The Cross-ratio Property and Its Use for Cryptanalysis of Round-reduced AES

Zhenzhen Bao[1,2,3], Jian Guo[4], Eik List[4], and Haoyang Wang[5]

[1] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
[2] Zhongguancun Laboratory, Beijing, China
[3] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, China
[4] Nanyang Technological University, Singapore, Singapore
[5] School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
zzbao@tsinghua.cn, {guojian, eiklist}@ntu.edu.sg, haoyang.wang@sjtu.edu.cn

Abstract. In this work, we propose three techniques for advancing cryptanalysis of round-reduced AES, two of which exploit the multiplicative inverse, and a third, structural, property that generalizes the S-box switch to multiple quartets.

Firstly, we formalize the cross-ratio property for tracing a nonlinear equation over $\mathbb{F}_{2^8}$ from the differences of four distinct inputs or their respective outputs through the key-wrapped multiplicative inverse. While the underlying properties of the multiplicative inverse have been well-studied, their usefulness for non-algebraic attacks has surprisingly remained unexamined. We demonstrate that it allows a more efficient matching between the sets of many related texts in a Demirci-Selçuk Meet-in-the-middle attack, leading to reductions in time and memory of both the online and offline phases for the seminal seven-round AES-128 by Derbez et al. from Eurocrypt 2013.

Secondly, we show that the cross-ratio property gives rise to a relevant special case: when its inputs to the multiplicative inverse span a two-dimensional space, one can almost always recover the input differences of a quartet from only their output differences, or, in an alternative formulation, even the key applied before the outputs. We show how this can lead to new reduced-data three-round distinguishers.

Thirdly, we define the mixture-quartet switch, an event that a mixture plaintext structure of 16 texts allows a partitioning into four quartets that all produce a related difference after two rounds. While most recent advances of the analysis of AES had focused on structural properties, we can trace partially active diagonals through a Super-S-box. Thus, by combining structural and algebraic properties, we describe a new distinguisher on four AES rounds with lower data complexity. Our applications do not threaten the security of the full AES, but advance the understanding of the building blocks of the AES further, and are likely applicable to similar settings and ciphers.

Keywords: Secret-key cryptography · block cipher · AES · differential cryptanalysis · meet-in-the-middle attack

1 Introduction

1.1 Cryptanalysis of The AES

The Advanced Encryption Standard (AES) [19,20] has been subject to enormous efforts of cryptanalysis, e.g. [24,25,27,30,31,49]. Its applications span a wide range of settings and models, as a primitive for encryption, authentication, authenticated encryption, hashing, or even as a one-way function in signature schemes. Yet, key-recovery attacks in the single secret key setting are its prime application. As the best attacks in the single secret-key setting still reach "only" seven out of 10 rounds for AES-128 [27], nine out of 12 rounds for AES-192 [29,46], and nine out of 14 rounds for AES-256 [29], there is high confidence in its classical security after more than 25 years of cryptanalysis.
However, after the research on finding improved attacks for the maximal number of rounds had peaked at the beginning of the 2010s [14,24,25,27,42], the research focus had shifted towards identifying new properties and improving practical attacks for fewer rounds. As a result, the past decade has brought many new insights and techniques to light. Grassi et al. proposed the subspace-trail notation [34] and later with Rechberger and Rønjom the multiple-of-n property [33]; Rønjom and Bardeh were at the forefront of applying yoyos and exchange attacks [5,6,48] to the AES; and Grassi introduced mixture differentials [32]. Those works paved the way for more efficient attacks on round-reduced versions, such as practical attacks on five rounds [4], improved six-round distinguishers [3,6,7,16], and six-round boomerangs [8,9]. Moreover, Leurent and Pernot presented new representations of the key schedule [40,41], showing that there are still interesting observations to be made. However, except for the latter on the key schedule, the new properties and improved attacks above all exploited only structural properties of the cipher. While much work was devoted to deepening the understanding of its S-box from an abstract view (e.g. [11,15,38]), those could rarely develop into new attacks. One exception is, e.g., the S-box linearization by Zhang et al. [50], which was later proven useful for improving attacks on the AES when used as a compression function in the context of block-cipher-based hashing [17]. It is natural that the extensive scrutiny that the AES has seen renders it challenging to identify new traits, such as the multiple-of-eight property that can be exploited in better attacks. However, given the widespread deployment of the AES, the relevance of finding new attack properties remains as high as or even more than it was 25 years ago.

1.2 Contribution

In this work, we propose three techniques for advancing the understanding of the AES further, two of which exploit the multiplicative inverse and one structural property that generalizes the S-box switch to multiple quartets.
Firstly, we formulate the *cross-ratio* property, which allows computation of a nonlinear equation over $\mathbb{F}_{2^8}$ independently from the differences of four inputs, as well as from the differences of their respective outputs from the multiplicative

inverse. While not unknown to earlier interpolation attacks and studies of t-wise independence, we demonstrate its usefulness for cryptanalytical attacks, such as for matching sets of many related texts in a Demirci-Selçuk Meet-in-the-middle (DS-MitM) key-recovery attack. Specifically, it allows us to slightly reduce the online time and the memory complexity in the seminal seven-round AES-128 from Eurocrypt 2013.

Secondly, we outline that the amount of information we obtain about the inputs to the multiplicative inverse increases in a special case when its inputs to the multiplicative inverse span a two-dimensional space. We show how to directly obtain the input differences or the final round keys of a quartet of texts from only their output differences, as well as a masking key added to the result after the layer. As a first intuitive application, we propose new three-round distinguishers with new time and data complexity records.

Moreover, we study mixture structures whose analysis has concentrated only on truncated differentials and, therefore, needed to wait for fully active diagonals to extend the differential shared by multiple pairs for two or more rounds. We study the probability of obtaining equal bytes across pairs after two AES rounds when plaintext pairs were constructed from a mixture structure. In combination with the cross-ratio property, we show a new lowest-data chosen-plaintext four-round distinguisher. Our results are compared to the literature in Tables 1–3.

2 Preliminaries

2.1 General Notations

We denote by $\mathbb{F}_2$ the finite field of prime characteristic 2, by $\mathbb{F}_{2^n}$ its extension and by $\mathbb{F}_{2^n}^* := \mathbb{F}_{2^n} \setminus \{0\}$ its elements without the zero element. We represent sets by calligraphic letters (such as $\mathcal{C}$). We employ typewriter font for hexadecimal values (such as `0xff`). Let $X, Y \in \mathbb{F}_2^n$ for some positive integer n in the following. We denote by $X \oplus Y$ their elementwise XOR, and by $\overline{X}$ the bitwise inverse of X. For a non-negative integer i, we write $[i..j]$ for $\{i, i+1, \ldots, j\}$, and similarly $x_{i..j}$ for a sequence of variables $(x_i, x_{i+1}, \ldots, x_j)$. $\langle i \rangle_m$ denotes the m-bit encoding of an integer i and $\mathsf{int}_m(x)$ the decoding of an m-bit string to the corresponding integer, e.g. $\langle 135 \rangle_8 = (10000111)$ and $\mathsf{int}_8((10000111)) = 135$. We omit m if it is clear from the context. We denote by $\mathbf{1}^m$ and $\mathbf{0}^m$ the vectors in $\mathbb{F}_2^m$ with their m elements being all ones and all zeroes, respectively. For a set $\mathcal{X}$ and random variables X_1, X_2, we indicate by $X_1, X_2 \leftarrow_{\$} \mathcal{X}$ that each X_i is sampled independently and uniformly at random from $\mathcal{X}$, i.e. with replacement; we write $X_1, X_2 \twoheadleftarrow_{\mathsf{wor}} \mathcal{X}$ that the X_i's are sampled uniformly at random from $\mathcal{X}$ without replacement. For a space $\mathcal{X}$, let $\mathcal{X}^\top$ denote its orthogonal space and for vectors $\mathbf{v}, \mathbf{w} \in \mathcal{X}^m$, let $\mathsf{span}(\mathbf{v}, \mathbf{w})$ denote their span and we write $(N)_k$ for the falling factorial $\prod_{i=0}^{k-1}(N-i)$.

2.2 The AES

We recall only those details of the AES that will be necessary in this work. The Advanced Encryption Standard [20,47] is a substitution-permutation network that transforms 16-byte plaintexts through 10, 12, or 14 rounds for key sizes of 128, 192, or 256 bits, respectively. Each state byte is interpreted as an element of the finite field $\mathbb{F}_{2^8}$. The state itself can be viewed either as a linear array of 16 elements in $\mathbb{F}_{2^8}^{16}$ or, also commonly, as a 4×4 matrix in $\mathbb{F}_{2^8}^{4\times 4}$. We use typewriter font, e.g. $\mathtt{2}$, also for constants in $\mathbb{F}_{2^8}$.

Almost each round consists of the operations SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey$[K^i]$ (AK$[K^i]$), i.e. a round-key addition with a round key K^i. Before the first round, an additional whitening key K^0 is XORed to the state; the final round omits the MixColumns operation. For compactness, we will sometimes write $\overline{\mathsf{SB}}$, $\overline{\mathsf{SR}}$, and $\overline{\mathsf{MC}}$ for SB^{-1}, SR^{-1}, and MC^{-1}, respectively. We write $X[i]$ for the i-th byte of an internal state X, and we interchangeably also use $X[(r,c)]$ to denote the byte in the r-th row and c-th column of state X with the usual byte ordering of

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} (0,0) & (0,1) & (0,2) & (0,3) \\ (1,0) & (1,1) & (1,2) & (1,3) \\ (2,0) & (2,1) & (2,2) & (2,3) \\ (3,0) & (3,1) & (3,2) & (3,3) \end{bmatrix}.$$

When using $X[(r,c)]$, we treat the indices as taken modulo four by default and use sets such as $X[(r,[i..j])]$ to denote $(X[(r,i)],\ldots,X[(r,j)])$ and similarly for sets of row indices. Note that we will use the notation $X[i_1,i_2]$ as a short form for $X[i_1]$, $X[i_2]$. We denote by X^i, Y^i, Z^i, and W^i the internal states in the i-th round directly after the application of AddRoundKey, SubBytes, ShiftRows, and MixColumns, respectively. We denote by $\mathsf{R}[K^i] \stackrel{\text{def}}{=} \mathsf{AK}[K^i] \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$ one AES round, and denote by $\widehat{\mathsf{R}} \stackrel{\text{def}}{=} \mathsf{AK}[K^i] \circ \mathsf{SR} \circ \mathsf{SB}$ the reduced final round. We denote by $\widehat{K}^i = \mathsf{MC}^{-1}(K^i)$ an equivalent key of K^i, and denote by $\widehat{W}^i$ the state after AddRoundKey if we exchange the order of MixColumns and AddRoundKey in the i-th round. We use $\mathbf{M}$ to denote the circular MixColumns matrix.

2.3 Subspace Notations

Grassi et al. [34] introduced column, diagonal, anti-diagonal, and mixed spaces that will alleviate some descriptions. Let $\{e_0, \ldots, e_{15}\}$ be unit vectors of $\mathbb{F}_{2^8}^{16}$, where e_j has a single 1 in the j-th byte. The column spaces $\mathcal{C}_i$ are defined as $\mathcal{C}_i = \mathsf{span}(e_{4i}, e_{4i+1}, e_{4i+2}, e_{4i+3})$, for $i \in [0..3]$. This generalizes naturally to higher dimensions. The diagonal spaces $\mathcal{D}_i$ and anti-diagonal spaces $\mathcal{ID}_i$ are defined as $\mathcal{D}_i = \mathsf{SR}^{-1}(\mathcal{C}_i)$ and $\mathcal{ID}_i = \mathsf{SR}(\mathcal{C}_i)$ and the mixed space $\mathcal{M}_i$ as $\mathcal{M}_i = \mathsf{MC}(\mathcal{ID}_i)$. For $\mathcal{I} \subseteq [0..3]$, $\mathcal{C}_\mathcal{I}$, $\mathcal{D}_\mathcal{I}$, $\mathcal{ID}_\mathcal{I}$, and $\mathcal{M}_\mathcal{I}$ are defined as

$$\mathcal{C}_\mathcal{I} \stackrel{\text{def}}{=} \bigoplus_{i\in\mathcal{I}} \mathcal{C}_i\,, \quad \mathcal{D}_\mathcal{I} \stackrel{\text{def}}{=} \bigoplus_{i\in\mathcal{I}} \mathcal{D}_i\,, \quad \mathcal{ID}_\mathcal{I} \stackrel{\text{def}}{=} \bigoplus_{i\in\mathcal{I}} \mathcal{ID}_i\,, \quad \mathcal{M}_\mathcal{I} \stackrel{\text{def}}{=} \bigoplus_{i\in\mathcal{I}} \mathcal{M}_i\,.$$

Given a state $X \in \mathbb{F}_{2^8}^{16}$, we define by $\mathcal{C}_{\mathcal{I}}(X) = X \cap \mathcal{C}_{\mathcal{I}}$ the columns in $\mathcal{I}$ of X, by $\mathcal{D}_{\mathcal{I}}(X)$ the diagonals in $\mathcal{I}$ of X, and by $\mathcal{ID}_{\mathcal{I}}(X)$ the anti-diagonals in $\mathcal{I}$ of X. We will often omit the set braces for brevity. For example, $\mathcal{C}_0(X) = X[0..3]$, $\mathcal{D}_0(X) = X[0, 5, 10, 15]$, or $\mathcal{ID}_0(X) = X[0, 7, 10, 13]$.

2.4 General S-box Properties

The patched inverse $\mathsf{Inv} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ for $n = 2^\kappa$ with positive integer κ, is the map $\mathsf{Inv}(x) =^{\text{def}} x^{2^n-2}$. The AES S-box $S : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ uses the patched inverse Inv over $\mathbb{F}_{2^8}$ as its core element and applies a linear layer over $(\mathbb{F}_2)^8$ to its output

$$S(x) \stackrel{\text{def}}{=} \mathbf{L} \cdot x^{254} \oplus \mathtt{0x63}$$

where the multiplication with $\mathbf{L} \in \mathbb{F}_2^{8\times 8}$ interprets the result of x^{254} as a vector in $\mathbb{F}_2^8$. We will also write L to denote the parallel application of $\mathbf{L}$ to all cells of an AES (partial) state. The following property is well-known and will be used in the further parts of this work.

Property 1 (Average #Solutions of Differentials Through the AES S-box). Let Δx, $\Delta y \in \mathbb{F}_{2^8}$. For the AES S-box $S : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$, there exists one value $x \in \mathbb{F}_{2^8}$ on average such that $S(x) \oplus S(x \oplus \Delta x) = \Delta y$. More precisely, for a fixed $\Delta x \neq 0$, there exist 126 differences Δy which have two solutions each, one difference Δy with four solutions, and 129 differences Δy with no solutions.

Thus, its differential uniformity is four, and all columns and rows in its DDT of nontrivial differentials contain exactly one entry of four. Moreover, all 4-entries in the DDT are formed by two distinct unordered input pairs with one pair having zero as one of its values (and their order-reversed pairs).

2.5 Mixtures and Mixed Pairs

In [32], Grassi introduced mixture cryptanalysis as a formalization of the conditional differentials observed in the multiple-of-eight studies [33]. In the following, assume that states $X \in \mathbb{F}_2^n$ can be partitioned into m b-bit elements, $X = (X[0], \ldots, X[m-1])$ for $m \cdot b = n$. In the context of AES-like ciphers, words can refer to bytes, rows, columns, diagonals, or inverse diagonals. We define $\mathsf{wt}(X) =^{\text{def}} |\{i : X[i] \neq 0\}|$ as the hamming-weight function and define $\mathsf{wt}_{\mathcal{B}}$, $\mathsf{wt}_{\mathcal{C}}$, and $\mathsf{wt}_{\mathcal{D}}$ as shorthand notations for weight functions that operate on bytes, columns, and diagonals, respectively.

Definition 1 (Exchange Function). Given $\mathbf{v} \in \mathbb{F}_2^m$ and a pair of states $X_0, X_1 \in \mathbb{F}_2^{m\cdot b}$, we define the exchange function $\rho : \mathbb{F}_2^m \times (\mathbb{F}_2^{m\cdot b})^2 \to \mathbb{F}_2^{m\cdot b}$ as

$$\rho^{\mathbf{v}}(X_0, X_1) \stackrel{\text{def}}{=} \left(X_{\mathbf{v}[0]}[0], X_{\mathbf{v}[1]}[1], \ldots, X_{\mathbf{v}[m-1]}[m-1]\right) .$$

We define $\rho_{\mathcal{B}}^{\mathbf{v}}$, $\rho_{\mathcal{C}}^{\mathbf{v}}$, and $\rho_{\mathcal{D}}^{\mathbf{v}}$ to operate on bytes, columns, and diagonals, respectively.

Definition 2 (Mixed Pair). For $\mathbf{v} \in \mathbb{F}_2^m$ and a pair of states $X_0, X_1 \in \mathbb{F}_2^{m \cdot b}$ with $X_0 \neq X_1$, we call each unordered pair $\left(\rho^{\mathbf{v}}(X_0, X_1), \rho^{\overline{\mathbf{v}}}(X_0, X_1)\right)$ a *mixed pair* of, or generated from, (X_0, X_1).

Note that $\rho^{\overline{\mathbf{v}}}(X_0, X_1) = \rho^{\mathbf{v}}(X_1, X_0)$ for all $\mathbf{v}$, X_0, and X_1.

Property 2 (Mixture-transition property of AES [32]). Let P_0 and P_1 be two plaintext states such that $\mathsf{wt}_{\mathcal{D}}(P_0 \oplus P_1) = 4$. For every $\mathbf{v} \in \mathbb{F}_2^4$, define the diagonally mixed plaintext $P_{\mathbf{v}} \stackrel{\text{def}}{=} \rho^{\mathbf{v}}_{\mathcal{D}}(P_0, P_1)$, Then, the eight complementary unordered pairs $(P_{\mathbf{v}}, P_{\mathbf{v} \oplus \mathbf{1}^4})$ all have the same difference after two rounds.

The mixture transition stems from the fact that two rounds of AES can be seen as the parallel application of four Super-S-boxes [21] whose results are combined only linearly by the MixColumns operation at the end of the second round.

3 The Cross-ratio Property of the Multiplicative Inverse

Let $n = 2^\kappa$ for a positive integer κ, and let $k_1, k_2 \in \mathbb{F}_{2^n}$ be independent secret keys. The basic setting of the key-wrapped multiplicative inverse is illustrated in Figure 1. Assume we are given four pairwise distinct values $x_0, x_1, x_2, x_3 \in \mathbb{F}_{2^n}$ that are mapped to four values $y_0, y_1, y_2, y_3 \in \mathbb{F}_{2^n}$ through a patched multiplicative inversion, wrapped by two key additions with k_1 before the inversion and with k_2 afterwards. We use the following notations:

$$x_i \xmapsto{\oplus k_1} u_i \xmapsto{\mathsf{Inv}(\cdot)} v_i \xmapsto{\oplus k_2} y_i\,,$$

that is, $u_i = x_i \oplus k_1$, $v_i = u_i^{-1}$, and $y_i = v_i \oplus k_2$ for $i \in [0..3]$, representing the results of the first key addition, the multiplicative inversion, and the second key addition, respectively. Furthermore, we define the input and output differences by $\alpha_i =^{\text{def}} x_0 \oplus x_i$ and $\beta_i =^{\text{def}} y_0 \oplus y_i$.

