IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2018, No. 4, pp. 102–127. DOI:10.13154/tosc.v2018.i4.102-127

New Yoyo Tricks with AES-based Permutations

Dhiman Saha[1], Mostafizar Rahman[2] and Goutam Paul[2]

[1] Department of Electrical Engineering & Computer Science,
Indian Institute of Technology, Bhilai 492015, India
dhiman@iitbhilai.ac.in
[2] Cryptology and Security Research Unit (CSRU), R. C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata 700108, India
mrahman454@gmail.com,goutam.paul@isical.ac.in

Abstract. In Asiacrypt 2017, Rønjom et al. reported some interesting generic properties of SPNs, leading to what they call the Yoyo trick, and applied it to find the most efficient distinguishers on AES. In this work, we explore the Yoyo idea in distinguishing public permutations for the first time. We introduce the notion of nested zero difference pattern which extends the Yoyo idea and helps to compose it using improbable and impossible differential strategies to penetrate higher number of rounds. We devise a novel inside-out application of Yoyo which enables us to start the Yoyo game from an internal round. As an application, we investigate the AES-based public permutation AESQ used inside the authenticated cipher PAEQ. We achieve the first deterministic distinguisher of AESQ up to 8 rounds and the first 9-round distinguisher of AESQ that start from the first round with a practical complexity of around 2^{26}. We manage to augment Yoyo with improbable and impossible differentials leading to distinguishers on $9, 10, 12$ rounds with complexities of about $2^2, 2^{28}, 2^{126}$ respectively. Further, with impossible differentials and a bi-directional Yoyo strategy, we obtain a 16-round impossible differential distinguisher with a complexity of 2^{126}. Our results outperform all previous records on AESQ by a substantial margin. As another application, we apply the proposed strategies on AES in the *known-key* setting leading to one of the best 8-round known-key distinguisher with a complexity of 2^{30}. Finally, this work amplifies the scope of the Yoyo technique as a generic cryptanalysis tool.

Keywords: AES · AESQ · Distinguisher · PAEQ · Yoyo · Differential Cryptanalysis

1 Introduction

Non-random behavior of a cryptographic construction has been historically seen as a sign of an inherent weakness waiting to be exploited. In this regard, devising distinguishers forms one of the fundamental aims of a cryptanalyst since they help exhibit non-randomness. A distinguisher generally constitutes a statistical or structural property of a crypto-primitive that is not expected to occur for an *equivalent* random function. The scope of distinguishers is further amplified by the probability of their possible conversion/extension to more stronger forms of attacks like key-recovery for block ciphers or collisions for hash functions. The SHA3 competition [oST] witnessed the Zero-Sum distinguisher (introduced by Aumasson and Meier [AM09]) which was one of the most studied attacks against the internal public permutation Keccak$-f$ of SHA3 winner Keccak. On the other hand, a multitude of distinguishing attacks have been reported on AES [DR02] both in the secret-key as well as known-key setting (introduced by Knudsen and Rijmen [KR07]). The known-key paradigm is of particular interest since it enables studying a cipher as a public permutation. Moreover, as argued by Knudsen and Rijmen, non-existence of a known-key distinguisher

Received: 2018-09-01, Accepted: 2018-11-01, Published: 2018-12-13

implies non-existence of a secret-key one, making it imperative to study the former. This work aims to explore distinguishing attacks on public permutations based on AES with the motivation that results reported here might lead to stronger attacks on constructions where these permutations are deployed as an internal transformation.

We investigate an interesting cryptanalytic tool called the Yoyo game which was recently shown to be very effective in devising distinguishers [RBH17] on the block cipher standard AES. The Yoyo strategy was first reported in crypto literature by Biham et al. who used it for the cryptanalysis of SKIPJACK [BBD+99]. In the Yoyo game, new pairs of plaintexts and ciphertexts are made adaptively from the original pairs. While making new pairs a certain property is kept invariant. A common strategy is the use of zero difference in the pairs. Suppose a pair of plaintext/ciphertexts have certain zero difference after some rounds of a cipher. In the Yoyo game, it is verified whether new pairs of plaintexts/ciphertexts that are formed by swapping bytes/words of the original pairs still hold the same zero difference after the same number of rounds of the cipher. Using the Yoyo game Biryukov et al. have found a 7-round distinguisher for Feistel networks[BLP16]. In Asiacrypt 2017, Rønjom et al. applied the Yoyo game to generic Substitution-Permutation (SP) networks [RBH17] and proposed a generic 2-SP round deterministic distinguisher. As a case-study they applied the strategy on variants of AES and found many practical distinguishers on up to 5 rounds of AES. They also reported a distinguisher for 6-round AES with data complexity $2^{122.83}$ and a key recovery attack on 5-round AES with complexity 2^{31} that requires $2^{11.3}$ plaintexts/ciphertexts pairs.

The current work intends to look at the Yoyo technique as a general cryptanalysis strategy specially in the light of public permutations. In particular, we look at AESQ, the AES-based internal permutation of CAESAR [cae] round 2 candidate PAEQ [BA14]. PAEQ, along with AESQ permutation was introduced by Biryukov and Khovratovich in ISC 2014 [BK14]. There are many variants of PAEQ but across all variants, the same permutation AESQ of width 512 bits is used. The designers themselves have done a lot of cryptanalysis on PAEQ and have shown a Constrained Input Constrained Output (CICO) attack with complexity 2^{32} on 8-round AESQ. They have also proposed a 12-round distinguisher with complexity 2^{256}. Bagheri et al. have reduced the complexity of the 12-round disgtinguisher to 2^{128}[BMS16]. They have extended their work for 16-round AESQ permutation and shown a distinguisher with complexity 2^{192} using 2^{128} memory. A key recovery attack has also been devised on PAEQ targeting the diffusion of the AESQ permutation by Saha et al. [SKMC16]. They have proposed a 8-round key recovery with complexity of 2^{48}.

This work reports a family of distinguishers on AESQ which primarily capitalize on the Yoyo game. This is the first time that Yoyo based distinguishers have been devised for a public permutation. The basic Yoyo idea is augmented with other cryptanalytic principles to penetrate a higher number of rounds. In doing so, the first practical 9-round distinguisher that works from the first round is achieved. The inside-out technique is leveraged up on to reach up to 10 rounds with practical complexities and extended to 12 rounds with 2^{126} queries. Further, we introduce the idea of bi-directional Yoyo game where two Yoyo games are played in opposite directions and connected using the properties of the linear layer of AESQ. This leads to the development of a 16-round distinguisher with a complexity of 2^{126}. We summarize our results in comparison to the previous works in Table 1. As can be seen, the current work outperforms all previous results by a huge margin while requiring negligible memory. Finally, to emphasize the scope of the devised techniques we apply them to AES which under the known-key setting also behaves like a public permutation. Applying bi-directional Yoyo on AES helps devise one of the best 8-round distinguishers with complexity of 2^{30} and negligible memory requirement, as shown in Table 3.

The rest of the work is organized as follows. In Section 2, a brief overview of AESQ permutation is given. In Section 3, tools used throughout the paper are discussed. In

Section 4.1, a deterministic distinguisher for 8-round AESQ is presented. This deterministic distinguisher is extended and a 9-round probabilistic distinguisher is illustrated in Section 4.2. In Section 5, a brief overview of improbable differential and inside-out technique and their application to AESQ is given. In Section 6, impossible Yoyo distinguishers for 12 and 16-round AESQ are demonstrated. In Section 7, impossible differential Yoyo and impossible differential bi-directional Yoyo techniques is applied to round-reduced AES in known-key setting. Experimental setup and results are briefly mentioned in Section 8 and elaborated in Appendix A. Arguments in favour of the validity of the work are discussed in Section 9. The concluding remarks are furnished in Section 10.

Table 1: Distinguishers on Round-Reduced AESQ

Rnd	Complexity			Technique	Reference
	Time	Memory	Success Prob.		
8	2^{32}			CICO	Designers [BA14]
8†	1	Negligible		**Yoyo**	**Section 4.1**
9	$2^{26.08}$	Negligible	0.71		**Section 4.2**
9†	5	Negligible	0.82	**Improbable Differential Yoyo**	**Section 5.2**
10†	2^{28}	Negligible	0.77		**Section 5.2**
12†	2^{126}	Negligible	0.84	**Impossible Differential Yoyo**	**Section 6.1**
	2^{256}	2^{256}	0.61	Rebound Attack	Designers [BA14]
	2^{128}	Negligible	0.83		Bagheri et al. [BMS16]
	$2^{102.4}$	$2^{102.4}$	0.83	Time-memory Trade-off	
	$2^{128-x/4}$	2^{x}			
16†	2^{192}	2^{128}	0.83	Rebound Attack	
	2^{188}	2^{128}	0.83	Multi Ltd.-Birthday Distinguisher	
	2^{192+x}	2^{128-x}		Time-memory Trade-off	
	2^{126}	Negligible	0.84	**Impossible Differential Bi-directional Yoyo**	**Section 6.2**

† Starting from round 2

2 Description of AESQ

PAEQ is an authenticated encryption scheme. In its core, PAEQ uses the 512-bit AESQ permutation. This can be viewed as four 128-bit registers with each register running two rounds of AES where XOR-ing the subkey operation is replaced with XOR-ing a round constant. In AESQ, a state is of 64 bytes. There are four groups of 16 bytes each. We call each of them as register and named them as A,B,C and D from left to right. Each columns of the registers is 32-bit words and is numbered from 0 to 3. Like, first and last column of register A is A[0] and A[3] respectively. Two rounds AES is run for each of the registers. Then a shuffling is done among all the registers. Shuffle mapping is shown in Table 2. In original AESQ, this operation is repeated 10 times. So, AESQ permutation consists of 20 AES rounds. Fig. 1 shows a 2-Round AESQ Permutation.

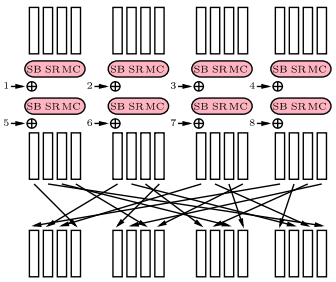


Figure 1: 2-Round AESQ Permutation

From	A[0]	A[1]	A[2]	A[3]
To	A[3]	D[3]	C[2]	B[2]
From	B[0]	B[1]	B[2]	B[3]
To	A[1]	D[1]	C[0]	B[0]
From	C[0]	C[1]	C[2]	C[3]
To	A[2]	D[2]	C[3]	B[3]
From	D[0]	D[1]	D[2]	D[3]
To	A[0]	D[0]	C[1]	B[1]

Table 2: Shuffle Table

3 Tools for the Analysis

In this section, we describe the necessary notations, definitions, and concepts that will be used in our subsequent analysis.

3.1 Notations

We enlist some notations frequently used in this work below.

