distinguishers as something "between secret-key model and hash function use-cases". For this reason, we conclude in Sect. 8, with a discussion of the results and a proposal of a refinement of the known-key model which restores its original intent (in which the role of the verifier gets back to being marginal).

2 Preliminary

2.1 Preliminary—description of AES

The Advanced Encryption Standard [13] is a *Substitution-Permutation network* that supports key sizes of 128, 192 and 256 bits. The 128-bit plaintext initializes the internal state as a 4×4 matrix of bytes that are interpreted 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 non-linearity in the cipher);
- *ShiftRows* (SR): cyclic shift of each row to the left;
- *MixColumns* (MC): multiplication of each column by a constant 4×4 invertible matrix M_{MC} (MC and SR provide diffusion in the cipher[2]);
- *AddRoundKey* (ARK): XORing the state with a 128-bit subkey.

[2] SR makes sure column values are spread, MC makes sure each column is mixed.

One round of AES can be described as $R(x) = K \oplus MC \circ SR \circ \text{S-Box}(x)$. In the first round an additional AddRoundKey operation (using a whitening key) is applied, and in the last round the MixColumns operation is omitted. Finally, as we do not use the details of the AES key schedule in this paper, we refer to [13] for a complete description.

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 subkey of the r-th round. If only the key of the final round is used, then we denote it by k to simplify the notation. Finally, we denote by R one round of AES, while we denote r rounds of AES by R^r. We sometimes use the notation R_K instead of R to highlight the round key K. As a last thing, in this paper we often use the term "partial collision" (or "*collision*") when two texts belong to the same coset of a given subspace X.

2.2 Preliminary—subspace trails

Invariant subspace cryptanalysis can be a powerful cryptanalytic tool, and subspace trails [18] – introduced at ToSC/FSE 2017—are a recent generalization of it.

Let F denote a round function in an iterative block cipher and let $V \oplus a$ denote a coset of a vector space V. Then if $F(V \oplus a) = V \oplus a$ we say that $V \oplus a$ is an *invariant coset* of the subspace V for the function F. This concept can be generalized to *trails of subspaces*.

Definition 1 ([18]) 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 $a_i \in V_i$, there exist $a_{i+1} \in V_{i+1}$ such that $F(V_i \oplus a_i) \subseteq V_{i+1} \oplus a_{i+1}$, then $(V_1, V_2, ..., V_{r+1})$ is a *subspace trail* of length r for the function F. If all the previous relations hold with equality, the trail is called a *constant-dimensional subspace trail*.

This means that if F^t denotes the application of t rounds with fixed keys, then $F^t(V_1 \oplus a_1) = V_{t+1} \oplus a_{t+1}$. We refer to [18] for more details about the concept of subspace trails. Our treatment here is however meant to be self-contained.

Subspace trails of AES

In this section, we recall the subspace trails of AES presented in [18]. For the following, we only work with vectors and vector spaces over $\mathbb{F}_{2^8}^{4\times 4}$, and we denote by $\{e_{0,0}, ..., e_{3,3}\}$ 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). We also recall that given a subspace X, the cosets $X \oplus a$ and $X \oplus b$ (where $a \neq b$) are *equal* (that is $X \oplus a \equiv X \oplus b$) if and only if $a \oplus b \in X$.

Definition 2 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, $\mathcal{C}_0$ corresponds to the symbolic matrix

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

Definition 3 The *diagonal spaces* $\mathcal{D}_i$ and the *inverse-diagonal spaces* $\mathcal{ID}_i$ are respectively defined as $\mathcal{D}_i = SR^{-1}(\mathcal{C}_i) \equiv \langle e_{0,i}, e_{1,i+1}, e_{2,i+2}, e_{3,i+3} \rangle$ and $\mathcal{ID}_i = SR(\mathcal{C}_i) \equiv \langle e_{0,i}, e_{1,i-1}, e_{2,i-2}, e_{3,i-3} \rangle$, where the indexes are taken modulo 4.

For instance, $\mathcal{D}_0$ and $\mathcal{ID}_0$ correspond to the symbolic matrices

$$\mathcal{D}_0 \equiv \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & x_4 \end{bmatrix}, \qquad \mathcal{ID}_0 \equiv \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 \\ 0 & 0 & x_3 & 0 \\ 0 & x_4 & 0 & 0 \end{bmatrix}.$$

Definition 4 The *i-th mixed spaces* $\mathcal{M}_i$ are defined as $\mathcal{M}_i = MC(\mathcal{ID}_i)$.

For instance, $\mathcal{M}_0$ corresponds to the symbolic matrix

$$\mathcal{M}_0 \equiv \begin{bmatrix} 0\text{x}02 \cdot x_1 & x_4 & x_3 & 0\text{x}03 \cdot x_2 \\ x_1 & x_4 & 0\text{x}03 \cdot x_3 & 0\text{x}02 \cdot x_2 \\ x_1 & 0\text{x}03 \cdot x_4 & 0\text{x}02 \cdot x_3 & x_2 \\ 0\text{x}03 \cdot x_1 & 0\text{x}02 \cdot x_4 & x_3 & x_2 \end{bmatrix}.$$

Definition 5 For $I \subseteq \{0, 1, 2, 3\}$, let $\mathcal{C}_I, \mathcal{D}_I, \mathcal{ID}_I$ and $\mathcal{M}_I$ be defined as

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

As shown in detail in [18]:

- for any coset $\mathcal{D}_I \oplus a$, there exists a unique $b \in \mathcal{C}_I^{\perp}$ such that $R(\mathcal{D}_I \oplus a) = \mathcal{C}_I \oplus b$;
- for any coset $\mathcal{C}_I \oplus a$, there exists a unique $b \in \mathcal{M}_I^{\perp}$ such that $R(\mathcal{C}_I \oplus a) = \mathcal{M}_I \oplus b$.

Theorem 1 *For each I and for each $a \in \mathcal{D}_I^{\perp}$, there exists one and only one $b \in \mathcal{M}_I^{\perp}$ such that*

$$R^2(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b. \tag{1}$$

We refer to [18] for a complete proof of this theorem. Observe that b depends on a (the constant that defines the initial coset of $\mathcal{D}_I$) and on the secret key k.

Moreover, note that if X is a generic subspace, $X \oplus a$ is a coset of X and x and y are two elements of the (same) coset $X \oplus a$, then $x \oplus y \in X$. It follows that:

Lemma 1 *For all x, y and for all $I \subseteq \{0, 1, 2, 3\}$:*

$$Prob(R^2(x) \oplus R^2(y) \in \mathcal{M}_I \mid x \oplus y \in \mathcal{D}_I) = 1. \tag{2}$$

As demonstrated in [18], we finally recall that for each $I, J \subseteq \{0, 1, 2, 3\}$:

$$\mathcal{M}_I \cap \mathcal{D}_J = \{0\} \qquad \text{if and only if} \qquad |I| + |J| \leq 4, \tag{3}$$

Theorem 2 *Let $I, J \subseteq \{0, 1, 2, 3\}$ such that $|I| + |J| \leq 4$. For all $x \neq y$:*

$$Prob(R^4(x) \oplus R^4(y) \in \mathcal{M}_I \mid x \oplus y \in \mathcal{D}_J) = 0. \tag{4}$$

We remark that all these results can be re-described using a more "classical" truncated differential notation, as formally pointed out in [8,23]. For example, if two texts t^1 and t^2 are equal except for the bytes in the i-th diagonal[3] for each $i \in I$, then they belong in the same coset of $\mathcal{D}_I$. A coset of $\mathcal{D}_I$ corresponds to a set of $2^{32 \cdot |I|}$ texts with $|I|$ active diagonals. Again, two texts t^1 and t^2 belong in the same coset of $\mathcal{ID}_I$ if the bytes that lie in the i-th anti-diagonal for each $i \notin I$ are equal to zero. Similar considerations hold for the column space $\mathcal{C}_I$ and the mixed space $\mathcal{M}_I$.

[3] The i-th diagonal of a 4×4 matrix A is defined as the elements that lie on row r and column c such that $r - c = i \bmod 4$. The i-th anti-diagonal of a 4×4 matrix A is defined as the elements that lie on row r and column c such that $r + c = i \bmod 4$.

3 Known-key security of block ciphers

Firstly, we give a formal definition of the known-key distinguisher scenario, recalling the one proposed in [16] by Gilbert as a starting point.

3.1 Definition of known-key distinguisher

Informally, a known-key distinguisher exploits the fact that it is in general harder for an adversary who does not know the key to derive an N-tuple of input blocks of a given block cipher E that is "abnormally correlated" with the corresponding N-tuple of output blocks than for one who knows the secret key. This difficulty is well expressed by the T-intractable definition, expressed by Gilbert as follows:

Definition 6 Let $E : (K, X) \in \{0,1\}^k \times \{0,1\}^n \rightarrow E_K(X) \in \{0,1\}^n$ denote a block cipher of block size n bits. Let $N \geq 1$ and $\mathcal{R}$ denote an integer and any relation over the set S of N-tuples of n-bit blocks. $\mathcal{R}$ is said to be T-intractable relatively to E if, given any algorithm $\mathcal{A}$ that is given an oracle access to a perfect random permutation Π of $\{0,1\}^n$ and its inverse, it is impossible for $\mathcal{A}$ to construct in time $T' \leq T$ two N-tuples $\mathcal{X} = (X_i)$ and $\mathcal{Y} = (Y_i)$ such that $Y_i = \Pi(X_i)$, $i = 1, ..., N$ and $\mathcal{X}\,\mathcal{R}\,\mathcal{Y}$ with a success probability $p \geq 1/2$ over Π and the random choices of $\mathcal{A}$. The computing time T' of $\mathcal{A}$ is measured as an equivalent number of computations of E, with the convention that the time needed for one oracle query to Π or Π^{-1} is equal to 1. Thus if q denotes the number of queries of $\mathcal{A}$ to Π or Π^{-1}, then $q \leq T'$.

Definition 7 Let $E : (K, X) \in \{0,1\}^k \times \{0,1\}^n \rightarrow E_K(X) \in \{0,1\}^n$ denote a block cipher of block size n bits. A known-key distinguisher $(\mathcal{R}, \mathcal{A})$ of order $N \geq 1$ consists of (1) a relation $\mathcal{R}$ over the N-tuples of n-bit blocks (2) an algorithm $\mathcal{A}$ that on the input of a k-bit key K produces in time $T_{\mathcal{A}}$, i.e. in a time equivalent to $T_{\mathcal{A}}$ computations of E, an N-tuple $\mathcal{X} = (X_i)$ $i = 1, ..., N$ of plaintext blocks and an N-tuple $\mathcal{Y} = (Y_i)$ $i = 1, ..., N$ of ciphertext blocks related by $Y_i = E_K(X_i)$ and by $\mathcal{X}\,\mathcal{R}\,\mathcal{Y}$. The two following conditions must be met:

- The relation $\mathcal{R}$ must be $T_{\mathcal{A}}$-intractable relatively to E;
- The validity of $\mathcal{R}$ must be efficiently checkable.

To formalize the last requirement, we incorporate the time for checking whether two N-tuples are related by $\mathcal{R}$ in the computing time $T_{\mathcal{A}}$ of algorithm $\mathcal{A}$.

We emphasize that while the algorithm $\mathcal{A}$ takes a random key K as input, *the relation $\mathcal{R}$ satisfied by the N-tuples of input and output blocks constructed by $\mathcal{A}$ or $\mathcal{A}'$ is the same for all values of K (in other words, it is independent of K) and must be efficiently checkable without knowing K.*

3.2 The known-key distinguisher scenario

To better understand these definitions, we propose and describe in more detail a generic scenario for a known-key distinguisher, which is depicted in Fig. 1. This scenario is composed of five characters, which are a key generator, an oracle, two players and a verifier. We assume that the oracle is instantiated by an *ideal cipher Π* defined as[4]

$$\Pi : (k, p) \in \{0,1\}^k \times \{0,1\}^n \rightarrow c = \Pi(k, p) \in \{0,1\}^n$$

[4] The parameters k and n are the same that defines the encryption scheme E, that is $E : (K, p) \in \{0,1\}^k \times \{0,1\}^n \rightarrow c = E_K(p) \in \{0,1\}^n$.

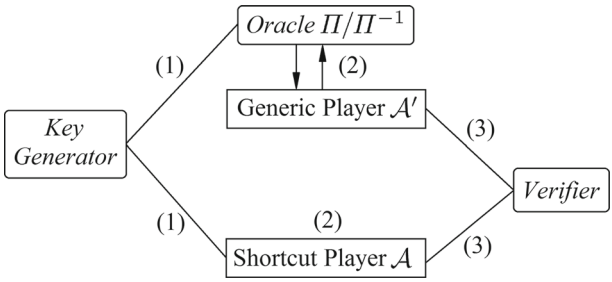

Fig. 1 A *Known-Key Distinguisher Scenario*. First, we assume a relationship $\mathcal{R}$ is chosen and fixed. Step (1): the secret key is given to the Oracle Π/Π^{-1}, to the Shortcut Player $\mathcal{A}$ and to the Verifier. Step (2): the Shortcut Player $\mathcal{A}$ and the Generic Player $\mathcal{A}'$ generate the N-tuples that satisfy the required relationship $\mathcal{R}$. Step (3): the Verifier receives the N-tuple and checks if $\mathcal{R}$ is satisfied or not. The fastest player to generate the N-tuple wins the "game"

such that $\Pi(k, \cdot)$ is a permutation for each fixed $k \in \{0,1\}^k$. Equivalently, Π is chosen uniformly at random among all ciphers with a k-bit key and an n-bit input/output. Moreover, we assume that the verifier knows the details both of E and of Π.

After fixing a relation $\mathcal{R}$ defined as in Def. 6, the known-key distinguisher scenario can be described as follows:

1st) step: the key generator generates a key, which is given to the oracle and to one of the two player. In the following:

- "*shortcut player*" denotes the player who knows the key and faces the encryption scheme E;
- "*generic player*" denotes the player who does not know the key and faces the ideal cipher Π.

Referring to the previous definitions by Gilbert, the generic player can be identified by the algorithm $\mathcal{A}'$, while the shortcut player can be identified by the algorithm $\mathcal{A}$;

2nd) step: the two players generate the N-tuple of (plaintexts, ciphertexts) which satisfy the required relation $\mathcal{R}$. Since the generic player does not know the key, he must ask the oracle (identified with Π and/or Π^{-1} in the previous definitions) for the encryption (resp. decryption) of chosen plaintexts (resp. ciphertexts). We stress that this step does not consist only in the generation of (plaintext, ciphertext) pairs, but also includes any computational cost that the player must do in order to find the N-tuple with the required property;

3rd) step: when a player finds the N-tuple which satisfies the required relation $\mathcal{R}$, he sends it to the verifier. The verifier finally checks if (1) the relation $Y'_i = E_K(X'_i)$ (case of shortcut player) or $Y'_i = \Pi(X'_i)$ (case of generic player) is satisfied for each i and if (2) the N-tuple satisfies the relation $\mathcal{R}$. The first/fastest player who sends the N-tuple with the required property $\mathcal{R}$ wins the "game".

A distinguisher is meaningful if the cost of the generic player—assuming that the cost of one oracle-query is equal to the cost of one encryption—*to generate the N-tuple is higher than the cost of the shortcut player, when the probability of success is equal for the two players.* Equivalently, a distinguisher is meaningful if the probability of the generic player to win the game is higher than the probability of the shortcut player, when the number of (plaintext, ciphertext) pairs that the two players can generate is fixed and equal for both

players. In other words, in the first version one considers the computational costs of the two players to generate the N-tuples with a fixed probability of success (equal for both the players). In the second version, the computational cost (equivalent to the number of oracle queries for the generic player and the number of N-tuple generated by the shortcut one) is fixed and one considers the probabilities of success of the two players to win the game.

Before going further, we emphasize that *the role of the verifier is only to prevent one or both of the two players from cheating*. In other words, in the case of honest players, the verifier can be omitted, and the winner of the game is simply the first/fastest player that claims to have found the N-tuple of (plaintexts, ciphertexts) which satisfy the required relation $\mathcal{R}$. We highlight that such *a verifier is implicitly present in all the distinguishers currently present in the literature*.

Verification Step. Both for the distinguishers that we are going to present and for Gilbert's one, the computational cost of the verification step is not negligible. To clarify, we identify *the verification cost*[5] *only as the cost to check that the relation $\mathcal{R}$ holds*. Thus, in order to compare our distinguishers to the others present in the literature, we define *the cost of the distinguisher as the sum of* the cost of the verification step (i.e. *the cost of the verifier*) *and* of the cost to construct the set of plaintexts/ciphertexts with the required property. For this reason, we assume for the following that *a relationship $\mathcal{R}$ is efficiently checkable if and only if the computational cost of the verifier is negligible with respect to the players' ones*. This implies that the cost of the distinguisher can be approximated with the computational cost of the shortcut player (since the cost of the other player is always higher in the case of a meaningful distinguisher).

What about the cost of the Generic Player? Since the generic player depends on the oracle to generate the N-tuple (i.e. he cannot work alone to generate it), two possible settings can be analyzed. In the first one, only the number of oracle queries is considered to determine the computational cost of this player, that is the number of encryptions/decryptions required by the generic player to the oracle. In the second one, both the number of oracle queries and any other computational cost of the generic player (which is in general not negligible) are considered. Intuitively this second setting is weaker than the first one, in the sense that a known-key distinguisher in the first setting works also in the second one but not vice-versa. In other words, one can expect that the required number N of tuples is in general higher (or at least equal) in the first setting than in the second one.

For the goal of this paper, in the following we limit ourselves to consider only the first scenario.

Role of the Ideal Cipher in the Known-Key Scenario. For completeness, note that the ideal cipher in the previous model can be replaced by the encryption scheme E faced by the shortcut player *if* such a cipher is a Strong PseudoRandom Permutation (see Appendix A for more details).

4 State of the art of known-key distinguishers for AES-128

Here we review the most relevant distinguishers for AES in the above scenario, with particular attention to the ones proposed by Gilbert in [16]. For simplicity, we assume that the relations

[5] In other words, the cost of checking that the relations $Y_i = E_K(X_i)$ (case of shortcut player) and $Y_i = \Pi(X_i)$ (case of generic player) are satisfied for each i is *not* considered/included. In the following, we assume that such relations are always satisfied.