3.1 The Cross Ratio

With the notation and setting above, we can introduce the cross ratio as a convenient notation that will serve as a unifying invariant in our analysis.

Definition 3 (Cross Ratio). For four pairwise distinct elements $x_0, x_1, x_2, x_3 \in \mathbb{F}_{2^n}$, the cross ratio is defined as

$$\mathsf{CR}(x_0, x_1, x_2, x_3) \stackrel{\text{def}}{=} \frac{(x_1 \oplus x_3)(x_0 \oplus x_2)}{(x_1 \oplus x_2)(x_0 \oplus x_3)}\,. \tag{1}$$

Proposition 1. Using the notation from above, let $x_i \neq k_1$ and $y_i \neq k_2$ for all $i \in \{0, 1, 2, 3\}$. The cross ratio is invariant under key addition and inversion.

$$\mathsf{CR}(x_0, x_1, x_2, x_3) = \mathsf{CR}(u_0, u_1, u_2, u_3) = \mathsf{CR}(v_0, v_1, v_2, v_3) = \mathsf{CR}(y_0, y_1, y_2, y_3). \tag{2}$$

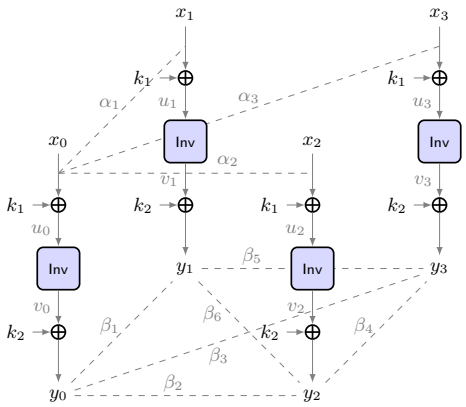


Fig. 1: The setting of the key-wrapped multiplicative inverse.

Proof. Since $u_i \oplus u_j = (x_i \oplus k_1) \oplus (x_j \oplus k_1) = x_i \oplus x_j$, we have

$$\mathsf{CR}(u_0, u_1, u_2, u_3) = \mathsf{CR}(x_0, x_1, x_2, x_3)\,, \tag{3}$$

and $\mathsf{CR}(y_0, y_1, y_2, y_3) = \mathsf{CR}(v_0, v_1, v_2, v_3)$ follows analogously. Moreover, for $i \neq j$, it holds that

$$v_i \oplus v_j = \frac{1}{u_i} \oplus \frac{1}{u_j} = \frac{u_i \oplus u_j}{u_i u_j}\,.$$

Thus

$$\mathsf{CR}(v_0, v_1, v_2, v_3) = \frac{\frac{u_1 \oplus u_3}{u_1 u_3} \cdot \frac{u_0 \oplus u_2}{u_0 u_2}}{\frac{u_1 \oplus u_2}{u_1 u_2} \cdot \frac{u_0 \oplus u_3}{u_0 u_3}} = \frac{(u_1 \oplus u_3)(u_0 \oplus u_2)}{(u_1 \oplus u_2)(u_0 \oplus u_3)} = \mathsf{CR}(u_0, u_1, u_2, u_3)\,. \tag{4}$$

The proof follows from the combination of Equations (3) and (4). □

Given an input set $\{x_0, x_1, x_2, x_3\}$ and a corresponding output set $\{y_0, y_1, y_2, y_3\}$, Proposition 1 yields an efficient distinguisher for the multiplicative inverse, computable in $O(1)$ operations that we call the *cross-ratio distinguisher*, and which can also be represented using only differences:

$$\mathsf{CR}(x_0, x_1, x_2, x_3) = \frac{(\alpha_1 \oplus \alpha_3)\alpha_2}{(\alpha_1 \oplus \alpha_2)\alpha_3} = \frac{(\beta_1 \oplus \beta_3)\beta_2}{(\beta_1 \oplus \beta_2)\beta_3} = \mathsf{CR}(y_0, y_1, y_2, y_3)\,. \tag{5}$$

Remark 1. The cross ratio remains invariant for any Möbius transformation

$$M(x) \stackrel{\text{def}}{=} \frac{ax + b}{cx + d}\,, \tag{6}$$

where inversion is an instance with $(a, b, c, d) = (0, 1, 1, 0)$ for all $x \in \mathbb{F}_{2^m} \setminus \{0\}$.

Proposition 2 (Generalized Cross Ratio). Let $\sigma_\kappa : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ be the κ-th Frobenius automorphism $\sigma_\kappa(x) =^{\text{def}} x^{2^\kappa}$. Then, for any Möbius transformation $M(x)$ as in Equation (6), and any $k \in [0..m-1]$, the semilinear map

$$f(x) = \sigma_\kappa\left(M(x)\right) = \left(\frac{ax+b}{cx+d}\right)^{2^\kappa} \tag{7}$$

satisfies

$$\mathsf{CR}\left(f(x_0), f(x_1), f(x_2), f(x_3)\right) = \sigma_\kappa\left(\mathsf{CR}(x_0, x_1, x_2, x_3)\right) = \left(\mathsf{CR}(x_0, x_1, x_2, x_3)\right)^{2^\kappa}$$

for all pairwise distinct x_0, x_1, x_2, x_3 with $cx_i + d \neq 0$. Thus, such maps preserve the cross ratio *up to* the field automorphism $z \mapsto z^{2^\kappa}$.

Example 1. One of the S-boxes of the Korean block-cipher standard ARIA [39] uses the monomial map $x \mapsto x^{247}$ as its internal nonlinear component. That is, it produces the map $x \mapsto x^{-2^3}$ for all nonzero $x \in \mathbb{F}_{2^8}^*$ and the projection $0 \mapsto 0$, which corresponds to an instance of Equation (7) with parameters $(a, b, c, d, \kappa) = (0, 1, 1, 0, 3)$. Thus, it holds for that S-box that

$$\left(\frac{(\alpha_1 \oplus \alpha_3)\alpha_2}{(\alpha_1 \oplus \alpha_2)\alpha_3}\right)^{2^3} = \frac{(\beta_1 \oplus \beta_3)\beta_2}{(\beta_1 \oplus \beta_2)\beta_3}\,. \tag{8}$$

3.2 Recovering A Two-dimensional Affine Subspace

Next, we consider the special case where the four inputs form a two-dimensional affine subspace spanned by two distinct nonzero elements $\alpha_1, \alpha_2 \in \mathbb{F}_{2^n}^*$:

$$(x_0, x_1, x_2, x_3) = x_0 \oplus \langle \alpha_1, \alpha_2 \rangle = (x_0, x_0 \oplus \alpha_1, x_0 \oplus \alpha_2, x_0 \oplus \alpha_1 \oplus \alpha_2)\,.$$

3.2.1 Recovering Input Differences from the Outputs

Using the notation from above (cf. Figure 1), a direct computation gives

$$\left.\begin{aligned} x_0 \oplus x_1 = x_2 \oplus x_3 = \alpha_1, \\ x_0 \oplus x_2 = x_1 \oplus x_3 = \alpha_2 \end{aligned}\right\} \quad \text{and thus} \quad x_1 \oplus x_2 = x_0 \oplus x_3 = \alpha_1 \oplus \alpha_2\,.$$

Since $x_i \oplus x_j = u_i \oplus u_j$, we can easily obtain from Equation (4):

$$\mathsf{CR}(x_0, x_1, x_2, x_3) = \frac{\alpha_2^2}{(\alpha_1 \oplus \alpha_2)^2}.$$

Let $s =^{\text{def}} \alpha_1 \oplus \alpha_2$ and $r =^{\text{def}} \mathsf{CR}(x_0, x_1, x_2, x_3)$. We have $\sqrt{r} = \alpha_2/s$. Hence

$$\begin{aligned} \alpha_2 &= \sqrt{r} \cdot s \\ \alpha_1 &= s \oplus \alpha_2 = (1 \oplus \sqrt{r})\, s\,. \end{aligned} \tag{9}$$

As shorthand notation, we introduce additional variables $\beta_4 =^{\text{def}} y_2 \oplus y_3$ and $\beta_5 =^{\text{def}} y_1 \oplus y_3$, and define $z_4 =^{\text{def}} \beta_4^{-1}$ and $z_5 =^{\text{def}} \beta_5^{-1}$. Moreover, we can easily derive for all $i \in [0..3]$:

$$\beta_i^{-1} = \left(\frac{1}{u_0} \oplus k_2 \oplus \frac{1}{u_0 \oplus \alpha_i} \oplus k_2\right)^{-1} = \frac{u_0(u_0 \oplus \alpha_i)}{\alpha_i} = \frac{u^2}{\alpha_i} \oplus u_0 \,. \tag{10}$$

We can then derive

$$\beta_4^{-1} = \frac{u_0^2 \oplus \alpha_2^2}{\alpha_1} \oplus u_0 \oplus \alpha_2 \quad \text{and} \quad \beta_5^{-1} = \frac{u_0^2 \oplus \alpha_1^2}{\alpha_2} \oplus u_0 \oplus \alpha_1 \tag{11}$$

as the inputs lie in a two-dimensional affine subspace.

Lemma 1. Let $(u_0, u_0 \oplus \alpha_1, u_0 \oplus \alpha_2, u_0 \oplus \alpha_1 \oplus \alpha_2)$ be elementwise distinct and zero-free. Then, it holds that

$$\beta_1^{-1} \oplus \beta_4^{-1} = \frac{\alpha_2^2}{\alpha_1} \oplus \alpha_2 \quad \text{and} \quad \beta_2^{-1} \oplus \beta_5^{-1} = \frac{\alpha_1^2}{\alpha_2} \oplus \alpha_1 \,.$$

Proof. The lemma follows from combining Equations (10) and (11). □

Multiplying the two identities in (11) gives

$$(\beta_1^{-1} \oplus \beta_4^{-1}) \cdot (\beta_2^{-1} \oplus \beta_5^{-1}) = \frac{\alpha_2}{\alpha_1}(\alpha_1 \oplus \alpha_2) \cdot \frac{\alpha_1}{\alpha_2}(\alpha_1 \oplus \alpha_2) = (\alpha_1 \oplus \alpha_2)^2 = s^2.$$

Thus

$$s = \sqrt{(\beta_1^{-1} \oplus \beta_4^{-1}) \cdot (\beta_2^{-1} \oplus \beta_5^{-1})}. \tag{12}$$

On the other hand, by Proposition 2, we may compute the cross ratio from the outputs

$$r \stackrel{\text{def}}{=} \mathsf{CR}(y_0, y_1, y_2, y_3) = \mathsf{CR}(x_0, x_1, x_2, x_3) \,. \tag{13}$$

Combining Equations (9), (12), and (13) yields values for the input differences. As a result, we obtain Proposition 3.

Proposition 3 (Input-difference Solver). If the four inputs $x_i \in \mathbb{F}_{2^n}$ are zero-free and span an affine two-dimensional subspace, that is, $(x_0, x_1, x_2, x_3) = (x_0, x_0 \oplus \alpha_1, x_0 \oplus \alpha_2, x_0 \oplus \alpha_1 \oplus \alpha_2)$. Then, the input differences α_1, α_2 can be determined from the outputs $y_i = (x_i \oplus k_1)^{-1} \oplus k_2$:

$$\alpha_1 = \left(1 \oplus \sqrt{\mathsf{CR}(y_0, y_1; y_2, y_3)}\right) \sqrt{(\beta_1^{-1} \oplus \beta_4^{-1}) \cdot (\beta_2^{-1} \oplus \beta_5^{-1})} \,, \tag{14}$$

$$\alpha_2 = \sqrt{\mathsf{CR}(y_0, y_1; y_2, y_3)} \sqrt{(\beta_1^{-1} \oplus \beta_4^{-1}) \cdot (\beta_2^{-1} \oplus \beta_5^{-1})} \,. \tag{15}$$

3.2.2 Recovering Secret Keys from the Outputs

We can also use the cross-ratio property to recover the key that masked the outputs when the inputs are distinct and in a two-dimensional subspace.

Definition 4 (Elementary Symmetric Polynomials). Let $u_0, \ldots, u_{r-1} \in \mathbb{F}_{2^n}$. For $\ell = 0, 1, \ldots, r$, the ℓ-th elementary symmetric polynomial is defined as

$$e_\ell(u_0, \ldots, u_{r-1}) \stackrel{\text{def}}{=} \bigoplus_{0 \le i_1 < \cdots < i_\ell \le r-1} u_{i_1} \cdots u_{i_\ell} \,,$$

with the convention $e_0(u_0, \ldots, u_{r-1}) =^{\text{def}} 1$.

Example 2. For example, the expanded forms for $\ell = 1$ and $\ell = 3$ for four variables are

$$e_1(u_0, u_1, u_2, u_3) \stackrel{\text{def}}{=} u_0 \oplus u_1 \oplus u_2 \oplus u_3 \quad \text{and}$$

$$e_3(u_0, u_1, u_2, u_3) \stackrel{\text{def}}{=} u_0u_1u_2 \oplus u_0u_1u_3 \oplus u_0u_2u_3 \oplus u_1u_2u_3 \,, \quad \text{respectively}\,.$$

Proposition 4 (Key Solver). Let $n = 2^\kappa$ for some positive integer κ. Suppose the pairwise distinct quadruple $(u_0, u_1, u_2, u_3) \in (\mathbb{F}_{2^n}^*)^4$ forms a zero-free two-dimensional affine subspace. For some arbitrary $k \in \mathbb{F}_{2^n}$, define $y_i = u_i^{-1} \oplus k$ for $i \in [0..3]$. Then, whenever $e_1(y_0, y_1, y_2, y_3) \neq 0$, one can recover the masking key k in closed form from the values y_i:

$$k = \left(\frac{e_3(y_0, y_1, y_2, y_3)}{e_1(y_0, y_1, y_2, y_3)} \right)^{2^{n-1}} \,,$$

where $x \mapsto x^{2^{n-1}}$ denotes the unique square-root map in $\mathbb{F}_{2^n}$ (the inverse of the Frobenius squaring map).

Proof. Using $v_i = y_i \oplus k$, the two-dimensional input condition yields

$$\bigoplus_{i=0}^{3} u_i = \bigoplus_{i=0}^{3} \frac{1}{v_i} = \bigoplus_{i=0}^{3} (y_i \oplus k)^{-1} = \left(\frac{F_k(y_0, y_1, y_2, y_3)}{(y_0 \oplus k)(y_1 \oplus k)(y_2 \oplus k)(y_3 \oplus k)} \right) = 0 \,,$$

where the numerator can be expanded to

$$F_k(y_0, y_1, y_2, y_3) = e_3(y_0, y_1, y_2, y_3) \oplus e_1(y_0, y_1, y_2, y_3) \cdot k^2 \tag{16}$$

over fields of characteristic two as the factors for other powers of k vanish. When Equation (16) is zero, we can rewrite it as

$$k^2 = \frac{e_3(y_0, y_1, y_2, y_3)}{e_1(y_0, y_1, y_2, y_3)} \,,$$

and taking the unique square root produces our claim. □

Equipped with those techniques, we show three applications in the remainder of this work. First, we combine them with the deterministic decomposition of two-round AES into four parallel Super-S-boxes to obtain three-round distinguishers with the lowest data and time complexities to date. Second, we illustrate how to extend them for the lowest-data distinguishers on four rounds. Third, we propose an improved DS-MitM attack on seven-round AES-128.

Table 1: Best low-data chosen-plaintext distinguishers on three-round AES. Time given in #encryption equivalents; data in #chosen plaintexts; memory requirements are negligible for each distinguisher. p_s = success probability. $^{(1)}$ = A three-round variant of [10] testing for a collision in any byte between text pairs.

Attack	Time	Data	p_s
Integral [18,37]	256	256	$\simeq 1$
Truncated differential [34]	20	20	0.95
Impossible differential$^{(1)}$ [10]	10	10	0.94
Mixture differential [35]	10	10	0.95
Mixture differential [Suppl. A]	**6**	**6**	**0.96**
Mixture differential [Sect. 4.2]	**4**	**4**	**0.96**

4 Three-round Distinguishers

In this section, we show how to use the input-difference solver from Section 3 for a chosen-plaintext distinguisher on three-round AES with only four chosen plaintexts. Another distinguisher based on the key solver with six chosen plaintexts is given in the Supplementary Material A. Our results are summarized and compared to previous ones in Table 1.

4.1 The Baseline: Impossible-differential Distinguisher

We can reduce Biham's well-known impossible-differential distinguisher on four-round AES [10] to three rounds. Suppose N plaintexts are pairwise distinct in exactly one byte and have the same randomly chosen values in all other bytes. Then, the texts will be pairwise distinct from each other in each byte after 2.5 rounds, before the final MixColumns operation in Round 3. In contrast, a random collision in any Byte j between any pair of states occurs with probability ca. 2^{-8} for outputs from a random permutation. Thus, the adversary can invert the final MixColumns operation and test for byte collisions. For a random permutation over $\mathbb{F}_{2^8}^{16}$, the probability that no collision occurs is approximately

$$p_{\text{rand}}(N) \approx \left(\frac{\left(2^8\right)_N}{\left(2^8\right)^N} \right)^{16},$$

which yields $p_{\text{rand}}(10) \approx 0.057$. In contrast, there is never a byte collision for the real 2.5-round AES. The only previous result coming close to it is the so-called mixture-integral distinguisher by Grassi and Schofnegger [35] on three-round AES, which suggests the same key for multiple mixed pairs, but does not exploit the internal structure of the AES S-box. New results should reduce this baseline.

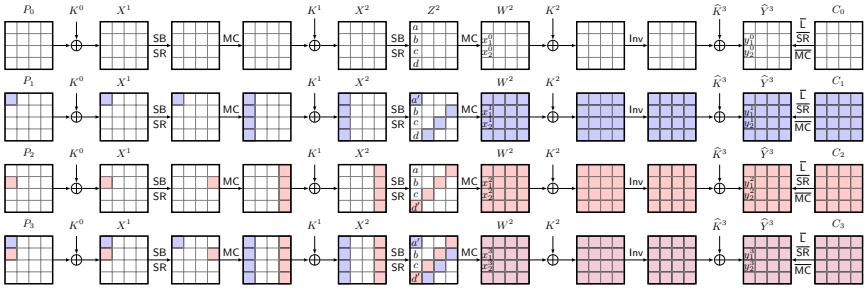


Fig. 2: The three-round distinguisher using four chosen plaintexts. Colored bytes indicate active differences.

4.2 A Distinguisher Using Four CPs and Proposition 3

Distinguisher Idea. We can reduce the data complexity further for distinguishing three-round AES to four chosen plaintexts. Let (P_0, P_1) differ in exactly one byte at some index j, e.g., $j = 0$: $P_0[0] \neq P_1[0]$ in the following. Let (P_2, P_3) be a second plaintext pair that differs from (P_0, P_1) only in one byte index $j' \neq j$ and $P_2[j'] = P_3[j']$. We choose $j' = 1$ as an example. Due to Property 2, both pairs will have the same difference $W^2_{2i} \oplus W^2_{2i+1} = X^3_{2i} \oplus X^3_{2i+1}$ for $i \in \{0, 1\}$ after two rounds.
When evaluating Proposition 3 for two bytes in the same inverse output diagonal of Z^3 (before the final MixColumns operation), one obtains four equations for two variables. The adversary inverts the final MixColumns operation of all texts and applies Proposition 3 for two bytes of its choice, e.g., $\widehat{Y}^3[1, 2]$. We define

- $a =^{\text{def}} Z^2_0[0] = Z^2_2[0] \neq a' =^{\text{def}} Z^2_1[0] = Z^2_3[0]$,
- $b =^{\text{def}} Z^2_i[1]$ and $c =^{\text{def}} Z^2_i[2]$ are constant for all $i \in [0..3]$, and
- $d =^{\text{def}} Z^2_0[3] = Z^2_1[3] \neq d' =^{\text{def}} Z^2_2[3] = Z^2_3[3]$.