$x \leftarrow y$	x gets the value of y
$\alpha \rightarrow \beta$	The transition from α to β
$\alpha \nrightarrow \beta$	α does not transit to β
$\Pr[\mathcal{E}]$	Probability of occurence of event $\mathcal{E}$
$wt(x)$	Weight of a vector x
x_i	i^{th} component of vector x
$\texttt{AESQ}_{i \rightarrow j}$	AESQ permutation from round i to round j

We, additionally, try to reuse some of the notations used in [RBH17]. So a generic permutation is assumed to be of the form of $\mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ where, $q = 2^k$ given by:

$$F(x) = S \circ L \circ S \circ L \circ S(x)$$

Here, S is considered as a large SBox to be visualized as a concatenation of smaller component SBoxes operating on $\mathbb{F}_q$. The linear layer over $\mathbb{F}_q^n$ is denoted by L. A *word* represents an element of $\mathbb{F}_q$ while the internal *state* is a vector of words $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_{n-1}) \in \mathbb{F}_q^n$. Based on this the authors in [RBH17] define the Zero Difference Pattern (ZDP) as below:

Definition 1. Zero Difference Pattern.[RBH17] Let, $\alpha \in \mathbb{F}_q^n$ for $q = 2^k$. The Zero Difference Pattern for α is

$$\nu(\alpha) = (z_0, z_1, ..., z_{n-1}),$$

where $\nu(\alpha)$ takes values in $\mathbb{F}_2^n$ and $z_i = 1$ if $\alpha_i = 0$ or $z_i = 0$ otherwise.

Interestingly, the zero-difference pattern does not consider the nature of separate words when they are non-zero and just classifies them into one category. Our aim is to look further into individual words i.e. we want to investigate the nature of α_i when $z_i = 0$. To

facilitate this, we define a *unit* as the element on which the smallest SBox of the cipher is defined. For e.g. for AES a unit is a byte. It can be noted that considering the smallest SBox a word is 8-bit, while considering the SuperSBox representation [DR06], a word is 32 bits. So, the AES state representation changes from $\mathbb{F}_{2^8}^{16}$ to $(\mathbb{F}_{2^8}^4)^4$. When a word uses multiple units, the zero difference pattern does not take into account the nature of these units and marks a word active even if at least one unit in the word is active. We want to study the activity of the units. So we introduce the notion of *Nested Zero Difference Pattern*.

Definition 2. Nested Zero Difference Pattern Let, $\alpha \in (\mathbb{F}_q^{\frac{n}{m}})^m$ for $q = 2^k$ and $\alpha_i \in \mathbb{F}_q^{\frac{n}{m}}$ and $\alpha_i = (\beta_{i0}, \beta_{i1}, \cdots, \beta_{i(\frac{n}{m}-1)})$, where β_{ij} is the unit. The Nested Zero Difference Pattern (ν^2) defined for α is

$$\nu^2(\alpha) = \{\nu^2(\alpha_0), \nu^2(\alpha_1), \cdots, \nu^2(\alpha_{n-1})\}, \quad \nu^2(\alpha_i) = (y_0, y_1, ..., y_{\frac{n}{m}-1}),$$
$$wt(\nu^2(\alpha)) = \sum_{i=0}^{n-1} wt(\nu^2(\alpha_i)),$$

where $\nu^2(\alpha_i)$ takes values in $\mathbb{F}_2^n$ and $y_i = 1$ if $\beta_{ij} = 0$ or $y_i = 0$ otherwise[1].

The following example will make things clearer.

Example 1. Here we show the different words of the AES state considering the inputs of the SuperSBox. Note that the words will change based on whether we are observing the SuperSBox input or output.

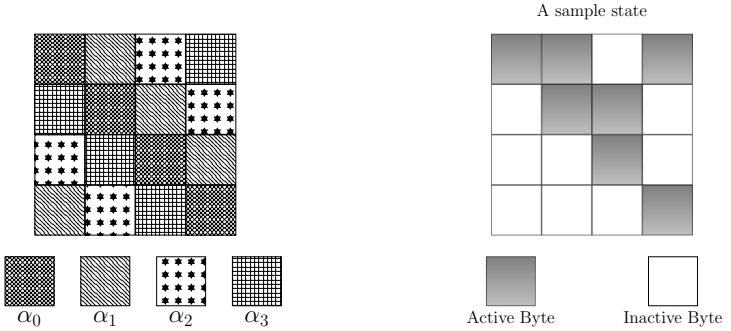


Figure 2: Different words and a sample state showing zero and non-zero bytes.

Let us consider the Zero Difference Pattern of the sample state (α) in Fig. 2:

$\nu(\alpha) = (0, 0, 1, 0)$ and $wt(\nu(\alpha)) = 1$. Thus ZDP considers only one word to be inactive. Let us now look at the Nested ZDP of the state α. It can be easily inferred that Nested ZDP gives more information pertaining to the active words. The idea of Nested ZDP will be useful when we will consider differentials over and above the Yoyo game.

$$\nu_1^2(\alpha_0) = (0,0,0,0), \quad \nu_2^2(\alpha_1) = (0,0,1,1), \quad wt(\nu^2(\alpha)) = 9.$$
$$\nu_3^2(\alpha_2) = (1,1,1,1), \quad \nu_4^2(\alpha_3) = (0,1,1,1),$$

3.2 Yoyo Analysis for Two Generic SP-Rounds

Rønjom et al. have carried out Yoyo analysis for two generic SP-rounds [RBH17]. Two generic SP-round is $G_2' = L \cdot S \cdot L \cdot S$, where L is the linear transform layer and S is the permutation layer. For simplicity, the final L layer is omitted and the modified two generic

[1] It is understood that $m|n$

SP round is denoted as $G_2 = S \cdot L \cdot S$. They have presented a deterministic distinguisher for G_2. For the explanation of the distinguisher and how it works, we have to go through some definitions originally defined in their paper. The next definition signifies how to swap between pairs of texts to form a new pair of texts.

Definition 3. [RBH17] Let, $\alpha, \beta \in \mathbb{F}_q^n$ be two states and $v \in \mathbb{F}_2^n$ be a vector, then $\rho^v(\alpha, \beta)$ is a new state in $\mathbb{F}_q^n$ created from α, β by swapping components among them. The i^{th} component of $\rho^v(\alpha, \beta)$ is defined as

$$\rho^v(\alpha, \beta)_i = \begin{cases} \alpha_i, & \text{if } v_i = 1; \\ \beta_i, & \text{if } v_i = 0. \end{cases} \tag{1}$$

The following theorem describes the deterministic distinguisher for 2 generic SP-rounds.

Theorem 1. *[RBH17] Let, $p^0, p^1 \in \mathbb{F}_q^n$, $c^0 = G_2(p^0)$ and $c^1 = G_2(p^1)$. For any vector $v \in \mathbb{F}_2^n$, $c'^0 = \rho^v(c^0, c^1)$ and $c'^1 = \rho^v(c^1, c^0)$. Then*

$$\nu(G_2^{-1}(c'^0) \oplus G_2^{-1}(c'^1)) = \nu(p'^0 \oplus p'^1) = \nu(p^0 \oplus p^1).$$

The trick is to choose any random pair of plaintexts with certain zero difference pattern and encrypt them using G_2. Then swap words/bytes between the produced ciphertexts and create a new pair of ciphertexts. Decrypt the new pair using G_2 and obtain a new pair of plaintexts. The zero difference pattern of these new pair of plaintexts will be same as the zero difference pattern of the original pairs of plaintexts. This event occurs with probability 1. This property of two generic SP-rounds can be exploited to distinguish it from a random construction.

3.3 SuperSBox and MegaSBox of `AESQ`

Let us consider round 2 and round 3 (before MixColumns) of `AESQ` permutation. We can consider the input to round 2 as 16 diagonals of 4 bytes each. In round 2, after SubBytes and ShiftRows operation each of the 16 diagonals aligns in a single column. Effect of MixColumns and shuffle operation is confined within the column. SubBytes and ShiftRows operation of round 3 dealigns the column into a inverse diagonal. From the above analysis we observe that a group of 4 bytes (a diagonal) in the input to round 2 affects only a group of 4 bytes (inverse diagonal) in the output of round 3 (before MixColumns). These operations can be grouped into a single 32-bit SBox called as SuperSBox. Therefore, round 2 and round 3 of `AESQ` permutation can be viewed as a single round with 16 parallel SuperSBoxes.

This concept can be further extended and four round `AESQ` permutation can be viewed as a single round with 128-bit MegaSBoxes [DLP+09, BA14]. We are now analysing round 2 to round 5 of `AESQ` permutation. `AESQ` state consists of 4 registers of 128 bits each. Consider four diagonals each from all the registers. After the SubBytes and ShiftRows operation each of them transforms into a column. MixColumns and adding a constant operation does not influence the other columns. Shuffling accumulates all the four columns into a single register where each of the registers undergoes two rounds of AES-like operation (round 3 and 4) which again does not influence the other registers. Shuffling disperse the columns from a single register to four registers. Round 5 SubBytes and ShiftRows operation dealigns the columns into inverse diagonals. These operations can be grouped into a single 128-bit MegaSBox and round 2 to 5 (before MixColumns) can be viewed as a single round with 4 parallel MegaSBoxes. The following MixColumns operation can be considered as mega-linear transformation on `AESQ` state (512-bits) and called as MegaMixColumns (MMC) operation. Fig. 4 shows how two rounds and four rounds of `AESQ` permutation exhibits the properties of SuperSBox and MegaSBox.

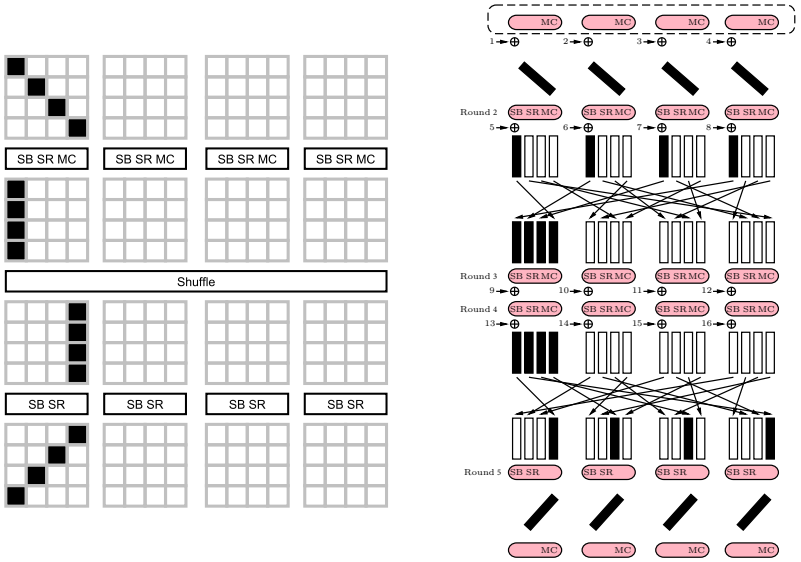


Figure 4: SuperSBox and MegaSBox [BA14] of AESQ. There are 16 SuperSBoxes covering 1.5 rounds and 4 MegaSBoxes covering 3.5 rounds

3.4 Data Complexity and Success Probability

For a distinguishing event, the data complexity and the success probability depend on the probabilities p and $p_0 = p(1+q)$ of the same event respectively in the random case and in the case of the algorithm under consideration. In [PR18], detailed analysis of various relations between data complexity of the distinguisher and the corresponding success probability is presented. We use the most general result from [PR18, Theorem 2], which involves no crude approximation such as ignoring the constant terms or assuming p, q to be small.