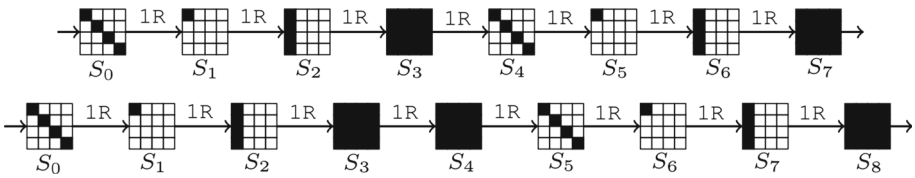

Fig. 2 7- and 8-round differential paths for AES-128

$Y_i = E_K(X_i)$ (case of shortcut player) and $Y_i = \Pi(X_i)$ (case of generic player) are always satisfied for each i, that is that the two players do not cheat about these relations.

4.1 Distinguishers based on the Rebound Technique

4.1.1 7- and 8-Round known-key distinguisher & the rebound attack

For the case of the 7- and 8-round known-key distinguishers proposed in [25] and [17], the goal of the two players is to find two pairs of (plaintexts, ciphertexts)—i.e. (p^1, c^1) and (p^2, c^2)—s.t. (1) the two plaintexts are equal in one fixed diagonal—equivalently, belong to the same coset of $\mathcal{D}_i$ for a fixed $i \in \{0, 1, 2, 3\}$ (i.e. $p^1 \oplus p^2 \in \mathcal{D}_i$)—and (2) the two ciphertexts are equal in one fixed anti-diagonal (if the final MixColumns operation is omitted)—equivalently, belong to the same coset of $\mathcal{M}_i$ for a fixed $i \in \{0, 1, 2, 3\}$ (i.e. $c^1 \oplus c^2 \in \mathcal{M}_i$).

In the above known-key distinguisher setting, the best technique that the shortcut player (i.e. the player who knows the key) can exploit to win the game is the *Rebound Attack*. The rebound attack is a differential attack and it was proposed in [26] for the cryptanalysis of AES-based hash functions. Since it is a differential attack, one needs a "good" (truncated) differential trail in order to exploit it. Examples of truncated differential trails used for 7- and 8-round AES are depicted in Fig. 2. The rebound attack consists of two phases, called inbound and outbound phase. In the first one, the attacker uses the knowledge of the key to find pairs of texts that satisfy the middle rounds of the truncated differential trail. In the second one, he propagates the solutions found in the first phase in the forward and in the backward directions, and checks if at least one of them satisfies the entire differential trail.

As proved in [17], for the AES case and using the rebound attack, the shortcut player needs approximately 2^{48} computations in order to find the two (plaintexts, ciphertexts) pairs (p_1, c_1) and (p_2, c_2) with the required properties (besides a memory cost of $16 \times 2^{32} = 2^{36}$ bytes). Instead, in the case of an ideal cipher, the generic player needs approximately 2^{64} operations in order to find them with the same probability.

4.1.2 Multiple limited-birthday 8-round known-key distinguisher

An improvement of the previous known-key distinguisher on 8-round of AES was proposed in [19]. Using the subspace trail notation, in this modified version of the 8-round known-key distinguisher, the goal of the two players is to find two pairs of (plaintexts, ciphertexts) such that the two plaintexts belong to the same coset of $\mathcal{D}_i$ for an *arbitrary* i and the two ciphertexts belong to the same coset of $\mathcal{M}_j$ for an *arbitrary* j, where i and j are not fixed in advance and it is not required that they are equal (i.e. no condition is imposed on i and j). A concrete example is depicted in Fig. 3. For arbitrary initial and final subspaces, the computational cost of the shortcut player is reduced from 2^{48} to 2^{44} (note that there are 4

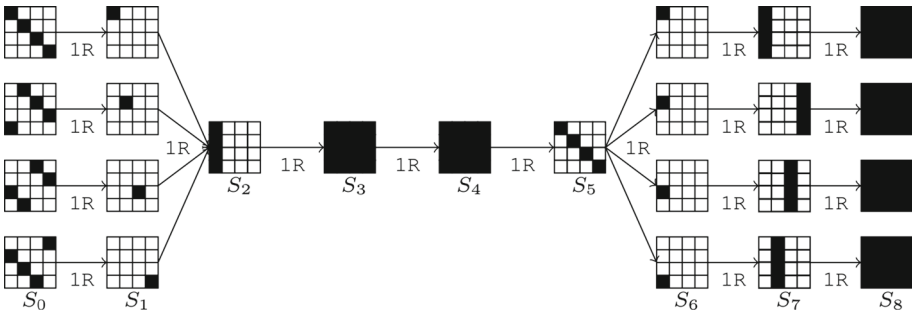

Fig. 3 8-round differential characteristic for known-key distinguisher of AES-128

initial and final different subspaces $\mathcal{D}_i$ and $\mathcal{M}_j$, for a total of $4^2 = 2^4$ possibilities) while the required memory is still 2^{32}, as shown in detail in [19]. In Appendix E.1 we show that the same technique can be exploited to improve the 7-round known-key distinguisher presented in [25].

4.2 Gilbert's known-key distinguishers

4.2.1 Uniform distribution 8-round known-key distinguisher

Another 8-round known-key distinguisher for AES is based on the uniform distribution property and it was proposed by Gilbert in [16]. In this case, the goal of the two players is to find a set of 2^{64} (plaintext, ciphertext) pairs—that is, (p^i, c^i) for $i = 0, ..., 2^{64} - 1$—such that the bytes of the plaintexts and of the ciphertexts are uniformly distributed:

- for each $j, k \in \{0, 1, 2, 3\}$ and for each $x \in \mathbb{F}_{2^8}$, there are 2^{56} plaintexts p^i for $i \in I \subseteq \{0, ..., 2^{64} - 1\}$ with $|I| = 2^{56}$ that satisfy $p^i_{j,k} = x$ for all $i \in I$;
- for each $j, k \in \{0, 1, 2, 3\}$ and for each $x \in \mathbb{F}_{2^8}$, there are 2^{56} ciphertexts c^i for $i \in I \subseteq \{0, ..., 2^{64} - 1\}$ with $|I| = 2^{56}$ that satisfy $c^i_{j,k} = x$ for all $i \in I$.

Using the subspace trail notation, it is possible to re-formulate the goal of the two players as follows: find a set of 2^{64} (plaintext, ciphertext) pairs—that is, (p^i, c^i) for $i = 0, ..., 2^{64} - 1$—such that

- for each $I \subseteq \{0, 1, 2, 3\}$ with $|I| = 3$ the plaintexts are uniformly distributed in cosets of the diagonal space $\mathcal{D}_I$, or equivalently, for each I with $|I| = 3$ and for each $a \in \mathcal{D}_I^\perp$ there are 2^{32} plaintexts p^j for $j \in J \subseteq \{0, ..., 2^{64} - 1\}$ with $|J| = 2^{32}$ such that $p^j \in \mathcal{D}_I \oplus a$ for all $j \in J$;
- for each $I \subseteq \{0, 1, 2, 3\}$ with $|I| = 3$ the ciphertexts are uniformly distributed in cosets of the mixed space $\mathcal{M}_I$, or equivalently, for each I with $|I| = 3$ and for each $a \in \mathcal{M}_I^\perp$ there are 2^{32} ciphertexts c^j for $j \in J \subseteq \{0, ..., 2^{64} - 1\}$ with $|J| = 2^{32}$ such that $c^j \in \mathcal{M}_I \oplus a$ for all $j \in J$.

If the final MixColumns is omitted, an equivalent condition holds on the ciphertexts by replaying the mixed space $\mathcal{M}_I$ with the inverse-diagonal one $\mathcal{ID}_I$. To be more formal:

Definition 8 Consider 2^{64} texts $t^i \in \mathbb{F}_{2^8}^{4\times 4}$ for $i = 0, ..., 2^{64} - 1$, and let $I \subseteq \{0, 1, 2, 3\}$ with $|I| = 3$ fixed. These 2^{64} texts t^i are "*uniformly distributed*" in cosets of $\mathcal{M}_I$ if

- for each coset $\mathcal{M}_I \oplus a$ for $a \in \mathcal{M}_I^{\perp}$, there exists a set $\mathcal{T}_a$ of 2^{32} texts $\mathcal{T}_a := \{t^j\}_{j=0,\ldots,2^{32}-1}$ such that $t^j \in \mathcal{M}_I \oplus a$ for each $t^j \in \mathcal{T}_a$;
- given sets $\mathcal{T}_a$ and $\mathcal{T}_b$ just defined for two different cosets $\mathcal{M}_I \oplus a$ and $\mathcal{M}_I \oplus b$ where $(a \oplus b) \in \mathcal{M}_I^{\perp}$, then $\mathcal{T}_a \cap \mathcal{T}_b = \emptyset$.

Before going further, we prove that the two previous formulations are equivalent, namely that the bytes of the plaintexts are uniformly distributed if and only if the plaintexts are uniformly distributed in cosets of the diagonal space $\mathcal{D}_J$ for each J with $|J| = 3$ (analogous for the ciphertexts). Consider the case in which the plaintexts are uniformly distributed in cosets of the diagonal space $\mathcal{D}_J$, which means that for each $a \in \mathcal{D}_{1,2,3}^{\perp} \equiv \mathcal{D}_0$, there are 2^{32} plaintexts $\hat{p}^i$ for $i \in I \subseteq \{0, ..., 2^{64} - 1\}$ and $|I| = 2^{32}$ that belong e.g. to the same coset of $\mathcal{D}_{1,2,3} \oplus a$ (analogous for the other spaces $\mathcal{D}_J$ with $|J| = 3$). For each fixed $a \in \mathcal{D}_{1,2,3}^{\perp} \equiv \mathcal{D}_0$, this means that

$$\forall i \in I,\ \forall j \in \{0, 1, 2, 3\}: \quad \hat{p}^i \in \mathcal{D}_{1,2,3} \oplus a \quad \text{if and only if} \quad \hat{p}^i_{j,j} = a_{j,j}.$$

Working at byte level, note that for each $x \in \mathbb{F}_{2^8}$ and for each index j, there are 2^{24} different $a \in \mathcal{D}_{1,2,3}^{\perp} \equiv \mathcal{D}_0$ s.t. $a_{j,j} = x$. It follows that there are $2^{24} \cdot 2^{32} = 2^{56}$ plaintexts p^i s.t. $p^i_{j,j} = x$, which means that the bytes of the given plaintexts are uniformly distributed. To complete the proof, it is sufficient to use a similar strategy and the definition of $\mathcal{D}$ in order to prove that, given plaintexts whose bytes are uniformly distributed, they are uniformly distributed in cosets of the diagonal space $\mathcal{D}_J$ for each J with $|J| = 3$.

Finally, for the follow-up we highlight that *the uniform distribution property implies the balance/zero-sum property*[6] both on the plaintexts and on the ciphertexts.

The strategy of the shortcut player. Here, we briefly re-propose the best strategy that the shortcut player can use to win the game using the subspace trails notation instead of the *Super-SB* notation ($Super\text{-}SB(\cdot) \equiv$ S-Box $\circ ARK \circ MC \circ$ S-Box$(\cdot)$)—see Appendix C for more details. The idea is to start in the middle with a set $\mathfrak{S}$ of texts defined as $\mathfrak{S} := \mathcal{D}_i \oplus \mathcal{M}_j \oplus c$ for a constant c, where $|\mathfrak{S}| = 2^{64}$. Observe that

$$\mathfrak{S} = \mathcal{D}_i \oplus \mathcal{M}_j \oplus c \equiv \bigcup_{b \in \mathcal{D}_i \oplus c} \mathcal{M}_j \oplus b = \bigcup_{a \in \mathcal{M}_j \oplus c} \mathcal{D}_i \oplus a, \tag{5}$$

i.e. the set $\mathfrak{S}$ can be re-written as the union of cosets of the space $\mathcal{D}_i$ or as the union of cosets of the space $\mathcal{M}_j$. The ciphertexts are given by the 4-round encryption of $\mathfrak{S}$, and the plaintexts by the 4-round decryption of $\mathfrak{S}$.

After encrypting $\mathfrak{S}$ for 4 rounds, the texts are uniformly distributed in each coset of $\mathcal{M}_I$ of dimension 12 (i.e. $|I| = 3$). That is, after 4 rounds, each coset of $\mathcal{M}_I$ for $|I| = 3$ contains exactly 2^{32} elements. Indeed, by Theorem 2 note that given two elements in the same coset of $\mathcal{D}_I$, they can not belong to the same coset of $\mathcal{M}_J$ for $|I| + |J| \leq 4$ after 4-round. Thus, given a coset of $\mathcal{D}_i$ with $|i| = 1$, after 4 rounds each element is distributed in a different cosets of $\mathcal{M}_J$ for $|J| = 3$. Since a coset of $\mathcal{D}_i$ contains 2^{32} elements and since there are exactly 2^{32} cosets of $\mathcal{M}_J$, the elements of $\mathcal{D}_i \oplus \mathcal{M}_j$ are uniformly distributed in each coset of $\mathcal{M}_I$. The same happens if one decrypts $\mathfrak{S}$ for 4 rounds. In this case, after decrypting $\mathfrak{S}$ for 4 rounds, the texts are uniformly distributed in each coset of $\mathcal{D}_I$ of dimension 12 (i.e. $|I| = 3$), that is each coset of $\mathcal{D}_I$ for $|I| = 3$ contains exactly 2^{32} elements.

On the meaningfulness of this distinguisher. What is the *minimum number* $N \equiv 2^{64} + M > 2^{64}$ *of random (plaintext, ciphertext) pairs* s.t. there is a subset of 2^{64} pairs whose bytes

[6] The set of texts $\{t^i\}_{i \in I}$ satisfies the balance property if and only if $\bigoplus_{i \in I} t^i = 0$.

are uniformly distributed both on the plaintexts and on the ciphertexts with non-negligible property?

First of all, note that 2^{64} texts satisfy the uniform distribution on each byte with probability

$$p = \left[\prod_{i=0}^{255}\binom{2^{64}-i\cdot 2^{56}}{2^{56}}\cdot\left(2^{-8}\right)^{2^{64}}\right]^{16} = \left(\frac{2^{64}!}{\left(2^{56}!\right)^{256}}\cdot\left(2^{-8}\right)^{2^{64}}\right)^{16}.$$

Indeed, consider the following problem. Given N texts and 2 sets, assume that each text belongs to one of the two sets with probability 2^{-1}. It follows that the N texts are uniformly distributed among the two sets with prob. $\binom{N}{N/2}\cdot 2^{-N}$. In a similar way, given $d \geq 2$ sets, they are uniformly distributed with probability[7] $\prod_{i=0}^{d-1}\binom{N-i\cdot N/d}{N/d}\cdot d^{-N}$.

Using Stirling's formula $n! \simeq n^n\cdot e^{-n}\cdot\sqrt{2\pi\cdot n}$, this probability is well approximated by

$$p = \left(\frac{2^{64}!}{\left(2^{56}!\right)^{256}}\cdot\left(2^{-8}\right)^{2^{64}}\right)^{16} \simeq \left(\frac{1}{2^{49}\cdot\pi}\right)^{128}\cdot(256!)^{-1/2} \simeq 2^{-7328.1} \equiv 2^{-2^{12.84}}. \quad (6)$$

In other words, given 2^{64} plaintexts whose bytes are uniformly distributed, this represents the probability that the bytes of the corresponding ciphertexts are uniformly distributed[8].

Given $2^{64}+M$ texts, it is possible to construct

$$\binom{2^{64}+M}{2^{64}} \simeq \frac{1}{\sqrt{2\pi\cdot M}}\cdot\left(\frac{2^{64}+M}{M}\right)^{M}$$

different sets of 2^{64} texts (where the approximation is given using Stirling's formula and by the assumption $M \ll 2^{64}$). This number is always bigger than $p^{-2} \equiv 2^{2^{13.84}}$ for each $M \geq 2^{12}$. Thus, given $2^{64}+2^{12}$ random pairs, there is a good probability to find 2^{64} (plaintext, ciphertext) pairs for which the bytes of the plaintexts and of the ciphertexts are uniformly distributed. It follows that if the cost of the generic player is approximated by the number of oracle queries, then his cost is approximately of $2^{64}+2^{12} \simeq 2^{64}$ encryptions versus 2^{64} encryption of the shortcut player.

So, *why is this distinguisher meaningful?* Instead of focusing on the cost of the two players, the idea is to show that the probability of the generic player to win the game given 2^{64} texts is negligible. To do this, authors of [16] claim that this probability is upper bounded by the probability of the following game: "*given* $2^{64}-1$ *(plaintext, ciphertext) pairs whose bytes are 'almost uniform'*—see the definition in the following, *find a text for which the bytes of the corresponding* 2^{64} *texts are uniformly distributed*". Since this probability is upper bounded by 2^{-127}—see proof of Prop. 4 of [16] – and since this second game is (strongly) "related" to the original one, the conclusion follows immediately. For completeness, we emphasize that *no formal proof is provided* in [16] that supports this second claim. In other words, it is not formally proved that the fact that this second game is "hard" implies the hardness of the original game, and/or viceversa.

Finally, we formally define what "almost uniform" means. Consider $2^{64}-1$ texts $t^i \in \mathbb{F}_{2^8}^{4\times 4}$ for $i = 0, ..., N-2$. We say that the bytes of $2^{64}-1$ texts t^i are "*almost uniform*" if for

[7] Consider the case $N = 2^{64}$ and $d = 256$. The product of the binomial coefficients is explained as follows. For each one of the 16 bytes, there must exist $2^{64}/256 = 2^{56}$ texts for each one of the 256 possible values. Thus, there are $\binom{2^{64}}{2^{56}}$ possible sets of 2^{56} texts for which the byte as value 0, $\binom{2^{64}-2^{56}}{2^{56}}$ possible sets of 2^{56} texts for which the byte as value 1 and so on.