For $\widehat{Y}^3[1]$, the subspace solver returns (α_1, α_2) such that $W^2_0[1] = W^2_1[1] \oplus \alpha_1 = W^2_2[1] \oplus \alpha_2 = W^2_3[1] \oplus \alpha_1 \oplus \alpha_2$ for

$$\begin{aligned}\alpha_1 &= (a \oplus 2b \oplus 3c \oplus d) \oplus (a' \oplus 2b \oplus 3c \oplus d) = a \oplus a' \,,\\ \alpha_2 &= (a \oplus 2b \oplus 3c \oplus d) \oplus (a \oplus 2b \oplus 3c \oplus d') = d \oplus d' \,,\end{aligned}$$

and for $\widehat{Y}^3[2]$, it returns (α'_1, α'_2) such that $W^2_0[2] = W^2_1[2] \oplus \alpha'_1 = W^2_2[2] \oplus \alpha'_2 = W^2_3[2] \oplus \alpha'_1 \oplus \alpha'_2$ for

$$\begin{aligned}\alpha'_1 &= (a \oplus b \oplus 2c \oplus 3d) \oplus (a' \oplus b \oplus 2c \oplus 3d) = a \oplus a' \,,\\ \alpha'_2 &= (a \oplus b \oplus 2c \oplus 3d) \oplus (a \oplus b \oplus 2c \oplus 3d') = 3(d \oplus d') \,.\end{aligned}$$

The notation is illustrated in Figure 2. Thus, the adversary can test, e.g., whether $\alpha_1 = \alpha'_1$, which holds with probability close to 0.97 for the real AES and 2^{-8} for a random permutation. A formal description is given in Algorithm 1.

Algorithm 1 Three-round distinguisher using Proposition 3 and four CPs.

function solveForDifference(y_0, y_1, y_2, y_3)
  $\beta_1 \leftarrow y_0 \oplus y_1$; $\beta_4 \leftarrow y_2 \oplus y_3$; $\beta_2 \leftarrow y_0 \oplus y_2$; $\beta_5 \leftarrow y_1 \oplus y_3$
  $s \leftarrow \sqrt{(\beta_1^{-1} \oplus \beta_4^{-1}) \cdot (\beta_2^{-1} \oplus \beta_5^{-1})}$
  $r \leftarrow \mathsf{CR}(y_0, y_1, y_2, y_3)$
  return $(1 \oplus r) \cdot s$

function A()
  Choose (P_0, P_1) that differ only in Byte 0
  Create a mixed pair (P_2, P_3) that differs from (P_0, P_1) only in Byte 1 and $P_2[1] = P_3[1]$
  Ask the challenger for their corresponding ciphertexts $C_0, \ldots, C_3$
  $\widehat{Y}_i^3 \leftarrow \overline{\mathsf{L}}(\overline{\mathsf{SR}}(\overline{\mathsf{MC}}(C_i)))$ for all $i \in [0..3]$
  for $c \leftarrow 0..3$ **do**
    $y_i \leftarrow \widehat{Y}_i^3[(1,c)]$, $y_i' \leftarrow \widehat{Y}_i^3[(2,c)]$ for $i \in [0..3]$
    if $|\{y_0, \ldots, y_3\}| < 4$ **or** $|\{y_0', \ldots, y_3'\}| < 4$ **then continue** ▷ Not pairwise distinct
    $\alpha_1 \leftarrow$ solveForDifference(y_0, y_1, y_2, y_3)
    $\alpha_1' \leftarrow$ solveForDifference(y_0', y_1', y_2', y_3')
    if $\alpha_1 = \alpha_1'$ **then return** 1 ▷ Real round-reduced AES
    else return 0 ▷ Random permutation
  return 0 ▷ Random permutation by default

Complexities and Probabilities. The distinguisher needs four chosen plaintexts, negligible memory, four three-round encryptions, four single-column decryptions, and four fast lookups to a table that fits into fast cache. It has a negligible computational complexity. In 100 000 experiments each with the real three-round AES, the distinguisher returned true with probability 0.962, both when testing for the first column of $\widehat{Y}^3$ only and when using any column where all texts y_i and y_i' were pairwise distinct. In contrast, in 100 000 more experiments, each with a truncated ideal permutation, the distinguisher returned true with probability 0.004 both when testing for the first column only, and also with probability 0.004 when using any column where all texts y_i and y_i' were pairwise distinct.

5 The Mixture-quartet Switch

In this section, we introduce the *mixture-quartet switch*, which describes that a local byte collision inside a mixture structure can force several quartets to have a byte collision that can be traced through the subsequent S-box layer. Those quartets provide the structure used later by the four-round distinguisher.

5.1 Mixture Structure and Quartet Partition

Definition 5 (Mixture Structure). Let (X_0, X_1) be a pair of distinct states of m elements. For a fixed $\mathbf{v} \in \mathbb{F}_2^m$, define the mixed state $X_{\mathbf{v}} \stackrel{\text{def}}{=} \rho^{\mathbf{v}}(X_0, X_1)$. A *mixture structure* $\mathcal{S}(X_0, X_1)$ is the set of all mixed unordered pairs

$$\mathcal{S}(X_0, X_1) \stackrel{\text{def}}{=} \{(X_{\mathbf{v}}, X_{\overline{\mathbf{v}}}) : \mathbf{v} \in \mathbb{F}_2^m\} .$$

This extends naturally to byte-mixture or diagonal-mixture structures, which operate on $m = 16$ bytes and $m = 4$ diagonals, respectively. In this work, a mixture structure usually refers to a diagonal-mixture structure.
When $d = \mathsf{wt}_{\mathcal{B}}(X_0 \oplus X_1)$ diagonals differ between X_0 and X_1, the structure contains $|\mathcal{S}(X_0, X_1)| = 2^{d-1}$ distinct unordered mixed pairs. For any $\mathbf{v} \in \mathbb{F}_2^4$, we identify $X_{\mathbf{v}}$ with $X_{\mathsf{int}_4(\mathbf{v})}$.

Definition 6 (Quartet Partition). For a fixed translation vector $\mathbf{h} \in \mathbb{F}_2^m \setminus \{\mathbf{0}^m, \mathbf{1}^m\}$, $\mathbb{F}_2^m$ can be partitioned into 2^{m-2} quartets $\mathcal{P}^{\mathbf{h}} \stackrel{\text{def}}{=} \{((\mathbf{v}, \overline{\mathbf{v}}), (\mathbf{v} \oplus \mathbf{h}, \overline{\mathbf{v}} \oplus \mathbf{h})) : \mathbf{v} \in \mathbb{F}_2^m\}$. We call $\mathcal{P}^{\mathbf{h}}$ a *quartet partition* of $\mathbb{F}_2^m$ induced by $\mathbf{h}$.

Due to the decomposability of two-round AES into four Super-S-boxes, one is particularly interested in mixture structures. Hence, we focus on diagonals in the following.

Definition 7 (Quartet Partition of a Mixture Structure). Assume X_0 and X_1 differ in all four diagonals: $\mathsf{wt}_{\mathcal{D}}(X_0 \oplus X_1) = 4$. Then, $\mathcal{S}_{\mathcal{D}}(X_0, X_1)$ contains exactly $2^{4-1} = 8$ mixed pairs that can be partitioned into a set of four quartets of two distinct mixed pairs each by a translation vector $\mathbf{h} \in \mathbb{F}_2^4 \setminus \{\mathbf{0}^4, \mathbf{1}^4\}$

$$\mathcal{Q}_{\mathcal{D}}^{\mathbf{h}}(X_0, X_1) \stackrel{\text{def}}{=} \left\{\{(X_{\mathbf{v}}, X_{\overline{\mathbf{v}}}), (X_{\mathbf{v}\oplus\mathbf{h}}, X_{\overline{\mathbf{v}}\oplus\mathbf{h}})\} : \mathbf{v} \in \mathbb{F}_2^4\right\} .$$

We call $\mathcal{Q}_{\mathcal{D}}^{\mathbf{h}}(X_0, X_1)$ a *quartet partition* induced by $\mathbf{h}$, where each quartet is defined by the exchange vectors in the coset $\mathbf{v} \oplus \{\mathbf{0}^4, \mathbf{1}^4, \mathbf{h}, \overline{\mathbf{h}}\}$.

Property 3. There exist $2^4 - 2$ exchange vectors $\mathbf{h} \notin \{\mathbf{0}^4, \mathbf{1}^4\}$. Since $\mathbf{h}$ and $\overline{\mathbf{h}}$ produce the same partition, there exist exactly seven quartet partitions for a mixture structure when X_0 and X_1 differ in all four diagonals.

Definition 8 (Good quartet). Let $Q = \{(X_{\mathbf{v}}, X_{\overline{\mathbf{v}}}), (X_{\mathbf{v}\oplus\mathbf{h}}, X_{\overline{\mathbf{v}}\oplus\mathbf{h}})\}$ be a quartet of two distinct (unordered) mixed pairs. We call Q a *good quartet* w.r.t. some byte index j if and only if $X_{\mathbf{v}}[j] \oplus X_{\overline{\mathbf{v}}}[j] = X_{\mathbf{v}\oplus\mathbf{h}}[j] \oplus X_{\overline{\mathbf{v}}\oplus\mathbf{h}}[j]$.

We are particularly interested in good quartets that have the same pairwise input difference before and after an S-box. With the AES S-box, the case having collisions among quartets has the highest probability of obtaining this condition.

5.2 Mixture-quartet Switch

Let $\mathcal{S}_{\mathcal{D}}(P_0, P_{15})$ be a mixture structure and let $X_i^3 \leftarrow \mathsf{R}^2(P_i)$ be the corresponding states after two AES rounds for $i \in [0..15]$. Fix one byte index j in $X_i^3[j]$ for all i. Since two rounds of AES can be seen as the parallel application of four Super-S-boxes, the value in any byte $X_{\mathbf{v}}^3[j]$ after two rounds can be written as

$$X_{\mathbf{v}}^3[j] = \psi_0(\mathcal{D}_0(P_{\mathbf{v}[0]})) \oplus \psi_1(\mathcal{D}_1(P_{\mathbf{v}[1]})) \oplus \psi_2(\mathcal{D}_2(P_{\mathbf{v}[2]})) \oplus \psi_3(\mathcal{D}_3(P_{\mathbf{v}[3]})) \oplus C$$

for four key-dependent functions $\psi_i : \mathbb{F}_{2^8}^4 \to \mathbb{F}_{2^8}$ and a key-dependent constant C. Define $\eta_i \stackrel{\text{def}}{=} (\psi_i(\mathcal{D}_i(P_0)) \oplus \psi_i(\mathcal{D}_i(P_{15})))$. It holds for any $\mathbf{v}$ that

$$X^3_{\mathbf{v}}[j] \oplus X^3_{\mathbf{v}\oplus\mathbf{h}}[j] = \bigoplus_{i\,:\,\mathbf{h}_i\neq 0} \eta_i \tag{17}$$

Consequently, whenever a nonzero $\mathbf{h}$ satisfies $\bigoplus_{i:h_i=1} \eta_i = 0$, we have $X^3_{\mathbf{v}}[j] = X^3_{\mathbf{v}\oplus\mathbf{h}}[j]$ for every representative $\mathbf{v}$. Hence, the same endpoint-wise equalities also hold after the following S-box layer: $S(X^3_{\mathbf{v}}[j]) = S(X^3_{\mathbf{v}\oplus\mathbf{h}}[j])$. Therefore, if $\mathbf{h} \notin \mathsf{span}(\mathbf{1}^4)$, the quartet partition $\mathcal{Q}^{\mathbf{h}}_{\mathcal{D}}(X^3_0, X^3_1)$ consists of four good quartets w.r.t. Byte j. We call this event the *mixture-quartet switch*.
We note that a mixture-quartet switch induced by $\mathbf{h} = \mathbf{1}^4$ is special as it implies that every original mixed pair has a zero difference in Byte j. Thus, any combination of two of the eight mixed pairs forms a good quartet, i.e., $\binom{8}{2} = 28$ good quartets in total. In contrast, directions $\mathbf{h} \neq \mathbf{1}^4$ produce the usual mixture-quartet switch of (at least) four good quartets.
From the nature of the AES round, the implications extend through the subsequent S-box layer. A zero difference $X^3_{\mathbf{v}}[j] = X^3_{\mathbf{v}\oplus\mathbf{h}}[j]$ for any $\mathbf{h} \neq \mathbf{0}^4$ implies a zero difference for eight such pairs in the mixture structure. It follows then that $S(X^3_{\mathbf{v}}[j]) = S(X^3_{\mathbf{v}\oplus\mathbf{h}}[j])$. Thus, a good quartet w.r.t. Byte j before SubBytes is also a good quartet after SubBytes.

Remark 2. The fact that a zero-byte difference in a mixed pair after two rounds yields a zero-byte difference in all eight pairs from a mixture structure is well-known as the event under the multiple-of-eight property [33]. That and subsequent works [4,32] demanded a zero difference to occur in a diagonal of one pair in order to extend it by two further rounds. In contrast, by considering all quartets from a mixture structure, we can combine events for different ways of combining collisions in the bytes of a diagonal after two rounds, and extend it with the cross-ratio property for a four-round distinguisher.

Definition 9 (Good-column Quartet Partition). Fix a quartet partition $\mathcal{Q}^{\mathbf{h}}_{\mathcal{D}}(X_0, X_1)$. Assume there exists some $c \in [0..3]$ such that the quartet partition produces a mixture-quartet switch, i.e., are four good quartets w.r.t. all $j \in [4c..4c+3]$. Then, we call the quartet partition a *good-column quartet partition*.

This event is used in the later four-round distinguisher. The same quartet partition must be compatible with the switch in all four byte positions of a column.

5.3 Switch Probabilities

Mixture-quartet Switch. For a valid active column, changing exactly one swap bit changes one Super-S-box contribution, so $\eta_i \neq 0$ for all i. Thus, no single-weight direction $\mathbf{h}$ can produce a zero-byte difference in a pair after two AES rounds. Among the 15 directions $\mathbf{h} \neq \mathbf{0}^4$, the admissible directions $\mathbf{h}$ have Hamming weights 2, 3, or 4, giving

$$\binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 11$$

possible directions. Under usual randomization assumptions, each admissible direction is zero with probability about 2^{-8}. Thus, the approximate probability that there exists an admissible $\mathbf{h}$ that induces a mixture-quartet switch is

$$\frac{11}{256} \approx 2^{-4.54}\,.$$

This includes the case $\mathbf{h} = \mathbf{1}^4$ that corresponds to the classical mixture-differential case. Later, we are interested in orthogonal directions $\mathbf{h}$, i.e., $\mathbf{h} \notin \{\mathbf{0}^4, \mathbf{1}^4\}$. A quartet partition is represented by the pair $\{\mathbf{v}, \overline{\mathbf{v}}\}$. Four such pairs have Hamming weights $(1, 3)$. Thus, only the weight-3 representative can be an orthogonal zero-difference direction. The other three pairs have weights $(2, 2)$, so either representative can be zero. The one-byte orthogonal switch probability is

$$4 \cdot \frac{1}{256} + 3 \cdot \frac{2}{256} = \frac{10}{256} \approx 2^{-4.68}.$$

For comparison, in a random permutation model, the probability that the eight mixed pairs can be partitioned into four good quartets is, approximately,

$$\frac{8!}{2^4 \cdot 4!} \cdot 2^{-8 \cdot 4} \approx 2^{-25.29}\,,$$

where $\frac{8!}{2^4 \cdot 4!}$ is the number of partitions of eight objects into four unordered pairs.

Good-column Quartet Partition. A similar way can be applied to derive the probability for good-column quartet partitions. For a fixed output column, one quartet partition must be compatible in all four byte positions. For the all-direct case, there is one assignment, counted only once. For the four weight-$(1, 3)$ partitions, each byte has two compatible choices, but the all-direct assignment is excluded. For the three weight-$(2, 2)$ partitions, each byte has three compatible choices, again excluding the all-direct assignment. Assuming independence between byte positions, we obtain

$$p'_{\text{wcol}} \approx \frac{1 + 4(2^4 - 1) + 3(3^4 - 1)}{256^4} = \frac{301}{2^{32}} \approx 2^{-23.77}. \tag{18}$$

Over all four columns, we obtain

$$p_{\text{wcol}} \approx 4p'_{\text{wcol}} \approx 2^{-21.77}\,.$$

This estimate already includes the seven quartet partitions. We confirmed all of the probabilities above experimentally.

Number of Good Quartets. For one tested byte, the structured AES setting permits only $0, 1, 4, 8, 12$, or 28 good quartets; the counts $2, 3, 5, 6, 7$ and $13, \ldots, 27$ do not occur. This follows from the coset propagation of switching-point equalities, together with the 4-differential-uniformity of the inverse S-box for isolated DDT entries. Supplementary Material C provides details on our probability derivation and summarizes our experimental results.

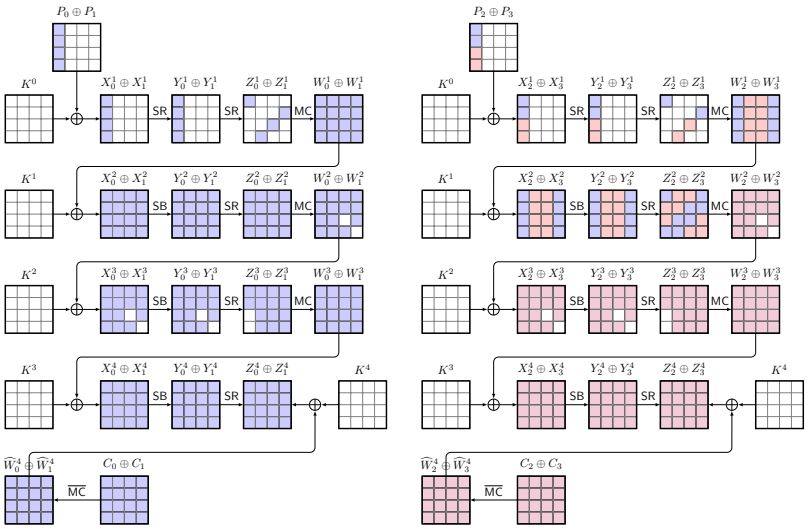


Fig. 3: Exemplary truncated-differential trails between two pairs of quartets in our distinguisher. The key solver is used to recover the input differences of bytes in the leftmost column through the final S-box layer.

6 A Low-data Four-round Distinguisher

This section combines the mixture-quartet switch with the key solver to build a low-data distinguisher for four-round AES.