Surprisingly, in the existing works with which we compare our results, only the data complexity is given and none of them explicitly mention the success probability. We have computed their success probabilities and we find that at the same success probability, we achieve much lower complexities. Moreover, for our new distinguishers, we explicitly mention the success probabilities along with the reported complexities.

4 Distinguishers using Direct Yoyo on AESQ

In order to adapt the Yoyo trick on AESQ, we need to first identify the $S \circ L \circ S$ construction embedded in the permutation. To do that one has to recall the notion of MegaSBox whereby 3.5 rounds[2] of AESQ starting from an even round can be depicted as independent computations of four 128-bit words (Ref. Fig. 4). These four MegaSBoxes constitute the first S layer of the generic SPN. The subsequent MegaMixColumns corresponds to L layer while the next iteration of four MegaSBoxes represent the last S layer thereby completing the $S \circ L \circ S$ sequence. Fig. 4 shows this construction starting from Round-2. So two generic SP-rounds map to 8 rounds of AESQ without the last MMC. So, the Yoyo distinguisher pertaining to two generic SP-rounds as discussed above directly applies to $\texttt{AESQ}_{2\to 9}$. In the next subsection we work out the details of this distinguisher which is the first *deterministic* 8-round distinguisher for the AESQ permutation.

[2]Without the MegaMixColumns of the last round

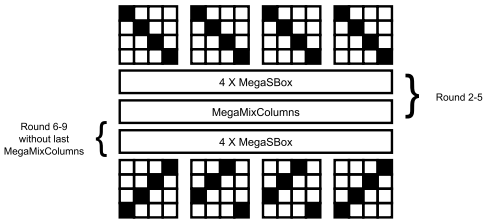


Figure 5: $\texttt{AESQ}_{2\to 9}$ as an $S \circ L \circ S$ construction.

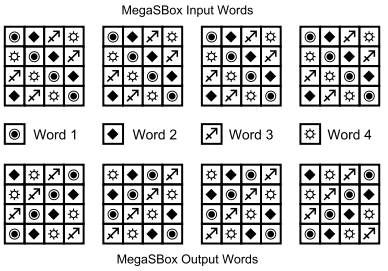


Figure 6: Word configuration for each MegaSBox

4.1 Distinguisher for 8 Rounds

Let us first look at the Yoyo game for AESQ which we will call as a subroutine for the distinguishing algorithm. The Yoyo game shown in Algorithm 1 is tailored w.r.t $\texttt{AESQ}_{i\to j}$ but will be analogous for any corresponding random permutation. The procedure is self-explanatory except for two things:

- The function MSwap is used to swap words between two states of AESQ. Apart from the states it accepts an argument DIRECTION which decides whether input or output words (Ref. Fig. 4) will be swapped. So, if DIRECTION = FORWARD, then output words will be swapped while for DIRECTION = BACKWARD, it will be done in accordance with input word pattern. This distinction takes into account the direction in which the game is being played. As will be seen later, we will need to play the game in the backward direction to penetrate a higher number of rounds. Moreover, MSwap can, at random, swap any one, two or three words of the states. As shown by the authors of [RBH17], all such word-swap configurations are equivalent and preserve the properties of the Yoyo game.
- The argument Mode is used to play either half or full of the YOYO game and respectively receives values MID or FULL. Later, in this work we will show how output of half of the game can be used to generate input states that help to distinguish up to 16 rounds of AESQ.

Algorithm 1 Yoyo Game for AESQ

procedure YOYO($p_1, p_2, \texttt{AESQ}_{i\to j}, \texttt{Mode}, \texttt{DIRECTION}$)
  $\{c_1 \leftarrow \texttt{AESQ}_{i\to j}(p_1); \quad c_2 \leftarrow \texttt{AESQ}_{i\to j}(p_2)$
  $(c_1', c_2') \leftarrow \texttt{MSWAP}(c_1, c_2, \texttt{DIRECTION})$ ▷ DIRECTION $\in$ {FORWARD, BACKWARD}
  if Mode = MID **then**
    return (c_1', c_2')
  else if Mode = FULL **then**
    $\{p_1' \leftarrow \texttt{AESQ}^{-1}_{i\to j}(c_1'); \quad p_2' \leftarrow \texttt{AESQ}^{-1}_{i\to j}(c_2')$
  end if
  return (p_1', p_2')
end procedure

With the Yoyo game in place, the 8-round distinguisher that uses it, is straightforward as shown in Algorithm 2. The distinguisher accepts a permutation PERMUTE. It chooses inputs p_1 and p_2 such that $\alpha = p_1 \oplus p_2$ has a particular ZDP (of weight at least one and at most three), say $\nu(\alpha) = (1, 0, 1, 0)$. Then it plays the Yoyo game generating two new

inputs p'_1 and p'_2 with $\Delta = p'_1 \oplus p'_2$. If $\texttt{PERMUTE} = \texttt{AESQ}_{2\to 9}$, it is ensured that ZDP of α is same as that of Δ.

Algorithm 2 Distinguisher for $\texttt{AESQ}_{2\to 9}$

Output: 1 for AESQ, -1 otherwise ▷ 8-Round AESQ without last MMC

1: **procedure** DistYoyo(PERMUTE)
2: $p_1, p_2 \leftarrow \left\{(m_1, m_2) : 3 \geq wt(\nu(m_1 \oplus m_2)) > 0\right\}$ ▷ At least one word active
3: $p'_1, p'_2 \leftarrow \texttt{Yoyo}(p_1, p_2, \texttt{PERMUTE}, \texttt{FULL}, \texttt{FORWARD})$
4: **if** $\nu(p'_1 \oplus p'_2) = \nu(p_1 \oplus p_2)$ **then** ▷ Holds for $\texttt{AESQ}_{2\to 9}$ deterministically
5: **return** 1
6: **else**
7: **return** -1
8: **end if**
9: **end procedure**

The pictorial description is captured by Fig. 7. It is intentionally shown that the Nested ZDP might differ which will definitely happen probabilistically. This is because the Yoyo principle guarantees that the ZDP will be preserved but has no claim on the activity pattern inside the active words. In the next subsection we will show how assuming a particular Nested ZDP enables us to extend the distinguisher to include the first round making it the first 9-round AESQ result that starts from round one.

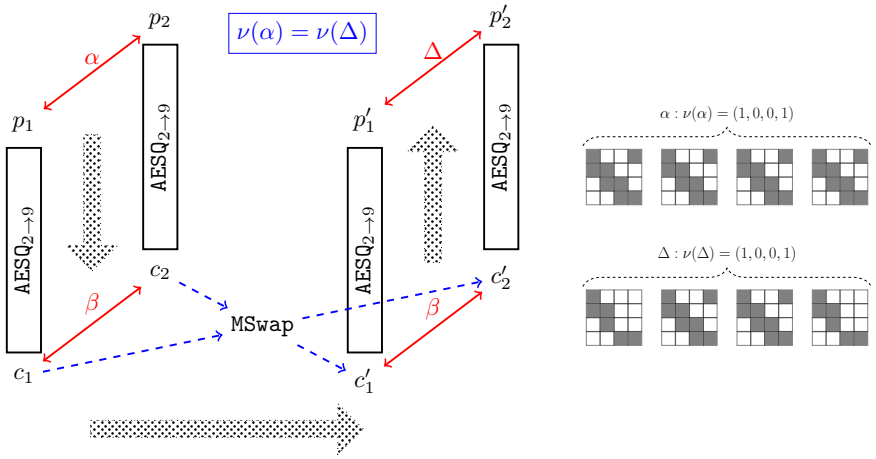


Figure 7: Deterministic 8-round Yoyo distinguisher. ZDP of input difference α is preserved by output difference Δ of the Yoyo game. A possible configuration of (α, Δ) is shown. Note that though $\nu(\alpha) = \nu(\Delta), wt(\nu^2(\alpha)) \neq wt(\nu^2(\Delta))$. We will use this kind of Nested ZDP to extend the Yoyo beyond 8 rounds.

4.2 Extension to 9-round AESQ

The inclusion of the first round relies on the notion of Nested ZDP that we introduced earlier. The basic idea is to:

- First leverage on the determinism of the 8-round Yoyo while imposing some restriction on both the input and output Nested ZDP, thereby making it probabilistic. If the input and output differences of the Yoyo are α and η respectively, then the restrictions are of the form:

1. Input: Exactly one byte active in one word.

$$wt(\nu(\alpha)) = 3 \text{ and } \exists i: \ wt(\nu^2(\alpha_i)) = 15.$$

2. Output: Exactly one byte *inactive* in one word.

$$wt(\nu(\eta)) \overset{\texttt{Yoyo}}{=} 3 \text{ and } \exists i: \ wt(\nu^2(\eta_i)) = 1.$$

Now, the probability that α leads to η is $\left[16 \times 2^{-8} \times (\frac{255}{256})^{15}\right] \approx 2^{-4.08}$.

- The second step is to find a one-round differential to connect with the input difference α. This is standard[3] and we can, with a probability 2^{-22}, find an input difference that conforms to the input restriction for the Yoyo game. This leads to the inclusion of round one in the forward direction.

- The last step is to include the round in the backward direction. One can easily note that if the output restriction is satisfied then, the extra round in the return path will automatically lead to a state that has four inactive bytes. More precisely, if the last difference is denoted by Δ then we have $\exists i: \ wt(\nu^2(\Delta_i)) = 4$. Also, the inactive bytes will belong to same diagonal due to the last inverse ShiftRows operation of Round 1.

Fig. 8 gives an overview of the entire extension strategy and also depicts a particular configuration of states that conform to the above statements while Algorithm 3 illustrates the distinguishing procedure.