[8] For comparison, note that given 2^{64} plaintexts whose sum is zero, then the sum of the corresponding ciphertexts is equal to zero with probability 2^{-128}.

each row and column $j, k = 0, 1, 2, 3$ (1) there exists $x \in \mathbb{F}_{2^8}$ s.t. there are $2^{56} - 1$ texts that satisfy $t^i_{j,k} = x$ and (2) for each $y \in \mathbb{F}_{2^8} \setminus x$, there are 2^{56} texts that satisfy $t^i_{j,k} = y$. More generally:

Definition 9 Consider $2^N - d$ texts $t^i \in \mathbb{F}_{2^8}^{4\times 4}$ for $i = 0, ..., N - d - 1$ for $d \geq 1$. The bytes of these $2^N - d$ texts t^i are "*almost uniform*" if for each row and column $j, k = 0, 1, 2, 3$:

- there exists a set $X \equiv \{x_1, ..., x_s \in \mathbb{F}_{2^8}\}$ with cardinality $s \leq d$ such that for each $x_l \in X$ with $1 \leq l \leq s$ there are $2^{N-8} - d \leq \hat{s}_l \leq 2^{N-8} - s$ texts that satisfy $t^i_{j,k} = x_l$ where $\sum_{l=1}^{s} \hat{s}_l = d$;
- for each $y \in \mathbb{F}_{2^8} \setminus X$, there are 2^{N-8} texts that satisfy $t^i_{j,k} = y$.

Note that, given a set of 2^N texts whose bytes are uniformly distributed, then the bytes of each subset of $2^N - d$ texts (for each $d \geq 1$) are "almost uniform" distributed w.r.t. the previous definition.

4.2.2 Extension to 10 rounds of AES

The previous distinguisher is the starting point used by Gilbert in order to set up the first 10-round known-key distinguisher for AES. The basic idea is to extend this 8-round distinguisher based on the uniform distribution property adding one round at the end and one at the beginning. In the known-key distinguisher scenario presented above, the players have to send to the verifier 2^{64} (plaintext, ciphertext) pairs, that is (p^i, c^i) for $i = 0, ..., 2^{64} - 1$, such that :

1. there exists a key k^0 s.t. the bytes of $\{R_{k^0}(p^i)\}_i$ are uniformly distributed, or equivalently s.t. the texts $\{R_{k^0}(p^i)\}_i$ are uniformly distributed among the cosets of $\mathcal{D}_I$ for each I with $|I| = 3$;
2. there exists a key k^{10} s.t. the bytes of $\{R^{-1}_{k^{10}}(c^i)\}_i$ are uniformly distributed, or equivalently s.t. the texts $\{R^{-1}_{k^{10}}(c^i)\}_i$ are uniformly distributed among the cosets of $\mathcal{M}_J$ for each J with $|J| = 3$.

We emphasize that *it is not required that k^0 and k^{10} are equal to the "real" subkeys (generated by the key-generator – see before) that define $E_K(\cdot)$, that is k^r can be different from the r-th subkey. In other words, it is only required that such keys exist, and not that they are equal to the real subkeys*[9]. The same assumption holds for all Gilbert's like distinguishers presented in this paper and in the literature. Moreover, in this game, the subkeys k^0 and k^{10} are assumed to be independent—no key-schedule holds (argumentation are given by Gilbert to show that the same distinguisher is applicable also to the case in which the key-schedule holds—we discuss this topic in details in the following).

Since uniform distribution implies balance property (vice-versa is not true in general), for the follow-up we highlight that if the plaintexts and the ciphertexts satisfy the previous properties, then they also have the zero-sum property respectively after one round encryption w.r.t. the key k^0 (that is, $\bigoplus_{i=0}^{2^{64}-1} R_{k^0}(p^i) = 0$) and after one round decryption w.r.t. the key k^{10} (that is, $\bigoplus_{i=0}^{2^{64}-1} R^{-1}_{k^{10}}(c^i) = 0$).

On the meaningfulness of this distinguisher. What is the probability that given a set of 2^{64} texts there exists a key $\hat{k}$ such that the bytes of 1-round encryption (resp. decryption) of such

[9] For this and the following distinguishers, we abuse the notation k^r to denote a key of a certain round r. In general, it is not required that in a Gilbert's-like distinguisher such subkey k^r is equal to the real secret subkey. In order to simplify the notation, we decided to abuse the notation k^r to denote both cases.

texts are uniformly distributed? Using the calculation proposed for the 8-round distinguisher and since there are 2^{128} different keys, this probability is equal to $2^{128} \cdot p \simeq 2^{128} \cdot 2^{-7328.1} = 2^{-7200.1} \equiv 2^{-2^{12.81}}$ where p is defined in (6). Similar to the 8-round case, it follows that $2^{64} + 2^{12} \simeq 2^{64}$ (plaintext, ciphertext) pairs are sufficient to have good probability to win the game.

Again, *why is this distinguisher meaningful?* Working as for the 8-round case and in order to support this distinguisher, authors of [16] show that the probability of the generic player to win the game given 2^{64} texts is negligible. To do this, a claim is made about the fact that this probability is upper bounded by the probability of the following game: "*given* $2^{64} - d$ *(plaintext, ciphertext) pairs for* $d \geq 5$ *– that is,* (p^i, c^i) *for each* $i = 0, ..., 2^{64} - d - 1$ *– with the property that there exist a set of keys* k^0 *and* k^{10} *for which the bytes of* $R_{k^0}(p^i)$ *and of* $MC^{-1} \circ R^{-1}_{k^{10}}(c^i)$ *(that is 1-round encryption of* p^i *and the 1-round decryption of the ciphertexts) are 'almost uniform' distributed, find the remaining d texts for which the bytes of the corresponding* 2^{64} *texts after 1-round encryption/decryption are uniformly distributed*". Since this probability is upper bounded by $(2^{128})^2 \times \left(\frac{5^{16}}{2^{128}-2^{64}+1}\right)^3 \simeq 2^{-16.5}$—see proof of Prop. 6 in [16]— and since this second game is "related" to the original one, the conclusion follows immediately.

Strategy of the verifier, of the shortcut player and of the generic one. Since the keys k^0 and k^{10} for which the relation $\mathcal{R}$ is satisfied can be different from the real subkeys, the verifier has no information of the keys for which the relation $\mathcal{R}$ is satisfied, and her task is to check if they exist. It follows that one must show that the above conditions are efficiently checkable. The only way to verify these requirements is to find these two subkeys in an efficient way, which is not possible using a brute force attack (k^0 and k^{10} have 128 bits). Under Gilbert's assumption—no key-schedule holds, the verifier can work independently on k^0 and k^{10}. Instead of checking all the $2 \cdot 2^{128} = 2^{129}$ possible values of k^0 and k^{10}, the idea proposed in [16] is to check uniform distribution working on single columns of $SR(c^i)$ and of $SR^{-1}(p^i)$ (the strategy proposed by Gilbert[10] is similar to the one proposed in Algorithm 1). In this way, the verifier must guess only 32 bits instead of 128, and she has to repeat this operation 4 times (one for each anti-diagonal/diagonal) for each key. In the following, we discuss a way to improve this procedure working independently on each byte of k^0 and k^{10} instead of entire anti-diagonal/diagonal. The idea is simply to use integral attack [11,21] to filter wrong keys (much) faster.

About the shortcut player (i.e. the one who knows the key), he can construct these 2^{64} (plaintext, ciphertext) pairs using the same strategy just discussed for the 8 rounds distinguisher (note that in this case the keys k^0 and k^{10} correspond to the secret sub-keys). As a result, the distinguisher can be considered meaningful (w.r.t. the definition given before) since (1) the probability that the generic player (i.e. the one who does not know the secret key) successfully outputs (input, output) pairs that satisfy the previous properties (both in the input and in the output) is upper bounded by $2^{-16.5}$ and since (2) the verifier can find the keys k^0 and k^{10} that satisfy the required property (if they exist) with a computational cost which is smaller than the cost of the two players.

[10] Algorithm 1 is presented in order to propose a 12-round distinguisher based on the uniform distribution property as extension of 10-round Gilbert's distinguisher. The difference between this algorithm and the one proposed in [16] is the fact that in our case some wrong-key candidates can be eliminated using the zero-sum property. In other words, in order to turn our algorithm into the one proposed in [16], it is sufficient to check all the keys $k \equiv (k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ from (0x00, 0x00, 0x00, 0x00) to $(0xff, 0xff, 0xff, 0xff)$, and not only the ones found by Algorithm 2.

4.2.3 Generic considerations on Gilbert's 10-round distinguisher

The previous 10-round distinguisher proposed in [16] is different from all the previous distinguishers up to 8 rounds present in the literature. For all distinguishers up to 8-round, the relation $\mathcal{R}$ that the N-tuple of (plaintexts, ciphertexts) must satisfy does not involve any operation of the block cipher E. As a consequence, it allows the verifier to check whether the N-tuple of (plaintexts, ciphertexts) satisfy the required relation $\mathcal{R}$ without knowing anything of the key. When $\mathcal{R}$ does not re-use operations of E, this provides some heuristic evidence that this distinguisher can be considered *meaningful*.

On the other hand, the previous 10-round distinguisher and the ones that we are going to propose do not satisfy this requirement, i.e. in these cases the relation $\mathcal{R}$ involves and re-uses some operations of E. The novelty of Gilbert's work is not just the possibility to extend the distinguisher up to 10-round AES, but rather the introduction of a new distinguisher model. Requiring the existence of round keys for which the 1-round encryption of the plaintexts (respectively, 1-round decryption of the ciphertexts) satisfy the relation $\mathcal{R}$, or in other words considering relations $\mathcal{R}$ that depend on some operations of E, allows to set up new distinguishers that penetrate more round of the block cipher. For a detailed discussion on the reasons why such known-key distinguishers should not be systematically ruled out as if they were *artificial* we refer to Sect. 3 of [16].

A variant of Gilbert's distinguisher. Before going further, we highlight a variant of Gilbert's distinguisher—that also applies to all our proposed distinguishers present in the paper – which allows to better understand it. Consider the case in which the two players have to send to the verifier the N-tuple that verify the required relation $\mathcal{R}$ *together with the subkeys* for which such a relation is satisfied. As an example, in the 10-round distinguisher just presented, the players have to send 2^{64} (plaintexts, ciphertexts) pairs (p^i, c^i) and the two subkeys k^0 and k^{10} such that the bytes of $R_{k^0}(p^i)$ and $MC^{-1} \circ R^{-1}_{k^{10}}(c^i)$ are uniformly distributed. Thus, since the task of the verifier is to check that the relation $\mathcal{R}$ is satisfied only for the keys she received, it follows that her computational cost is negligible. On the other hand, we show in details in Appendix B that such variant of the distinguisher is meaningless, since it can be set up for any number of rounds of AES.

4.2.4 Another strategy for the verifier

In order to extend Gilbert's distinguisher on 12-round AES by exploiting the uniform distribution property, we present another possible strategy that allows to check the existence of keys k^0 and k^{10} for which the required property $\mathcal{R}$ is verified. The goal of the following strategy is *not* to improve the computational cost of the verifier, but to show the possibility to check the existence of such keys *working independently on each byte* of the keys instead of combinations of 4 bytes. The idea is simply to first filter wrong key candidates using the integral attack [11,21]: in this way, the verifier limits herself to check the uniform distribution property only on the keys that satisfy the zero-sum property. In other words, *instead of checking directly the uniform distribution property as done in* [16]*, we first filter wrongly guessed key by checking the zero-sum property.*

In more detail, instead of working on 4 bytes of the subkeys k^0 and k^{10} simultaneously (as proposed in [16]), we highlight that it is actually possible to work at byte level, finding k^0 and k^{10} on single bytes (independently of the others) as in a classical integral/zero-sum attack. The idea is to exploit the fact that *uniform distribution implies zero-sum property*:

Data: 2^{64} texts t^i for $i = 0, ..., 2^{64} - 1$
Result: One anti-diagonal of k - e.g. $(k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ - s.t. each byte of $MC^{-1} \circ R_k(t^i)$ is uniformly distributed
Let $A[0, ..., 2^{32} - 1]$ and $B^1[0, ..., 255]$, $B^2[0, ..., 255]$, $B^3[0, ..., 255]$, $B^4[0, ..., 255]$ five arrays initialized to zero;
for i from *0 to* $2^{64} - 1$ **do**
  $x \leftarrow t^i_{0,0} + 2^8 \cdot t^i_{3,1} + 2^{16} \cdot t^i_{2,2} + 2^{24} \cdot t^i_{3,1}$;
  $A[x] \leftarrow A[x] + 1$;
end
Use Algorithm 2 to find $k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1}$ - i.e. to filter wrong candidates;
for *each* $k \equiv (k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$ *found using Algorithm 2* **do**
  for s from $(0x00, 0x00, 0x00, 0x00)$ *to* $(0xff, 0xff, 0xff, 0xff)$ **do**
    Let $s \equiv (s^0, s^1, s^2, s^3) \in \mathbb{F}^4_{2^8}$ be a column of 4 bytes;
    Compute $x \equiv MC^{-1} \circ R_k(s)$; `// partial decryption of s w.r.t. to k - note: x ≡ (x^1, x^2, x^3, x^4) ∈ F^4_{2^8} is a column of 4 bytes`
    Increment B^1, B^2, B^3, $B^4[x]$: $B^j[x^j] \leftarrow B^j[x^j] + A[x]$ for each $j = 1, 2, 3, 4$;
  end
  if *uniform distribution - i.e.* $B^j[x] = 2^{56}$ *for each* $x = 0, ..., 255$ *and for each* $j = 1, 2, 3, 4$ **then**
    identify k as possible candidate;
  end
end
return candidates for $(k_{0,0}, k_{1,3}, k_{2,2}, k_{3,1})$.

Algorithm 1: *Verifier Strategy*: find one anti-diagonal (e.g. the first one) of the last round-key k - equivalent for the other anti-diagonals and for the first round key - such that the bytes of $MC^{-1} \circ R_k^{-1}(t^i)$ are uniformly distributed. For simplicity, we omit the final MixColumns.

Data: 2^{64} texts t^i for $i = 0, ..., 2^{64} - 1$
Result: One byte of k - e.g. $k_{0,0}$ - s.t. $\bigoplus_i$ S-Box$^{-1}(p^i_{0,0} \oplus k_{0,0}) = 0$
Let $A[0, ..., 2^8 - 1]$ an array initialized to zero;
for i from *0 to* $2^{64} - 1$ **do**
  $A[t^i_{0,0}] \leftarrow (A[t^i_{0,0}] + 1) \bmod 2$; `// A[x] denotes the value stored in the x-th address of the array A`
end
for k from $0x00$ *to* $0xff$ **do**
  $x \leftarrow 0$;
  for i from 0 *to* 255 **do**
    $x \leftarrow x \oplus A[i] \cdot$ S-Box$^{-1}(i \oplus k)$; `// A[i] can only be 0 or 1`
  end
  if $x = 0$ **then**
    identify k as candidate for $k_{0,0}$;
  end
end
return candidates for $k_{0,0}$.

Algorithm 2: *First Part of Verifier Strategy:* working on each byte of the key independently of the others, filter wrong key candidates using zero-sum property.

1st) step/filter: the verifier first looks for subkeys k^0 and k^{10} that satisfy $\bigoplus_{i=0}^{2^{64}-1} R_{k^0}(p^i) = 0$ and $\bigoplus_{i=0}^{2^{64}-1} R^{-1}_{k^{10}}(c^i) = 0$ working independently on each byte;
2nd) step/filter: only for keys that satisfy zero-sum, she then checks if the uniform property is verified, working simultaneously on 4 bytes of the subkeys.

We emphasize that if zero-sum is not satisfied, then also uniform distribution is not satisfied. Moreover, we highlight that the number of subkeys that satisfy zero-sum is very small compared to the number of all possible keys. Indeed, note that since zero-sum is satisfied with prob. 2^{-128} and since there are only 2^{128} keys, on average only one key passes the first step/filter. This also implies that "checking uniform distribution once that zero-sum property is satisfied" has negligible cost compared to the total cost. A pseudo-code of this strategy is proposed in Algorithm 2.

Just for completeness, we mention that using this proposed strategy[11], the verification cost is a little smaller than the one given in the original strategy proposed in [16] (approximately[12] $2 \cdot 2^{64}$ vs $10 \cdot 2^{64}$ look-ups table, that is $2^{57.36}$ vs $2^{59.7}$ ten-round encryptions assuming 1 S-Box look-up $\approx$ 1 table look-up). One more time, we emphasize that the goal of this strategy is to show the possibility to work on single byte of the key independently of the others in order to find k^0 and k^{10}: this allows us in the following to set up a distinguisher on 12-round AES.

4.3 Statistical integral distinguisher with multiple structures

Finally, we mention for completeness that at ACISP 2017 the distinguishers proposed by Gilbert in [16] has been improved by T. Cui, L. Sun, H. Chen and M. Wang [10]. In this paper, authors turn both the 8- and 10-round Gilbert's distinguishers into "statistical integral ones" [32] with the goal to reduce the data/time complexity.

5 New 10-round distinguisher of AES—*Full* AES-128

Using the same strategy proposed by Gilbert in [16], we set up our 10-round distinguisher by extending the 8-round one presented in [19] and recalled in Sect. 4.1.2 both at the beginning and at the end. In the above defined known-key distinguisher scenario, the players have to send to the verifier $n \geq 64$ different tuples of (plaintext, ciphertext) pairs, that is $\{(p_i^1, c_i^1), (p_i^2, c_i^2)\}$ for $i = 0, ..., n-1$, s.t.:

1. there exists a key k^0 s.t. for each tuple there exists j for which the two plaintexts belong to the same coset of $\mathcal{D}_j$ after one round, that is

$$\exists\, k^0 \quad \text{s.t.} \quad \forall i \in \{0, ..., n-1\}, \quad \exists j \in \{0, ..., 3\} \quad \text{s.t.} \quad R_{k^0}(p_i^1) \oplus R_{k^0}(p_i^2) \in \mathcal{D}_j;$$

2. there exists a key k^{10} s.t. for each tuple there exists l for which the two ciphertexts belong to the same coset of $\mathcal{M}_l$ one round before, that is

$$\exists\, k^{10} \quad \text{s.t.} \quad \forall i \in \{0, ..., n-1\}, \quad \exists l \in \{0, ..., 3\} \quad \text{s.t.} \quad R_{k^{10}}^{-1}(c_i^1) \oplus R_{k^{10}}^{-1}(c_i^2) \in \mathcal{M}_l.$$

We stress that the *keys k^0 and k^{10} must be equal for all the tuples*. In other words, if there exist two different tuples (c_0, c_1) and (c_2, c_3) such that $R_k^{-1}(c_0) \oplus R_k^{-1}(c_1) \in \mathcal{M}_l$ and $R_{\tilde{k}}^{-1}(c_2) \oplus R_{\tilde{k}}^{-1}(c_3) \in \mathcal{M}_{\tilde{l}}$ for two different keys $k \neq \tilde{k}$, then the above defined relation $\mathcal{R}$ is not satisfied. Note that without this request on the secret keys k^0 and k^{10}, it is extremely easy

[11] Note that all the arrays A defined in Algorithms 1 and 2 can be computed simultaneously and stored, and that we expect that only few (on average only one) keys pass Algorithms 2. Thus, the cost of the verification step is well approximated by the cost to compute the array A.