6.1 Distinguishing Event

From a mixture structure of plaintexts, the adversary tests the seven possible quartet partitions of its eight mixed pairs. In the real AES, the switch can create a hidden good-column quartet partition in the states Z^3, before the MixColumns operation in Round 3. After the following MC operation, this gives two byte positions in which all four quartets form zero-free affine subspaces, which are then detected by the key solver.

Definition 10 (Two-dimensional Two-byte Quartet Partition). Fix a quartet partition $\mathcal{Q}_{\mathcal{D}}^{\mathbf{h}}(X_0,\, X_1)$. For all quartets defined by the directions $\mathbf{v}, \mathbf{h}$, let the set of bytes at some $r, c \in [0..3] \times [0..3]$ in the quartet be

$$\mathcal{B}_{r,c}(Q_{\mathbf{v},\mathbf{h}}) \stackrel{\text{def}}{=} \{X_{\mathbf{v}}[(r,c)], X_{\overline{\mathbf{v}}}[(r,c)], X_{\mathbf{v}\oplus\mathbf{h}}[(r,c)], X_{\overline{\mathbf{v}}\oplus\mathbf{h}}[(r,c)]\}\,.$$

We call the quartet partition a *two-dimensional two-byte quartet partition* if there exists some $c \in [0..3]$ and at least two distinct byte indices $r, r' \in [0..3]$ such that for all quartets in the partition, the values in $\mathcal{B}_{r,c}$ and $\mathcal{B}_{r',c}$ are zero-free, pairwise distinct, and form a two-dimensional affine subspace.

Table 2: Best low-data chosen-plaintext distinguishers on four-round AES. Mem. = memory (in states), succ. = success probability.

Attack	Data	Time	Mem.	Succ.
Multiple-of-8 [33]	2^{33}	$2^{33.7}$	2^{32}	≥ 0.95
Integral [18]	2^{32}	2^{32}	negl.	≥ 0.95
Mixture differential [32]	2^{17}	$2^{17.9}$	2^{17}	≥ 0.95
Impossible differential [10]	$2^{16.25}$	$2^{17.1}$	$2^{17.25}$	≥ 0.95
Mixture differential [Sect. 6]	$\mathbf{2^{14.0}}$	$\mathbf{2^{26.2}}$	$\mathbf{2^{14}}$	≥ 0.95

The distinguisher uses two bytes since a single byte would give a visible signal, but a smaller distance to the probability of a random permutation, while three bytes would require more data for a comparable success probability. In three-round AES, the event is produced by exploiting the mixture-quartet switch in all four bytes of a diagonal, and tracing the results after a mixture switch through the subsequent S-box layer. The mixture-quartet switch in a diagonal ensures that the outputs after three rounds will always sum to zero (as MixColumns is linear), one has to pay attention to have two bytes that are zero-free and whose values are pairwise distinct in every quartet.

Proposition 5. A mixture structure induces the event in Definition 10 after three rounds of AES with probability approximately $2^{-21.83}$.

Proof. Equation (18) gives the probability of a good-column quartet partition for a fixed column. Among the 301 compatible direction assignments, the 11 assignments where all four byte positions use the same zero-difference direction remain one-dimensional after MixColumns and are unusable for the key solver. The remaining 290 assignments give a two-row rank-two and zero-free factor $p_{\geq 2} \simeq 0.9964$. The full proof is given in Supplementary Material C.1. Testing all four output columns gives

$$p_{2\text{byte}} \approx 4 \cdot \frac{290}{2^{32}} \cdot p_{\geq 2} \approx 2^{-21.83}.$$

We verified the probability experimentally. □

Random-permutation Baseline. For a random permutation, applying the key solver of Proposition 4 to four unrelated quartet values returns essentially independent byte candidates. For one fixed partition, column, and row, treating the first candidate as the reference, equality of all four costs 2^{-24}. Requiring the same in two rows costs 2^{-48}. The distinguisher scans seven partitions, four columns, and six unordered row pairs. For N_{str} mixture structures, the expected number of random hits is therefore

$$N_{\text{str}} \cdot 7 \cdot 4 \cdot 6 \cdot 2^{-48}.$$

With $N_{\text{str}} = 2^{23.42}$, this is about $2^{-17.19}$, which is why any hit can be safely interpreted as four-round AES in the following attack.

6.2 Distinguisher

Data structure. Choose one active column, prepare a structure of $2^{13.98}$ plaintexts that iterate over all combinations of 9, 10, 12, and 15 values in the four active bytes. Pairs from this plaintext structure can combine into about

$$\binom{9}{2}\binom{10}{2}\binom{12}{2}\binom{15}{2} \approx 2^{23.42}$$

mixture structures. After encryption, the adversary applies the inverse of the final linear layer, including MC^{-1} and the inverse affine layer of the S-box, so that the key solver sees the key-wrapped inverse output.

Algorithm. The distinguisher proceeds as follows.

1. Ask for the ciphertexts corresponding to the $2^{13.98}$ plaintexts from the plaintext structure. For each ciphertext, invert the linear transformation in the last round, including the affine layer of the S-box, and store the transformed state used by the key solver.
2. Each mixture structure is obtained by choosing two values from each of the four active-byte sets. The resulting 16 texts define eight mixed pairs.
3. For every mixture structure, scan the seven quartet partitions of the eight mixed pairs into four quartets. For each partition and column c, apply the key solver on all four quartets and all four rows of c.
4. If there is a column and an unordered row pair $\{r, r'\}$ such that all four quartets suggest the same equivalent key byte in row r and also in row r', output "AES". If no hit is found, output "random".

The test uses the key solver only for detecting a structural event. It does not guess a key byte: the byte is derived from the four observed values and then compared across quartets.

Complexity. With $p_{2\text{byte}} \approx 2^{-21.83}$, the standard independence estimate gives a success probability of $1 - (1 - p_{2\text{byte}})^{N_{\text{str}}}$. Setting this value to 0.95 gives $N_{\text{str}} \approx 2^{23.41}$. We use $2^{13.98}$ plaintexts, which produce $2^{23.42}$ mixture structures. For each mixture structure, there are seven quartet partitions, four possible columns, and 16 key-solver calls per partition-column pair. This gives 448 solver calls per structure. Over $2^{23.42}$ structures, this is about $2^{32.23}$ solver calls. As each solver call can be implemented by a small $\mathbb{F}_{2^8}$ table lookup and a few byte operations, we count it as about 2^{-6} four-round encryption equivalents. The resulting time is $2^{26.2}$ four-round encryption equivalents.
The memory is dominated by the transformed ciphertext states used by the key solver. These require $2^{13.98}$ AES state equivalents. The $2^{23.42}$ mixture structures are enumerated on the fly from the four active-byte sets and are not stored. Apart from small byte lookup tables, the total memory is therefore about 2^{14} states. The overall complexity is given in Table 2.

Table 3: Best key-recovery attacks on seven-round AES-128. Enc. = encryption equivalents, MA = memory accesses to huge tables of size $\geq 2^{52}$ states, Mem. = memory (in states). $^{(*)}$ = uses one output differential, complexity from [41].

	Computations						
	Key recovery		Offline		Data collection		
Attack	Enc.	MAs	Enc.	MAs	Enc.	MAs	Mem.
DS-MitM [30]	2^{116}	2^{108}	2^{116}	2^{112}	2^{116}	$2^{120.3}$	2^{116}
Imp. differential [13]$^{(*)}$	$2^{112.1}$		–		$2^{106.1}$		$2^{74.1}$
Zero difference [5]	$2^{110.2}$		–		$2^{110.2}$		$2^{110.2}$
Imp. differential [40,41]	$2^{110.9}$		–		$2^{104.9}$		$2^{71.9}$
DS-MitM [26,27]	2^{99}	2^{96}	2^{92}	2^{88}	2^{105}	2^{105}	2^{90}
DS-MitM [27]	2^{99}	2^{96}	2^{100}	2^{96}	2^{97}	$2^{100.3}$	2^{98}
DS-MitM [Sect. 7]	$\mathbf{2^{92.4}}$	$\mathbf{2^{88.3}}$	$\mathbf{2^{96.6}}$	$\mathbf{2^{92.8}}$	$\mathbf{2^{97}}$	$\mathbf{2^{99.6}}$	$\mathbf{2^{92}}$

7 Improved DS-MitM Attack on Seven-round AES-128

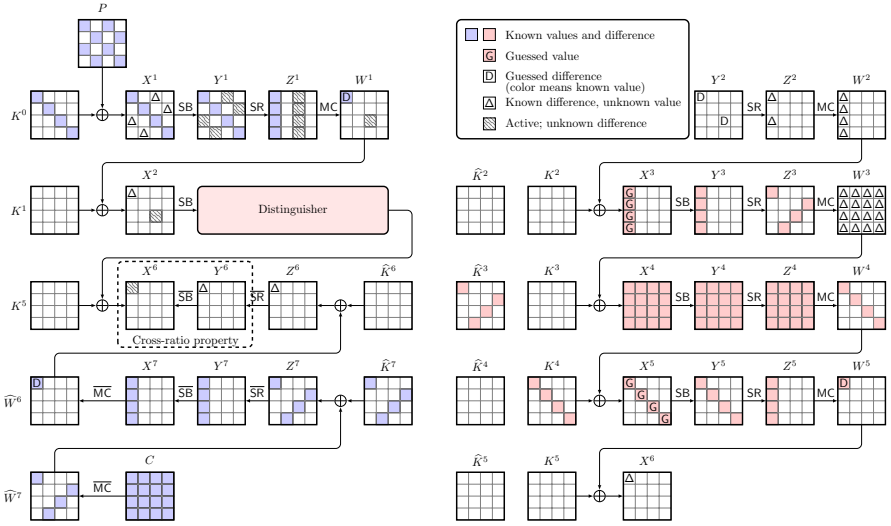
In the following, we improve the attack on seven-round AES-128 from [27] with the cross-ratio distinguisher and several small but effective techniques:

1. We use the cross-ratio property to match efficiently through the connecting round without guessing the corresponding byte of $\widehat{K}^6$.
2. Saving this guess of $\widehat{K}^6$ enables us to swap the online and offline phases, thereby reducing the memory complexity.
3. In the offline phase, we show that the tables from [27] are rotationally invariant. Thus, we can reduce the computational complexities by eliminating redundant computations.
4. We further exploit the independence of Super-S-boxes to compute the offline trails more efficiently.

One of the distinguishers and the corresponding key-recovery part that maps two active bytes to a single active byte through the distinguisher is shown in Figure 4. Our result and the comparison to previous attacks on seven-round AES-128 is given in Table 3. In the following, we give a high-level overview. To aid readers, an introduction to DS-MitM attacks is given in Supplementary Material D.

7.1 High-level Overview of The Original Attack

DS-MitM attacks exploit that, over a few rounds, a sequence of related texts depends on only a limited number of key and state variables. Hence, the sequence can assume far fewer values than expected for a random permutation. Following Dunkelman et al. [30], one can start from a pair that follows a (truncated)



(a) Online key-recovery phase. (b) Offline distinguisher trail.

Fig. 4: Key-recovery rounds and distinguisher on seven-round AES-128 [27] for one of 2^8 differential trails in the distinguisher with the 2-to-1 active-byte mapping. Colored and marked bytes are active.

differential trail through a subcipher to further reduce the internal values and computations. We call this pair the *reference pair*, and the text from which the related texts are defined the *reference text*. The original attack used a set of related texts that iterated over all values in an input byte and left all other bytes constant w.r.t. the reference text. This set, including the reference text, is usually called a δ-*set*. In an offline phase, all possible δ-sets are traced through the subcipher (Rounds 2–5 in Figure 4b), and all possible output sets are stored.

The Offline Phase. The distinguisher uses two families of 2^8 truncated differential trails in total. We call a trail a 2-*to*-1 *trail* if the active-byte pattern from ΔY^2 to ΔW^5 has $2 \to 4 \to 16 \to 4 \to 1$ active bytes after the successive round operations of the middle subcipher. There are 2^6 such trails: four choices for the active input diagonal in ΔY^2 and 16 choices for the single active byte in ΔW^5. Figure 4b depicts one of these trails, where the δ-set is constructed over $W^1[0]$. The attack also uses 1-*to*-2 *trails*, whose active-byte pattern from ΔY^2 to ΔW^5 is $1 \to 4 \to 16 \to 4 \to 2$. For the leftmost active column at ΔX^2, there are four choices for the active byte and $4 \cdot \binom{4}{2}$ choices for the two active bytes in ΔW^5, giving 96 trails. The same number of trails is obtained when the active byte at X^2 lies in the third column.

The Online Phase. In the online phase, the adversary builds 2^s structures of 2^{64} plaintexts, each iterating over all values in two plaintext diagonals. The ad-

versary looks for pairs that can potentially satisfy the differential trails through the middle, and discards all other pairs. For the potential pairs, It guesses outer key material to compute the pair to the beginning of the distinguisher, here ΔX^2, and derives the corresponding δ-set of related texts. The related texts are traced backwards to plaintexts, whose ciphertexts are then looked up. The adversary next guesses the key material needed to compute to the end of the distinguisher, forms the candidate output set, and tests whether this set appears in the offline tables. If a match is found, the current key guess is output. Figure 4a illustrates this online phase, i.e., the outer key-guessing rounds corresponding to the distinguisher trail in Figure 4b. Further details can be found in [27].

7.2 Matching with the Cross-ratio Property

Derbez et al.'s attack [27] used multisets, following Dunkelman et al. [30], to avoid guessing K^1. The price is that the order of the related texts is lost between the two sides of the meet-in-the-middle. In their matching step, one therefore has to guess one byte of $\widehat{K}^6$ to compute the multiset of differences at W^5.
We replace this matching value with a sequence of cross-ratio values. The cross-ratio property from Section 3 lets us match through the key-wrapped inverse S-box without guessing the connecting byte of $\widehat{K}^6$. The only requirement is that both sides agree on three anchor texts; each remaining related text is then used as the fourth input to the cross ratio. For clarity, we describe the case where the matching byte has index 0; the other byte positions are analogous. Let

$$\mathbf{x} = (x, x', x_1, \ldots, x_{255}) = (W^5[0], W'^5[0], W_1^5[0], \ldots, W_{255}^5[0])$$

denote the values on the W^5-side, and let

$$\mathbf{y} = (y, y', y_1, \ldots, y_{255}) = (\widehat{Y}^6[0], \widehat{Y}'^6[0], \widehat{Y}_1^6[0], \ldots, \widehat{Y}_{255}^6[0])$$

denote the corresponding values on the $\widehat{Y}^6$-side, after applying the inverse affine map of the AES S-box output layer. Without guessing K^1, the related texts outside the reference pair are not ordered consistently across the two sides. Thus, in general, x_i need not correspond to y_i.

Choosing Anchor Texts. To compute a canonical cross-ratio sequence, we need three anchor texts whose correspondence is known on both sides. The reference pair provides two anchors: we set $\widehat{x}_0 = x$, $\widehat{x}_1 = x'$, and use the corresponding $\widehat{y}_0 = y$, $\widehat{y}_1 = y'$ on the $\widehat{Y}^6$-side. It remains to choose a third anchor in a way that is independent of the unknown ordering of the related texts. We choose $\widehat{x}_2$ as a value of highest multiplicity in $\mathbf{x}$. If several values share the highest multiplicity, say r values, we break ties by choosing the one that gives the lexicographically smallest sequence of cross-ratio values with respect to the other tied values. The same rule can be applied on the $\widehat{Y}^6$-side: multiplicities are invariant under reordering, and the tie-breaking values are cross ratios, hence invariant through the key-wrapped inverse. The detailed procedure is given in Algorithm 5 in Supplementary Material F. It costs at most $3r^2$ memory accesses to small tables.

As experimentally evaluated in Supplementary Material F, this is less than 48 memory accesses on average.
Once $\widehat{x}_0, \widehat{x}_1, \widehat{x}_2$ have been fixed, the W^5-side computes, for every remaining value x_i, $\mathsf{CR}(\widehat{x}_0, \widehat{x}_1, \widehat{x}_2, x_i)$, and stores the resulting sorted sequence. The $\widehat{Y}^6$-side computes the analogous sequence $\mathsf{CR}(\widehat{y}_0, \widehat{y}_1, \widehat{y}_2, y_i)$. By Proposition 1, the two sequences match for the correct key guess without guessing the connecting byte of $\widehat{K}^6$, except for the usual zero cases of the patched inverse.
For the 24 1-to-2 trails, the orientation of the reference pair is not fixed in order to save a factor 2 in the offline computation. In this case, we apply the anchor-selection procedure twice, once with $(\widehat{x}_0, \widehat{x}_1) = (x, x')$ and once with $(\widehat{x}_0, \widehat{x}_1) = (x', x)$, and keep the lexicographically smaller sequence. The same convention is used on the $\widehat{Y}^6$-side.

Cost of Constructing a Matching Sequence. We use the following cost estimate for constructing one canonical cross-ratio matching sequence for each δ-set. Let m be the number of byte positions that have to be evaluated for all texts before the matching byte sequence is available, and let $\tau \in \{0, 1\}$ indicate whether the orientation of the reference pair must also be tested in the swapped order. Then, the cost is

$$C_{\mathrm{CR}}(m, \tau) = 257m + 2 \cdot 255 + 48(1 + \tau) + 254 + 2 \cdot 254 \tag{19}$$

memory accesses (MAs) to small byte-sized tables. The terms correspond to computing the relevant byte values for the reference pair and the 255 related texts, bucket-sorting to find a highest-multiplicity anchor, anchor selection by Algorithm 5, computing the 254 cross-ratio values, and bucket-sorting the resulting sequence, respectively.
On the W^5-side, only one byte sequence is constructed, so $m = 1$. For the 2-to-1 trails the reference-pair orientation is fixed, giving $C_{\mathrm{CR}}(1, 0) \approx 2^{10.6}$. For the 1-to-2 trails, we also test the swapped orientation, giving $C_{\mathrm{CR}}(1, 1) \approx 2^{10.7}$.

Swapping Phases. The cross-ratio property does not by itself require changing the order of the phases. Its effect is that the matching information no longer carries an additional factor 2^8 for the connecting byte of $\widehat{K}^6$. This makes it effective to swap the online and offline phases, as in [12,28]: we first run the online phase, store its matching sequences, and then generate the offline trail sets and look them up in these online tables. Without removing the $\widehat{K}^6$ guess, this phase swap would not reduce the memory complexity of the attack from [27].