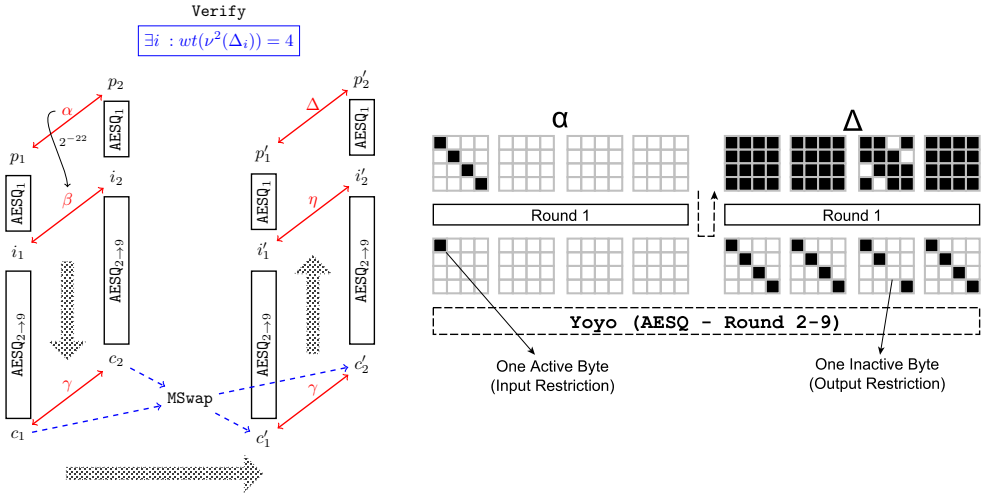


Figure 8: Probabilistic 9-round Yoyo distinguisher. A configuration of (α, Δ) is shown.

Now, the final probability $\Pr\left[\exists i: \ wt(\nu^2(\Delta_i)) = 4\right]$ for $\texttt{AESQ}_{1\to9}$ and a corresponding random permutation $\mathcal{R}$ is given by $p_0 = 2^{-26.08}$ and $p = 16 \times 2^{-32} \times (\frac{255}{256})^{60} \approx 2^{-28.34}$ respectively, giving us a success rate of 71% at the data complexity of $2^{26.08}$.

In the next section, we introduce the notion of improbable differential Yoyo whereby we try to compose the Yoyo game with improbable differentials capitalizing on the *inside-out* strategy.

[3]Recall, the classical $4 \to 1$ transition though AES MixColumns.

Algorithm 3 Distinguisher for $\texttt{AESQ}_{1\to 9}$

Output: 1 for AESQ, -1 otherwise ▷ 9-Round AESQ without last MMC

procedure ImpDistYoyo(PERMUTE)
  count $\leftarrow 0$
  while count $\leq 2^{26.08}$ **do**
    $i_1, i_2 \leftarrow \Big\{(m_1, m_2) : \alpha = m_1 \oplus m_2; wt(\nu(\alpha)) = 3 \wedge \exists i : \ wt(\nu^2(\alpha_i)) = 12\Big\}$
      ▷ All four *active* bytes should belong to the same diagonal
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\Big(i_1, i_2, \texttt{PERMUTE}, \texttt{FULL}, \texttt{FORWARD}\Big)$
    if $\exists i : \ wt(\nu^2(\alpha_i)) = 4$ **then return** 1
      ▷ All four *inactive* bytes should belong to the same diagonal
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return -1
end procedure

5 Improbable Differential Yoyo

In Indocrypt 2010, Tezcan introduced the notion of improbable differential cryptanalysis [Tez10]. The idea is to find a differential which is less probable for a given permutation (say, $\mathcal{P}$) in comparison to a random permutation (say, $\mathcal{R}$). So if $Pr_{\mathcal{P}}(\Delta_{in} \to \Delta_{out}) = p_0$ while $Pr_{\mathcal{R}}(\Delta_{in} \to \Delta_{out}) = p$, then for the *improbability* criteria we must have $p_0 < p$, where $Pr_{\mathcal{P}}$ and $Pr_{\mathcal{R}}$, represent the probabilities of the input difference Δ_{in} and output difference Δ_{out} to occur for $\mathcal{P}$ and $\mathcal{R}$ respectively. Tezcan argued that improbability led to the well-known *impossibility* criteria where $p_0 = 0$. He further proposed an idea known as the *expansion technique* [Tez10] to devise improbable differential by connecting (multiple) differentials with an impossible differential. The expansion technique is briefly stated below:

$Pr_{\mathcal{P}}(\Delta_{mid} \to \Delta_{out}) = 0$ Impossible Differential
$Pr_{\mathcal{P}}(\Delta_{in} \to \Delta_{mid}) = p'$ Connecting Differential
Let $Pr_{\mathcal{R}}(\Delta_{in} \to \Delta_{out}) = p$
Let $Pr_{\mathcal{P}}(\Delta_{in} \to \Delta_{out}) = p_0$ if $\Delta_{in} \not\to \Delta_{mid}$

Then probability of the *improbable* differential for a given permutation $\mathcal{P}$ is given by:

$$Pr_{\mathcal{P}}(\Delta_{in} \to \Delta_{out}) = p_0 = (1 - p') \times p + p' \times 0 = (1 - p')p \tag{2}$$

Our idea is to:

- First, use our notion of *Nested ZDP* with properties of the MixColumns to devise an impossible differential.
- Then, we find a differential that connects the starting ZDP of the Yoyo game with input difference of the impossible differential.

Overall, by virtue of the expansion technique, we are able to devise an improbable differential leading to higher number of rounds than that covered by the Yoyo game. In order to do this, we will use the *inside-out* technique described next.

5.1 The Inside-Out Technique

The inside-out technique has been used extensively in distinguishing public permutations like the Keccak-f permutation [AM09], whereby the idea is to start from an intermediate state to generate a set of inverted initial states that preserve a certain property (e.g. the zero-sum property in case of Keccak). Here we try to adapt the technique to incorporate the Yoyo game. The idea is as follows:

- Play the Yoyo game from an intermediate round to generate pairs of input states.
- In the return path of the Yoyo game, we extend the number of rounds using an improbable/impossible differential.

In the next subsection we will show different distinguishers based on the following claim which leads to impossible differences at various rounds of `AESQ`.

Claim 5.1 (Impossible Difference)**.** *Let the input difference before the r^{th} round (where r denotes an odd round) MegaMixColumns be δ and let exactly one word of δ (say δ_i) be active i.e. $wt(\nu(\delta)) = 3$, then the following will hold*

1. *If $wt(\nu^2(\delta_i)) = 0$, then all 64 SBoxes of the state will be active before $(r+1)^{th}$ round Mega-MixColumns.*
2. *If $wt(\nu^2(\delta_i)) = 0$, then all 16 SuperSBoxes of the state will be active before $(r+2)^{th}$ round Mega-MixColumns.*
3. *Unconditionally, all 4 MegaSBoxes of the state will be active before $(r+4)^{th}$ round Mega-MixColumns.*

Proof. The proof proceeds as below:

1. $wt(\nu(\delta)) = 3$ and $wt(\nu^2(\delta_i) = 0)$ implies that every byte in word δ_i is active. Since only one word is active, we have a byte active in every column. The word-configuration will be in accordance with MegaSBox output words (See Fig. 4). Now, the claim follows directly from the property of `AES` MixColumns. *It is well known that the total number of active bytes before and after a MixColumns operation cannot be less than 5.* Since, the input to every MixColumns operation in the r^{th} round has 1 byte active, then the output has all 4 bytes of the column active with a probability 1. So the entire state is active after r^{th} round MMC. Thus in the $(r+1)^{th}$ round, all SBoxes of the state have non-zero input difference and hence cannot have a zero output difference and hence will be all active deterministically up to the next MMC operation. If the output difference is denoted by η, then
$$\Pr\Big[wt(\nu^2(\eta)) = 1\Big] = 0 \qquad \leftarrow \text{Impossible Difference}$$
2. The argument remains the same with the only difference that after the r^{th} round MMC all 16 SuperSBoxes are activated that span 1.5 rounds ending just before $(r+1)^{th}$ round MMC. Since, all SuperSBoxes are active, it is impossible to have a zero-difference at the output of any of them.
3. For MegaSBox, the restriction $wt(\nu^2(\delta_i)) = 0$ is no longer required. So, we only need $wt(\nu(\delta)) = 3$. Irrespective of the status of the bytes inside the word, after r^{th} round MMC, all 4 MegaSBoxes will be activated and will span 3.5 rounds. Thus, at the end of $(r+4)^{th}$ round MMC, we cannot have the case that any of the words signifying the output of the MegaSBoxes is inactive. If the output difference is denoted by γ, then
$$\Pr\Big[wt(\nu(\gamma)) = 1\Big] = 0 \qquad \leftarrow \text{Impossible Difference}$$

□

The implications of Claim 5.1 are captured by Fig. 9 which shows the particular case of $r = 9$ relevant for this work. So, we have three impossible differentials covering 10, 10 to 11 and 10 to 13 rounds respectively. We next show how we convert the first two into improbable differentials to get a 9-round and 10-round distinguisher with practical complexities. Later, using the third one, we will devise an impossible differential distinguisher for 12 rounds.

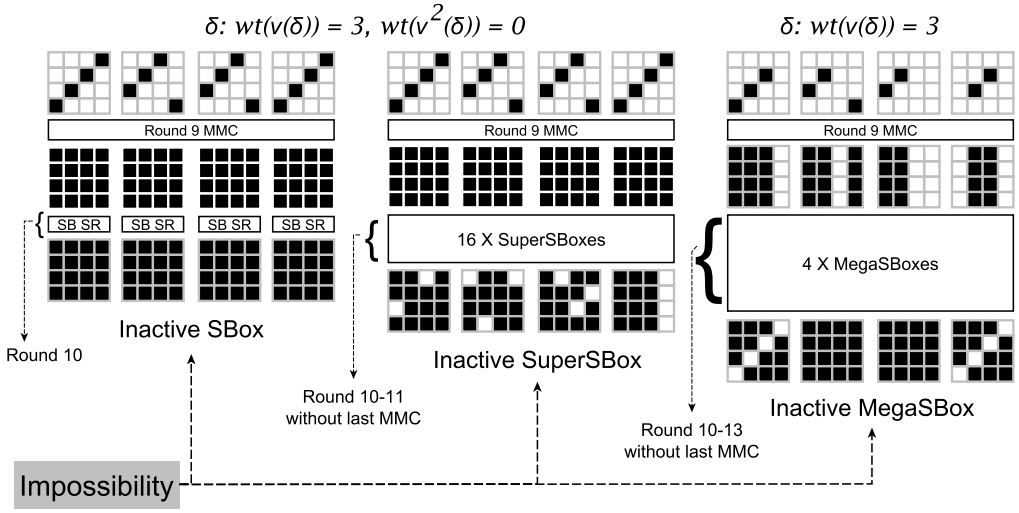


Figure 9: Different State Configurations Conforming to Claim 5.1

5.2 Improbable Differential Yoyo Distinguisher for 9-round and 10-round AESQ

As per the requirement of the expansion technique explained earlier, we need an impossible differential and a *connecting* differential that conforms to its input. Now, Claim 5.1 (1,2) already gives us the impossible differential. We are interested particularly for the case when $r = 9$. So, for $r = 9$, we have 1 and 2 round impossible differential (without last MMC as shown in Fig. 9). We now use the Yoyo game to generate the connecting differential. The strategy is demonstrated in Fig. 10. The probabilities can be derived as below:

$$\alpha : wt(\nu(\alpha)) = 3$$
$$\Pr[wt(\nu(\delta)) = 3] = 1 \quad [\because \nu(\alpha) \overset{\text{Yoyo}}{=} \nu(\delta)]$$
$$\Pr[wt(\nu(\delta)) = 3 \wedge wt(\nu^2(\delta_i)) = 0] = \left(\frac{255}{256}\right)^{16} [\delta_i \leftarrow \text{Only active word}]$$

So, the technique is to use the Yoyo game to generate an "arbitrary" number of inputs pairs (p'_1, p'_2) such that the output difference of these pairs over $\texttt{AESQ}_{2\to 10}$ or $\texttt{AESQ}_{2\to 11}$ can never have an inactive SBox or SuperSBox respectively. To ascertain the data complexity one needs to find the probabilities of these events occurring for a random permutation. In the next subsection we find the data-complexity.