[12] Note that both Algorithm 1 and 2 can be optimized such that it is possible to compute the array A simultaneously for each row and column of the text t^i, for a total cost of 2^{64} table look-ups. It follows that the cost of our strategy corresponds to the cost to prepare the array A for the two algorithm, that it 2^{65} table look-ups.

to construct tuples such that the two ciphertexts belong to the same coset of $\mathcal{M}_l$ one round before. Indeed, as we are going to show, given two ciphertexts c^1 and c^2, on average there exist $4 \cdot (2^8)^4 = 2^{34}$ different keys k such that $R_k^{-1}(c^1) \oplus R_k^{-1}(c^2) \in \mathcal{M}_l$ for a certain l. As a result, it is straightforward to construct n different tuples that satisfy the above defined relationship $\mathcal{R}$ without any condition on the key k.

We also observe that the claim "*the transposition of our technique to the 8-round distinguisher of* [17] *does not allow to derive a valid 10-round distinguisher*" made in [16] is justified only when just $n = 1$ tuple of pairs is used and/or no assumption on the key k is done. In other words, the above defined relation $\mathcal{R}$—for which we consider $n \geq 64$ different tuples of pairs of texts—together with the requirement of *uniqueness of the key k* allows to extend the 8-round distinguisher of [17] exploiting the same strategy proposed in [16].

Key-schedule *vs* **Independent Subkeys.** Before we go on, it is also important to emphasize that no condition on the keys k^0 and k^{10} is imposed, except that they exist and they are equal for all the tuples. That is, *we do not require that these keys are equal to the real secret subkeys*. The same consideration holds also for the next distinguishers presented in this paper, and for the 10-round distinguisher presented by Gilbert in [16].

Moreover, as in [16], two possible scenarios can be considered and studied:

1. no key-schedule holds—k^0 and k^{10} are independent;
2. AES key-schedule among k^0 and k^{10}.

Intuitively, the second case (i.e. with key schedule) is harder than the first one (i.e. without key schedule) for the generic player, since a further property must be verified. In other words, the time required by this player to generate the tuples for the second scenario is not smaller than for the first one, that is the probability of success in the second scenario is not higher than in the first one.

In the following, we limit ourselves to consider the case of independent subkeys. To justify this choice, we recall the strategy adopted by Gilbert in [16] to set up his 10-round distinguisher. First he considers the case of AES with independent subkeys (denoted by $\text{AES}^\star_{10}$), and he presents a 10-round known-key distinguisher for $\text{AES}^\star$. Then, he simply observes that this known-key distinguisher on $\text{AES}^\star_{10}$ "*is obviously applicable without any modification to* AES_{10}*, i.e. the full AES-128*" (see [16, Sect. 4.2 - page 221]). Using the same argumentation, we can easily conclude that also our distinguisher can be applied to real AES, i.e. to the case in which the key schedule holds. Indeed, as we are going to highlight in the following, nothing changes for the shortcut player (i.e. the one who knows the key), while this scenario is more complicated for the generic player, since a further condition on k^0 and k^{10} (that is, the key schedule) is imposed.

About the "Number n of Different Tuples of (plaintext, ciphertext) Pairs". In the following we present the distinguisher in the case of independent subkeys. To obtain a suitable value for n, we consider the best strategy that the generic player can adopt to win the game.

A value of n is suitable when the computational cost of the generic player is worse than the one of the other player. To find such a value, one has to consider the numbers of oracle-queries done by the two players (and potentially any further cost of the generic player). In particular, if only the number of oracle-queries is taken in account, then n must be equal or greater than 8, which implies that the computational cost for the shortcut player is of 2^{47} and for the generic player is (approximately) of $2^{48.9}$. In order to make the advantage of the shortcut player more significant, we have chosen an (arbitrary) value of $n = 64$, which implies a cost for the shortcut player of 2^{50} computations and (approximately) of $2^{65.6}$ computations for the generic player.

5.1 The verifier

Given n tuples, the verifier has to check the existence of keys k^0 and k^{10} as defined previously. Since the subkeys are independent (no key schedule is considered), the idea is simply to work independently on the plaintexts (in order to find k^0) and on the ciphertexts (in order to find k^{10}).

Let's work for simplicity on the ciphertexts (analogous for the plaintexts). The idea is to find the key k^{10} (if it exists) using the the low-data truncated differential attack[13] on 3-round AES-128 presented in [18]. In the following, we briefly recall such an attack, opportunely modified with respect to the one presented in [18] due to the different scope of this work. In particular, here we describe the attack on 3 rounds presented in [18] as an attack on a single round.

Truncated differential attack [18]. Consider three texts in the same coset of $\mathcal{M}_i$ for $i \in \{0, 1, 2, 3\}$ and the corresponding ciphertexts after one round, that is (p^j, c^j) for $j = 1, 2, 3$ where $c^j = R(p^j)$ and $p^j \in \mathcal{M}_i \oplus a$ for an arbitrary (fixed) $a \in \mathcal{M}_i^{\perp}$. The goal of the attack is to find the key k such that the ciphertexts belong to the same coset of $\mathcal{M}_i$ one round before, that is k has to satisfy the following condition:

$$R_k^{-1}(c^1) \oplus R_k^{-1}(c^2) \in \mathcal{M}_i \qquad \text{and} \qquad R_k^{-1}(c^1) \oplus R_k^{-1}(c^3) \in \mathcal{M}_i. \tag{7}$$

For simplicity, we assume that the final MixColumns operation is omitted (otherwise one simply switches the final MixColumns and the final AddRoundKey operation, as usual in the literature).

Since each column of $\mathcal{M}_i$ depends on different and independent variables, the idea of the attack is to work independently on each column of $\mathcal{M}_i$ (and so of $SR^{-1}(k)$), and to exploit the relationships that hold among the bytes that lie in the same column of $\mathcal{M}_i$.

Without loss of generality, we assume $i = \{0\}$ and we present the attack only for the first column of $SR^{-1}(k)$ (analogous for the others). As showed in [18], the conditions (7) are fulfilled if the bytes of the first column of $SR^{-1}(k)$ satisfy the following relations:

$$s^h_{0,0} = 0\text{x}02 \cdot s^h_{1,3}, \qquad s^h_{2,2} = s^h_{1,3}, \qquad s^h_{3,1} = 0\text{x}03 \cdot s^h_{1,3}, \tag{8}$$

where $s^h_{i,j} = \text{S-Box}^{-1}(c^1_{i,j} \oplus k_{i,j}) \oplus \text{S-Box}^{-1}(c^h_{i,j} \oplus k_{i,j})$ for $h = 2, 3$. For each value of $k_{1,3}$ (2^8 possible values in total), the idea is to find the values of $k_{0,0}$, $k_{2,2}$ and $k_{3,1}$ that satisfy the previous relationships. On average, 2^8 combinations of these four bytes (i.e. one for each possible value of $k_{1,3}$) satisfy the relations (8) for each pair of the ciphertexts. In other words, given two texts c^1 and c^2, on average there are $(2^8)^4 = 2^{32}$ keys k for which the condition $R_k^{-1}(c^1) \oplus R_k^{-1}(c^2) \in \mathcal{M}_i$ is satisfied (as already mentioned before). The idea is to test them using the second pair of ciphertexts: on average, only the right combination passes the test. The same procedure is used for the others columns of $SR^{-1}(k)$.

The total computational cost of the attack is well approximated by the cost of the first phase, that is by the cost to find (on average) the 2^8 combinations of $k_{0,0}$, ..., $k_{3,1}$ that satisfy (8) for the first column—similar for the others (the cost of checking them with the second pair of texts is negligible). In particular, the computational cost of this attack using 3 chosen plaintexts can be approximated by $2^{17.1}$ S-Box look-ups (and negligible memory cost), or approximately $2^{11.6}$ table look-ups and a memory cost of $16 \times 2^{12} = 2^{16}$ using a precomputation phase. We refer to [18] for all the details.

Finally, we emphasize that the same attack works exactly in the same way also in the decryption direction (chosen ciphertexts attack) with the same complexity. In this case, the

[13] We emphasize that *this attack has been practical verified* (see [18] for details).

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 (i.e. $k_{i,i}$ for each $i = 0, ..., 3$).
(*Note*: the same procedure with the *same* ciphertexts can be used to recover the other diagonals of the key.)
for all values *of* $k_{1,3}$ **do**
  for all values *of* $k_{0,0}$ **do**
    check if $s^h_{0,0} = 0\text{x}02 \cdot s^h_{1,3}$ is satisfied for both pairs of ciphertexts, where $s^h_{i,j} = \text{S-Box}^{-1}(c^1_{i,j} \oplus k_{i,j}) \oplus \text{S-Box}^{-1}(c^h_{i,j} \oplus k_{i,j})$ for $h = 2, 3$
    if *satisfied* **then**
      identify candidates for $k_{1,3}$ and $k_{0,0}$;
      repeat the same procedure for $k_{2,2}$ and $k_{3,1}$, that is check if the equivalence $s^h_{2,2} = s^h_{1,3}$ and $s^h_{3,1} = 0\text{x}03 \cdot s^h_{1,3}$ are satisfied;
    end
  end
end
return candidate of the first diagonal of k

Algorithm 3: *Key-recovery Attack - Pseudo Code*. For simplicity, in this pseudo-code, we show how to find only the first diagonal of the secret key that verify relationship $\mathcal{R}$. To recover the entire key, it is sufficient to repeat the same attack for the other diagonals using the same pairs of ciphertexts. For more details, see Algorithm 5 of [18].

idea is to look for a key such that the corresponding plaintexts belong to the same coset of $\mathcal{D}_i$ after one round (see [18] for details).

The verifier strategy. The verifier simply applies the previous strategy in order to find key k^{10} (analogous for k^0). First of all, given a single tuple, there exist on average $4 \cdot (2^8)^4 = 2^{34}$ keys of the final round such that the two ciphertexts belong to the same coset of $\mathcal{M}_l$ one round before for a certain $l \in \{0, 1, 2, 3\}$. Since the probability that two keys k^{10} are equal is 2^{-128} and given n tuples, the probability that at least one key k^{10} exists (for which the previous requirements are satisfied) is given by:

$$2^{34n} \cdot 2^{-128(n-1)} = 2^{-94 \cdot n + 128}.$$

By this preliminary analysis, it is already possible to deduce that the number of tuples should be at least 2 (i.e. $n \geq 2$). Indeed, for $n = 1$ such a key always exists (which implies that using a random tuple it is possible to win the game), while for $n = 2$ the probability that such key exists for two random tuples is only 2^{-60}.

Thus, assume that the verifier receives $n \geq 2$ tuples. Given the first tuple and working independently on each column as described before, the attacker finds 2^8 combinations for each column of $SR^{-1}(k)$ and checks them immediately with the second tuple. Since she repeats this attack for each possible $\mathcal{M}_l$ (i.e. 4 times), the cost of this step is of $4 \cdot 2^{17.1} = 2^{19.1}$ S-Box look-ups. In this way, the verifier finds on average only one key (if it exists). If at least one possible key is found using two tuples, she simply checks if the other $n - 2$ tuples satisfy the relation $\mathcal{R}$ for this found key. The cost of this operation is well approximated by $2 \cdot 16 = 2^5$ S-Box look-ups for each tuple (note that she must decrypt one round for two ciphertexts).

In conclusion, since the verifier applies the same attack on the plaintexts and on the ciphertexts, given $n \geq 2$ tuples, the cost of the verifier is well approximated by $2 \times [2^{19.1} + (n - 2) \cdot 2^5]$ S-Box look-ups, that is approximately $2^{12.5}$ 10-round encryptions if $n \ll 2^{14}$.

5.2 The shortcut player

The shortcut player can simply use the rebound attack described in [19] and in Sect. 4.1.2 for the known-key distinguisher on 8 rounds in order to find the n tuples that satisfy the above defined relation $\mathcal{R}$. Indeed, it is straightforward to prove that all the properties are satisfied, since for each tuple the two plaintexts belong to the same coset of $\mathcal{D}_i$ (for a certain i) after 1-round encryption (w.r.t. the real subkey k^0) and the two ciphertexts belong to the same coset of $\mathcal{M}_l$ (for a certain l) 1-round decryption (w.r.t. the real subkey k^{10}) by construction. Since the computational cost to build one tuple is of 2^{44} encryptions, the cost to construct n tuples is well approximated[14] by $n \cdot 2^{44}$.

Before going further, we mention that the same strategy works even if a key-schedule is considered (that is, if the subkeys are independent), since the texts generated in the previous way satisfied the required property w.r.t. the real subkeys k^0 and k^{10}.

5.3 The generic player

Here we analyze and present the (intuitively) best strategy that the generic player can use in order to find n tuples with the required properties, and the corresponding computational cost. Intuitively, the best strategy for this player is to choose tuples for which the condition on the plaintexts is fulfilled with probability 1. Then, the player asks the oracle for the corresponding ciphertexts. The idea is to check if there exists a subkey k^{10} and n tuples such that the two ciphertexts of each of these n tuples belong to the same coset of $\mathcal{M}_l$ one round before. We remember that it is not necessary that the subkey for which this condition is satisfied is the real one (similar for the plaintext). This process is repeated until the n tuples of pairs of texts that satisfy the required relation $\mathcal{R}$ are found.

Working on the Plaintexts. To do this, the generic player must fix a random key $\hat{k}$, and computes for a certain $j \in \{0, ..., 3\}$ and for a random $a \in \mathcal{D}_j^{\perp}$ the following set:

$$D_a := R_{\hat{k}}^{-1}(\mathcal{D}_j \oplus a). \tag{9}$$

The idea is to choose plaintexts in the set D_a we just defined. In this way, the property on the plaintexts is (obviously) satisfied. The corresponding ciphertexts are simply got by oracle-queries. Since the cardinality of a coset of $\mathcal{D}_j$ is 2^{32}, the computation of a set D_a requires $16 \cdot 2^{32} = 2^{36}$ S-Box look-ups for each coset $\mathcal{D}_j \oplus a$. Note that if the player needs more than 2^{32} (plaintext, ciphertext) pairs, he simply chooses another $a' \in \mathcal{D}_j^{\perp}$ (or/and another j) and, using the *same* key $\hat{k}$, he computes the corresponding set $D_{a'}$ defined as before. We emphasize that the player must always use the same key $\hat{k}$ to compute these sets, in order to fulfill the property on the plaintexts.

Working on the Ciphertexts. As we have already seen, given a single tuple there exist on average 2^{34} keys such that the two ciphertexts belong to the same coset of $\mathcal{M}_j$ one round before. To set up a meaningful distinguisher, a value of n is suitable if the number of oracle-queries (hence, the cost) of the generic player is higher than the cost of the shortcut player. By previous observations, given a set of n tuples, the probability that at least a key exists for which the property on the ciphertexts is satisfied is $2^{-94n+128}$. Thus, the idea is to estimate

[14] We do not exclude the possibility of some trade-offs that could allow to reduce the computational cost to construct n tuples, i.e. such that the total computational cost which increases less than linearly. However, for our results, the "roughly linear" approximation is sufficient.

the number of (plaintext, ciphertext) pairs that this player has to generate in order to win the game (that is, in order to find with high probability n tuples with the required property). If this number is higher than $2^{44} \cdot n$ (for a fixed n), then the other player wins the game.

Since each set D_a defined as before contains 2^{32} different plaintexts, it is possible to construct approximately 2^{63} different couples $\{(p^1, c^1), (p^2, c^2)\}$. Given t different sets D_a, it is possible to construct $s = 2^{63} \cdot t$ different couples. It follows that one can construct approximately

$$\binom{s}{n} \approx \frac{s^n}{n!}$$

different sets of n different tuples (i.e. n different couples $\{(p^1, c^1), (p^2, c^2)\}$), where the previous approximation holds for $n \ll s$. Since the probability that a set of n tuples satisfies the above defined relation $\mathcal{R}$ is $2^{-94n+128}$, the generic player must consider at least s different couples such that $s^n/n! \simeq 2^{94n-128}$ or equivalently

$$s \simeq 2^{94-\frac{128}{n}} \cdot (n!)^{\frac{1}{n}}. \tag{10}$$

By this formula, for $n = 8$ this player has to consider approximately $2^{79.9}$ different tuples, or equivalently $2^{48.9}$ (plaintext, ciphertext) pairs (that is, $2^{16.9}$ initial different sets D_a). In other words, given $2^{16.9}$ initial different sets D_a, it is possible to construct approximately $2^{16.9} \cdot 2^{63} = 2^{79.9}$ different couples, that is approximately 2^{624} different sets of 8 tuples. Since each one of these sets satisfies the required properties with probability $2^{-94 \cdot 8+128} = 2^{-624}$, he has a good probability to find 8 different tuples that satisfy the required relation. The cost to generate these $2^{48.9}$ (plaintexts, ciphertexts) pairs is of $2^{48.9}$ oracle-queries (with the assumption 1 oracle-query $\simeq$ 1 encryption), while the cost to generate these 8 tuples for the shortcut player is of $8 \cdot 2^{44} = 2^{47}$ (which is smaller). Since the cost of the generic player is higher than the cost of the shortcut player for each n s.t. $n \geq 8$, we finally choose an (arbitrary) value of $n = 64$ in order to make the advantage of the shortcut player more significant.

Finally, in the case in which a key-schedule holds, the generic player has to repeat the previous strategy until the subkeys k^0 and k^{10}—for which the texts satisfy the required property $\mathcal{R}$—satisfy the key-schedule. Since a further property must be satisfied, the game becomes harder for the generic player (while, as we have seen before, nothing changes for the shortcut player).