7.3 Online Phase

The adversary builds $2^s = 2^{33}$ structures of 2^{64} plaintexts each, where each structure iterates over all values in the two diagonals $\mathcal{D}_{0,2}$. The online procedure is summarized in Algorithm 2.
After the differential filtering, we expect $2^{s+34.6} + 2^{s+33} \approx 2^{s+35} = 2^{68}$ surviving pairs on average. The first term comes from the 2-to-1 trails, and the second from the 1-to-2 trails. Each surviving pair suggests about 2^{20} key candidates:

Algorithm 2 Online phase of the DS-MitM attack on AES-128.

function OnlinePhase()
  for each of the 2^s structures of 2^{64} plaintext-ciphertext pairs **do**
    Compute $\widehat{W}^7 \leftarrow \overline{\mathsf{MC}}(C)$ for all ciphertexts C
    Store the $\widehat{W}^7$ values in a table $\mathcal{Q}$, indexed by the 64-bit value of the active plaintext diagonals $\mathcal{D}_{0,2}(P)$
    for all $\binom{4}{2}$ choices $k_1, k_2 \in [0..3]$ **do**
      Store the $\widehat{W}^7$ values in a table $\mathcal{Q}'_{k_1,k_2}$, indexed by the values in $\mathcal{ID}_{k_1,k_2}$
      Find pairs satisfying the ciphertext-side condition for 2-to-1 trails
        ▷ one active inverse diagonal after an additional fast check
      Find pairs satisfying the ciphertext-side condition for 1-to-2 trails
        ▷ two active inverse diagonals
      for each surviving pair **do**
        Look up all candidates for the active diagonal of K^0 that map the plaintext pair to an admissible single-byte difference after the first round
        for all candidates K^0 **do**
          Form the δ-set of 255 related texts in the active byte $\Delta W^1[(r_y, c_y)]$
          Trace these related texts backwards to plaintexts
          Look up their corresponding $\widehat{W}^7$ values from $\mathcal{Q}$
          Look up all candidates for the relevant inverse diagonal of $\widehat{K}^7 = \mathsf{MC}^{-1}(K^7)$ that map the ciphertext-side difference to an admissible difference at $\widehat{Y}^6$
          for all candidates $\widehat{K}^7$ **do**
            Compute the corresponding values at $\widehat{Y}^6[(r_w, c_w)]$
            Select the three anchors and build the cross-ratio sequence at $\widehat{Y}^6[(r_w, c_w)]$ using Algorithm 5
            Store the anchors and the cross-ratio sequence in $\mathcal{T}_{r_y,r_w,c_w}$ for 2-to-1 trails, or in $\mathcal{T}'_{r_y,r_w,r'_w,c_w}$ for 1-to-2 trails

- about $4 \cdot 2^8 = 2^{10}$ keys of $K^0[0, 5, 10, 15]$ on average (or $K^0[2, 7, 11, 12]$ for pairs following the 1-to-2 trails and have only the third plaintext diagonal active) to map the plaintexts to one of four possible differences after the first round.
- about $4 \cdot 2^8 = 2^{10}$ keys of one inverse diagonal of $\widehat{K}^7$, depending on the active inverse diagonal of ΔZ^7 that maps the ciphertext inverse diagonal to one out of four single-byte differences in the leftmost active column of ΔZ^6.

Data-collection Complexities. The attack uses $2^s = 2^{33}$ structures of 2^{64} texts, which results in 2^{97} chosen plaintexts and that amount of encryptions performed by the oracle. The probability of 2-to-1 distinguisher trails (see Figure 4) is $64 \cdot 2^{-48} \cdot 2^{-120} = 2^{-162}$ (the factor 2^{-48} results from the two $4 \to 1$ transitions in the MixColumns of round 1, 2^{-120} arises from the five $4 \to 1$ transitions in the MixColumns of rounds 6 and 7, and 64 is the number of trails considered). Thus, the $2^{33} \cdot \binom{2^{64}}{2} \approx 2^{160}$ possible pairs cover roughly 0.25 of the number of pairs

needed on average. The probability of the 1-to-2 distinguisher trails is about $96 \cdot 2^{-24} \cdot 2^{-112} \approx 2^{-129.4}$. Since each structure can construct $2 \cdot 2^{32} \cdot \binom{2^{32}}{2} \approx 2^{96}$ such pairs, the $2^{33} \cdot 2^{96} \approx 2^{129}$ considered pairs cover roughly 0.75 of the number of pairs needed on average; thus, both types of trails can be seen to produce one pair on average, with a success probability of approximately $1 - e^{-1} \approx 0.637$.
The states $\widehat{W}^7$ are stored in tables to find pairs satisfying the ciphertext-side conditions. We use a table $\mathcal{Q}$, indexed by the active plaintext-diagonal values, to retrieve the corresponding $\widehat{W}^7$ values for the current structure. We also use tables $\mathcal{Q}'_{k_1,k_2}$ for the $\binom{4}{2}$ choices $k_1, k_2 \in [0..3]$, indexed by the values in inverse diagonals $\mathcal{ID}_{k_1,k_2}$. These tables identify pairs with the required inactive inverse diagonals; the 2-to-1 case, which needs only one active inverse diagonal, is handled by an additional check in fast memory. Only one table $\mathcal{Q}'_{k_1,k_2}$ needs to be kept at a time. Thus, the pair-finding phase costs six random accesses to huge tables per ciphertext, plus a sequential access to $\mathcal{Q}$. We approximate this by $6 \cdot 2^{97} \approx 2^{99.6}$ MAs to huge tables.

Computational Complexity. The online filtering leaves about 2^{68} surviving pairs. For each pair and each K^0 candidate, the adversary derives the corresponding δ-set and retrieves its ciphertext-side values. This gives $2^{68} \cdot 2^{10} = 2^{78}$ such computations, each costing $255 \cdot 4$ S-box computations and 255 accesses to the huge table $\mathcal{Q}$.
For each pair and each key candidate $(K^0, \widehat{K}^7)$, the adversary constructs the matching sequence on the $\widehat{Y}^6$-side. Here, the guessed inverse diagonal of $\widehat{K}^7$ gives four byte positions to evaluate, so we use $m = 4$ in the cost estimate in Equation (19). The orientation issue adds only the small 48-MA term, and we approximate both trail families by $C_{\mathrm{CR}}(4, \tau) \approx 2^{11.2}$. Since there are about 2^{20} key candidates per surviving pair and about 2^{68} surviving pairs, this gives 2^{88} matching computations.
Counting 112 byte S-box/table operations as one seven-round encryption equivalent, the online computational complexity is therefore $2^{78} \cdot 255 \cdot 4/112 + 2^{88} \cdot 2^{11.2}/112 \approx 2^{92.4}$ seven-round encryption equivalents. The number of MAs to huge tables are approximately $2^{78} \cdot 255 + 2^{88} \approx 2^{88.3}$.

Memory Complexity. In the online phase, we guess the outer keys for all surviving pairs, store the results, and then conduct the offline phase, where we look up the corresponding sets in the online tables. It needs memory for 2^{64} plaintext-ciphertext pairs each in the tables $\mathcal{Q}$ and $\mathcal{Q}'_{k_1,k_2}$, i.e., 2^{66} state equivalents. In the attack, the adversary expects $2^{s+35} = 2^{68}$ ciphertext pairs that may follow the trails in the distinguisher. For each pair, approximately 2^{20} key candidates are guessed. For each pair and each key candidate, the adversary collects 257 bytes at the matching point $\widehat{Y}^6$ and stores them in 16 tables $\mathcal{T}_{r_y,r_w,c_w}$ and 24 tables $\mathcal{T}'_{r_y,r_w,r'_w,c_w}$ in total. However, the approximately 2^{88} entries on average are distributed over all tables. Thus, the memory complexity in the online phase is ca. $2^{66} + 2^{68+20} \cdot 257/16 \approx 2^{92}$ state equivalents.

7.4 Offline Phase

7.4.1 Previous Offline-table Construction

We recall the offline-table construction from [27], since our later optimizations reduce the number of trail sets and the cost of constructing each of them. For concreteness, we describe the 2-to-1 trails; the 1-to-2 trails are analogous.
Each offline table contains multisets of differences of one byte in ΔW^5, and the δ-set is constructed from a text at Y^2 that belongs to a pair (Y^2, Y'^2) that follows one of the differential trails. Originally, the adversary builds 2^6 tables $\mathcal{T}_{r_y,r_w,c_w}$, for all $(r_y, r_w, c_w) \in [0..3]^3$, each constructed from guesses of 11 bytes related to the reference pair and text following a differential trail, i.e., $255^3 \cdot 2^{64} \approx 2^{88}$ possibilities of

$$\Delta Y^2[(r_y, 0)], \quad \Delta Y^2[(r_y + 2, 2)], \quad \mathcal{C}_{4-r_y}(X^3), \quad \mathcal{D}_{c_w}(X^5), \quad \Delta W^5[(r_w, c_w)].$$

These variables are used to compute from ΔY^2 to ΔX^4 and from ΔW^5 backwards to ΔY^4, from which one can derive the full X^4 based on Property 1. Consequently, the adversary obtains $4 + 16 + 4$ byte parameters $\mathcal{C}_{4-r_y}(X^3)$, X^4, and $\mathcal{D}_{c_w}(X^5)$, which allow to compute the multiset $\{\Delta W_1^5[(r_w, c_w)], \ldots, \Delta W_{255}^5[(r_w, c_w)]\}$, where $\Delta W_i^5 = W^5 \oplus W_i^5$ is the difference of the i-th text w.r.t. the reference text. Moreover, it allows the adversary to derive the keys $\mathcal{ID}_{4-r_y}(\widehat{K}^3)$ and $\mathcal{D}_{c_w}(K^4)$.
In the following, we improve this procedure in several aspects:

1. We do not store the offline-phase results; instead, we refer to the trails in a table as a *trail set*.
2. We show that only a fraction of less than 1/4 of the trail sets is necessary.
3. We reduce the computations necessary for each trail.
4. We modify the multisets for efficient matching.

We outline the basic construction of the trail set of $\mathcal{T}_{0,0,0}$ in Algorithm 3.

7.4.2 Eliminating Redundant Trails

For 2-to-1 trails, Proposition 6 states that the multisets in the 2^4 trail sets $\mathcal{T}_{0,*,*}$ contain all other multisets for $r_y \neq 0$ due to rotational equivalences in the trails. Its proof and auxiliary notations are deferred to Supplementary Material E.

Proposition 6. For 2-to-1 trails, each entry in $\mathcal{T}_{r_y,r_w,c_w}$ with $r_y \neq 0$ exists also in $\mathcal{T}_{0,r_w-r_y,c_w}$.

A similar observation applies to 1-to-2 trails. For those, we added $c_y \in \{0, 2\}$ to indicate the column of the active byte in ΔY^2 and $r'_w \neq r_w$ for the row of the second active byte in $\Delta W^5[(r_w, c_w)]$ and $\Delta W^5[(r'_w, c_w)]$ and used four variables as indices.

Proposition 7. For 1-to-2 trails, each entry in $\mathcal{T}_{c_y,r_y,r_w,r'_w,c_w}$ with $r_y \neq 0$ exists also in $\mathcal{T}_{0,r_w-r_y,r'_w-r_y,c_w}$.

Algorithm 3 Computations of $\mathcal{T}_{0,0,0}$.

```
1: function compute()
2:     T_{0,0,0} ← ∅
3:     for all guesses of ΔY^2[(0,0)], ΔY^2[(2,2)] and C_0(X^3) do
4:         Compute from Y^2 and ΔY^2 to ΔX^4 with the known variables
5:         for all guesses of ΔW^5[(0,0)] and C_0(W^5) do
6:             Compute from C_0(Z^5) and ΔW^5 to ΔY^4 with the known variables
7:             if there exists a solution for ΔX^4 → ΔY^4 through the S-box then
8:                 Deduce the full states X^4 and X'^4
9:             else discard the current guess   ▷ 1 solution on average, cf. Property 1
10:            for i ← 1..255 do
11:                Derive Y_i^2 s.t. Y_i^2[0] ⊕ Y^2[0] = ΔY_i^2[0] = ⟨i⟩_8 and Y_i^2[j] = Y^2[j] for
                   all j ≠ 0
12:                Compute ΔC_0(X_i^3) and C_0(X_i^3) = ΔC_0(X_i^3) ⊕ C_0(X^3)
13:                Compute ΔX_i^4 and X_i^4 = ΔX_i^4 ⊕ X^4
14:                Compute ΔD_0(X_i^5) and D_0(X_i^5) = ΔD_0(X_i^5) ⊕ D_0(X^5) and ΔW_i^5[0]
15:            Build a multiset of differences ΔW_i^5[0], and add it to T_{0,0,0}
16:    return T_{0,0,0}
```

As a consequence of Propositions 6 and 7, the number of trails that need computation is reduced to $16 + 24 + 24 = 64$. Moreover, we observe that for each 1-to-2 trail starting from the third plaintext diagonal, there exists an equivalent trail in the existing sets of 1-to-2 trails from the first plaintext diagonal.

Proposition 8. For 1-to-2 trails, each entry in $\mathcal{T}_{2,r_y,r_w,r'_w,c_w}$ with $r_y \neq 0$ exists also in $\mathcal{T}_{0,r_w-r_y,r'_w-r_y,c_w+2}$.

Thus, one has to compute only the 24 1-to-2 trails with $c_y = 0$, and needs only trail sets $\mathcal{T}_{r_y,r_w-r_y,r'_w-r_y,c_w}$. Therefore, the attack can reduce the total number of trail sets from 2^8 to $16 + 24 = 40$.

Reference-invariant Trails. We can further reduce the size of the computations corresponding to 1-to-2 trails. Instead of guessing $255 \cdot 256$ combinations of $\Delta Y^2[(r_y, 0)]$ and $X^3[0]$ for the reference text, we can consider only the ca. 2^{15} guesses such that it holds that $Y^3[0] < Y'^3[0]$. These are combined with the approximately $2^{9 \cdot 8}$ guesses of $Y^3[1, 2, 3]$, $\mathcal{C}_{c_w}(Z^5)$, and $\Delta W^5[(\{r_w, r'_w\}, c_w)]$, so that we only need to cover 2^{87} guesses in those 24 trail sets. Note that this approach cannot be applied to the 2-to-1 trails, as their texts in the δ-set must share one of the two values diagonals with the reference text, while that diagonal differs in the other reference text. Hence, the total number of multisets to be computed in the offline phase reduces to $16 \cdot 2^{88} + 24 \cdot 2^{87} \approx 2^{92.8}$.

7.4.3 Accelerating the Offline Computations

Derbez et al. estimated the average computation of each text in the offline phase by 2^{-4} encryption equivalents [26,27, Sect. 3.2]. However, we can reuse the com-

putation of some multisets. The space of 2^{88} values of the parameters can be partitioned into approximately 2^{84} subsets, each containing 2^4 values. For each subset, the computation for 15 multisets of them can be derived by simply mixing the values at X^5 and X'^5 from the reference pair. Similarly, the computational effort can be further reduced by constructing mixtures from X^4 and X'^4. The details are provided in Supplementary Material G. As a result, each partial encryption in the offline phase is equivalent to less than $2^{-6.7}$ seven-round encryption equivalents.
In total, we obtain approximately $(2^{10.6} \cdot 16 \cdot 2^{88} + 2^{10.7} \cdot 24 \cdot 2^{87}) \cdot 1/112 \approx 2^{96.6}$ encryption equivalents for the offline phase. It needs $16 \cdot 2^{88} + 24 \cdot 2^{87} \approx 2^{92.8}$ MAs to find a match.

Success Probability. We expect a filtering quality in the order of that of Derbez et al., since we effectively have unordered multisets, which should yield only the correct key. However, matching with the cross-ratio distinguisher through the key-wrapped S-box comes at a probabilistic price of hitting zero. Thus, the attack is successful if and only if (1) the data contains a pair that follows one of the differential trails, and (2) the texts from that pair and its 255 related texts do not hit zero at the matched S-box at X^6. The first probability is well-known to be approximately $1 - e^{-1}$. Under the assumption that the values at X^6 are uniformly distributed, the probability of the second term is $(255/256)^{257} \approx 0.366$. Therefore, we obtain a lower bound of $2^{-2.11} \approx 0.23$ for the attack to succeed vs. the previous attack of $1 - e^{-1} \approx 0.63$. Note that the second probability could potentially be improved by matching on an appropriate subset of x, x', $\widehat{x}_2$, and only those elements that occur with sufficient frequency, e.g., more than once.

8 Conclusion and Discussion

We brought two properties of the multiplicative inverse to light to improve actual attacks on round-reduced AES plus generalized the S-box switch to multiple pairs. Our properties are versatile, which is demonstrated by our applications in very detrimental settings. The cross-ratio property could slightly reduce the adversary's guessing effort in the online phase of the DS-MitM attack on seven-round AES-128 and reduce its memory complexity. This improves the best attack on the most widespread cipher for the first time after more than a decade. Moreover, the mixture-quartet switch provides cryptanalysts with another tool in the box, reducing the data complexity for chosen-plaintext distinguishers on four rounds. Every technique possesses limitations, and our proposals are no exception. The cross-ratio property needs quartets to work, which is given in the DS-MitM attack on seven-round AES-128, but not the case in the state-of-the-art attacks on other versions and not necessarily in other meet-in-the-middle settings, e.g., for compression-function cryptanalysis. For key recovery, the most relevant limitation is the need to study all quartets, i.e., pairs of pairs, instead of only texts, which may increase the complexity exponentially in terms of available data. We see possible extensions of our four-round distinguisher for key recovery.

References

1. Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *LNCS*, pages 103–119. Springer, 2008.
2. Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 70–89. Springer, 2009.
3. Zhenzhen Bao, Jian Guo, and Eik List. Extended Truncated-differential Distinguishers on Round-reduced AES. *IACR Trans. Symmetric Cryptol.*, 2020(3):197–261, 2020.
4. Achiya Bar-On, Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. Improved Key Recovery Attacks on Reduced-Round AES with Practical Data and Memory Complexities. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO II*, volume 10992 of *LNCS*, pages 185–212. Springer, 2018.
5. Navid Ghaedi Bardeh and Vincent Rijmen. New Key-Recovery Attack on Reduced-Round AES. *IACR Trans. Symmetric Cryptol.*, 2022(2):43–62, 2022.
6. Navid Ghaedi Bardeh and Sondre Rønjom. Practical Attacks on Reduced-Round AES. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi, editors, *Africacrypt*, volume 11627 of *LNCS*, pages 297–310. Springer, 2019.
7. Navid Ghaedi Bardeh and Sondre Rønjom. The Exchange Attack: How to Distinguish Six Rounds of AES with $2^{88.2}$ Chosen Plaintexts. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT III*, volume 11923 of *LNCS*, pages 347–370. Springer, 2019.
8. Augustin Bariant, Orr Dunkelman, Nathan Keller, Gaëtan Leurent, and Victor Mollimard. Improved Boomerang Attacks on 6-Round AES. *IACR Cryptol. ePrint Arch.*, page 977, 2024.
9. Augustin Bariant and Gaëtan Leurent. Truncated Boomerang Attacks and Application to AES-Based Ciphers. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 3–35. Springer, 2023.
10. Eli Biham and Nathan Keller. Cryptanalysis of Reduced Variants of Rijndael. http://csrc.nist.gov/archive/aes/round2/conf3/ papers/35-ebiham.pdf, 2001.
11. Aurélien Boeuf, Anne Canteaut, and Léo Perrin. Propagation of Subspaces in Primitives with Monomial Sboxes: Applications to Rescue and Variants of the AES. *IACR Trans. Symmetric Cryptol.*, 2023(4):270–298, 2023.
12. Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum Security Analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.
13. Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the Impossible Possible. *J. Cryptology*, 31(1):101–133, 2018.
14. Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT I*, volume 8873 of *LNCS*, pages 179–199. Springer, 2014.
15. Claude Carlet. On the vector subspaces of $\mathbb {F}_{2^n}$ over which the multiplicative inverse function sums to zero. *Des. Codes Cryptogr.*, 93(4):1237–1254, 2025.
16. Chengcheng Chang, Meiqin Wang, Ling Sun, and Wei Wang. Improved Truncated Differential Distinguishers of AES with Concrete S-Box. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT*, volume 13774 of *Lecture Notes in Computer Science*, pages 422–445. Springer, 2022.