We directly use Eq. (2) to derive the probability of the combined differential. For the 9-round attack, the probability of observing at least one inactive SBox for a random permutation is $1 - \left(\frac{255}{256}\right)^{64} \approx 0.22$. Similarly, for 10 rounds the probability of observing

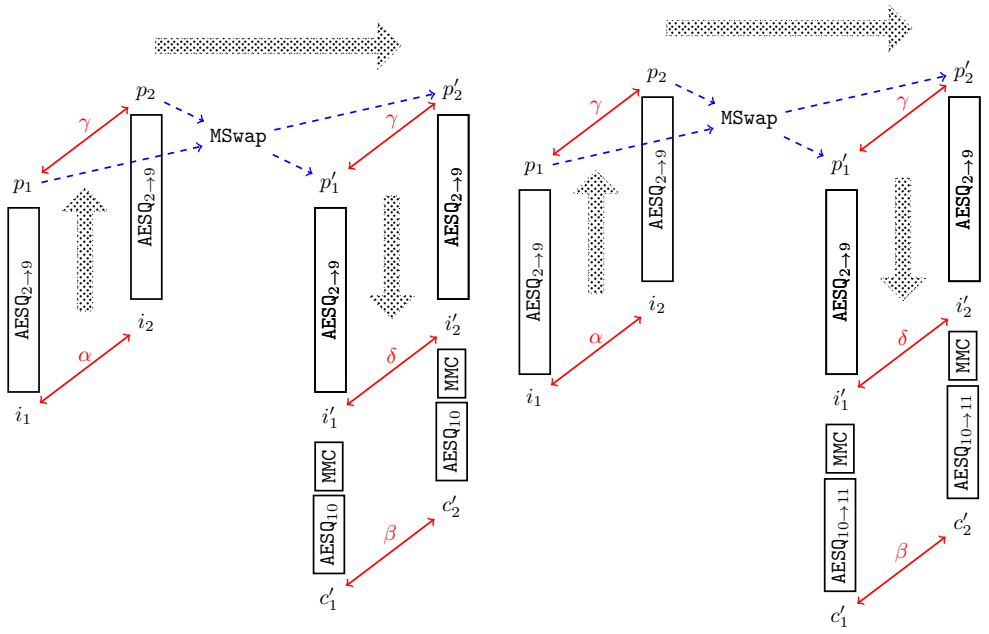


Figure 10: Improbable Differential Yoyo distinguisher for 9/10-round AESQ

an inactive SuperSBox is $p = 16 \times \frac{1}{2^{32}} \times \left(\frac{2^{32}-1}{2^{32}}\right)^{15} \approx 2^{-28}$. As per notations of Eq. (2), we have:

- For 9 rounds

$$p \approx 0.22$$
$$p' = \left(\frac{255}{256}\right)^{16} \approx 0.94$$
$$p_0 = (1-p')p \approx 0.0132$$

- For 10 rounds

$$p \approx 2^{-28};$$
$$p' = \left(\frac{255}{256}\right)^{16} \approx 0.94$$
$$p_0 = (1-p')p \approx 0.06 \times 2^{-28}$$

Thus the numbers of input pairs needed to distinguish $\texttt{AESQ}_{2\to10}$ and $\texttt{AESQ}_{2\to11}$ are around $\frac{1}{0.22} \approx 5$ and 2^{28} respectively with a success probability of 82% and 77% respectively, thereby leading to the first practical distinguishers for these rounds of AESQ. For merely 5 samples in case of 9 rounds, the Normal approximation used in [PR18, Theorem 2] does not hold and so we perform direct calculation of false positive and false negative errors in computing the theoretical estimate of the success probability. Algorithm 4 captures both 9/10 round distinguishers at the same time. In the next section, we introduce the notion of impossible differential Yoyo in the inside-out setting. Based on that we develop two distinguishers on 12 and 16 rounds of AESQ.

6 Impossible Differential Yoyo

Impossible differential has been shown to be a special case of improbable differential [Tez14, TS16]. Now it is easy to note that if the connecting differential used in the expansion technique occurs with a probability 1, then the combined differential becomes impossible. The basic idea is to use the determinism of the Yoyo game along with the inside-out technique to arrive at the input of an impossible differential. We do this in two ways: The first way leverages upon the third part of Claim 5.1. The second way tries to combine two Yoyo games in two directions.

Algorithm 4 Distinguisher for $\texttt{AESQ}_{2\to10}/\texttt{AESQ}_{2\to11}$

Output: 1 for AESQ, -1 otherwise ▷ 9/10-Round AESQ without last MMC

procedure ImprDistYoyo(PERMUTE, RNDS)
  if RNDS $= 9$ **then**
    CMPLXTY $\leftarrow 5$
    COND $\leftarrow wt(\nu^2(c'_1 \oplus c'_2)) > 0$
  else
    CMPLXTY $\leftarrow 2^{28}$
    COND $\leftarrow wt(\nu^2(c'_1 \oplus c'_2)) = 4$ ▷ Inactive bytes ∈ same SuperSBox
  end if
  count $\leftarrow 0$
  while count $\leq \lceil$CMPLXTY$\rceil$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{MID}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1); \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    if COND $=$ TRUE **then return** -1
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return 1
end procedure

6.1 Impossible Differential Yoyo Distinguisher for 12-round AESQ

This attack is similar to the ones described in the previous section with only difference that we no longer have restriction on the connecting differential due to Claim 5.1 (3). So what we have is an impossible differential spanning 3.5 rounds due to the MegaSBox and the connecting differential that hold with probability 1 due to the Yoyo game. Fig. 11a illustrates the strategy while the procedural details are covered by Algorithm 5.

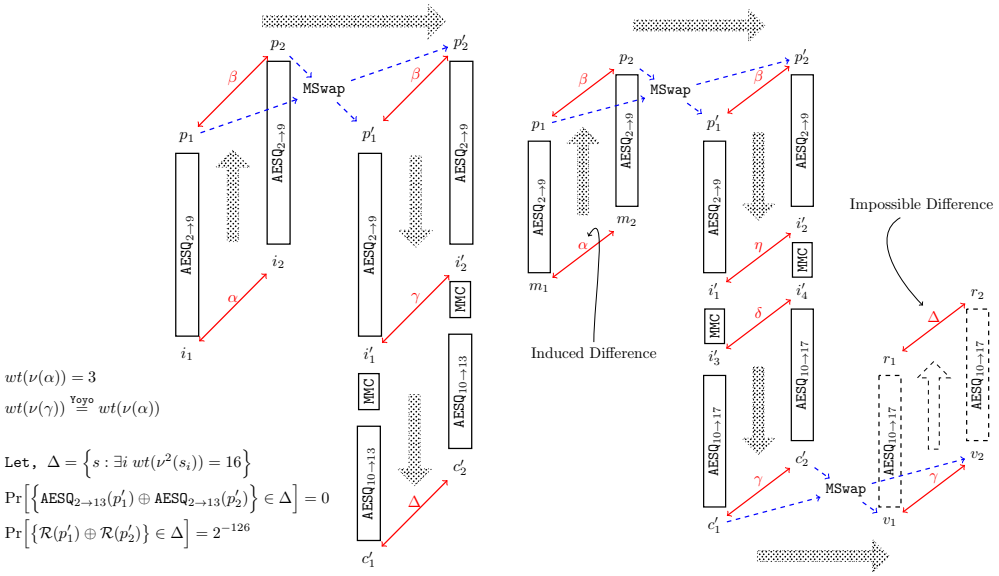
Algorithm 5 Distinguisher for $\texttt{AESQ}_{2\to13}$

Output: 1 for AESQ, -1 otherwise ▷ 12-Round AESQ without last MMC

procedure ImpDistYoyo(PERMUTE)
  count $\leftarrow 0$
  while count $\leq 2^{126}$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{MID}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1) \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    if $wt(\nu(c'_1 \oplus c'_2)) = 1$ **then return** -1 ▷ Impossible difference for $\texttt{AESQ}_{2\to13}$
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return 1
end procedure

The probability that any one of the words corresponding to the MegaSBoxes is active

for a random permutation is $p = \frac{4}{2^{128}}$. Further, we have $p' = 1$ and therefore $p_0 = 0$, resulting in a data complexity of 2^{126} with a success probability of 84%. Next we show an interesting way to combine two Yoyo games to come up with a 16-round distinguisher starting from round 2.



(a) Impossible Differential Yoyo **(b)** Impossible Differential Bi-directional Yoyo

Figure 11: Impossible Differential Yoyo Distinguishers on $\texttt{AESQ}_{2\to13}$ and $\texttt{AESQ}_{2\to17}$

6.2 Impossible Differential Bi-directional Yoyo Distinguisher for 16-round AESQ

As the name suggests, *bi-directional* Yoyo combines two Yoyo games. These games are played in opposite directions and employ the inside-out strategy. The ZDP requirements of the two games are different as stated below:

- Game 1 is played with $\texttt{AESQ}^{-1}_{2\to9}$ without last MMC. Only one word should be active in the input differential. So the weight of ZDP needs to be 3.

  $$i'_1, i'_2 \leftarrow \texttt{Yoyo}\Big(m_1, m_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{FULL}, \texttt{BACKWARD}\Big) : wt(\nu(m_1 \oplus m_2)) = 3$$

- Game 2 is played with $\texttt{AESQ}_{10\to17}$ without last MMC. All words should be active in the input differential. So the weight of ZDP needs to be 0.

  $$r_1, r_2 \leftarrow \texttt{Yoyo}\Big(i'_3, i'_4, \texttt{AESQ}_{10\to17}, \texttt{FULL}, \texttt{FORWARD}\Big) : wt(\nu(i'_3 \oplus i'_4)) = 0$$

- In order to connect Game 1 and Game 2, we will use an MMC operation. One can visualize this as the MMC of Round 9 which is excluded while playing Game 1. So the claim is as follows:

  Claim 6.1. *If* $i'_3 = \text{MMC}(i'_1)$ *and* $i'_4 = \text{MMC}(i'_2)$, *then*

  1. $wt(\nu(i'_3 \oplus i'_4)) = 0$ *and*
  2. $\Pr\big[wt(\nu(r_1 \oplus r_2)) = 1\big] = 0$

Proof. The first claim follows from the fact that due to Game 1, $wt(\nu(i'_1 \oplus i'_2)) = 3$. So, we have exactly one word active in $(i'_1 \oplus i'_2)$. This also implies that due to the word configuration (Recall Fig. 4) we can have exactly one byte active in each column of $(i'_1 \oplus i'_2)$. Due to the property of MixColumns, every single active byte in $(i'_1 \oplus i'_2)$ will lead to a fully (all four bytes) active column in $(i'_3 \oplus i'_4)$. Since, the minimum number of active bytes in $(i'_1 \oplus i'_2)$ is one, so after after MMC on i'_1 and i'_2, we will have at least one column active in $(i'_3 \oplus i'_4)$. Now, as each byte in the active column belongs to a different word, so an active column implies four active words i.e. $wt(\nu(i'_3 \oplus i'_4)) = 0$.