6 New 12-round distinguisher of AES

As one of the major contributions of this paper, in this section we present the first known-key distinguisher for 12-round AES. This distinguisher is obtained by extending the previous 10-round distinguisher both at the end and at the beginning, or equivalently by extending two times at the end and at the beginning the 8-round known-key distinguisher presented in [19] and in Sect. 4.1.2. We highlight that this result provides a counterexample to the claims made in [16].

In the known-key distinguisher scenario, the players have to send to the verifier $n \geq 2^{38}$ different tuples of (plaintext, ciphertext) pairs, that is $\{(p_i^1, c_i^1), (p_i^2, c_i^2)\}$ for $i = 0, ..., n-1$, such that:

1. there exist keys k^0, k^1 s.t. for each tuple there exists j for which the two plaintexts belong to the same coset of $\mathcal{D}_j$ after two rounds, that is

$$\exists\, k^0, k^1 \text{ s.t. } \forall i \in \{0, ..., n-1\}, \quad \exists j \in \{0, ..., 3\} \text{ s.t. } R^2_{k^0,k^1}(p_i^1) \oplus R^2_{k^0,k^1}(p_i^2) \in \mathcal{D}_j;$$

2. there exist keys k^{11}, k^{12} s.t. for each tuple there exists l for which the two ciphertexts belong to the same coset of $\mathcal{M}_l$ two rounds before, that is

$$\exists k^{11}, k^{12} \text{ s.t. } \forall i \in \{0, ..., n-1\},\ \exists l \in \{0, ..., 3\} \text{ s.t. } R^{-2}_{k^{11},k^{12}}(c_i^1) \oplus R^{-2}_{k^{11},k^{12}}(c_i^2) \in \mathcal{M}_l;$$

where $R^2_{k^0,k^1}(\cdot) = R_{k^1}(R_{k^0}(\cdot))$ and $R^{-2}_{k^{11},k^{12}}(\cdot) = R^{-1}_{k^{11}}(R^{-1}_{k^{12}}(\cdot))$.

As for the known-key distinguisher for 10-round AES, here we limit ourselves to present a known-key distinguisher for 12-round of AES with independent subkeys (that is, no key schedule is considered). However, using similar argumentation as before, we claim that the same distinguisher can be applied to the case in which the key schedule holds (in this case, nothing changes for the shortcut player, while the challenge becomes much harder for the other player).

The strategy used by the players and by the verifier is very similar to the one presented for the 10-round distinguisher in the case of no key-schedule. For this reason, we refer to the previous section for all the details, and we limit ourselves here to highlight the major differences.

6.1 The shortcut and the generic player

Exactly as before, the shortcut player can generate n tuples of texts with the required properties exploiting the Rebound attack, for a cost of $n \cdot 2^{44}$ computations.

The generic player exploits a strategy similar to the one proposed for the 10-round distinguisher with no key-schedule. First he fixes random keys $\hat{k}^0$, $\hat{k}^1$ and $\hat{k}^{12}$, and using the keys $\hat{k}^0$ and $\hat{k}^1$, he computes the set $D_a = R^{-1}_{\hat{k}^0}(R^{-1}_{\hat{k}^1}(\mathcal{D}_j \oplus a))$. Similar to the previous case, the idea is to work with plaintexts in the same set D_a. He then gets the corresponding ciphertexts by oracle-queries, and he simply decrypts them using the key $\hat{k}^{12}$. Then, using the same strategy proposed for the 10-round distinguisher, he can construct n tuples that satisfy the relation $\mathcal{R}$, that is he is able to find n tuples of texts for which a common key k^{11} exists such that the requirement on the ciphertexts is satisfied.

By analogous calculation as before, at least $n \geq 8$ tuples are sufficient to set up a meaningful distinguisher when only the number of oracle-queries is considered.

6.2 The verifier

When the verifier receives the n tuples, she can work as in the case of the 10-round distinguisher in order to check if the required properties are satisfied or not. First of all, since there is no key schedule, the verifier can work independently on k^0, k^1 (that is on the plaintexts) and on k^{11}, k^{12} (that is on the ciphertexts). Similarly to the previous case, also for this 12-round distinguisher the idea is to exploit a truncated differential key-recovery attack to find (if they exist) the four keys k^0, k^1 and k^{11}, k^{12}. Such truncated differential attack—presented on 4-round AES in [18] and recalled in Appendix D—is obtained by extending the attack presented in the previous section for the 10-round case.

In the following, we limit ourselves to work on the ciphertexts (analogous for the plaintexts). Given the first tuple and using the strategy described in Appendix D, the verifier first guesses eight bytes of the final subkey k^{12} (two diagonals) and partially decrypts the texts. Exploiting the same strategy proposed for the 3-round key-recovery attack described in Sect. 5.1, she finds 2^{34} values for eight bytes of k^{11}, for a total of $2^{34} \cdot 2^{64} = 2^{98}$ candidates for eight bytes of k^{11} and of k^{12}. Then, she eliminates wrong candidates by testing them against

the other tuples of texts—to reduce the computational cost, she can work independently on each column of k^{11}. Since the probability that the subkeys k^{11} and k^{12} satisfy the required property for another tuple of texts is $4 \cdot 2^{-32} = 2^{-30}$, using other four tuples the verifier finds approximately only one pair of subkeys k^{11} and k^{12} for which the property on the ciphertexts is satisfied (note $2^{98} \cdot (2^{-30})^4 = 2^{-22}$). The cost of this step is of 2^{76} table look-ups (using the pre-computation phase)—see Appendix D or in [18] for details. The remaining eight bytes of k^{11} and of k^{12} can be found in a similar way.

As a result, given 5 different tuples, the total cost for this attack is approximately of $4 \cdot 2^{76} = 2^{78}$ table look-ups (using the pre-computation phase). When the verifier has found a possible candidate for the four subkeys, she checks that also the other $n - 5$ tuples satisfy the relation $\mathcal{R}$ for the found keys. In conclusion, given $n \geq 5$ tuples, the total cost for the verifier can be approximated at $2 \cdot (2^{78} + 2^6 \cdot (n-5))$ table look-ups in order to find the four required subkeys. If $n \ll 2^{72}$, then the computational cost of the verifier is approximately of $2^{71.1}$ twelve-round encryptions.

6.3 About the "number n of different tuples of (plaintext, ciphertext) pairs"

Due to the previous analysis, the distinguisher can be considered meaningful for $n \geq 8$. However, if $n = 8$ then the cost of the shortcut player (2^{47} computations) is much smaller than the cost of the verifier ($2^{71.1}$ computations), which is not consistent with the given definition of known-key distinguisher (see Sect. 3.1). Indeed, by definition the verification cost must be less than the cost of the shortcut player (and so less than the cost of the generic player). In order to fulfill this condition, it is sufficient to choose a number of tuple n that satisfy the condition $n \cdot 2^{44} \gg 2^{71.1}$ (and $n \ll 2^{72}$). It follows that a good (arbitrary) choice for this distinguisher[15] could be $n \geq 2^{38}$.

In conclusion, to win the game, the two players have to send 2^{38} tuples of (plaintext, ciphertext) pairs with the required properties. The cost for the shortcut player is of 2^{82} computations, while the verification cost is of $2^{71.1}$ computations.

7 Extending Gilbert's distinguisher to 12-round AES

In this section, we show that Gilbert's 10-round distinguisher can be extended to 12-round AES still exploiting the uniform distribution property. The main argumentation made in [16] about the impossibility of such extension regards the impossibility to efficiently check the relationship $\mathcal{R}$ when more than a single round is added a the beginning (resp. at the end) of the 8-round distinguisher[16]. To solve this problem, we make use of the verification strategy proposed in Sect. 4.2.4.

In the following, we first formally define the 12-round distinguisher based on the uniform distribution property, and – after showing that $\mathcal{R}$ is efficiently checkable—we prove that this new 12-round distinguisher is meaningful using the same argumentation proposed in [16] for the 10-round case.

[15] By previous analysis, we remember that the cost of the shortcut player is always lower than the cost of the generic player for each value of n that satisfies $n \geq 8$.

[16] Observe that in [16] the verifier works simultaneously on 4 bytes of the key. If one adds another round, it follows that the only way to decrypt two rounds to check a particular property is to guess one full subkey. This implies that the cost of the verifier is higher than 2^{128}, that is higher than the costs of the two players.

7.1 Gilbert's distinguisher (based on uniform distribution property) on 12-round AES

Using Gilbert's 10-round distinguisher as a starting point, a formal definition of the 12-round known-key distinguisher based on the uniform distribution property is given in the following. In the known-key distinguisher scenario, the players have to send to the verifier $n \geq 4$ *different* sets of 2^{64} (plaintext, ciphertext) pairs, that is (p_i^j, c_i^j) for $i = 0, ..., 2^{64} - 1$ and $j = 0, ..., n - 1$, such that

1. there exist keys k^0, k^1 such that for all $j = 0, ..., n - 1$ the texts $\{R_{k^1}(R_{k^0}(p_i^j))\}_i$ are uniformly distributed among the cosets of $\mathcal{D}_I$ for each $I \subseteq \{0, 1, 2, 3\}$ with $|I| = 3$, or equivalently s.t. for all $j = 0, ..., n - 1$ the bytes of the texts $\{R_{k^1}(R_{k^0}(p_i^j))\}_i$ are uniformly distributed;
2. there exist keys k^{11}, k^{12} such that for all $j = 0, ..., n-1$ the texts $\{R^{-1}_{k^{11}}(R^{-1}_{k^{12}}(c_i^j))\}_i$ are uniformly distributed among the cosets of $\mathcal{M}_J$ for each $J \subseteq \{0, 1, 2, 3\}$ with $|J| = 3$, or equivalently s.t. for all $j = 0, ..., n - 1$ the bytes of the texts $\{R^{-1}_{k^{11}}(R^{-1}_{k^{12}}(c_i^j))\}_i$ are uniformly distributed.

As for Gilbert's distinguisher, we assume that all subkeys are independent, that is no key-schedule holds. However, due to the same argumentation given before, we recall that the same distinguisher works exactly in the same way also in the case in which a key-schedule holds. Moreover, we emphasize that the keys k^0, k^1, k^{11} and k^{12} for which the previous properties are satisfied must be the same for all n sets of 2^{64} texts. In other words, given n sets s.t. the first set satisfies the property of uniform distribution for some keys k^0, k^1, the second for some other keys $\hat{k}^0, \hat{k}^1$ and so on, then the required properties can not be considered fulfilled.

As we are going to show, this distinguisher is meaningful for $n \geq 2$: our choice of $n = 4$ is due in order to make the advantage of the shortcut player more significant. Using the same strategy proposed for the 8-round distinguisher and for the 10-round one, the cost of the shortcut player to construct the $n = 4$ sets of 2^{64} (plaintext, ciphertext) pairs with the required properties is of $4 \cdot 2^{64} = 2^{66}$ encryptions. In the following, we show that (1) the verification cost is smaller than 2^{66} encryptions and that (2) the probability of victory of the shortcut player using 2^{66} oracle queries is negligible.

7.2 The verifier

The main problem in order to extend Gilbert's 10-round distinguisher regards the verification process. For this reason, here we analyze this problem in detail. In order to set up the verification process in an efficient way, we exploit the strategy that we proposed in Sect. 4.2.4:

- in the first step, the verifier filters wrong keys by checking zero-sum property;
- in the second step, the verifier checks if the uniform distribution property is satisfied *only* for those keys for which the balance property holds.

Integral attack. Consider the case in which the final MixColumns operation is omitted (if it is not omitted, it is sufficient to swap the final MixColumns and the final AddRoundKey) and the case in which the attacker looks for keys k^{11}, k^{12} (analogous for k^0, k^1). Using a classical integral attack with the extension at the end (see [13] for all the details) the verifier guesses 5 bytes of the keys, that is 4 bytes (i.e. one anti-diagonal) of the last subkey k^{12} and 1 byte of the subkey k^{11}. She first uses the 4 bytes of the last key to partially decrypt one

round, and then she simply checks that the zero-sum property holds 2-round before (working on each byte independently of the others).

The complete pseudo-code of this attack is given in Algorithm 4, where the verifier looks for 4 bytes of the last subkey k^{12} and (simultaneously) for 4 of the subkey k^{11}. Working on 4 bytes of k^{11}, the zero-sum property is satisfied with prob. 2^{-32}. Since the verifier tests 2^{32} values of 4 bytes of k^{12} and 2^{32} of 4 bytes of k^{11}, we expect that using a single set (i.e. $n = 1$) of 2^{64} texts, then $2^{64} \cdot 2^{-32} = 2^{32}$ combinations of these 8 bytes pass this first test (analogous for the other 4 combinations of 8 bytes). The idea is to test these 2^{32} combinations using a second set of texts (thus, $n \geq 2$). It follows that *on average* only 1 combination of these 8 bytes of the keys passes the test. Using the same procedure, the verifier is able to find keys k^{12} and k^{11} for which the zero-sum property is satisfied. Finally, as for the 10-round case, the idea is to work as described in Algorithm 1 in order to check that the uniform distribution property is satisfied.

Computational Cost of the Verifier. First of all, given 2 sets of 2^{64} texts the verifier must compute the array A as defined in Algorithm 4 and Algorithm 1. The cost of this step is given by $2 \cdot 2^{64}$ look-ups table (note that all these arrays can be computed and stored at the same time).

In order to compute the arrays B of Algorithm 4, the verifier must compute $2^{32} \cdot 2^{32} \cdot 4 = 2^{66}$ S-Box and $2^{32} \cdot 2^{32} \cdot 2 = 2^{65}$ look-ups table. Given the arrays B, the cost to find the candidates for 4 bytes of the key k^{12} and 4 bytes of key k^{11} is given by

$$\underbrace{2^{32}}_{\text{due to } k^{12}} \cdot \underbrace{4 \cdot 2^{8}}_{\text{due to } k^{11}} \cdot 2^{8} \cdot 2 = 2^{51} \text{ S-Boxes}$$

and 2^{51} table look-ups. Assuming that the cost of 1 S-Box look-up is equivalent to the cost of 1 table look-up[17], the total cost of the verifier to compute Algorithm 4 and to find the entire keys (i.e. k^0, k^1 and k^{11}, k^{12}) is given by $2 \cdot 4 \cdot (2^{66} + 2^{65} + 2 \cdot 2^{51}) \simeq 2^{69.6}$ table look-ups, or equivalently $2^{61.8}$ 12-round encryptions.

For the (few) candidates of the key that satisfy zero-sum, the verifier tests the uniform distribution property. Given n sets of texts, the *total* cost is well approximated by $2^{69.6} + n \cdot 2^{64}$ table look-ups (note that the cost to check the uniform distribution property is basically negligible w.r.t. the cost of finding the keys for which the zero-sum is satisfied), that is $2^{61.8} + n \cdot 2^{56.1}$ 12-round encryptions. It follows that the cost of the verifier is lower than the costs of the two players (which is higher than $n \cdot 2^{64}$ encryptions).

Partial-sum attack. Finally, we mention that the verifier can use the "Partial-Sum" key-recovery attack [15,30] in order to find the keys k^0, k^1 and k^{11}, k^{12} (much) faster than using the interpolation attack described before. On the other hand, we highlight that this is out of the scope of the paper[18]. Indeed, as we are going to show, the verification cost is already smaller than the costs of the two players using the integral attack, which means that the distinguisher can be set up using a classical integral attack for the verification phase.

[17] We highlight that even if this approximation is not formally correct - the size of the table of an S-Box look-up is lower than the size of the table used for our proposed distinguisher, it allows to give a comparison between our proposed distinguisher and the others currently present in the literature. At the same time, we note that the same approximation is largely used in the literature, as for example in [16].

[18] Since *the goal of this paper* is just to prove that a 12-round Gilbert's like distinguisher is potentially possible, we do not present such strategy in details (and we refer to [15,30] for more details).

Data: 2 sets of 2^{64} texts t^i and $\hat{t}^i$ for $i = 0, ..., 2^{64} - 1$
Result: One anti-diagonal of k^{12}—e.g. $(k^{12}_{0,0}, k^{12}_{1,3}, k^{12}_{2,2}, k^{12}_{3,1})$—and one column of k^{11}—$(k^{11}_{0,0}, k^{11}_{1,0}, k^{11}_{2,0}, k^{11}_{3,0})$—s.t. $\bigoplus_i R^{-1}_{k^{11}} \circ R^{-1}_{k^{12}}(p^i)_{j,k} = 0$ for each $j = k$
Let $A_j[0, ..., 2^{32} - 1]$ and $B^0_j[0, ..., 255]$, $B^1_j[0, ..., 255]$, $B^2_j[0, ..., 255]$, $B^3_j[0, ..., 255]$ ten arrays initialized to zero for $j = 0, 1$;
for i from *0 to* $2^{64} - 1$ **do**
  $x \leftarrow t^i_{0,0} + 2^8 \cdot t^i_{1,3} + 2^{16} \cdot t^i_{2,2} + 2^{24} \cdot t^i_{3,1}$;
  $A_0[x] \leftarrow (A_0[x] + 1) \bmod 2$; `// A[x] denotes the value stored in the x-th address of the array A` $\hat{x} \leftarrow \hat{t}^i_{0,0} + 2^8 \cdot \hat{t}^i_{1,3} + 2^{16} \cdot \hat{t}^i_{2,2} + 2^{24} \cdot \hat{t}^i_{3,1}$;
  $A_1[\hat{x}] \leftarrow (A_1[x] + 1) \bmod 2$;
end
for k^{12} from $(0x00, 0x00, 0x00, 0x00)$ *to* $(0xff, 0xff, 0xff, 0xff)$ **do**
  for s from $(0x00, 0x00, 0x00, 0x00)$ *to* $(0xff, 0xff, 0xff, 0xff)$ **do**
    Let $s \equiv (s^0, s^1, s^2, s^3) \in \mathbb{F}^4_{2^8}$ is a column of 4 bytes;
    Compute $x \equiv MC^{-1} \circ R_{k^{12}}(s)$; `// partial decryption of s w.r.t. to k - note:` $x \equiv (x^1, x^2, x^3, x^4) \in \mathbb{F}^4_{2^8}$ `is a column of 4 bytes`
    $B^k_j[x^k] \leftarrow (B^k_j[x^k] + A_j[i]) \bmod 2$ for each $j = 0, 1$ and $k = 0, 1, 2, 3$;
  end
  for $k^{11}_{0,0}$ from $0x00$ *to* $0xff$ **do**
    $x \leftarrow \bigoplus_{i=0}^{256} B^0_0[i] \cdot \text{S-Box}^{-1}(i \oplus k^{11}_{0,0})$; `// B[i] can only be 0 or 1`
    $\hat{x} \leftarrow \bigoplus_{i=0}^{256} B^0_1[i] \cdot \text{S-Box}^{-1}(i \oplus k^{11}_{0,0})$;
    if $x = \hat{x} = 0$ **then**
      Find $k^{11}_{1,0}$, $k^{11}_{2,0}$ and $k^{11}_{3,0}$ in a similar way (by repeating the last step);
      if *zero-sum satisfied* **then**
        Identify one anti-diagonal of k^{12} and one column of k^{11} as possible key;
      end
    end
  end
end
return candidates for k^{12} and k^{11}.