17. Shiyao Chen, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang. Diving Deep into the Preimage Security of AES-Like Hashing. In Marc Joye and Gregor Leander, editors, *EUROCRYPT I*, volume 14651 of *Lecture Notes in Computer Science*, pages 398–426. Springer, 2024.
18. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *FSE*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.
19. Joan Daemen and Vincent Rijmen. AES proposal: Rijndael, 1998.
20. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
21. Joan Daemen and Vincent Rijmen. Understanding Two-Round Differentials in AES. In Roberto De Prisco and Moti Yung, editors, *SCN*, Lecture Notes in Computer Science, pages 78–94. Springer, 2006.
22. Hüseyin Demirci and Ali Aydin Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.
23. Hüseyin Demirci, Ihsan Taskin, Mustafa Çoban, and Adnan Baysal. Improved Meet-in-the-Middle Attacks on AES. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *LNCS*, pages 144–156. Springer, 2009.
24. Patrick Derbez and Pierre-Alain Fouque. Exhausting Demirci-Selçuk Meet-in-the-Middle Attacks Against Reduced-Round AES. In Shiho Moriai, editor, *FSE*, volume 8424 of *LNCS*, pages 541–560. Springer, 2013.
25. Patrick Derbez and Pierre-Alain Fouque. Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO II*, volume 9815 of *Lecture Notes in Computer Science*, pages 157–184. Springer, 2016.
26. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. *IACR Cryptol. ePrint Arch.*, page 477, 2012. Version from 2012-08-21.
27. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 371–387. Springer, 2013.
28. Patrick Derbez and Léo Perrin. Meet-in-the-Middle Attacks and Structural Analysis of Round-Reduced PRINCE. In Gregor Leander, editor, *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 190–216. Springer, 2015.
29. Xiaoli Dong, Jun Liu, Yongzhuang Wei, Wen Gao, and Jie Chen. Meet-in-the-middle attacks on AES with value constraints. *Des. Codes Cryptogr.*, 92(9):2423–2449, 2024.
30. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 158–176. Springer, 2010.
31. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. *J. Cryptol.*, 28(3):397–422, 2015.
32. Lorenzo Grassi. Mixture Differential Cryptanalysis: a New Approach to Distinguishers and Attacks on round-reduced AES. *IACR Transactions on Symmetric Cryptology*, 2018(2):133–160, 2018.
33. Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. A New Structural-Differential Property of 5-Round AES. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT II*, volume 10211 of *LNCS*, pages 289–317, 2017.

34. Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace Trail Cryptanalysis and its Applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2):192–225, 2017.
35. Lorenzo Grassi and Markus Schofnegger. Mixture Integral Attacks on Reduced-Round AES with a Known/Secret S-Box. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 312–331. Springer, 2020.
36. Thomas Jakobsen and Lars R. Knudsen. The Interpolation Attack on Block Ciphers. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.
37. Lars R. Knudsen and David A. Wagner. Integral Cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.
38. Nikolay Kolomeec and Denis Bykov. On the image of an affine subspace under the inverse function within a finite field. *Des. Codes Cryptogr.*, 92(2):467–476, 2024.
39. Daesung Kwon, Jaesung Kim, Sangwoo Park, Soo Hak Sung, Yaekwon Sohn, Jung Hwan Song, Yongjin Yeom, E-Joong Yoon, Sangjin Lee, Jaewon Lee, et al. New block cipher: ARIA. In *ICISC*, pages 432–445. Springer, 2003.
40. Gaëtan Leurent and Clara Pernot. New Representations of the AES Key Schedule. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT I*, volume 12696 of *Lecture Notes in Computer Science*, pages 54–84. Springer, 2021.
41. Gaëtan Leurent and Clara Pernot. New representations of the AES key schedule. *J. Cryptol.*, 38(1):2, 2025.
42. Leibo Li, Keting Jia, and Xiaoyun Wang. Improved Single-Key Attacks on 9-Round AES-192/256. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *LNCS*, pages 127–146. Springer, 2014.
43. Rongjia Li and Chenhui Jin. Meet-in-the-middle attacks on 10-round AES-256. *Des. Codes Cryptogr.*, 80(3):459–471, 2016.
44. Tianren Liu, Stefano Tessaro, and Vinod Vaikuntanathan. The t-wise Independence of Substitution-Permutation Networks. In Tal Malkin and Chris Peikert, editors, *CRYPTO IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 454–483. Springer, 2021.
45. Jiqiang Lu and Wenchang Zhou. Improved meet-in-the-middle attack on 10 rounds of the AES-256 block cipher. *Des. Codes Cryptogr.*, 92(4):957–973, 2024.
46. Jiqiang Lu and Wenchang Zhou. Improved Meet-in-the-Middle Attacks on Nine Rounds of the AES-192 Block Cipher. In Elisabeth Oswald, editor, *CT-RSA*, volume 14643 of *Lecture Notes in Computer Science*, pages 136–159. Springer, 2024.
47. National Institute of Standards and Technology. FIPS 197. *National Institute of Standards and Technology, November*, pages 1–51, 2001.
48. Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT I*, volume 10624 of *Lecture Notes in Computer Science*, pages 217–243. Springer, 2017.
49. Bing Sun. Provable Security Evaluation of Block Ciphers Against Demirci-Selçuk's Meet-in-the-Middle Attack. *IEEE Trans. Inf. Theory*, 67(7):4838–4844, 2021.
50. Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic Attacks on Round-Reduced Rain and Full AIM-III. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT III*, volume 14440 of *Lecture Notes in Computer Science*, pages 285–310. Springer, 2023.

Algorithm 4 Three-round distinguisher using Proposition 4 with six CPs.

function $\mathsf{solveForKey}(y_0, y_1, y_2, y_3)$
  return $(e_3(y_0, y_1, y_2, y_3)/e_1(y_0, y_1, y_2, y_3))^{2^7}$

function $\mathbf{A}()$
  Choose (P_0, P_1) with $\mathsf{wt}_{\mathcal{D}}(P_0 \oplus P_1) \geq 3$
  Generate two new distinct unordered mixed pairs (P_2, P_3) and (P_4, P_5) from (P_0, P_1)
  Ask the challenger for their corresponding ciphertexts $C_0, \ldots, C_5$
  $\widehat{Y}_i^3 \leftarrow \overline{\mathsf{L}}(\overline{\mathsf{SR}}(\overline{\mathsf{MC}}(C_i)))$ for all $i \in [0..5]$
  for $j \leftarrow 0..15$ **do**
    $(y_0, \ldots, y_5) \leftarrow (\widehat{Y}_0^3[j], \ldots, \widehat{Y}_5^3[j])$
    if $|\{y_0, \ldots, y_5\}| < 6$ **then continue** ▷ $(y_0, \ldots, y_5)$ are not pairwise distinct
    $k \leftarrow \mathsf{solveForKey}(y_0, y_1, y_2, y_3)$
    $k' \leftarrow \mathsf{solveForKey}(y_0, y_1, y_4, y_5)$
    if $k = k'$ **then return** 1 ▷ Real round-reduced AES
    else return 0 ▷ Random permutation
  return 0 ▷ Random permutation by default

Supplementary Material

A A Distinguisher Using Six CPs and Proposition 4

Distinguisher Idea. The high-level idea is as follows. The adversary **A** chooses a plaintext pair (P_0, P_1) that differ in at least three diagonals. Define the difference vector $\mathbf{v} \in \mathbb{F}_2^4$ such that $\mathbf{v}[i] = 1$ iff $\mathcal{D}_i(P_0) \neq \mathcal{D}_i(P_1)$ and 0 otherwise. **A** generates two pairwise distinct unordered mixed pairs (P_2, P_3) and (P_4, P_5) from (P_0, P_1) by exchanging diagonals. Let the corresponding states after two rounds be X_i^3 for $i \in [0..5]$. Due to Property 2, all pairs will have the same difference $X_{2i}^3 \oplus X_{2i+1}^3$ for $i \in [0..2]$ after two rounds. Therefore, for any fixed byte index j, it will hold that $X_0^3[j] \oplus X_1^3[j] = \alpha_1$ for some α_1.
Thus, any combination of two pairs $X_{2i}^3, X_{2i+1}^3, X_{2i'}^3, X_{2i'+1}^3$ for $i \neq i'$ spans a subspace of dimension at most two. While the subsequent S-box may destroy this property, the adversary can use Proposition 4 to recover it. **A** inverts the final MixColumns and ShiftRows operations, as well as the affine layer of the S-box. Let the resulting states be $\widehat{Y}_i^3$ and the outputs from the S-box for the first two pairs in some Byte j be y_i for $i \in [0..5]$. The adversary chooses the first such byte such that all six values y_i are pairwise distinct. Then, using Proposition 4, the key solver will return a suggestion for the key. Whenever all values in one byte are pairwise distinct and nonzero before the final XOR with an equivalent key, which is satisfied with high probability, all quartets suggest the same key. In contrast, the probability of the event is close to 2^{-8} for a random permutation. A formal description of the distinguisher is given in Algorithm 4.

Complexities and Probabilities. The distinguisher needs six chosen plaintexts, negligible memory, six three-round encryptions, six single-column decryp-

tions, and six fast lookups to a table that fits into fast cache. It has a negligible computational complexity of computing e_3 and e_1 twice each. In 100 000 experiments each with three-round AES, the distinguisher returned true

- with probability 0.944, when testing for the first byte only, and
- with probability 0.962, when using any byte where all six texts were pairwise distinct, if existing.

In contrast, in 100 000 more experiments, each with a truncated ideal permutation, the distinguisher returned true

- with probability 0.011, when testing for the first byte only, and
- with probability 0.015, when using any byte where all six texts were pairwise distinct, if existing.

B Related Work

We note that the cross-ratio distinguisher is a variant of the observation by Jakobsen and Knudsen in their seminal work that introduced interpolation attacks [36]. They showed in their attacks on a simplified variant of SHARK that an output y of a key-alternating cipher with the multiplicative inverse as its round function can be expressed as a quotient of its input x and three key-dependent constants c_i: $y = (x \oplus c_1)(c_2 x \oplus c_3)$. In [44, App. B], Liu et al. revisited essentially the same property. Both works showed that four texts would suffice to obtain a b-bit equation system through the multiplicative inverse that would recover the constants and test on their equality. However, the focus of their works differed from ours significantly; [36] had both in- and outputs available at the same time, and the goal of [44] was to advance the constructive understanding of the AES' t-wise independence, and not on developing attacks. Thus, neither work made the step towards developing an efficiently testable relation such that both sides depended only on either the inputs or outputs. Our contributions are (1) a formalization that allows for processing the differences from each side separately, with only either input or output differences from four texts, which allows to derive (2) generalizations and special cases relevant to attacks, and (3) demonstrate their usefulness in several distinguisher and key-recovery attacks on round-reduced AES.

C Further Discussion of Good Quartet Counts

We further analyze the four good quartets in more detail, considering good quartets that may have overlapping mixed pairs rather than forming a strict partition.
The simplest case occurs when the byte difference at the switching point X^3 is zero for all mixed pairs, resulting in $\binom{8}{2} = 28$ good quartets. The remaining cases with nonzero input difference are analyzed below. For clarity, we denote the

probability of the mixture-quartet switch occurring with nonzero input difference at X^3 as

$$P_n = (1 - 2^{-8}) \cdot (4/256 + 3/128) \simeq 2^{-4.68}\,.$$

12 **Good Quartets.** Consider the four good quartets from a mixture-quartet switch. When two of these quartets share identical pairs at X^3, and the remaining two quartets also share identical pairs at X^3 (but with different output differences than the first two quartets), we obtain additional good quartets. Since two identical quartets contribute $\binom{4}{2} = 6$ good quartets, we have a total of 12 good quartets.
Since each of the original four good quartets consists of two identical pairs, when two good quartets are identical, the remaining two good quartets must also be identical by the symmetry. The probability that two quartets are same is around $P_d = 1/256$. Thus, the probability for 12 good quartets is:

$$P_n \cdot P_d \simeq 2^{-12.68}\,.$$

28 **Good Quartets.** When the output differences from the 12 good quartets described above are identical, all $\binom{8}{2} = 28$ possible pairs form good quartets. Additionally, the most straightforward case yielding 28 good quartets occurs when the input difference at the switching point is zero, which ensures that all mixed pairs have zero output differences. The probability is about 2^{-8}.

Eight Good Quartets. For the four good quartets in a mixture-quartet switch, the probability that they are mutually different is $1 - P_d$. Under this condition, if two of these quartets contain pairs that correspond to a 4-entry in the DDT, then the output differences at Z^3 for the pairs in these two quartets are identical. Since each quartet contains two pairs, the combination of two such quartets yields $\binom{4}{2} = 6$ good quartets. Including the other two good quartets, we obtain eight good quartets in total. The probability is approximately

$$P_n \cdot (1 - P_d) \cdot \binom{4}{2} \cdot \frac{4}{256} \cdot \frac{2}{256} \simeq 2^{-15.10}\,.$$

Four Good Quartets. The four good quartets are mutually different, and all pairs correspond to 2-entry in the DDT. The probability is approximately

$$P_n \cdot (1 - P_d) \cdot \left(1 - \binom{4}{2} \cdot \frac{4}{256} \cdot \frac{2}{256}\right) \simeq 2^{-4.69}\,.$$

Apart from the cases derived from the mixture-quartet switch, there are two other cases of good quartets.

Table 4: Probabilities for different numbers of occurring good quartets Q for one byte from a mixture structure for three-round AES and random values. Experiments use 2^{32} mixture structures. MQ is the random variable that denotes the occurrence of a non-direct mixture-quartet switch.

	Q						
Case	0	1	4	8	12	28	MQ
3-round AES theory	0.95	$2^{-8.26}$	$2^{-4.69}$	$2^{-15.10}$	$2^{-12.68}$	$2^{-8.00}$	$2^{-4.68}$
3-round AES experiment	0.95	$2^{-8.24}$	$2^{-4.70}$	$2^{-15.07}$	$2^{-12.69}$	$2^{-8.00}$	$2^{-4.69}$
Random experiment	0.90	$2^{-3.31}$	$2^{-14.94}$	2^{-31}	–	–	$2^{-25.57}$

One Good Quartet. All the eight mixed pairs are mutually different, but two of them have the same output difference, that is, the values correspond to a 4-entry in the DDT. Thus, there is only one good quartet, and the probability is approximately

$$(1-2^{-8}) \cdot \left(1 - \frac{4}{256} - \frac{3}{128}\right) \cdot \binom{8}{2} \cdot \frac{4}{256} \cdot \frac{2}{256} \simeq 2^{-8.26}\,.$$

No Good Quartet. All the eight mixed pairs are mutually different, and none of them have the same output difference. Thus, we obtain no good quartet, and the probability is approximately

$$(1-2^{-8}) \cdot \left(1 - \frac{4}{256} - \frac{3}{128}\right) \cdot \left(1 - \binom{8}{2} \cdot \frac{4}{256} \cdot \frac{2}{256}\right) \simeq 0.95\,.$$

We verified all probabilities by experiments. For comparison, we also provide experimental results for a random permutation. The complete theoretical and experimental results are presented in Table 4.

C.1 Proof of Proposition 5

This subsection gives the counting behind Proposition 5. We start from the probability of a good-column quartet partition for a fixed column in Equation (18). For one output column, this event has probability

$$p^{(1)}_{\mathrm{wcol}} \approx \frac{301}{2^{32}}.$$

The number 301 counts the compatible direction assignments for the four byte positions of that column. A weight-$(1,3)$ partition offers two admissible directions in each byte position, namely the direct direction and the weight-3 non-direct direction; over four such partitions this gives $4(2^4-1)$ assignments after removing the all-direct assignment. A weight-$(2,2)$ partition offers three admissible directions, namely the direct direction and the two weight-2 non-direct

directions; over three such partitions this gives $3(3^4 - 1)$ assignments. Adding the all-direct assignment once gives

$$1 + 4(2^4 - 1) + 3(3^4 - 1) = 301.$$

Not all of these assignments are useful after the next MixColumns. If all four byte positions use the same zero-difference direction, then every quartet stays one-dimensional after MixColumns and cannot trigger the two-dimensional key-solver condition. There are

$$\binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 11$$

such common-direction assignments, corresponding to the admissible directions of weights 2, 3, and 4. Thus the useful assignments are the remaining

$$301 - 11 = 290$$

non-common direction assignments.

We now count the probability that one of these 290 assignments becomes useful for at least two output rows. Fix one quartet. Its four points are indexed by $t = (t_1, t_2) \in \mathbb{F}_2^2$. In one input byte, saying that the quartet contains exactly two values paired in direction ℓ means that the byte value has the form

$$z_j(t) = a_j + b_j \ell_j(t), \qquad b_j \neq 0,$$

where $\ell_j : \mathbb{F}_2^2 \to \mathbb{F}_2$ is one of the three nonzero linear forms. The case $b_j = 0$ is the degenerate one-value case. If all four input bytes use the same form ℓ_j, then every output byte after MixColumns has the form $A + B\ell_j(t)$ and is still one-dimensional. This is precisely why the 11 common-direction assignments are removed.

For a non-common assignment, write the four input-byte values of a quartet as $z_j(t) = a_j + b_j \ell_j(t)$ for $j = 0, \ldots, 3$. For output row r of the MixColumns operation,

$$w_r(t) = \bigoplus_{j=0}^{3} m_{r,j} z_j(t) = A_r \oplus \bigoplus_{j=0}^{3} c_{r,j} \ell_j(t), \qquad c_{r,j} = m_{r,j} b_j \neq 0,$$

because all MixColumns coefficients $m_{r,j}$ are nonzero. Choose coordinates on the quartet so that the three possible pairings are represented by t_1, t_2, and $t_1 \oplus t_2$. If these three forms occur with multiplicities (n_1, n_2, n_3) among the four input bytes, and if S_i is the XOR sum of the coefficients $c_{r,j}$ attached to the i-th form, then

$$w_r(t_1, t_2) = A_r \oplus U_r t_1 \oplus V_r t_2, \qquad U_r = S_1 \oplus S_3, \quad V_r = S_2 \oplus S_3.$$

This output quartet is a rank-two affine plane in the byte if and only if

$$U_r \neq 0, \qquad V_r \neq 0, \qquad U_r \neq V_r.$$

It remains to count the distribution of the sums S_i. For $s \in \mathbb{F}_{2^8}$, define

$$N_n(s) = \#\{(x_1, \ldots, x_n) \in (\mathbb{F}_{2^8}^*)^n : x_1 \oplus \cdots \oplus x_n = s\}.$$

Thus $N_n(s)$ is the number of ways that n nonzero byte coefficients can XOR to s. The standard character-sum count gives

$$N_n(0) = \frac{255^n + 255(-1)^n}{256}, \qquad N_n(a) = \frac{255^n - (-1)^n}{256} \quad (a \neq 0).$$

For a fixed multiplicity triple (n_1, n_2, n_3), the rank-two probability in one output row is therefore

$$p_{\mathrm{rk2}}(n_1, n_2, n_3) = \frac{\sum_{\substack{s_1, s_2, s_3 \in \mathbb{F}_{2^8} \\ s_1 \oplus s_3 \neq 0,\ s_2 \oplus s_3 \neq 0,\ s_1 \neq s_2}} N_{n_1}(s_1) N_{n_2}(s_2) N_{n_3}(s_3)}{255^{n_1+n_2+n_3}}.$$

Only three non-common multiplicity types can occur, giving

(n_1, n_2, n_3)	p_{rk2}	number of direction assignments
$(3, 1, 0)$	$16451834/16581375 \simeq 0.99219$	104
$(2, 2, 0)$	$16387064/16581375 \simeq 0.98828$	78
$(2, 1, 1)$	$16387318/16581375 \simeq 0.98830$	108.