The second claim can be easily inferred from Game 2. Since, the input difference of Game 2 has four active words, so we cannot have an *inactive* word in the output difference $r_1 \oplus r_2$. □

The entire bi-directional game is captured by Fig. 11b. Once Game 1 and Game 2 are connected, we can appreciate the fact that the combination of the second half of Game 1, the connecting MMC layer and the first half of Game 2 actually behaves like $\texttt{AESQ}_{2\to17}$ without the last MMC. This leads us in the direction of the distinguishing strategy described in Algorithm 6. So one can arbitrarily generate pairs of inputs for 16 round AESQ starting from round 2 excluding last MMC. The corresponding outputs under MSwap when subjected to $\texttt{AESQ}^{-1}_{10\to17}$ without Round 10 MMC can never lead to output difference having one *inactive* word. For a random permutation this happens with a probability of 2^{-126}. So, the data complexity and the success probability remain the same as the 12-round distinguisher.

Algorithm 6 Distinguisher for $\texttt{AESQ}_{2\to17}$

Output: 1 for AESQ, -1 otherwise ▷ 16-Round AESQ without last MMC

procedure ImpDistBiYoyo(PERMUTE)
  count $\leftarrow 0$
  while count $\leq 2^{126}$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{MID}, \texttt{BACKWARD}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1) \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    $(v_1, v_2) \leftarrow \texttt{MSwap}(c'_1, c'_2, \texttt{FORWARD})$ ▷ Excludes possibility of trivial extension
    $\begin{cases} r_1 \leftarrow \texttt{AESQ}^{-1}_{10\to17}(v_1) \\ r_2 \leftarrow \texttt{AESQ}^{-1}_{10\to17}(v_2) \end{cases}$ ▷ 9-Round AESQ without last MMC
    if $wt(\nu(r_1 \oplus r_2)) = 1$ **then** ▷ Impossible difference for $\texttt{AESQ}_{2\to17}$
      return -1
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return 1
end procedure

In the next section, we investigate the Known-Key security of AES in the light of the impossible differential Yoyo strategies developed above.

7 Applications to AES in the Known-Key Setting

Rønjom et al. have already shown application of Yoyo on AES in the secret key paradigm and argued that the maximum penetration was up to 6 rounds. In contrast, here we are more interested in public permutations which is motivated by our need to engage strategies like inside-out and start-in-the-middle which are implicitly inhibited in the secret-key setting. So an obvious direction would be to look at the known-key notion under which AES behaves as a public permutation and which opens up the avenue to expose AES to our extension strategies. As suggested, *known-key* refers to the scenario where the attacker has access to the key. Introduced by Knudsen and Rijmen [KR07] in Asiacrypt 2007, the idea was mainly motivated by the fact that non-existence of known-key distinguishers would imply non-existence of secret-key ones. Additionally, since block-ciphers are often used as primitives in hash functions where key-input could be totally or partially controllable, such kind of known-key analysis is imperative. The known-key security of block ciphers has received a lot of attention with Andreeva et al. attempting to formalize it first [ABM13] and later being systematically treated by Mennink and Preneel [MP15] in the context of hash functions. Below, we explore how some of the techniques introduced so far adapt to AES in this setting. In the process, we are able to device the one of the *most efficient 8-round known-key distinguisher* in terms of overall cost. It is assumed that the reader is familiar with AES and the notations used here are analogous to AESQ. The basic approach, as also taken in [RBH17] and earlier in this work for AESQ, is to capitalize on the well-known AES SuperSBox.

Impossible Differential Yoyo for 6-round AES The first idea is to apply the basic impossible differential Yoyo technique described in Section 6.1. So we use the inside-out philosophy to devise a connecting differential as per the last part of Claim 5.1 which is easily adapted to be applicable on AES. So we initiate the Yoyo game such that weight of ZDP is three. By virtue of the game, we get back the same ZDP at the end of 3.5 rounds. Now due to MixColumns (MC) of fourth round, all SuperSBoxes get activated. Thus, propagating forward for two rounds, due to the SuperSBox property, we cannot have the case, that the output difference has at least one *inactive* SuperSBox. The same for a random permutation would occur with a probability of 2^{-30}.

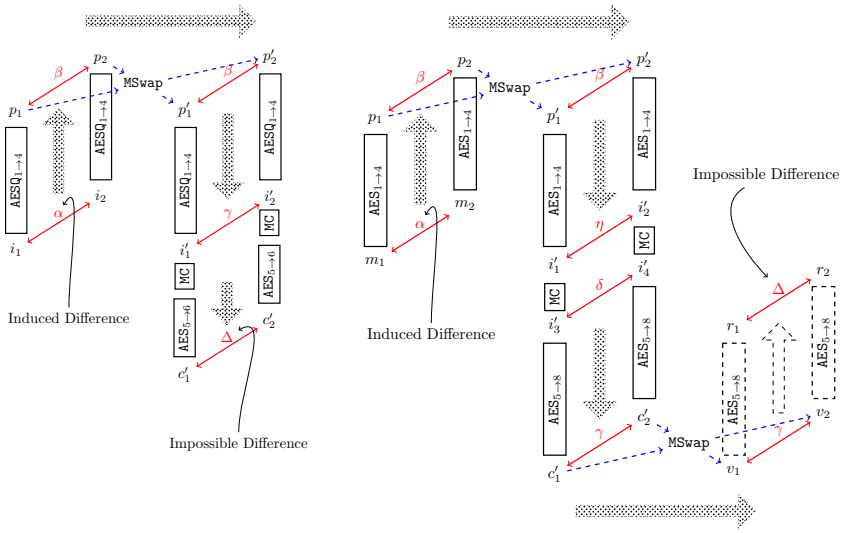


Figure 12: Impossible Differential Yoyo based Known-Key distinguisher for 6/8-round AES

Impossible Differential Bi-directional Yoyo for 8-round AES The bi-directional Yoyo trick introduced in the last section extends easily to the known-key model of AES. Since a single $S \circ L \circ S$ instance covers 4 rounds barring the last MixColumns (MC), two back-to-back Yoyo games with MC in between extends the attack to 8 rounds. As argued earlier, since the same impossible differential is used here, we are able to devise an 8-round known-key distinguisher with a complexity of 2^{30} with negligible memory. Fig. 12 depicts both the 6 and 8 round distinguishers.

Since, the introduction of known-key model, AES, in particular, has been analyzed extensively. Below we look at the state-of-the-art in devising 8-round known-key distinguishers on AES. Our inclination to 8 rounds stems from the urge to make a direct comparison with the maximum rounds we are able to penetrate here. This is captured in Table 3, where all the complexities (including ours) correspond to a success probability of 84%. Grassi and Rechberger [GR17] provide a near exhaustive analysis of known-key distinguishers while improving most of the available ones and also reporting new ones. Their main contribution was to show that the idea proposed by Gilbert [Gil14] is not limited to 10 rounds and can be further extended to 12 rounds. Going back to results on 8 rounds, we can see from Table 3 that the result reported here is only superseded by the extended multiple differential trail attack by Grassi and Rechberger while incurring some extra memory complexity.

Table 3: 8-round Known-Key Distinguishers on AES

Time Complexity	Memory Complexity	Property	Reference
2^{64}	2^{64}	Uniform Distribution	[Gil14]
2^{48}	2^{32}	Differential Trail	[GP10]
2^{44}	2^{32}	Multiple Differential Trail	[JNP13]
2^{30}	**negligible**	**Impossible Differential Bi-directional Yoyo**	Section 7
2^{23}	2^{16}	Extended 7-Round Multiple Differential Trail	[GR17]

8 Practical Verification

All distinguishers having practical complexities, i.e., $\mathtt{AESQ}_{2\to 9}$, $\mathtt{AESQ}_{1\to 9}$, $\mathtt{AESQ}_{2\to 10}$, $\mathtt{AESQ}_{2\to 11}$, $\mathtt{AES}_{1-6}$ and $\mathtt{AES}_{1-8}$ were implemented on a Java based hyper-threaded environment and verified to be conforming to the expected results. The details are furnished in Appendix A. The experimental details of the success probabilities computed for the distinguishers are provided in Appendix A.1. We now provide a discussion on all the distinguishing strategies introduced in this work.

9 Discussion

Distinguishing public permutations has always been seen as tricky due to the *unkeyed* nature of these crypto primitives. Two important things that are needed to be ensured to make a distinguisher in this setting *meaningful* are *non-triviality* and *randomness*.

A distinguisher should not be trivial in the sense that it should not be trivially extendible meaning that it is not supposed to work for any arbitrary number of rounds. Let us now discuss all distinguishers presented here in the light of this intended property. It is easily noticeable that the limitation of the Yoyo principle to hold only for $(S \circ L \circ S)$ is the first

line of defence against non-triviality. Thus rounds covered based on only a single Yoyo game cannot be extended beyond any 8 rounds of AESQ excluding last MMC while starting from an even round. The same is true for 4 rounds of AES without the last MC. As regards the strategies that were composed with Yoyo, they mostly rely on differentials that work over certain specified rounds and hence are not arbitrarily extendible. The only exception comes with the bi-directional Yoyo distinguisher where the last verification might seem non-standard. However, the non-triviality is ensured by the last MSwap operation (for example, Step 7 of Algorithm 6 for AESQ). Without that operation the distinguisher would be trivial because one could append any number of rounds as a part of first half of the second Yoyo game and invert the *same* number of rounds in the verification step.

The requirement of randomness is fundamental to devising distinguishers in general and for public permutations in particular. This is primarily because due to the unkeyed nature one could easily enumerate the permutation and employ the inverse to have a *trivial* verification. The distinguishing strategy should allow in principle sufficient randomness in choosing the inputs. In this respect, all distinguishers developed in the current work allow for that. Most of the distinguishers use first half of the Yoyo game as a subroutine and can generate almost arbitrary number of inputs which conform to certain input differences. These inputs lead to certain required differences in the middle either deterministically by virtue of the Yoyo technique or probabilistically augmenting Yoyo with probable, improbable or impossible differentials.