Algorithm 4: *First Part of Verifier Strategy:* filter wrong key candidates using zero-sum property.

7.3 The generic player: on the meaningfulness of this distinguisher

The last problem that we have to face regards the cost of the generic player. In particular, we have to show that the shortcut player can generate the n sets of 2^{64} texts (with the required property) in a more efficient way than the generic player.

What is the probability that given a set of 2^{64} texts there exist keys $\hat{k}^1$ and $\hat{k}^2$ such that the bytes of those texts after 2-round encryptions (resp. decryptions) are uniformly distributed? Due to similar calculation provided in Sect. 4.2 and since there are $(2^{128})^2 = 2^{256}$ different keys, this probability is equal to $2^{256} \cdot p \simeq 2^{256} \cdot 2^{-7328.1} = 2^{-7072.1} \equiv 2^{-2^{12.78}}$ where p is defined in (6).

More generally, given 2^{64} random pairs of texts, the probability that keys k^0, k^1 and k^{11}, k^{12} exist for which the bytes of the plaintexts/ciphertexts are uniformly distributed after 2-round encryption/decryption is $2^{512} \cdot 2^{-7328.1} = 2^{-6816.1} \equiv 2^{-2^{12.73}}$. Thus, similar to the 8-round case, it follows that $2^{64} + 2^{12} \simeq 2^{64}$ random (plaintext, ciphertext) pairs are sufficient

to have good probability to win the game[19]. In other words, $n \cdot (2^{64} + 2^{12}) \simeq n \cdot 2^{64}$ oracle queries – with random plaintexts/ciphertets—are sufficient for the generic player in order to win the game (note that—as done in [16]—this number does not take into account the cost to find the required pairs of texts).

Thus, *why is this distinguisher meaningful?* As for the 8- and the 10-round cases, instead of focusing on the cost of the players and using similar argumentation to the ones proposed by Gilbert, we show that the probability of the generic player to win the game given $n \geq 2$ sets of 2^{64} texts is negligible.

To do this, we claim that this probability is upper bounded by the probability of the following "related" game. Assume $n = 2$ and consider 2 sets of $2^{64} - d$ (plaintext, ciphertext) pairs for $d \geq 5$, that is (p^i, c^i) for each $i = 0, ..., 2^{64} - d - 1$, with the following property: there is a set of keys k^0, k^1 and k^{11}, k^{12}—which can correspond to the set of the entire keys—such that for each one of the two sets, the bytes of $R_{k^1} \circ R_{k^0}(p^i)$ and of $R^{-1}_{k^{11}} \circ R^{-1}_{k^{12}}(c^i)$ (that is 2-round encryption of p^i and the 2-round decryption of the ciphertexts) are "almost uniform" w.r.t. the definition given before. The goal of the player is to find $2 \cdot d$ texts such that the bytes of the 2^{64} texts of each set after 2-round encryption/decryption are uniformly distributed. The conclusion follows immediately since—as we are going to show – this probability is upper bounded by 2^{-25} and since this second game is "related" to the original one (as assumed in [16]).

More formally, and based on the same argumentation proposed by Gilbert—see proof of Prop. 4 of [16], it is possible to prove the following statement.

Proposition 1 *For any oracle algorithm $\mathcal{A}$ that makes $\leq N = 2 \cdot 2^{64} = 2^{65}$ oracle queries to a perfect random permutation Π or Π^{-1} of $\{0,1\}^{128}$, the probability that $\mathcal{A}$ outputs $n \geq 2$ sets of 2^{64}-tuple (X_i, Y_i) for $i = 0, ..., 2^{64} - 1$ that satisfy $Y_i = \Pi(X_i)$ and also satisfy $\mathcal{R}$ defined previously is upper bounded by $\binom{10}{5} \times 2^{512} \times \left(\frac{5^{16}}{2^{128}-(2^{64}-5)}\right)^6 \approx 2^{-25}$.*

Proof If at least one of the N pairs (X_i, Y_i) output by $\mathcal{A}$ does not result from a query X_i to Π or a query Y_i to Π^{-1}, then the probability that for this pair $Y_i = \Pi(X_i)$ and consequently the success probability of $\mathcal{A}$ is upper bounded by $\frac{1}{2^{128}-(N-1)}$. So from now on we only consider the opposite case, i.e. all the (X_i, Y_i) result from queries to Π or Π^{-1}.

As we have already said, a set of N texts is uniformly distributed if any subset is "almost" uniformly distributed w.r.t. the definition given before. Following the same argumentation provided by Gilbert for the 10-round case, we consider 2 sets of $2^{64} - 5$ (plaintext, ciphertext) pairs which are "almost" uniformly distributed for a set of keys k^0, k^1 and k^{11}, k^{12} after 2-rounds decryption/encryption, and we study the probability of the generic player to find the remaining $2 \cdot 5 = 10$ pairs such that there exist keys k^0, k^1 and k^{11}, k^{12} for which the bytes of the 2 corresponding sets of 2^{64} are uniformly distributed after 2-round encryption/decryption.

As shown in [16, Prop. 6], for each one of the two sets the probability that 5 pairs satisfy this condition is upper bounded by $\left(\frac{5^{16}}{2^{128}-(2^{64}-5)}\right)^3$. Moreover, observe that the player does 10 oracle queries, which can be divided in $\binom{10}{5} = 252$ different sets of 5 elements. Since the 2^{512} four subkeys are considered to be independent and must be equal for the two sets, one gets the claimed upper bound about the total probability of $\binom{10}{5} \times 2^{512} \times \left(\frac{5^{16}}{2^{128}-(2^{64}-5)}\right)^6 \approx 2^{-25}$. □

[19] Note that if the keys schedule holds (i.e. the subkeys are not independent), then the number of different subkeys is 2^{128} and not 2^{512}.

The same strategy applies for each $n \geq 2$. In particular, if $n = 4$ the probability becomes

$$\binom{20}{5} \times \binom{15}{5} \times \binom{10}{5} \times 2^{512} \times \left(\frac{5^{16}}{2^{128} - (2^{64} - 5)}\right)^{12} \approx 2^{-544.7}.$$

8 Discussion of results and proposal for a "New" model

In this paper, we improve all the known-key distinguishers (or present conjectures for such known-key distinguishers) currently present in the literature for AES from 7 up to 10 rounds of AES and we set up the first known-key distinguishers on 12 rounds of AES, by extending distinguishers based on truncated differential trails and uniform distribution property (using the technique proposed by Gilbert in [16]).

In order to extend Gilbert's distinguisher based on the uniform distribution property up to 12-round AES, we propose a different strategy that can be used by the verifier in order to check that the required relation $\mathcal{R}$ holds, and we present a formal proof which is based on the same argumentation proposed by Gilbert in order to justify the 8- and the 10-round distinguisher presented in [16]. For our new distinguishers using truncated-differential properties the situation is different: The problem to formally prove that no generic attack is better than those conjectured before remains open.

A "New" model: "Classical" known-key distinguisher

Taking a step back from the concrete results, what we also showed is that the gap between the known-key model and the chosen-key model may be even larger. Among the possibilities to remedy this counter-intuitive situation, we propose to define a new model that better capture the desire to have something "in-between" the chosen-key and the known-key model. Our proposal is to distinguish "classical" known-key distinguisher—where the verifier can directly verify the relation $\mathcal{R}$ on the plaintexts and ciphertexts without guessing any key material—and the "Gilbert" known-key distinguisher.

In particular, *characterizing a meaningful—or non-trivial—known-key distinguisher for a concrete cipher E remains an open problem.* Informally, a known-key distinguisher can be considered meaningful if the relation $\mathcal{R}$ (that defines it) has no "obvious connection" with the specification of E and is independent of the value of the key.

More generally, the relation $\mathcal{R}$ should not "extensively" re-use the operations that define E. Indeed, note that *if one considers a relation $\mathcal{R}$ that depends on the details of the internal primitives $E_K(\cdot)$, then any concrete implementable cipher* (like the AES instantiated by a *known* key) *can be trivially distinguished from an ideal cipher.* For instance, consider the following straightforward distinguishability attack. Assume the goal is to distinguish if an oracle is instantiated by a cipher $E_K(\cdot)$ or by an ideal cipher $\Pi(K, \cdot)$ under a known/chosen key K. Given a query X, one gets Y (which can be $Y = E_K(\cdot)$ or $Y = \Pi(K, X)$). Since the details of $E_K(\cdot)$ and the key K are known, one can simply compute $Y' = E_K(X)$. If $Y' = Y$, one can conclude that the oracle is instantiated by $E_K(\cdot)$.

"Classical" Known-Key Distinguisher: About the Relation $\mathcal{R}$. In order to achieve our goal, let us first introduce a set $\mathfrak{D}$ of distinguishers D defined as follows:

$\mathfrak{D}$ Set of Distinguishers: *$\mathfrak{D}$ denotes the set of all distinguishers D for which the relation $\mathcal{R}$ (that defines it) has no "obvious connection" with the specification*

of E (e.g. the details of the S-Box or/and of the round-constants etc.) *and it is independent of the value of the key*.

For a concrete example, note that a distinguisher that exploits the relation $X\mathcal{R}Y$ defined as $Y = E_X(X)$—as the one presented before—does not belong to $\mathfrak{D}$. Indeed, one has to know e.g. the details of the S-Box in order to check the relation $\mathcal{R}$ previously defined. The same happens for Gilbert's like distinguishers like [16], [10] and the ones introduced in this paper. Instead, a distinguisher that exploits the relation $(X_1, X_2)\mathcal{R}(Y_1, Y_2)$ as $X_1 \oplus X_2 \in \mathcal{X}$ and $Y_1 \oplus Y_2 \in \mathcal{Y}$ for particular subspaces $\mathcal{X}$ and $\mathcal{Y}$ (equivalently, X_1 and X_2 are equal in certain bits/bytes/words—similar for Y_1 and Y_2) belongs in $\mathfrak{D}$, since such a relation does not exploit any detail of $E(\cdot)$. Moreover, note that in this last case, $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ satisfy two *independent* properties, namely the fact that $X_1 \oplus X_2 \in \mathcal{X}$ is independent from the fact that $Y_1 \oplus Y_2 \in \mathcal{Y}$ (for every non-trivial $E(\cdot)$).

A possible way to formally define the set $\mathfrak{D}$ is to fix the relation $\mathcal{R}$ in advance, as done e.g. in [27]. In particular, consider a cipher $E_K(\cdot) : \mathbb{F}_2^N \to \mathbb{F}_2^N$ for a certain $N \in \mathbb{N}$. In [27], authors limit themselves to work with the class of known-key distinguishers whose relation $\mathcal{R}$ is defined as follows: given $\varphi, \psi \subseteq \{0, 1, ..., N-1\}$, the players have to send to the verifier n different (plaintext, ciphertext) pairs (p_i, c_i) for $i = 0, ..., n-1$ such that

$$\mathrm{Bit}_\varphi\big(p_0 \oplus ... \oplus p_{n-1}\big) = 0 \qquad \text{and} \qquad \mathrm{Bit}_\psi\big(c_0 \oplus ... \oplus c_{n-1}\big) = 0 \tag{11}$$

where $\mathrm{Bit}_\chi(x)$ outputs a string consisting of all bits of x whose index is in $\chi \subseteq \{0, 1, ..., N-1\}$. This relation allows to cover several known-key distinguisher in the literature, including the zero-sum one initial proposed by Knudsen and Rijmen in [20] (recalled in Appendix C.1) and the one based on the truncated diff./rebound attack proposed in [17,19,25] (recalled in Sect. 4.1.1-4.1.2). Moreover, it satisfies the definition previously given, since such a relation is independent of the specification of E. On the other hand, not all known-key distinguishers in the literature can be simply described using the property (11), as for example the known-key distinguisher on 8-round AES based on the uniform distribution property proposed in [16] (and recalled in Sect. 4.2.1) or the ones based on linear cryptanalysis (e.g. [9]).

As a result, *the problem to formalize—with a proper mathematical definition—the set* $\mathfrak{D}$ *of all distinguishers* D *for which the relation* $\mathcal{R}$ *(that defines it) has no "obvious connection" with the specification of* E *is still open for future research*.

"Classical" known-key distinguisher indifferentiability. With this in mind, we can define what "classical known-key indifferentiability" is:

Definition 10 Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher (where $(K, p) \mapsto c = E(K, p) = E_K(p)$), and let Π an ideal block cipher. Let $D \in \mathfrak{D}$ be a distinguisher with oracle access to a permutation and its inverse, and returning a single bit. The "*classical known-key indifferentiability*" (class)Inf-KK advantage of D is defined as

$$Adv^{\text{(class)Inf-KK}}(D) = \Big| Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot), E_K^{-1}(\cdot)}(K) = 1\big] + \\ - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot), \Pi^{-1}(K,\cdot)}(K) = 1\big]\Big|.$$

For integers q_D and t, the (class)Inf-KK advantage of E is defined as

$$Adv^{\text{(class)Inf-KK}}(q_D, t) = \max_{D \in \mathfrak{D}} Adv^{\text{(class)Inf-KK}}(D)$$

where the maximum is taken over all distinguishers (for which the relations $\mathcal{R}$ (that define them) has no "obvious connection" with the specification of E) making at most q_D oracle queries and running in time at most t. E is a (q, t, ε) (class)Inf-KK if $Adv^{\text{(class)Inf-KK}}(q_D, t) \leq \varepsilon$.

We point out that this definition is similar to the "known-key indifferentiability" one—denoted Inf-KK – proposed in [1, Def. 1]. In there, authors consider *known-key distinguishers for block ciphers based on idealized primitives such as randomly drawn functions or/and permutations*, e.g. an Even-Mansour construction $EM_r : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ defined as

$$EM_r(K, x) := k_r \oplus \pi(...\pi(k_1 \oplus \pi(x \oplus k_0)...)$$

for a fixed and *ideal* permutation π on n bits, where $k_0, ..., k_r$ denote the round keys derived from the master key K using some key-schedule.

However, here we point out the most important differences between these two definitions:

1. in Def. 1 of [1], the randomness is due to the fixed and *ideal* permutation π that defines $EM_r(K, \cdot)$; conversely, note that in our definition the first probability does not contain any randomness, but *there's a time complexity involved in* D;
2. since the "round function" π that defines $EM_r(K, \cdot)$ is ideal in [1, Def. 1], the problem to formalize the set of distinguisher $\mathfrak{D}$ (previously discussed) does not arise in this case (roughly speaking, since π is a random permutation, it is not possible to have any "obvious connection" between it and the relation $\mathcal{R}$ that defines the distinguisher). On the other hand, *the security against known-key distinguishers as defined in* [1, Def. 1] *is meaningless from a practical point of view*, since in practice we deal with ciphers with fixed and known/public round functions (as for the AES case). From this point of view, our work is more practically oriented.

"Classical" Known-Key Distinguisher: Some (useful) Properties/Considerations. Finally, we point out that

- if a cipher is a Strong PseudoRandom Permutation, then the ideal cipher in the (class)Inf-KK definition can be replaced by the encryption scheme instantiated with an unknown secret key;
- if a cipher is (class)Inf-KK secure, then it is also SPRP secure.

More formally:

Definition 11 Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher (where $(K, p) \mapsto c = E(K, p) = E_K(p)$), and let Π be an ideal block cipher. Let D be a distinguisher with oracle access to a permutation and its inverse, and returning a single bit. The (Strong PseudoRandom Permutation) SPRP-advantage of D is defined as

$$\begin{aligned} Adv^{\text{SPRP}}(D) = \big| &Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot), E_K^{-1}(\cdot)} = 1\big] \\ &- Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot), \Pi^{-1}(K,\cdot)} = 1\big]\big|. \end{aligned}$$

For integers q_D and t, the SPRP-advantage of E is defined as

$$Adv^{\text{SPRP}}(q_D, t) = \max_D Adv^{\text{SPRP}}(D)$$

where the maximum is taken over all distinguishers making at most q_D oracle queries and running in time at most t. E is a $(q, t, \varepsilon)-$SPRP if $Adv^{SPRP}(q_D, t) \leq \varepsilon$.

Proposition 2 *Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher which satisfies the SPRP ("Strong Pseudo-Random Permutation") definition. Then, E is (q, t, ε) (class)Inf-KK if and only if*

$$\max_{D \in \mathfrak{D}} \big| Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot), E_K^{-1}(\cdot)}(K) = 1\big] - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot), E_K^{-1}(\cdot)} = 1\big]\big| \leq \varepsilon$$

where the maximum is taken over all distinguishers making at most q_D oracle queries and running in time at most t.

Proposition 3 *If $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is (class)Inf-KK secure, then it also satisfies the SPRP ("Strong Pseudo-Random Permutation") definition.*

Informally, *if it is not possible to distinguish $E_K(\cdot)$ from Π when the key K is known, then it is not possible to distinguish them when the key is secret*. Vice-versa this is not true in general. As example, the best secret-key distinguishers on AES (independent of the key) covers 5/6 rounds [24,28], while the best known-key distinguisher covers 8 rounds (12 if one allows Gilbert's strategy).

Acknowledgements Authors thank Bart Mennink and Pooya Farshim for fruitful discussions and suggestions. Authors also thank anonymous reviewers for their valuable comments.

A "Classical" known-key security—details

A.1 Role of the ideal cipher in the known-key scenario

As already pointed out, if a cipher is a Strong PseudoRandom Permutation, then the ideal cipher in the (class)Inf-KK definition can be replaced by the encryption scheme instantiated with an unknown secret key. Informally, the ideal cipher is indistinguishable from the block cipher for which the key has been chosen at random.