The last column is obtained as follows. For the four weight-$(1, 3)$ partitions, each byte position has two admissible directions; after deleting the two common-direction assignments, the remaining patterns have types $(3, 1, 0)$ and $(2, 2, 0)$ with counts 8 and 6. For the three weight-$(2, 2)$ partitions, each byte position has three admissible directions; after deleting the three common-direction assignments, the remaining patterns have types $(3, 1, 0)$, $(2, 2, 0)$, and $(2, 1, 1)$ with counts 24, 18, and 36. Hence the totals are

$$4(8, 6, 0) + 3(24, 18, 36) = (104, 78, 108),$$

which sum to 290.
For a fixed output row, all four quartets must be rank two. Averaging over the 290 non-common direction assignments gives

$$p_{\mathrm{row,rk2}} = \frac{104 p_{310}^4 + 78 p_{220}^4 + 108 p_{211}^4}{290} \simeq 0.95941,$$

where p_{310}, p_{220}, and p_{211} denote the three rank-two probabilities in the table. A rank-two byte plane

$$\{A, A \oplus U, A \oplus V, A \oplus U \oplus V\}$$

contains zero for exactly four choices of the offset A. Hence, for fixed nonzero and distinct U, V, the plane is zerofree for 252 of the 256 offsets. The zerofree factor for four quartets in one row is therefore $(252/256)^4$, and

$$p_{\mathrm{row}} = p_{\mathrm{row,rk2}} \left(\frac{252}{256}\right)^4 \simeq 0.90083.$$

Definition 10 requires at least two output rows in the same column. Counting the complement of having zero or one good row gives

$$p_{\geq 2} = 1 - (1 - p_{\text{row}})^4 - 4p_{\text{row}}(1 - p_{\text{row}})^3 \simeq 0.99639.$$

A direct one-column experiment over the same 290 non-common assignments gives $p_{\geq 2} = 0.99636$, in agreement with this value.
Combining the pieces, for one fixed output column we obtain

$$p_{\text{fixed col}} \approx \frac{290}{2^{32}} \cdot p_{\geq 2}.$$

The distinguisher accepts if any of the four output columns satisfies the two-byte event. Since this event is rare, the column scan contributes a factor 4, giving

$$p_{\text{2byte}} \approx 4 \cdot \frac{290}{2^{32}} \cdot p_{\geq 2} \approx 2^{-21.825} \approx 2^{-21.83}.$$

D Demirci-Selçuk MitM Attacks

We present a brief introduction to Demirci-Selçuk meet-in-the-middle (DS-MitM) attacks [22] in this section. We start with the definition of δ-sequences (and -sets), which are well-known since the square attack [18]. Since their elements can be ordered, we consider the term of a δ-sequence.

Definition 11 (δ-sequence). For positive integers b, m, and a set $\mathcal{J} \subseteq [0..m-1]$, let $d := 2^{b \cdot |\mathcal{J}|}$ and let $\mathcal{X} = (X^0, X^1, \ldots, X^{d-1}) \subseteq (\mathbb{F}_{2^b}^m)^d$ be a sequence of pairwise distinct states. Let $X^i[\mathcal{J}] := \|_{j \in \mathcal{J}} X^i[j]$. We call $\mathcal{X}$ a δ-sequence if and only if for all distinct $X^i, X^{i'} \in \mathcal{X}$, it holds that

$$X^i[j] = X^{i'}[j] \quad \text{for all } j \notin \mathcal{J} \quad \text{and} \quad X^i[\mathcal{J}] \neq X^{i'}[\mathcal{J}]\,.$$

We call $\mathcal{X}$ a δ-set if it is an unordered set.

We refer to the set of state cells whose indices in $\mathcal{J}$ as the active cells and to its complement as the set of passive cells.

D.1 The General Demirci-Selçuk MitM Attack

For non-empty sets $\mathcal{K}$, $\mathcal{M}$, and $\mathcal{C}$, a DS-MitM attack decomposes a keyed function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ into three segments: $E = E_2 \circ E_d \circ E_1$. The attack consists of two phases, usually called the offline phase and the online phase, as the former can often be precomputed before the online querying phase. During the offline phase, a table $\mathcal{T}_{\mathsf{off}}$ is constructed that describes the possible mappings of a set of (partial) states from X to Y through E_d.
Given an input δ-sequence $\mathbf{X} = (X^0, X^1, \ldots, X^{d-1})$, the corresponding output sequences through E_d, $f(\theta, \mathbf{X}) = \mathbf{Y}_\theta = (Y^0, Y^1, \ldots, Y^{d-1})$, are computed for all instances of internal parameters θ. The resulting output sequences are then

stored in $\mathcal{T}_{\mathsf{off}}$. DS-MitM attacks exploit that the number of internal parameters, and therefore, the number of possible output sequences, is considerably lower for a real cipher than for an ideal construction. The adversary uses the precomputed table of possible output sequences to recover key bits in the online phase.
However, as in other MitM attacks, there is no need to fully separate the phases. Instead, DS-MitM attacks can also be seen as a variant of Aoki and Sasaki's splice-and-cut framework [1,2], where E_1 and E_2 represent one chunk, computed in the same direction, which wraps the second chunk E_d. The oracle $\mathcal{O}$ that maps plaintexts to ciphertexts acts can be used to splice the computational paths from E_1 to E_2 (or vice versa).

D.2 Previous Improvements

Standard Time-data Trade-off. Demirci and Selçuk [22] proposed a straightforward time-data-memory trade-off: instead of computing the full offline table for all, let us define $T_{\mathsf{off}} = 2^t$, necessary guesses of possible internal values and storing them, they would compute and store only a fraction of 2^{-c}, where the definition of c-bit constraints in the trail is a comprehensible, reproducible, and straightforward way of deciding which ones to store. On average, the adversary must collect a factor of 2^c times the original amount of data during the online phase to find a text (or pair) that conforms to a trail computed and stored in the offline table. This also implied that they have to repeat the online and key-recovery phases 2^c times since they considered no a-priori method to detect which data would be stored. Let T_{off}, T_{on}, M_{off}, and M_{on} denote the previous time and memory complexities for off- and online phases, respectively, and D the data complexity. Then, the standard trade-off can reduce $T'_{\mathsf{off}} = T_{\mathsf{off}} \cdot 2^{-c}$ and $M'_{\mathsf{off}} = M_{\mathsf{off}} \cdot 2^{-c}$ at the price of increasing $T'_{\mathsf{on}} = T_{\mathsf{on}} \cdot 2^c$ and $D' = D \cdot 2^c$, while the online memory usually remains unaffected: $M'_{\mathsf{on}} = M_{\mathsf{on}}$.

Differences Instead of Values. Instead of considering the sequence of values, the adversary can further reduce the complexity by considering the sequence of differences w.r.t. a **reference text** that is chosen from the sequence of values [22]. This reduction avoids guessing the key bytes added before the distinguisher's output states.

Multisets. Dunkelman et al. [30,31] proposed using unordered multisets instead of ordered sequences. They showed that multisets still contain sufficient information to filter out mismatches and avoid false positives during the matching between the online and offline computations.

Partial Sets. While multisets can reduce the complexity by not requiring certain key guesses, the adversary loses the information about which texts in a δ-set are mapped to which outputs between online and offline phases. In contrast, if the mapping is known, the stronger filter allows a different improvement proposed by Demirci et al. in [23]: instead of computing the trails for the full δ-set

in the offline phase, the adversary uses only a subset of the first m texts, e.g. 32 instead of all 256, of it such that its filtering probability suffices so that the key-recovery phase that tests the surviving key candidates will not dominate the overall attack complexity. While Dunkelman et al. [30,31] revisited [23], the approach had been used in many later DS-MitM attacks on AES-192 and -256, such as in [29,42,43,45,46].

Differential Enumeration. In [30,31], Dunkelman et al. proposed to construct the δ-sequence from a pair of texts that follows a differential trail $\Delta X \xrightarrow{E_d} \Delta Y$ through E_d. This approach could significantly reduce the number of potential values for the internal parameters and thus the size of the offline table. On the other hand, the adversary has to query chosen plaintexts to find at least one pair that fulfills a trail $\Delta P \xrightarrow{E_1} \Delta X \xrightarrow{E_d} \Delta Y$ in the online phase, which increases the online complexity. Throughout this work, we will refer to this pair as the **reference pair** of a set of dependent texts, and the element in the reference pair is called the **reference text**.

Multiple Trails. Instead of only using a single computational trail in the offline phase, Dunkelman et al. [30,31] used multiple differentials to reduce the data complexity, which was adopted by Derbez et al. [27].

Swapping Off- and Online Phase. Derbez and Perrin [28] and later Bonnetain et al. [12] proposed to reverse what is offline and what is online phase, as is usual in many other MitM attacks, to store the part with the lower memory requirements. Then, both phases are conducted online.

E Rotational Equivalence

E.1 Rotational Differences and Invariants in The AES

To denote row- and column-wise rotated states, we define cell-wise rotated differences and differentials. Let n_r, n_c be positive integers and let $\mathbb{F} = \mathbb{F}_{p^b}^{n_r \times n_c}$ in the following.

Definition 12 (Cell-wise rotated state and rotational difference). Let $(X, X') \in \mathbb{F}^2$. We say that X' is $(\Delta r, \Delta c)$-cell-wise rotated w.r.t. X if $X'[r + \Delta r][c + \Delta c] = X[r][c]$ for all $r \in [0..n_r - 1]$ and all $c \in [0..n_c - 1]$. Then, we write $X' = X \ggg (\Delta r, \Delta c)$ and call $(\Delta r, \Delta c)$ the cell-wise rotational difference.

Definition 13 (Cell-wise rotational differential). Let $F : \mathbb{F} \to \mathbb{F}$ be a function. Let $\Delta r, \Delta r' \in [0..n_r - 1]$ and $\Delta c, \Delta c' \in [0..n_c - 1]$. We write $(\Delta r, \Delta c) \xrightarrow[p]{F} (\Delta r', \Delta c')$ and call it a cell-wise rotational differential with probability p if

$$\Pr_{X \in \mathbb{F}}[F(X \ggg (\Delta r, \Delta c)) = F(X) \ggg (\Delta r', \Delta c')] = p\,.$$

We omit writing p if $p = 1$, i.e. if the differential holds for all inputs $X \in \mathbb{F}$ to F. For a function $F : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$, we write $(\Delta r, \Delta c) \xrightarrow[p]{F[(\Delta r'', \Delta c'')]} (\Delta r', \Delta c')$ (again, we omit writing p if $p = 1$) if for all $K \in \mathbb{F}$

$$\Pr_{X \in \mathbb{F}} \left[F\left(K \ggg (\Delta r'', \Delta c''), X \ggg (\Delta r, \Delta c)\right) = F(K, X) \ggg (\Delta r', \Delta c') \right] = p\,.$$

We can formulate well-known rotational invariants through the operations of the AES round function.

Property 4 (Rotational Invariants of AES Round Operations). For all $(\Delta r, \Delta c) \in [0..3] \times [0..3]$, it holds that:

$$(\Delta r, \Delta c) \xrightarrow{\mathsf{SB}^{\pm}} (\Delta r, \Delta c) \tag{2a}$$

$$(\Delta r, \Delta c) \xrightarrow{\mathsf{SR}} (\Delta r, \Delta c - \Delta r) \tag{2b}$$

$$(\Delta r, \Delta c) \xrightarrow{\mathsf{SR}^{-1}} (\Delta r, \Delta c + \Delta r) \tag{2c}$$

$$(\Delta r, \Delta c) \xrightarrow{\mathsf{MC}^{\pm}} (\Delta r, \Delta c) \tag{2d}$$

$$(\Delta r, \Delta c) \xrightarrow{\mathsf{AK}[(\Delta r, \Delta c)]} (\Delta r, \Delta c)\,. \tag{2e}$$

The proof is straightforward, using the fact that the MixColumns matrix $\mathbf{M}$ is circular, and is therefore omitted.

E.2 Proof of Proposition 6

We consider the computations of the sequences for the two representations:

- First, that of a base sequence from $\mathbf{Y}^2 = (Y^2, Y'^2, Y_1^2, \ldots, Y_{255}^2)$ to $\mathbf{W}^5 = (W^5[(r_w, c_w)], W'^5[(r_w, c_w)], W_1^5[(r_w, c_w)], \ldots, W_{255}^5[(r_w, c_w)])$.
- Second, that of $\widetilde{\mathbf{Y}}^2 = (\widetilde{Y}^2, \widetilde{Y'}^2, \widetilde{Y}_1^2, \ldots, \widetilde{Y}_{255}^2)$ to $\widetilde{\mathbf{W}}^5 = (\widetilde{W}^5[(r_w, c_w)], \widetilde{W'}^5[(r_w, c_w)], \widetilde{W}_1^5[(r_w, c_w)], \ldots, \widetilde{W}_{255}^5)[(r_w, c_w)]$.

We define $\widetilde{r}_a = r_y + \Delta$ for some $\Delta \in [1..3]$. The proof is shown in three steps in Lemmas 2–4.
Lemma 2 traces the rotational difference $\Delta Y^2 \ggg (\Delta, 0)$ to X^4.

Lemma 2 (Computing Forwards from ΔY^2 to X^4). Let $\Delta Y^2, Z^3 \in \mathbb{F}_{2^8}^{4\times 4} \times \mathbb{F}_{2^8}^{4\times 4}$ be fixed. Let $\Delta\widetilde{Y}^2 = \Delta Y^2 \ggg (\Delta, 0)$ for arbitrary $\Delta \in [0..3]$. Then, there exists $\widetilde{Y}^3 \in \mathbb{F}_{2^8}^{4\times 4}$ such that $\widetilde{X}^4 = X^4 \ggg (\Delta, 2\Delta)$, $\widetilde{X'}^4 = X'^4 \ggg (\Delta, 2\Delta)$, and $\widetilde{X^4}_i = X_i^4 \ggg (\Delta, 2\Delta)$, for all $i \in [1..255]$.

Proof. Denote $\Delta Y_i^2 = Y_i^2 \oplus Y^2$ for all i. Having a different active row in Y^2 means that $\Delta\widetilde{Y}^2 = \Delta Y^2 \ggg (\Delta, 0)$. Properties (2b) and (2d) ensure that

$$\begin{aligned}\Delta\widetilde{W}_i^2 &= \mathsf{MC}(\mathsf{SR}(\Delta\widetilde{Y}_i^2)) \\ &= \mathsf{MC}(\mathsf{SR}(\Delta Y_i^2 \ggg (\Delta, 0))) = \Delta W_i^2 \ggg (\Delta, -\Delta)\,.\end{aligned}$$

Next, we choose $\widetilde{X}^3 = X^3 \ggg (\Delta, -\Delta)$. Note that $\Delta\widetilde{W}_i^2 = \Delta\widetilde{X}_i^3$, Property (2e) ensures that

$$\begin{aligned}\widetilde{X}_i^3 &= \widetilde{X}^3 \oplus \Delta\widetilde{X}_i^3 \\ &= \widetilde{X}^3 \oplus \Delta\widetilde{W}_i^2 = (X^3 \ggg (\Delta, -\Delta)) \oplus (\Delta W_i^2 \ggg (\Delta, -\Delta)) \\ &= X_i^3 \ggg (\Delta, -\Delta),\end{aligned}$$

and $\Delta\widetilde{X}_i^3 = \Delta X_i^3 \ggg (\Delta, -\Delta)$.
Properties (2a) (2b) and (2d) ensure that

$$\begin{aligned}\Delta\widetilde{W}_i^3 &= \mathsf{MC}(\mathsf{SR}(\mathsf{SB}(\Delta\widetilde{X}_i^3))) \\ &= \mathsf{MC}(\mathsf{SR}(\mathsf{SB}(\Delta X_i^3 \ggg (\Delta, -\Delta)))) = \Delta W_i^3 \ggg (\Delta, -2\Delta)\,.\end{aligned}$$

Finally, we choose $\widetilde{X}^4 = X^4 \ggg (\Delta, -2\Delta)$. Note that $\Delta\widetilde{W}_i^3 = \Delta\widetilde{X}_i^4$, Property (2e) ensures that

$$\begin{aligned}\widetilde{X}_i^4 &= \widetilde{X}^4 \oplus \Delta\widetilde{X}_i^4 = \widetilde{X}^4 \oplus \Delta\widetilde{W}_i^3 = (X^4 \ggg (\Delta, -2\Delta)) \oplus (\Delta W_i^3 \ggg (\Delta, -2\Delta)) \\ &= X_i^4 \ggg (\Delta, -2\Delta) = X_i^4 \ggg (\Delta, 2\Delta)\,,\end{aligned}$$

where the last equality follows trivially since all rotational row and column differences are modulo 4.

Next, Lemma 3 propagates the rotational difference from $\Delta X^6 \ggg (\Delta, 0)$ backwards to Y^4 (which is identical to that at X^4) and shows that it matches that of Lemma 2 for the reference pair.

Lemma 3 (Computing Backwards from ΔX^6 to Y^4). Let $\Delta X^6, W^5 \in \mathbb{F}_{2^8}^{4\times4} \times \mathbb{F}_{2^8}^{4\times4}$ be fixed. Let $\Delta\widetilde{X}^6 = \Delta X^6 \ggg (\Delta, 0)$ for arbitrary $\Delta \in [0..3]$. Then, there exists $\widetilde{W}^5 \in \mathbb{F}_{2^8}^{4\times4}$ such that $\widetilde{Y}^4 = Y^4 \ggg (\Delta, 2\Delta)$ and $\widetilde{Y'}^4 = Y'^4 \ggg (\Delta, 2\Delta)$.

Proof. Let $\Delta\widetilde{W}^5 = \Delta W^5 \ggg (\Delta, 0)$, and we have $\Delta\widetilde{W}_i^5 = \Delta W_i^5 \ggg (\Delta, 0)$. Property (2d) ensures that

$$\Delta\widetilde{Z}_i^5 = \Delta Z_i^5 \ggg (\Delta, 0).$$

We choose $\widetilde{Z}^5 = Z^5 \ggg (\Delta, 0)$, property (2e) ensures that

$$\widetilde{Z}_i^5 = \widetilde{Z}^5 \oplus \Delta\widetilde{Z}_i^5 = (Z^5 \ggg (\Delta, 0)) \oplus (\Delta Z_i^5 \ggg (\Delta, 0)) = Z_i^5 \ggg (\Delta, 0).$$

Applying Properties (2c) and (2a), we obtain

$$\widetilde{X}_i^5 = \mathsf{SB}^{-1}(\mathsf{SR}^{-1}(\widetilde{Z}_i^5)) = \mathsf{SB}^{-1}(\mathsf{SR}^{-1}(Z_i^5 \ggg (\Delta, 0))) = X_i^5 \ggg (\Delta, \Delta).$$

We choose $\widetilde{K}^4 = K^4 \ggg (\Delta, \Delta)$, Property (2e) yields

$$\widetilde{W}_i^4 = \widetilde{X}_i^5 \oplus \widetilde{K}^4 = (X_i^5 \ggg (\Delta, \Delta)) \oplus (K^4 \ggg (\Delta, \Delta)) = (W_i^4 \ggg (\Delta, \Delta)).$$

Finally, Properties (2d) and (2c) again yields

$$\widetilde{Y}_i^4 = \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(\widetilde{W}_i^4)) = \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(W_i^4 \ggg (\Delta, \Delta))) = Y_i^4 \ggg (\Delta, 2\Delta).$$

Finally, Lemma 4 shows that the set of states for all values of a derived δ-set is preserved under the in- and output rotational differences above.