Table 4: Distinguishers reported in this work

	#R	Start → End	Complexity	Strategy	Remarks
AESQ	8	2 → 9	1	Yoyo	Basic Yoyo
	9	1 → 9	$2^{26.08}$	Yoyo + Nested ZDP	First 9 round Distinguisher starting from Round 1
	9	2 → 10	5	Improbable Differential Yoyo	Uses the inside-out technique
	10	2 → 11	2^{28}		
	12	2 → 13	2^{126}	Impossible Differential Yoyo	
	16	2 → 17	2^{126}	Bi-directional Impossible Differential Yoyo	Uses inside-out with bi-directional Yoyo
AES	6	1 → 6	2^{30}	Impossible Differential Yoyo	Uses the inside-out technique
	8	1 → 8	2^{30}	Bi-directional Impossible Differential Yoyo	Uses inside-out with bi-directional Yoyo

This work explores many ways to extend the Yoyo game. The authors in [RBH17], have shown attacks on 3/5 rounds AES, where they extend the basic Yoyo game. However, with the exception of the $\texttt{AESQ}_{2\to9}$ and $\texttt{AESQ}_{1\to9}$ distinguishers, the strategies reported here differ from the ones shown in [RBH17]. This is mostly because of the inside-out philosophy used here which becomes inapplicable in the secret-key setting. The main contribution of this work comes in the form of the idea of using the inside-out technique to partially deploying the Yoyo game as an input generator. The notion of *Nested ZDP* introduced

here seems to work nicely as a combiner of Yoyo and classical differential cryptanalysis. Along with MixColumns, the techniques used here exploit the properties of SuperSBoxes and MegaSBoxes. The bi-directional Yoyo game is the most effective strategy leading to doubling of the number of rounds penetrated. One might look critically at the last verification which uses $\texttt{AESQ}^{-1}_{10\to17}$. However, usage of such kind of verification is available in literature of distinguishers on Feistel schemes [LWZ15]. Moreover, as argued earlier, the strategy ensures non-triviality. Except the 12 and 16 round distinguishers of AESQ, all other distinguishers of AESQ and AES rely on practical data complexities and negligible memory. The closest comparable results for AESQ are due to Bagheri et al. [BMS16] who report rebound and time-memory trade-off attacks. Though the maximum number of rounds is same, the current work exponentially outperforms the former both in terms of data and memory requirements.

In case of 8-round AES, in the known-key setting, with the exception of [GR17], our result beats all other works, while being the only one that requires negligible memory. Table 4 summarizes the attacks presented here. It should be noted however that comparing attacks in the known-key model only by their complexity is not completely fair, as one has to take into consideration also the rate of *simplicity* of the found non-random property, which may affect the chances to extend the distinguisher to more rounds or to more powerful attacks. In this respect, our attack is not directly comparable to several of the previous results, as the non-random property we find is somewhat complex.

10 Conclusion

In this work we explored the impact of the Yoyo cryptanalytic strategy on public permutation AESQ as well as AES in the known-key model. We deployed the basic Yoyo technique to get a deterministic 8-round distinguisher for AESQ and extended it using our notion of Nested ZDP to include the first round using around 2^{26} queries. In addition to this we used the inside-out strategy to augment Yoyo using classical, improbable and impossible differentials to reach 9, 10, 12 rounds starting from round 2 with data complexities of about $2^2, 2^{28}$ and 2^{126} respectively. The final strategy devised here allows us to combine two Yoyo games giving a 16-round distinguisher using 2^{126} queries. The impossible difference based Yoyo strategies when applied to AES lead to known-key distinguishers for 6 and 8 rounds with a complexity of 2^{30}. One may note that all improbable distinguishers reported can be converted to impossible ones while paying some extra cost in terms of data complexity. The success probabilities of the attacks have been computed to be high enough and all distinguishers with practical complexities were verified using computer simulations.

It can be noted that attacks on AESQ presented here exclude the last MMC and except the attack on $\texttt{AESQ}_{1\to9}$ all of them start from round 2 due to the reliance on the MegaSBox. It would be interesting to overcome these limitations by looking further into the design of AESQ or possibly the Yoyo game itself. The bi-directional Yoyo game warrants further attention and might become a valuable generic cryptanalytic tool for analyzing other public crypto primitives.

References

[ABM13] Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards Understanding the Known-Key Security of Block Ciphers. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 348–366, 2013.

[AM09] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list, 2009. http://www.131002.net/data/papers/AM09.pdf.

[BA14] Khovratovich D Biryukov A. PAEQ v1. http://competitions.cr.yp.to/round1/paeqv1.pdf, 2014.

[BBD+99] Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In Stafford Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, pages 362–375, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[BK14] Alex Biryukov and Dmitry Khovratovich. PAEQ: Parallelizable Permutation-Based Authenticated Encryption. In Sherman S. M. Chow, Jan Camenisch, Lucas C. K. Hui, and Siu Ming Yiu, editors, *Information Security*, pages 72–89, Cham, 2014. Springer International Publishing.

[BLP16] Alex Biryukov, Gaëtan Leurent, and Léo Perrin. Cryptanalysis of Feistel Networks with Secret Round Functions. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography – SAC 2015*, pages 102–121, Cham, 2016. Springer International Publishing.

[BMS16] Nasour Bagheri, Florian Mendel, and Yu Sasaki. Improved Rebound Attacks on AESQ: Core Permutation of CAESAR Candidate PAEQ. In *Proceedings, Part II, of the 21st Australasian Conference on Information Security and Privacy - Volume 9723*, pages 301–316, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[cae] CAESAR Competition. https://competitions.cr.yp.to/caesar.html.

[Dew] Rishi Dewan. Advanced-Encryption-Standard-Algorithm. https://github.com/rishidewan33/Advanced-Encryption-Standard-Algorithm.

[DLP+09] Joan Daemen, Mario Lamberger, Norbert Pramstaller, Vincent Rijmen, Frederik Vercauteren, Esat Cosic, K U Leuven, Louvain , and Belgium . Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. 85:85–104, 06 2009.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag, Berlin, Heidelberg, 2002.

[DR06] Joan Daemen and Vincent Rijmen. Understanding Two-Round Differentials in AES. In *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, pages 78–94, 2006.

[Gil14] Henri Gilbert. A Simplified Representation of AES. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 200–222, 2014.

[GP10] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, pages 365–383, 2010.

[GR17] Lorenzo Grassi and Christian Rechberger. New and Old Limits for AES Known-Key Distinguishers. Cryptology ePrint Archive, Report 2017/255, 2017. https://eprint.iacr.org/2017/255.

[JNP13] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple Limited-Birthday Distinguishers and Applications. In *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 533–550, 2013.

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 315–324, 2007.

[LWZ15] Li Lin, Wenling Wu, and Yafei Zheng. Improved Meet-in-the-Middle Distinguisher on Feistel Schemes. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 122–142, 2015.

[MP15] Bart Mennink and Bart Preneel. On the Impact of Known-Key Attacks on Hash Functions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 59–84, 2015.

[oST] National Institute of Standards and Technology. SHA-3 : Cryptographic hash algorithm competition. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[PR18] Goutam Paul and Souvik Ray. On data complexity of distinguishing attacks versus message recovery attacks on stream ciphers. *Des. Codes Cryptography*, 86(6):1211–1247, 2018.

[RBH17] Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 217–243, Cham, 2017. Springer International Publishing.

[SKMC16] Dhiman Saha, Sourya Kakarla, Srinath Mandava, and Dipanwita Roy Chowdhury. Gain: Practical Key-Recovery Attacks on Round-reduced PAEQ. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 194–210, Cham, 2016. Springer International Publishing.

[Tez10] Cihangir Tezcan. The Improbable Differential Attack: Cryptanalysis of Reduced Round CLEFIA. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, pages 197–209, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Tez14] Cihangir Tezcan. Improbable differential attacks on Present using undisturbed bits. *J. Computational Applied Mathematics*, 259:503–511, 2014.

[TS16] Cihangir Tezcan and Ali Aydin Selçuk. Improved improbable differential attacks on ISO standard CLEFIA: Expansion technique revisited. *Inf. Process. Lett.*, 116:136–143, 2016.

A Experimental Verification

Most of the distinguishers presented in this paper have practical complexities. These have been performed experimentally and their complexities have been verified. All the experiments have been performed on a system with Intel core i7-6700 CPU@3.40 × 8 and memory 16GB. For programming, we have used Java `openjdk` version `1.8.0_181`. For implementing `AES` functionalities, we have used publicly available code [Dew].

The distinguisher for $\mathtt{AESQ}_{2\to 9}$ is deterministic in nature and this attack is performed in negligible time using Algorithm 2. The complexity of the distinguisher for $\mathtt{AESQ}_{1\to 9}$ is $2^{26.05}$. For the attack, pairs of plaintexts having only one word difference have been chosen at random. Plaintexts pairs whose differences have not mitigated into a byte after the first round of `AESQ` are filtered out. Among the remaining pairs, all possible swapping are done between the corresponding ciphertexts. Algorithm 3 describes the distinguisher. This attack has been performed using a single thread and it took 17435867($\approx 2^{24.05}$) iterations in 557 seconds for successfully finding a pair of plaintexts and a swap vector which conforms to our claim. The following pair of texts conform to our claim when swapped after forward permutation using the given vector.

$$\begin{bmatrix} \mathtt{fa} & \mathtt{b1} & \mathtt{5a} & \mathtt{2f} & \mathtt{00} & \mathtt{68} & \mathtt{a1} & \mathtt{e5} & \mathtt{b3} & \mathtt{55} & \mathtt{81} & \mathtt{01} & \mathtt{c3} & \mathtt{3d} & \mathtt{4c} & \mathtt{8a} \\ \mathtt{b7} & \mathtt{64} & \mathtt{0e} & \mathtt{f1} & \mathtt{92} & \mathtt{d5} & \mathtt{10} & \mathtt{b0} & \mathtt{89} & \mathtt{cb} & \mathtt{6f} & \mathtt{51} & \mathtt{b1} & \mathtt{63} & \mathtt{6c} & \mathtt{00} \\ \mathtt{f8} & \mathtt{9f} & \mathtt{22} & \mathtt{15} & \mathtt{97} & \mathtt{1a} & \mathtt{44} & \mathtt{7b} & \mathtt{2d} & \mathtt{c3} & \mathtt{64} & \mathtt{c6} & \mathtt{67} & \mathtt{64} & \mathtt{b2} & \mathtt{43} \\ \mathtt{28} & \mathtt{87} & \mathtt{32} & \mathtt{25} & \mathtt{18} & \mathtt{df} & \mathtt{4f} & \mathtt{98} & \mathtt{c6} & \mathtt{b0} & \mathtt{c7} & \mathtt{a4} & \mathtt{28} & \mathtt{2a} & \mathtt{59} & \mathtt{9d} \end{bmatrix}$$