Proposition 4 *Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher which satisfies the SPRP ("Strong Pseudo-Random Permutation") definition. Then, E is (q,t,ε) (class)Inf-KK if and only if*

$$\max_{D\in\mathfrak{D}} \left| Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K)=1\right] - Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}=1\right]\right| \leq \varepsilon$$

where the maximum is taken over all distinguishers making at most q_D oracle queries and running in time at most t.

Proof First, we prove that if E is (q,t,ε) (class)Inf-KK, then the claim holds:

$$\begin{aligned}
&\left| Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K)=1\right]\right.\\
&\quad \left. - Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}=1\right]\right|\\
&\leq \left| Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K)=1\right]\right.\\
&\quad \left. - Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot),\Pi^{-1}(K,\cdot)}(K)=1\right]\right|\\
&\quad + \left| Prob\left[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}=1\right]\right.
\end{aligned}$$

$$- Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot),\Pi^{-1}(K,\cdot)} = 1\big]\big|$$

The second term in the l.h.s. is smaller than ε since E is a SPRP. It follows that

$$\begin{aligned}
&\big|Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K) = 1\big] \\
&\quad - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)} = 1\big]\big| \\
&\leq \big|Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K) = 1\big] \\
&\quad - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot),\Pi^{-1}(K,\cdot)}(K) = 1\big]\big| + \varepsilon.
\end{aligned}$$

Using the same strategy, one can prove that

$$\begin{aligned}
&\big|Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K) = 1\big] \\
&\quad - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{\Pi(K,\cdot),\Pi^{-1}(K,\cdot)}(K) = 1\big]\big| \\
&\leq \big|Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)}(K) = 1\big] \\
&\quad - Prob\big[K \xleftarrow{\$} \{0,1\}^k; D^{E_K(\cdot),E_K^{-1}(\cdot)} = 1\big]\big| + \varepsilon.
\end{aligned}$$

Since the previous two equalities work for all $\varepsilon \geq 0$, the thesis follows. □

A.2 "(class)Inf-KK secure" implies SPRP

Proposition 5 *If $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is (class)Inf-KK secure, then it also satisfies the SPRP ("Strong Pseudo-Random Permutation") definition.*

Proof We are going to prove that if E is not a SPRP, then it is not inf-KK secure. If E is not a SPRP, then by definition there exists a distinguisher $\hat{D}$ such that

$$\begin{aligned}
Adv^{\text{SPRP}}(q_D, t) \geq Adv^{\text{SPRP}}(\hat{D}) = \big|Prob\big[K \xleftarrow{\$} \{0,1\}^k; \hat{D}^{E_K(\cdot),E_K^{-1}(\cdot)} = 1\big] + \\
- Prob\big[K \xleftarrow{\$} \{0,1\}^k; \hat{D}^{\Pi(K,\cdot),\Pi^{-1}(K,\cdot)} = 1\big]\big| \geq \varepsilon.
\end{aligned}$$

The idea is to build an (class)Inf-KK distinguisher D using $\hat{D}$ that has the same advantage in breaking E. Distinguisher D simulates the environment for $\hat{D}$ as follows: firstly, a random key $K \xleftarrow{\$} \{0,1\}^k$ is selected uniformly and D runs on the input K; then it forward all queries by $\hat{D}$—which is independent of K—to its own oracle. If $\hat{D}$ succeeds in distinguishing E and π, then D succeeds as well. In particular, we have $Adv^{\text{(class)Inf-KK}}(q_D, t) \geq Adv^{\text{(class)Inf-KK}}(D) = Adv^{\text{SPRP}}(\hat{D}) \geq \varepsilon$. □

B A possible variant of Gilbert's distinguisher—details

In Sect. 4.2, we proposed a possible variant of Gilbert's distinguisher—that also applies to all our proposed distinguishers present in the paper—which allows to better understand it. Consider the case in which the two players have to send to the verifier the N-tuple that verify the required relation $\mathcal{R}$ together with the subkeys for which such a relation is satisfied (here we assume that the relationship $\mathcal{R}$ depends on the existence of subkey(s) such that the

required property is not directly verified on the plaintexts or/and on the ciphertexts but one (or more) round(s) before/after[20]).

In this modified variant, the task of the verifier is to check if the relation $\mathcal{R}$ is satisfied (or not) *only* for the subkeys she received by the players. It follows that her computational cost is negligible, in the sense that it is comparable to the computational cost of the 8-round integral distinguisher presented in [16] where the required property $\mathcal{R}$ can be directly verified on the plaintexts/ciphertexts (or equivalently comparable to the computational costs of the other known-key distinguishers present in the literature up to 8 rounds). Here we show in details why such a distinguisher is meaningless.

The main problem regards the fact that such a distinguisher can be set up for any number of rounds. To explain this problem, consider our known-key distinguisher on $r = 8 + 2 \cdot r'$ rounds of AES, for $r' \geq 1$ (the same considerations apply e.g. to Gilbert distinguisher). The players have to send to the verifier n different tuples of (plaintext, ciphertext) pairs, that is $\{(p_i^1, c_i^1), (p_i^2, c_i^2)\}$ for $i = 0, ..., n-1$, *and* $2 \cdot r'$ subkeys $k^0, ..., k^{r'-1}$ and $k^r, ..., k^{r-r'+1}$ such that

1. for each tuple there exists $j \in \{0, ..., 3\}$ for which the two plaintexts belong to the same coset of $\mathcal{D}_j$ after r' rounds, that is

$$\forall i \in \{0, ..., n-1\},\ \exists j \in \{0, ..., 3\} \quad \text{s.t.} \quad R_{k^0,...,k^{r'-1}}(p_i^1) \oplus R_{k^0,...,k^{r'-1}}(p_i^2) \in \mathcal{D}_j;$$

2. for each tuple there exists $l \in \{0, ..., 3\}$ for which the two ciphertexts belong to the same coset of $\mathcal{M}_l$ r' rounds before, that is

$$\forall i \in \{0, ..., n-1\},\ \exists l \in \{0, ..., 3\} \quad \text{s.t.} \quad R^{-1}_{k^r,...,k^{r-r'+1}}(c_i^1) \oplus R^{-1}_{k^r,...,k^{r-r'+1}}(c_i^2) \in \mathcal{M}_l,$$

where $R_{k^0,...,k^{r'-1}}(\cdot) = R_{k^{r'-1}} \circ ... \circ R_{k^0}(\cdot)$ and $R^{-1}_{k^r,...,k^{r-r'+1}}(\cdot) = R^{-1}_{k^{r-r'+1}} \circ ... \circ R^{-1}_{k^r}(\cdot)$.

Consider now the costs of the verifier and of the two players. As we have already said, the cost of the verifier is negligible, since she has to check if the relation $\mathcal{R}$ is satisfied only for the received subkeys. The cost of the shortcut player is approximately of $n \cdot 2^{44}$ computations for n tuples, since she can use the rebound attack to find them (see Sect. 4.1.1 for details). The generic player instead can use the strategy proposed in details Sect. 5 for the 10 rounds case and in Sect. 6 for the 12 rounds one in order to win the game. Such strategy allows the player to find plaintexts (or ciphertexts) that satisfy the required condition with negligible computational cost. However, the only way to satisfy both the conditions (i.e. the relation $\mathcal{R}$) is to test the texts found in the first step by brute force. It follows that when the number n of required tuples increases, the computational cost of the generic player grows faster than the cost of the shortcut player. In other words, even if we do not exclude that a better strategy exists, it seems hard that the cost of the generic player can be lower than the cost of the shortcut one. By definition of known-key distinguisher given in Sect. 3.1, it follows that such a distinguisher can be set up for any number of rounds (of AES).

[20] A concrete example is obviously given by the 10-round known-key distinguisher proposed by Gilbert and based on the balance property.

C Known-key distinguishers for 7- and 8-round AES based on the uniform distribution and on the balance property

C.1 Known-key distinguisher based on balance property

The 7- and the 8-round known-key distinguisher based on the balance property are a direct application of the 3- and 4-round zero-sum secret-key distinguishers used in an inside-out fashion.

First of all, we recall some definitions. Given a set of texts, we say that a byte X could be:

- Active (A): Every value in $\mathbb{F}_{2^8}$ appears the same number of times in X;
- Balanced (B): The XOR of all values in X is 0;
- Constant (C): The value is fixed to a constant for all texts in X.

First, we formally define the 7- and the 8-round known-key distinguisher based on the balance property. Assume there are two players—one knows the key and the other not, and the verifier. To win the game, the players have to send to the verifier 2^n (plaintext, ciphertext) pairs, that is (p^i, c^i) for $i = 0, ..., 2^n - 1$, such that the balance property holds both on the plaintexts and on the ciphertexts: $\bigoplus_{i=0}^{2^n-1} p^i = \bigoplus_{i=0}^{2^n-1} c^i = 0$. A suitable value of n is 56 for 7 rounds of AES and 64 for 8 rounds case.

What is the best strategy that the shortcut player can use to win the game? Consider 2^{32} plaintexts with one active diagonal (i.e. 4 bytes), and all the others 12 bytes constants. It is a well-known fact that the sum of 2^{32} corresponding ciphertexts after four rounds is equal to zero. A similar property holds in the decryption direction, that is the following integral properties hold:

$$\begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix} \xleftarrow{R^{-3}} \begin{bmatrix} A & C & C & C \\ A & C & C & C \\ A & C & C & C \\ A & C & C & C \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A & C & C & C \\ C & A & C & C \\ C & C & A & C \\ C & C & C & A \end{bmatrix} \xrightarrow{R^{4}} \begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix}.$$

Equivalent, this means that if one takes a coset of $\mathcal{D}_i$ for a certain i, then the sum of the corresponding ciphertexts after 4 rounds is equal to zero. Again, if one takes a coset of $\mathcal{C}_j$ for a certain j as the set of ciphertexts, the sum of the corresponding plaintexts 3 rounds before is equal to 0. Thus, starting in the middle with a coset of $\mathcal{D}_i \oplus \mathcal{C}_j$ for a certain i and j, then the sum of the corresponding plaintexts 3 rounds before and the ciphertexts after 4 rounds is equal to 0:

$$\begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix} \xleftarrow{R^{-3}} \underbrace{\begin{bmatrix} A & C & C & C \\ A & A & C & C \\ A & C & A & C \\ A & C & C & A \end{bmatrix}}_{\equiv \text{ a coset of } \mathcal{D}_i \oplus \mathcal{C}_j} \xrightarrow{R^{4}} \begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix}.$$

This distinguisher on 7 rounds AES was proposed for the first time by Knudsen and Rijmen in [20].

Since a coset of $\mathcal{C}_j$ is mapped into a coset of $\mathcal{M}_j$ after one round with prob. 1, then given a coset of $\mathcal{M}_j$ for a certain j as the set of ciphertexts, the sum of the corresponding plaintexts 4 rounds before is equal to 0. Equivalently, starting in the middle with a coset of $\mathcal{D}_i \oplus \mathcal{M}_j$ for a certain i and j, then the sum of the corresponding plaintexts 4 rounds before and of the

ciphertexts after 4 rounds is equal to 0:

$$\begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix} \xleftarrow{R^{-4}} \mathcal{D}_i \oplus \mathcal{M}_j \oplus a \xrightarrow{R^4} \begin{bmatrix} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{bmatrix}$$

for a constant a.

C.2 About the validity of zero-sum known-key distinguishers

Here we analyze the validity of a zero-sum distinguisher in the scenario described in Sect. 3. In this case, the goal of the two players is to find an N-tuple of (plaintexts, ciphertexts) (p^i, c^i) for $i = 0, ..., N-1$ such that the sums of the plaintexts and of the ciphertexts are equal to zero, i.e.

$$\bigoplus_{i=0}^{N-1} p^i = \bigoplus_{i=0}^{N-1} c^i = 0.$$

Note that we do not require uniform-distribution of the plaintexts and/or of the ciphertexts.

As we are going to show, this problem cannot be used in order to set up meaningful distinguishers, since the cost of the generic player (to generate the N-tuple with the required zero-sum property) is similar to the cost of the shortcut player in the case in which both players have the same probability of success (assuming the cost of 1 oracle-query is equal to the cost of 1 encryption).

Shortcut Player. Assume that there exist two subspaces of texts $X = \{x^i\}_i$ and $Y = \{y^i\}_i$ such that $\dim(X \oplus Y) = n$ and such that for each φ, ψ the following properties hold:

$$\bigoplus_{x \in X \oplus \varphi \equiv \{x^i \oplus \varphi\}_i} R^s(x) \quad \text{and} \quad \bigoplus_{y \in Y \oplus \varphi \equiv \{y^i \oplus \psi\}_i} R^{-(r-s)}(y) = 0$$

for $0 < s < r$. Since

$$X \oplus Y = \bigcup_{y \in Y} X \oplus y = \bigcup_{x \in X} Y \oplus x$$

and since the shortcut player can work with intermediate states, he simply chooses texts in a coset of $X \oplus Y$ (e.g. $X \oplus Y \oplus \alpha$ for an arbitrary α) and simply defines the plaintexts as the $r - s$ rounds decryption of $X \oplus Y \oplus \alpha$ and the corresponding ciphertexts as the s rounds encryptions of $X \oplus Y \oplus \alpha$. Both the plaintexts and the ciphertexts defined in this way have the zero-sum property.

Generic player. One possible strategy that the generic player can use is the one proposed by Wagner in [31] in order to solve the k-sum problem. Given a function f on n bits, the k-sum problem is to find $x_1, \ldots, x_k$ such that $\bigoplus_{i=1}^{k} f(x_i) = 0$. A solution to this problem is given in [31] with a running time of $\mathcal{O}(N \cdot 2^{k/(1+\log_2 N)})$. This strategy has been used by Knudsen and Rijmen in [20]—the *first* authors that propose a zero-sum distinguisher—in order to estimate the computational cost of the generic player.

A more efficient way to find a solution of the analyzed problem is inspired by the attack against XHASH proposed in [3]. Assume we are looking for a set $\mathcal{Z} = \{z_i\}$ of N elements in $\mathbb{F}_{2^n}$ such that $\bigoplus_i z_i = \bigoplus_i f(z_i) = 0$. As first step, one consider N random value $x_i \in \mathbb{F}_{2^n}$

and compute $\mathcal{X} = \{x_i \| f(x_i)\}_i$ where $x_i \| f(x_i) \in \mathbb{F}_{2^{2n}}$. Let

$$S = \bigoplus_{\mathcal{X}} [x_i \| f(x_i)] \equiv [\bigoplus_{\mathcal{X}} x_i \| \bigoplus_{\mathcal{X}} f(x_i)].$$

If S is equal to zero (prob. 2^{-2n}), then the problem is solved.

If $S \neq 0$, for a certain $M \geq 2n$, the idea is to consider other M random elements $y_i \in \mathbb{F}_{2^n}$ and compute $\{y_i || f(y_i)\}_i$. Then, one look for binary coefficients $\{a_i\}_{i=1,\ldots,M}$ and M random elements y_i s.t. the following inequality

$$\bigoplus_{i=0}^{M} a_i \cdot ([x_i \| f(x_i)] \oplus [y_i \| f(y_i)]) = \bigoplus_{i=0}^{M} a_i \cdot [(x_i \oplus y_i) \| (f(x_i) \oplus f(y_i))] = S. \quad (12)$$

is satisfied. Given $M = 2n + \varepsilon$, the probability that no solution is found decreases exponentially by increasing ε.

Given a solution of the previous equality, the set $\mathcal{Z} = \{z_i\}$ defined as

$$z_i \equiv \begin{cases} a_i \cdot y_i \oplus (1 \oplus a_i) \cdot x_i & \text{if } i \leq M \\ x_i & \text{if } i > M \end{cases}$$

has the required zero-sum property: $\bigoplus_{z_i \in \mathcal{Z}} z_i = \bigoplus_{z_i \in \mathcal{Z}} f(z_i) = 0$. Indeed, since $a_i \cdot y_i \oplus (1 \oplus a_i) \cdot x_i$ is equal to y_i for $a_i = 1$ and equal to x_i otherwise, it follows that

$$\bigoplus_i [z_i \| f(z_i)] = \bigoplus_{i \leq M} (a_i \cdot [y_i \| f(y_i)] \oplus (1 \oplus a_i) \cdot [x_i \| f(x_i)]) \oplus \bigoplus_{i > M} [x_i \| f(x_i)] =$$
$$= \bigoplus_{i \leq M} (a_i \cdot ([y_i \| f(y_i)] \oplus [x_i \| f(x_i)]) \oplus \bigoplus_i [x_i \| f(x_i)] = S \oplus S = 0.$$

It follows that the computational cost is well approximated by:

- $N + 2n + \varepsilon$ encryptions;
- solve a linear system of $2n + \varepsilon$ equations[21].

If $N \gg 2n + \varepsilon$ (as in the case of AES, where $n = 128$ and $N = 2^{64}$), such a cost is well approximated by N computations/encryptions, that is it is "similar" to the cost of the shortcut player. As a result, it follows that such a distinguisher is not meaningful.

Final remark: zero-sum partition

Even if the zero-sum problem cannot be used in order to set up a meaningful distinguisher, a possible way to set up a known-key distinguisher based on the balance property is by considering the following slightly modified (but *much stronger*) problem:

Definition 12 (Zero-Sum Partition— [6]) Let P be a permutation from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$. A zero-sum partition for P of size $K = 2^k$ is a collection of $K = 2^k$ disjoint sets $\{X_1, X_2, ..., X_K\}$ sets with the following properties:

- $X_i = \{x_i^1, ..., x_i^{2^{n-k}}\} \subseteq \mathbb{F}_{2^n}$ for each $i = 1, ..., K$ s.t. $\bigcup_i X_i = \mathbb{F}_{2^n}$;
- for each $i = 1, .., K$:

$$\bigoplus_{x \in X_i} x = \bigoplus_{x \in X_i} P(x) = 0.$$

[21] The computational cost for inverting a $m \times m$ matrix is $\mathcal{O}(m^3)$. Since $2n + \varepsilon \ll N$, we emphasize that the cost of this step is negligible with respect to the cost of the first step.