Lemma 4 (Computing Forwards from X_i^4 to ΔX_i^6). Let ΔY^2, Z^3, ΔX^6, $W^5 \in (\mathbb{F}_{2^8}^{4\times 4})^4$ be fixed. Let $\Delta\widetilde{Y}^2 = \Delta Y^2 \ggg (\Delta, 0)$ and $\Delta\widetilde{X}^6 = \Delta X^6 \ggg (\Delta, 0)$ for arbitrary $\Delta \in [0..3]$. Then, there exist $\widetilde{Z}^3, \widetilde{W}^5 \in (\mathbb{F}_{2^8}^{4\times 4})^2$ such that $\widetilde{W}_i^5 = W_i^5 \ggg (\Delta, 0)$ for all $i \in [1..255]$.

Proof. Using the parameters chosen by Lemma 2 and 3, the transitions $\widetilde{X}_i^4 \xrightarrow{\mathsf{SB}} \widetilde{Y}_i^4$ are valid through $\mathsf{SubBytes}$ since those $X_i^4 \xrightarrow{\mathsf{SB}} Y_i^4$ are valid and the rotational differences are equal on both sides.
Properties (2b) and (2d) give

$$\widetilde{W}_i^4 = \mathsf{MC}(\mathsf{SR}(\widetilde{Y}_i^4)) = \mathsf{MC}(\mathsf{SR}(Y_i^4 \ggg (\Delta, 2\Delta))) = W_i^4 \ggg (\Delta, \Delta)\,.$$

Lemma 3 had used $\widetilde{K}^4 = K^4 \ggg (\Delta, \Delta)$. From Property (2e) follows that

$$\widetilde{X}_i^5 = \widetilde{K}^4 \oplus \widetilde{W}_i^4 = (K^4 \ggg (\Delta, \Delta)) \oplus (W_i^4 \ggg (\Delta, \Delta)) = X_i^5 \ggg (\Delta, \Delta)\,.$$

Applying Properties (2a), (2b), and (2d) yields

$$\widetilde{W}_i^5 = \mathsf{MC}(\mathsf{SR}(\mathsf{SB}(\widetilde{X}_i^5))) = W_i^5 \ggg (\Delta, 0)\,,$$

which yields our claim.

F Experiments on The Maximal Multiplicity

Algorithm 5 requires r iterations, each involving $r-1$ lookups to compute the cross-ratio values, $r-1$ writes, and $r-1$ reads to sort a sublist $\mathcal{B}'$, as well as r MAs to find the lexicographically smallest sequence $\mathcal{B}'$. This results in approximately $3r^2$ MAs to small tables.
We experimentally evaluated the distribution of what is the multiplicity of the highest multiplicity in a set of 255 random values from $\mathbb{F}_{2^8}$ in 10^6 experiments; with probabilities 0.47, 0.65, 0.75, and 0.84, it is at most 1, 2, 3, or 4, and the observed mean of $3r^2$ is approximately 32.05. With probability approximately $2r'/255$, exactly one element of x or x_1 has the maximal multiplicity and the list either becomes smaller or one has to build the list of the next highest multiplicity; we observed a mean of $3r^2 \approx 69.0$ for this case. Moreover, with probability approximately $(r'/255)^2$, both x_0 and x_1 differ but both have maximal multiplicity and the list is either reduced by two elements or one has to build the list of the next highest multiplicity. In this case, we observed a mean of $3r^2 \approx 136.9$. Even when using r' as high as 10 in those events, which is highly unlikely, we obtain for $32.05 + 2r'/255 \cdot 69 + (r'/255)^2 \cdot 136.9 < 37.7 < 3 \cdot 4^2$. Therefore, we can use $3r^2 < 48$ as an upper bound for the average number of memory accesses to identify $\mathcal{B}'$.

Algorithm 5 Determine the third text $\widehat{x}_2$ of the cross-ratio distinguisher for 1-to-2 trails. $\prec$ refers to lexicographically smaller for sorted sequences.

function determineThirdText$(x, x', (x_1, \ldots, x_{255}))$
  Initialize a list $\mathcal{B}[i] \leftarrow 0$ for all $i \in [0..255]$
  $\widehat{x}_0 \leftarrow x$; $\widehat{x}_1 \leftarrow x'$; $\alpha_1 \leftarrow \widehat{x}_0 \oplus \widehat{x}_1$
  if $\alpha_1 = 0$ **then return** $\perp$
  for $i \leftarrow 1..255$ **do** $\mathcal{B}[x_i] \leftarrow \mathcal{B}[x_i] + 1$
  $\mathcal{B}[\widehat{x}_0] \leftarrow 0$; $\mathcal{B}[\widehat{x}_1] \leftarrow 0$
  Find $b_1, \ldots, b_r$ s.t. $\mathcal{B}[b_1] = \cdots = \mathcal{B}[b_r] = \max_i\{\mathcal{B}[i]\}$
  $\mathcal{B}'_{\min} \leftarrow \emptyset$
  for $i \leftarrow 1..r$ **do**
    $\mathcal{B}' \leftarrow \emptyset$; $\widehat{x}_2 \leftarrow b_i$; $\alpha_2 \leftarrow \widehat{x}_0 \oplus \widehat{x}_2$
    for all $j \in [1..r] \setminus \{i\}$ **do**
      $\sigma_j^i \leftarrow \frac{(\alpha_1 \oplus \alpha_2)(\widehat{x}_0 \oplus b_j)}{(\alpha_1 \oplus \widehat{x}_0 \oplus b_j)\alpha_2}$
      $\mathcal{B}' \leftarrow \mathcal{B}' \| \sigma_j^i$
    Sort $\mathcal{B}'$
    if $\mathcal{B}'_{\min} = \emptyset$ **or** $\mathcal{B}' \prec \mathcal{B}_{\min}$ **then** $\mathcal{B}'_{\min} \leftarrow \mathcal{B}'$; $b \leftarrow b_i$
  return b

G Accelerating Offline Computations for The 7-round Attack on AES-128

Mixtures of X^5 and X'^5. Given a value of the 11-byte vector that produces a valid differential trail thorugh the distinguisher, we can derive the first diagonals $\mathcal{D}_0(X^5)$ and $\mathcal{D}_0(X'^5)$ to create a mixture set of 2^4 four-byte vectors[6]:

$$\{\rho^{\alpha}(\mathcal{D}_0(X^5), \mathcal{D}_0(X'^5)) : \alpha \in \mathbb{F}_2^4\}.$$

Since there is a bytewise bijective relation between $\mathcal{D}_0(X^5)$ and $\mathcal{C}_0(Z^5)$, we can partition the internal parameters into subsets of 2^4 each.
Leaving the other internal parameters constant, we compute the output set at $W^5[0]$ for each vector of the mixture set in the following.

1. We first compute an output set at $W^5[0]$ in full from $Y^2[0]$ using the guess $\mathcal{G}$ and thus the vector $\mathcal{D}_0(X^5)$ from the mixture set. This costs 24 S-box evaluations per text, i.e. $24 \cdot (2 + 255)$ S-box computations.
2. We define a Gray code $\varphi : \mathbb{Z}_{16} \times \mathbb{F}_2^4 \to \mathbb{F}_2^4$ that iterates over $\mathbb{F}_2^4$. In the i-th iteration of φ, we update the previous vector to

$$\rho^{\varphi(i,\alpha)}(\mathcal{D}_0(X^5), \mathcal{D}_0(X'^5)),$$

   by swapping one byte between $\mathcal{D}_0(X^5)$ and $\mathcal{D}_0(X'^5)$.
3. Since swapping bytes does not affect $\Delta\mathcal{D}_0(X^5)$, ΔY^4 remains unchanged and does not affect the derived parameters X^4 and Y^4. Thus, for each iteration, the output set at $\mathcal{D}_0(W^4)$ is the same as the one in Step 1, we only have to recompute the affected byte from $\mathcal{D}_0(X^5)$ to $W^5[0]$ for all texts.

[6] The 4-byte values in $\Delta\mathcal{D}_0(X^5)$ are nonzero due to the MDS matrix of AES.

Algorithm 6 Computation of trail set $\mathcal{T}_{0,0,0}$ using mixtures of X^5 and X'^5.

function constructTable$(0, 0, 0)$
  Initialize a lookup table $\mathcal{T}$
  for all values of the seven bytes $(\Delta Y^2[0, 10], Y^3[0..3], \Delta X^6[0])$ **do**
    Compute differences of ΔX^4
    for each mixture set of $\mathcal{D}_0(X^5)$ **do**
      Take the value $\mathcal{D}_0(X^5)$ from the reference pair, compute ΔY^4 and deduce X^4 and X'^4
      Initialize multisets $\mathcal{M}_x \leftarrow \emptyset$ and $\mathcal{M}_w \leftarrow \emptyset$
      for $Y_i^2[0]$ in the δ-set **do**
        Compute $\mathcal{D}_0(X_i^5)$ and $\Delta X_i^6[0] = W_i^5[0] \oplus W_0^5[0]$, and add them to $\mathcal{M}_x$ and $\mathcal{M}_w$, respectively
      Add $\mathcal{M}_w$ to $\mathcal{T}$
      for the remaining 15 vectors in the mixture set **do**
        Empty the multiset $\mathcal{M}_w$
        Swap $X^5[5j]$ and $X'^5[5j]$ for some $j \in [0..3]$ by Gray code
        for each element in $\mathcal{M}_x$ **do**
          Update $X_i^5[5j] = X_i^5[5j] \oplus \Delta X^5[5j]$
          Compute $\Delta X_i^6[0]$ with one S-box operation and add it to $\mathcal{M}_w$
        Add $\mathcal{M}_w$ to $\mathcal{T}$
  return $\mathcal{T}$

Assume the parameter bytes $X^5[5j]$ and $X'^5[5j]$ are swapped for some $j \in [0..3]$ in the current iteration, then all the values $X_i^5[5j]$ in the output set of the last iteration can be updated to $X_i^5[5j] \oplus \Delta X^5[5j]$. The computation of $W_i^5[0]$ from the updated $X_i^5[5j]$ costs one S-box evaluation and one multiplication by a constant in the MixColumns operation per text. The steps are also described in Algorithm 6.
In sum, the computation of the multiset of $\Delta X_i^6[0]$ for each mixture set needs $256 \cdot 24$ S-box for the first four-byte vector and $15 \cdot 256 \cdot 1$ S-box operations for the other 15 four-byte vectors, i.e. $257 \cdot (1 \cdot 24 + 15 \cdot 1) \simeq 2^{13.3}$ instead of the naive $16 \cdot 257 \cdot 24 \simeq 2^{16.6}$ S-boxes.

Mixtures of X^4 and X'^4. We can save computational effort also by constructing mixtures from X^4 and X'^4 where the states are fully active in the distinguisher.
Assume a fixed differential $(\Delta X^4, \Delta Y^4)$ through the S-box layer, if there exists a solution (X^4, X'^4) that fulfills the differential, i.e. $\mathsf{SB}(X^4) \oplus \mathsf{SB}(X^4 \oplus \Delta X^4) = \Delta Y^4$, we can construct mixture sets of X^4 and X'^4 by taking all possible solutions for $X^4[i]$ such that $S(X^4[i]) \oplus S(X^4[i] \oplus \Delta X^4[i]) = \Delta Y^4[i]$ over all $i \in [0..15]$. To quantify the size of such a mixture set, we define

$$\zeta(\Delta X^4, \Delta Y^4) \stackrel{\text{def}}{=} |\mathcal{S}_{\mathsf{sol}}(\Delta X^4, \Delta Y^4)| = \prod_{i=0}^{15} |\mathcal{S}_{\mathsf{sol}}(\Delta X^4[i], \Delta Y^4[i])|$$

with $\zeta(\Delta X^4, \Delta Y^4) \geq 2^{16}$ since $|\mathcal{S}_{\mathsf{sol}}(\Delta X^4[i], \Delta Y^4[i])| \in \{0, 2, 2^2\}$ implies that

$$\zeta(\Delta X^4, \Delta Y^4) = \begin{cases} 0 & \text{if there exists } i : |\mathcal{S}_{\mathsf{sol}}(\Delta X^4[i], \Delta Y^4[i])| = 0 \\ 2^k & \text{for some } k \in [16..32] \text{ otherwise}. \end{cases}$$

Thus, we can partition the space of all 2^{88} values of the internal parameters into mixture sets, such that each represents all the values of X^4 defined by $\Delta X^4, \Delta Y^4$, which is represented by the set

$$\{\rho^\alpha(X^4, X'^4) : \alpha \in \mathbb{F}_2^k\}.$$

In the following, we write ζ for $\zeta(\Delta X^4, \Delta Y^4)$. Note that we leave all the 11-byte vectors of parameters constant. The size ζ of the mixture set compensates for the fact that for the vast majority of values of the 11-byte vectors, there exists no solution and if there exist solutions, it comes in batches of ζ solutions.

We compute the output set at $W^5[0]$ for each vector of the mixture set in the following.

1. We first compute an output set at $W^5[0]$ in full from $Y^2[0]$ using the vector X^4 from the mixture set, which costs 24 S-boxes per text, and $24 \cdot 256$ in total.
2. We use a Gray code $\varphi : \mathbb{Z}_\zeta \times \mathbb{F}_2^k \to \mathbb{F}_2^k$ such that every newly computed vector of the mixture set updates only a single byte $X^4[j]$ with respect to the previously computed vector. For each iteration:
   (a) Swap the values of $X^4[j]$ and $X'^4[j]$, and do the same for $Y^4[j]$ and $Y'^4[j]$.
   (b) For all $i \in [0..255]$, update $X_i^4[j]$ to $\widetilde{X}_i^4[j] = X_i^4[j] \oplus \Delta X^4[j]$ and $Y_i^4[j]$ to $\widetilde{Y}_i^4[j] = S(\widetilde{X}_i^4[j])$.
   (c) For all $i \in [0..255]$, update $\widetilde{W}_i^4[\ell] = W_i^4[\ell] \oplus \gamma \cdot (\widetilde{Y}_i^4[j] \oplus Y_i^4[j])$, where ℓ is the byte position index corresponding to the affected byte in W^4 and X^5, and γ be the relevant coefficient in the MixColumns matrix.
   (d) Update the key byte $K^4[\ell]$ to $\widetilde{K}^4[\ell] = K^4[\ell] \oplus \gamma \cdot \Delta Y^4[j]$ to keep the parameter $X^5[\ell]$ unchanged. The reason is that we leave all the 11-byte vectors constant, each iteration changes only the values in the middle (mostly fully active) round but not in the wrapping rounds.
   (e) For all $i \in [0..255]$, update $\widetilde{X}_i^5[\ell] = \widetilde{W}_i^4[\ell] \oplus \widetilde{K}^4[\ell]$ and update $\widetilde{W}_i^5[0]$ by computing another S-box and a multiplication with the subsequent MixColumns factor.

Algorithm 7 Construction of the table $\mathcal{T}_{0,0,0}$ using mixtures of X^4 and X'^4.

function ConstructTable$(0, 0, 0)$
  Initialize a lookup table $\mathcal{T}$
  for all values of the 11 bytes $(\Delta Y^2[0, 10], Y^3[0..3], Z^5[0..3], \Delta X^6[0])$ **do**
    Compute ΔX^4 and ΔY^4
    if the solution set of $(\Delta X^4, \Delta Y^4)$ is non-empty **then** ▷ The probability is 2^{-16}
      Take one solution (X^4, Y^4) from the solution set, which is the mixture set of size ζ
      Initialize a multiset $\mathcal{M} \leftarrow \emptyset$
      for $Y_i^2[0]$ in the δ-set **do**
        Compute to $\Delta X_i^6[0]$, add it to $\mathcal{M}$, and record the values X_i^4, Y_i^4, Y_i^5
      Add $\mathcal{M}$ to $\mathcal{T}$
      for the remaining $\zeta - 1$ values in the mixture set **do**
        Empty the multiset $\mathcal{M}$
        Update the previous X^4 by the Gray code and the corresponding byte $K^4[\ell]$
        for the recorded values $\{X_i^4, Y_i^4, Y_i^5\}$ **do**
          Compute the changed byte of Y_i^4, and then $W_i^4[\ell]$ and $X_i^5[\ell]$
          Compute $W_i^5[0]$, and finally the difference $\Delta X_i^6[0]$ and add it to $\mathcal{M}$
        Add $\mathcal{M}$ to $\mathcal{T}$
  return $\mathcal{T}$

The steps are also described in Algorithm 7. As a result, the computation of the multiset of $\Delta X_i^6[0]$ for each mixture set needs $256 \cdot 24$ S-boxes for the first vector in the mixture set and $(\zeta - 1) \cdot 256 \cdot 2$ (and two multiplication lookups) for all the other vectors. Thus, each partial encryption is equivalent to

$$(256 \cdot 24 + (\zeta - 1) \cdot 256 \cdot 2)/(\zeta \cdot 256 \cdot 112) \leq 2^{-5.8}$$

seven-round encryption equivalents since $\zeta \geq 2^{16}$.
We can combine both mixture approaches above. Then, we can divide the space into $2^4 \cdot \zeta$ disjoint partitions of ζ mixtures of X^4 and X'^4 times 2^4 mixtures of X^5 and X'^5. For each partition, we have the following computations:

- For the first output set at $W^5[0]$, we have to compute at most $256 \cdot 24$ S-boxes.
- We derive a Gray code to iterate over the mixtures of X^4 and X'^4. For each update, we have to compute two S-boxes for 256 texts based on the first output set. Thus, $(\zeta - 1) \cdot 256 \cdot 2$ S-boxes.
- For each of the ζ mixtures of X^4 and X'^4, we can derive another sub-mixture set of X^5 and X'^5 of size 2^4 in total. We have already computed the $256 \cdot 24$ S-boxes for the reference pair in this mixture set of X^5 and X'^5. For each of the $(2^4 - 1)$ remaining values, we need to update a single S-box per text, i.e., $(2^4 - 1) \cdot 256 \cdot 1$ S-boxes.

This combined approach needs at most

$$1 \cdot 256 \cdot 24 + (\zeta - 1) \cdot 256 \cdot 2 + (\zeta \cdot (2^4 - 1) \cdot 256 \cdot 1) \text{ S-boxes}$$

to compute $2^4 \cdot \zeta$ multisets of $\Delta X_i^6[0]$, while $2^4 \cdot \zeta \cdot 256 \cdot 112$ S-boxes are required considering all S-boxes in seven-round AES. As a result, each partial encryption in the offline phase is equivalent to less than $2^{-6.7}$ seven-round encryptions, which is very close to the minimum of $1/112 \simeq 2^{-6.8}$ encryption equivalents if we managed to compute only a single S-box per text.