The zero-sum partition problem has been largely used in the literature in order to set up *partition* zero-sum known-key distinguishers on Keccak-f [2,7,12] and on other permutations like PHOTON [33]. Note that the shortcut player can use the same strategy proposed before for the zero-sum problem in order to solve this partition zero-sum problem: it is indeed sufficient to consider all possible disjoint coset of $X \oplus Y$. For completeness, we mention that the Keccak Team published a note [4] where they confirm the validity of such distinguishers: "*[...] the zero-sum distinguishers of* [2,6] *are valid, albeit with a very small advantage*".

C.3 Known-key distinguisher based on uniform distribution

Finally, another possibility is to set up a known-key distinguisher on 7- and 8-round AES based on the uniform distribution property, as done in [16]. The idea is simply to connect two 4-round trails in the middle and to choose a middle space $\mathcal{D}_i \oplus \mathcal{M}_j$ for i and j fixed (with $|i| = |j| = 1$). In the middle, the set $\mathcal{D}_i \oplus \mathcal{M}_j$ can be re-written as

$$\bigcup_{b \in \mathcal{D}_i} \mathcal{M}_j \oplus b = \bigcup_{a \in \mathcal{M}_j} \mathcal{D}_i \oplus a,$$

that is as the union of cosets of the space $\mathcal{D}_i$ or as the union of cosets of the space $\mathcal{M}_j$.

Forward direction. If one encrypts $\mathcal{D}_i \oplus a$ for four rounds ($a \in \mathcal{M}_j$), then the set $R^{(4)}(\mathcal{D}_i \oplus a)$ is a set of $(2^8)^4 = 2^{32}$ ciphertexts where each ciphertext belongs to a different coset of a mixed space $\mathcal{M}_I$ of dimension 12. Thus if one encrypts all 2^{32} cosets of $\mathcal{D}_i$, we get all the 2^{32} cosets of $\mathcal{M}_I$, where each coset contains exactly 2^{32} ciphertexts. For completeness, if the final MixColumns operation is omitted, then the encryption of all 2^{32} cosets of $\mathcal{D}_i$ results in all the 2^{32} cosets of $\mathcal{ID}_I$, where each coset contains exactly 2^{32} ciphertexts.

Indeed, note that by Theorem 2 two elements that belong to the same coset of $\mathcal{D}_I$ can not belong to the same coset of $\mathcal{M}_J$ for $|I| + |J| \leq 4$. Thus, given a coset of $\mathcal{D}_i$ with $|i| = 1$, after 4 rounds each element is distributed in a different coset of $\mathcal{M}_J$ for $|J| = 3$. Note that $\mathcal{D}_i \oplus \mathcal{M}_j = \bigcup_{a \in \mathcal{M}_j} \mathcal{D}_i \oplus a$. Thus, since the coset of $\mathcal{D}_i$ contains 2^{32} elements and since there are exactly 2^{32} cosets of $\mathcal{M}_J$, the elements of $\mathcal{D}_i \oplus \mathcal{M}_j$ are uniformly distributed in each coset of $\mathcal{M}_I$.

Backward Direction. If one decrypts $\mathcal{M}_j \oplus b$ for four rounds ($b \in \mathcal{D}_i$), then—due to Theorem 2—the set $R^{(-4)}(\mathcal{M}_j \oplus b)$ is a set of 2^{32} plaintexts where each plaintext belongs to a different coset of a diagonal space $\mathcal{D}_J$ of dimension 12. If one decrypts all 2^{32} cosets of $\mathcal{M}_j$, one gets all the 2^{32} cosets of $\mathcal{D}_J$, where each coset contains exactly 2^{32} plaintexts.

D Truncated differential attack [18] on 4-round AES

To set up the first 12-round known-key distinguisher on AES, we recall (a modified version of) the low-data complexity truncated differential attack on 4-round of AES-128 proposed in [18], obtained by extending the attack on 3 rounds presented in Sect. 5.1 at the end. We refer to [18, Algorithm 6] for a complete description of the attack (we assume the final MixColumns is omitted).

Consider plaintexts in the same coset of $\mathcal{M}_i$ for $|i| = 1$ and the corresponding ciphertexts after two rounds. The goal of the attack is to find the key such that the ciphertexts belong to the same coset of $\mathcal{M}_i$ two rounds before. The idea of the attack is to guess two columns of $SR^{-1}(k^2)$, where k^2 is the final key. Given 5 plaintexts and the corresponding ciphertexts

(p^j, c^j) for $j = 1, ..., 5$, for each one of the 2^{64} possible values of these two columns of $SR^{-1}(k^2)$, the idea is to partially decrypt these 5 ciphertexts one round, that is to compute the eight bytes $s^j := R_{k^2}^{-1}(c^j)$ for each $i = 1, ..., 5$. Due to the ShiftRows operation, these 8 bytes are distributed in two columns. Thus, the idea is simply to repeat the previous attack on 3 rounds. However, since the eight bytes of s^i are uniformly distributed in the four columns (i.e., two bytes for each column), for each column one can only exploit the relationship that holds among these two bytes (see [18] for details).

Using two pairs of ciphertexts (e.g. (c^1, c^2) and (c^1, c^3)), it is possible to find (on average) at most one combination of eight bytes of k^2 for each possible guess of the eight bytes of k^2, for a total of 2^{64} possibilities. The idea is to test these found values against other pairs of ciphertexts, that is to check if the relationships among the bytes of the keys hold also for these other pairs of ciphertexts[22]. Since each relationship is satisfied with probability 2^{-32} (there are four relationships, each one satisfied with probability 2^{-8}), it is sufficient to test the found values of k^1 and k^2 against only other two pairs of ciphertexts, in order to eliminate all the wrong candidates with high probability. Thus, using 5 chosen plaintexts (i.e. 4 pairs with a common plaintext[23]), it is possible to recover 8 bytes of k^1 and of k^2. The full key is discovered by repeating the same procedure on the last two columns of k^2.

As shown in [18], the computational cost of this attack is well approximated by 2^{81} S-Box look-ups (with negligible cost of memory) or 2^{76} table look-ups and a memory cost of $16 \cdot 2^{12} = 2^{16}$ bytes. Moreover, the same attack works also in the decryption direction, with the same complexity. In particular, given ciphertexts in the same coset of $\mathcal{D}_i$ for $|i| = 1$ and the corresponding plaintexts two rounds before, the idea is to look for the keys such that the plaintexts belong to the same coset of $\mathcal{D}_i$ after two rounds.

E New 7- and 8–round AES known-key distinguishers

In this section, we propose a new 8-round known-key distinguisher for AES, which are obtained extending at the end or/and at the beginning a 7-round known-key distinguisher for AES. The strategy to set up them is the same used in Sect. 5. For this reason, we refer to those sections for all the details.

E.1 7-Round known-key distinguisher

For the follow-up, we briefly recall the current best known distinguisher on 8 rounds of AES (proposed in [19] and already presented in Sect. 4.1.2). This distinguisher is obtained starting from the 8-round distinguisher presented in [17], and depicted in Fig. 2. Using the subspace trail notation and the known-key distinguisher scenario, the goal of the two players in this distinguisher is to find a pair of (plaintexts, ciphertexts)—i.e. (p^1, c^1) and (p^2, c^2)—with the following properties: the two plaintexts belong to the same coset of $\mathcal{D}_i$—i.e. $p^1 \oplus p^2 \in \mathcal{D}_i$—

[22] This step is different from the one proposed in [18]. In that case, the idea is to find the right key by a brute force attack in order to keep the data complexity as low as possible. For our distinguisher, we propose to test the found key against other pairs of plaintexts and ciphertexts, since it is not possible to use a brute force attack.

[23] Note that 4 different pairs can be obtained by 3 chosen plaintexts. However, such pairs are not useful for the attack. Indeed, note e.g. that if $R_k^{-1}(c^1) \oplus R_k^{-1}(c^2) \in \mathcal{M}_i$ and $R_k^{-1}(c^1) \oplus R_k^{-1}(c^3) \in \mathcal{M}_i$, it follows that also $R_k^{-1}(c^2) \oplus R_k^{-1}(c^3) \in \mathcal{M}_i$ since $\mathcal{M}_i$ is a subspace. We refer to [18] for a complete and detailed explanation.

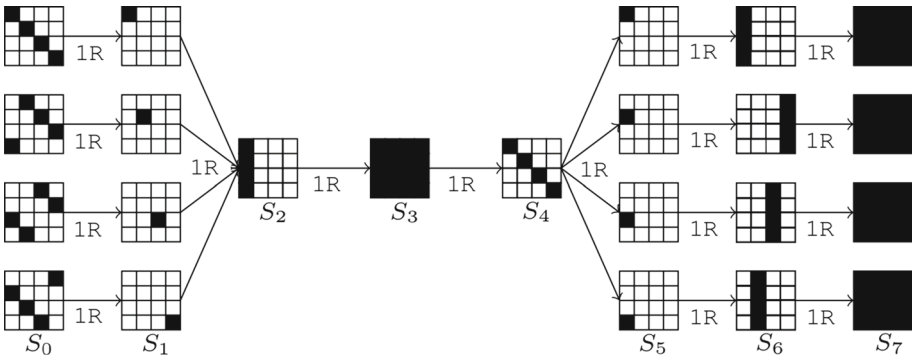

Fig. 4 7-round differential characteristic for known-key distinguisher of AES-128

and the two ciphertexts belong to the same coset of $\mathcal{M}_i$—i.e. $c^1 \oplus c^2 \in \mathcal{M}_i$, where the index i is fixed. The idea proposed in [19] to improve this distinguisher is simply to not fix the initial subspace $\mathcal{D}_i$ and the final one $\mathcal{M}_j$, that is to leave i and j be completely arbitrary (i.e. they can take any possible values). It follows that the probability that a solution of the inbound phase of the rebound attack satisfies the outbound phase is higher, which implies that a complexity of 2^{44} is sufficient (instead of 2^{48}) for the shortcut player.

The same strategy can be applied to the 7 rounds distinguisher presented in [25] and recalled in Sect. 4.1.1. In particular, using the same argumentation of [19], the computational cost of the distinguisher illustrated in Fig. 4 is 2^{20} instead of 2^{24}. Indeed, note that for free $\mathcal{D}_i$ and $\mathcal{M}_j$, the probability that a solution of the inbound phase satisfies the outbound phase increases by a factor 16.

E.2 8-Round known-key distinguisher

A possible 8-round known-key distinguisher can be set up starting from the 7-round distinguisher just presented and extending it at the end (or at the beginning) using a similar technique presented in Sect. 5 for the 10-round distinguisher. We refer to that section for a complete discussion of this technique, and we limit ourselves here to give a formal definition of the distinguisher and to do some considerations about the data and the computational cost.

In the known-key distinguisher scenario, the two players have to send to the verifier n different tuples of (plaintext, ciphertext) pairs, that is $\{(p_i^1, c_i^1), (p_i^2, c_i^2)\}$ for $i = 0, ..., n-1$, such that

1. for each tuple, there exists j s.t. the two plaintexts belong to the same coset of $\mathcal{D}_j$, that is

$$\forall i \in \{0, ..., n-1\}, \quad \exists j \in \{0, ..., 3\} \quad \text{s.t.} \quad p_i^1 \oplus p_i^2 \in \mathcal{D}_j;$$

2. there exists a key k s.t. for each tuple there exists l for which the two ciphertexts belong to the same coset of $\mathcal{M}_l$ one round before, that is

$$\exists! \, k \quad \text{s.t.} \quad \forall i \in \{0, ..., n-1\}, \quad \exists l \in \{0, ..., 3\} \quad \text{s.t.} \quad R_k^{-1}(c_i^1) \oplus R_k^{-1}(c_i^2) \in \mathcal{M}_l.$$

Note that the only difference with the 10-round distinguisher presented in Sect. 5 regards the strategy exploited by the generic player in order to choose the plaintexts. In particular, instead of choosing plaintexts in sets $D_a = R^{-1}(\mathcal{D}_j \oplus a)$ as defined in (9), in this case the

generic player has to consider plaintexts in cosets of $\mathcal{D}_j$. Moreover, the verifier can direct check the property on the plaintexts (which is independent of the key).

If only the number of oracle-queries is considered, it is possible to prove that $n \geq 3$ tuples are sufficient to set up this distinguisher. Indeed, using the same argumentation of Sect. 5, the generic player has to consider approximately $2^{52.18}$ different couples (see (10)), that is approximately $2^{26.59}$ different (plaintexts, ciphertexts) pairs, for a cost of $2^{26.6}$ oracle-queries, in order to have good probability to construct 3 tuples with the required properties. On the other hand, the cost for the shortcut player is only of $3 \cdot 2^{20} = 2^{21.6}$ computations. In order to make the advantage of the shortcut player more significant, we choose an (arbitrary) value of $n = 8$, which implies a cost for the shortcut player of 2^{23} computations and of $2^{48.9}$ computations for the generic player.

For both cases, the verifier uses the same strategy presented in Sect. 5, and her cost is well approximated by $2^{11.6}$ eight-round encryptions.

References

1. Andreeva, E., Bogdanov, A., Mennink, B.: Towards Understanding the Known-Key Security of Block Ciphers. In: FSE 2013, volume **8424** of LNCS, pp. 348–366, (2013)
2. Aumasson, J.-P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi, 2009. In: Presented at the Rump Session of Cryptographic Hardware and Embedded Systems—CHES (2009)
3. Bellare, M., Micciancio, D.: A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. In: EUROCRYPT 1997, vol. 1233 of LNCS, pp. 163–192 (1997).
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Note on zero-sum distinguishers of Keccak-f. 2010. Unpublished, http://keccak.noekeon.org/NoteZeroSum.pdf.
5. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: CRYPTO 2009, volume **5677** of LNCS, pp. 231–249, (2009)
6. Boura C., Canteaut A.: A zero-sum property for the KECCAK-f permutation with 18 rounds. IEEE Int. Symp. Inf. Theory **2010**, 2488–2492 (2010).
7. Boura, C., Canteaut, A., De Cannière, C.: Higher-Order Differential Properties of Keccak and Luffa. In:' FSE 2011, volume 6733 of LNCS, pp. 252–269 (2011).
8. Blondeau C., Leander G., Nyberg K.: Differential-linear cryptanalysis revisited. J. Cryptol. **30**(3), 859–888 (2017).
9. Blondeau, C., Peyrin, T., Wang, L.: Known-Key Distinguisher on Full PRESENT. In: CRYPTO 2015, volume **9215** of LNCS, pp. 455–474, (2015)
10. Cui, T., Sun, L., Chen, H., Wang, M.: Statistical Integral Distinguisher with Multi-structure and Its Application on AES. In: ACISP 2017, volume **10342** of LNCS, pp. 402–420, (2017)
11. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In: FSE 1997, volume **1267** of LNCS, pp. 149–165, (1997)
12. Duan M., Lai X.J.: Improved zero-sum distinguisher for full round Keccak-f permutation. Chin. Sci. Bull. **57**(6), 694–697 (2012).
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES—The Advanced Encryption Standard. Information Security and CryptographySpringer, New York (2002).
14. Fouque, P.-A., Jean, J., Peyrin, T.: Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In: textitCRYPTO 2013, volume **8042** of LNCS, pp. 183–203, (2013)
15. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In: FSE 2000, volume **1978** of LNCS, pages 213–230, (2001)
16. Gilbert, H.: A Simplified Representation of AES. In: ASIACRYPT 2014, volume **8873** of LNCS, pp. 200–222, (2014)
17. Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In: FSE 2010, volume **6147** of LNCS, pp. 365–383, (2010)
18. Grassi L., Rechberger C., Rønjom S.: Subspace trail cryptanalysis and its applications to AES. IACR Trans. Symmetric Cryptol. **2016**(2), 192–225 (2017).
19. Jean, J., Naya-Plasencia, M., Peyrin, T.: Multiple limited-birthday distinguishers and applications. In: SAC 2013, volume **8282** of LNCS, pp. 533–550, (2014)

20. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In: ASIACRYPT 2007, volume **4833** of LNCS, pp. 315–324, (2007)
21. Knudsen, L., Wagner, D.: Integral Cryptanalysis. In: *FSE 2002*, volume 2365 of LNCS, pp. 112–127, (2002)
22. Lamberger M., Mendel F., Schläffer M., Rechberger C., Rijmen V.: The rebound attack and subspace distinguishers: application to whirlpool. J. Cryptol. **28**(2), 257–296 (2015).
23. Leander, G., Tezcan, C., Wiemer, F.: Searching for subspace trails and truncated differentials. IACR Trans. Symmetric Cryptol. **2018**(1), 74–100 (2018).
24. Lorenzo, G., Christian, R., Rønjom, S.: A New Structural-Differential Property of 5-Round AES. In: EUROCRYPT 2017, volume **10211** of LNCS, pp. 289–317, (2017)
25. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In: SAC 2009, volume 5867 of LNCS, pages 16–35, (2009)
26. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. I:n FSE 2009, volume 5665 of LNCS, pp. 260–276, (2009)
27. Mennink, B., Preneel, B.: On the impact of known-key attacks on hash functions. In: ASIACRYPT 2015, volume 9453 of LNCS, pp. 59–84 (2015)
28. Rønjom, S., Bardeh, N.G., Helleseth, T.: Yoyo Tricks with AES. In: ASIACRYPT 2017, volume **10624** of LNCS, pp. 217–243, (2017)
29. Sasaki, Y., Yasuda, K.: Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes. In: FSE 2011, volume **6733** of LNCS, pp. 397–415, (2011)
30. Tunstall M.: Improved "Partial Sums"-based Square Attack on AES. SECRYPT **2012**, 25–34 (2012).
31. Wagner, D.: A Generalized Birthday Problem. In: CRYPTO 2002, volume 2442 of LNCS, pages 288–303, (2002)
32. Meiqin, W., Tingting, C., Huaifeng, C., Ling, S., Long, W., Andrey, B.: Integrals Go Statistical: Cryptanalysis of Full Skipjack Variants. In: *FSE 2016*, volume 9783 of *LNCS*, pp. 399–415, 2016.
33. Wang, Q., Grassi, L., Rechberger, C.: Zero-Sum Partitions of PHOTON Permutations. In: Topics in Cryptology - CT-RSA 2018, volume 10808 of LNCS, pp. 279–299, (2018)
34. Wei, L., Peyrin, T., Sokołowski, P., Ling, S., Pieprzyk, J., Wang, H.: On the (In)Security of IDEA in Various Hashing Modes. In: FSE 2012, volume **7549** of LNCS, pp. 163–179, (2012)