$$\begin{bmatrix} \mathtt{2e} & \mathtt{b1} & \mathtt{5a} & \mathtt{2f} & \mathtt{00} & \mathtt{68} & \mathtt{a1} & \mathtt{e5} & \mathtt{b3} & \mathtt{55} & \mathtt{81} & \mathtt{01} & \mathtt{c3} & \mathtt{3d} & \mathtt{4c} & \mathtt{8a} \\ \mathtt{b7} & \mathtt{70} & \mathtt{0e} & \mathtt{f1} & \mathtt{92} & \mathtt{d5} & \mathtt{10} & \mathtt{b0} & \mathtt{89} & \mathtt{cb} & \mathtt{6f} & \mathtt{51} & \mathtt{b1} & \mathtt{63} & \mathtt{6c} & \mathtt{00} \\ \mathtt{f8} & \mathtt{9f} & \mathtt{f2} & \mathtt{15} & \mathtt{97} & \mathtt{1a} & \mathtt{44} & \mathtt{7b} & \mathtt{2d} & \mathtt{c3} & \mathtt{64} & \mathtt{c6} & \mathtt{67} & \mathtt{64} & \mathtt{b2} & \mathtt{43} \\ \mathtt{28} & \mathtt{87} & \mathtt{32} & \mathtt{4c} & \mathtt{18} & \mathtt{df} & \mathtt{4f} & \mathtt{98} & \mathtt{c6} & \mathtt{b0} & \mathtt{c7} & \mathtt{a4} & \mathtt{28} & \mathtt{2a} & \mathtt{59} & \mathtt{9d} \end{bmatrix}$$

$$vector = \begin{bmatrix} \mathtt{01} & \mathtt{01} & \mathtt{01} & \mathtt{00} & \mathtt{01} & \mathtt{01} & \mathtt{00} & \mathtt{01} & \mathtt{01} & \mathtt{01} & \mathtt{00} & \mathtt{01} & \mathtt{01} & \mathtt{01} & \mathtt{01} & \mathtt{00} \end{bmatrix}$$

- Initial difference of two input states:

$$\begin{bmatrix} \mathtt{d4} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{14} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{d0} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{69} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \end{bmatrix}$$

- Difference of two states after one round

$$\begin{bmatrix} \mathtt{02} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{00} \end{bmatrix}$$

- Two states before swapping (before last MixColumns and ShiftRows)

$$\begin{bmatrix} \mathtt{4e} & \mathtt{12} & \mathtt{6c} & \mathtt{05} & \mathtt{2c} & \mathtt{ab} & \mathtt{29} & \mathtt{96} & \mathtt{c7} & \mathtt{76} & \mathtt{26} & \mathtt{7c} & \mathtt{68} & \mathtt{a6} & \mathtt{b2} & \mathtt{e1} \\ \mathtt{b2} & \mathtt{38} & \mathtt{9b} & \mathtt{7a} & \mathtt{f3} & \mathtt{68} & \mathtt{e0} & \mathtt{02} & \mathtt{10} & \mathtt{f1} & \mathtt{40} & \mathtt{f6} & \mathtt{dc} & \mathtt{42} & \mathtt{0c} & \mathtt{59} \\ \mathtt{67} & \mathtt{97} & \mathtt{9b} & \mathtt{0a} & \mathtt{6c} & \mathtt{61} & \mathtt{25} & \mathtt{0e} & \mathtt{0f} & \mathtt{f4} & \mathtt{92} & \mathtt{cd} & \mathtt{59} & \mathtt{33} & \mathtt{18} & \mathtt{d3} \\ \mathtt{53} & \mathtt{d2} & \mathtt{4c} & \mathtt{06} & \mathtt{2e} & \mathtt{da} & \mathtt{e9} & \mathtt{29} & \mathtt{88} & \mathtt{fa} & \mathtt{5f} & \mathtt{2d} & \mathtt{b1} & \mathtt{ae} & \mathtt{b7} & \mathtt{be} \end{bmatrix}$$

$$\begin{bmatrix} \mathtt{24} & \mathtt{2c} & \mathtt{e7} & \mathtt{0d} & \mathtt{40} & \mathtt{4d} & \mathtt{a4} & \mathtt{20} & \mathtt{92} & \mathtt{30} & \mathtt{a8} & \mathtt{fa} & \mathtt{b6} & \mathtt{0e} & \mathtt{8a} & \mathtt{2d} \\ \mathtt{8b} & \mathtt{89} & \mathtt{a0} & \mathtt{1a} & \mathtt{14} & \mathtt{c5} & \mathtt{e7} & \mathtt{e9} & \mathtt{c6} & \mathtt{84} & \mathtt{2e} & \mathtt{86} & \mathtt{49} & \mathtt{32} & \mathtt{13} & \mathtt{a5} \\ \mathtt{c1} & \mathtt{31} & \mathtt{6b} & \mathtt{2f} & \mathtt{a1} & \mathtt{87} & \mathtt{1c} & \mathtt{cf} & \mathtt{ea} & \mathtt{ca} & \mathtt{ce} & \mathtt{cd} & \mathtt{e0} & \mathtt{46} & \mathtt{c3} & \mathtt{1f} \\ \mathtt{03} & \mathtt{c4} & \mathtt{c4} & \mathtt{e5} & \mathtt{6f} & \mathtt{e5} & \mathtt{f3} & \mathtt{ad} & \mathtt{cf} & \mathtt{8d} & \mathtt{be} & \mathtt{5a} & \mathtt{8c} & \mathtt{ad} & \mathtt{7a} & \mathtt{bd} \end{bmatrix}$$

- Difference of two permuted states before swapping

$$\begin{bmatrix} 6a & 3e & 8b & 08 & 6c & e6 & 8d & b6 & 55 & 46 & 8e & 86 & de & a8 & 38 & cc \\ 39 & b1 & 3b & 60 & e7 & ad & 07 & eb & d6 & 75 & 6e & 70 & 95 & 70 & 1f & fc \\ a6 & a6 & f0 & 25 & cd & e6 & 39 & c1 & e5 & 3e & 5c & 00 & b9 & 75 & db & cc \\ 50 & 16 & 88 & e3 & 41 & 3f & 1a & 84 & 47 & 77 & e1 & 77 & 3d & 03 & cd & 03 \end{bmatrix}$$

- Two states after swapping (before last MixColumns and ShiftRows)

$$\begin{bmatrix} 4e & 12 & 6c & 0d & 2c & ab & a4 & 96 & c7 & 76 & a8 & 7c & 68 & a6 & b2 & 2d \\ b2 & 38 & 9b & 1a & f3 & 68 & e7 & 02 & 10 & f1 & 2e & f6 & dc & 42 & 0c & a5 \\ 67 & 97 & 9b & 2f & 6c & 61 & 1c & 0e & 0f & f4 & ce & cd & 59 & 33 & 18 & 1f \\ 53 & d2 & 4c & e5 & 2e & da & f3 & 29 & 88 & fa & be & 2d & b1 & ae & b7 & bd \end{bmatrix}$$

$$\begin{bmatrix} 24 & 2c & e7 & 05 & 40 & 4d & 29 & 20 & 92 & 30 & 26 & fa & b6 & 0e & 8a & e1 \\ 8b & 89 & a0 & 7a & 14 & c5 & e0 & e9 & c6 & 84 & 40 & 86 & 49 & 32 & 13 & 59 \\ c1 & 31 & 6b & 0a & a1 & 87 & 25 & cf & ea & ca & 92 & cd & e0 & 46 & c3 & d3 \\ 03 & c4 & c4 & 06 & 6f & e5 & e9 & ad & cf & 8d & 5f & 5a & 8c & ad & 7a & be \end{bmatrix}$$

- Difference of two permuted states after swapping

$$\begin{bmatrix} 6a & 3e & 8b & 08 & 6c & e6 & 8d & b6 & 55 & 46 & 8e & 86 & de & a8 & 38 & cc \\ 39 & b1 & 3b & 60 & e7 & ad & 07 & eb & d6 & 75 & 6e & 70 & 95 & 70 & 1f & fc \\ a6 & a6 & f0 & 25 & cd & e6 & 39 & c1 & e5 & 3e & 5c & 00 & b9 & 75 & db & cc \\ 50 & 16 & 88 & e3 & 41 & 3f & 1a & 84 & 47 & 77 & e1 & 77 & 3d & 03 & cd & 03 \end{bmatrix}$$

- Difference of states after 1st round (during inverse permutation)

$$\begin{bmatrix} d7 & 00 & 00 & 00 & d9 & 00 & 00 & 00 & 46 & 00 & 00 & 00 & 75 & 00 & 00 & 00 \\ 00 & c3 & 00 & 00 & 00 & 06 & 00 & 00 & 00 & 15 & 00 & 00 & 00 & c8 & 00 & 00 \\ 00 & 00 & ec & 00 & 00 & 00 & e6 & 00 & 00 & 00 & 65 & 00 & 00 & 00 & 16 & 00 \\ 00 & 00 & 00 & ec & 00 & 00 & 00 & \mathbf{00} & 00 & 00 & 00 & 8f & 00 & 00 & 00 & 02 \end{bmatrix}$$

- Difference of two states after inverse permutation

$$\begin{bmatrix} b5 & d0 & e8 & 68 & 6c & b4 & 07 & \mathbf{00} & 90 & 19 & 58 & 4e & 4c & b1 & ff & b1 \\ 06 & 1a & 63 & cb & \mathbf{00} & 48 & 08 & cf & 9a & 77 & 37 & 1b & bf & 0d & 35 & c5 \\ e4 & f5 & 61 & de & d1 & \mathbf{00} & de & fb & a7 & 65 & 6c & fa & 34 & 2d & 4a & ed \\ 1c & c0 & 83 & d6 & bf & 9f & \mathbf{00} & d4 & 25 & a3 & c1 & 67 & 18 & 2e & f0 & 0f \end{bmatrix}$$

For performing distinguishing attack on $\texttt{AESQ}_{2\to 11}$, 2^{28} pairs of plaintexts have been used. To reduce the running time, 16 threads have been used. It is an improbable differential distinguisher. For $\texttt{AESQ}_{2\to 11}$, the distinguisher took 5047.616 seconds to run for all 2^{28} iterations and have not found the improbable differential property; while for the random permutation it took 225885057 ($\approx 2^{27.75}$) iterations to find the improbable differential property; and thus it conforms to our claim. The distinguisher for $\texttt{AESQ}_{2\to 10}$ is also improbable differential in nature with a complexity of 5. In negligible time, its property has been verified. Algorithm 4 describes both of these distinguishers. We did not implement distinguishers for $\texttt{AESQ}_{2\to 13}$ and $\texttt{AESQ}_{2\to 17}$ as their complexities are impractical.

The distinguishing attacks on $\texttt{AES}_{1\to 8}$ introduced in this paper have practical complexity. This distinguishing attack is similar to Algorithm 6 with reduced complexity. The distinguishing algorithm ran for 2^{30} iterations in 2977.575 seconds in the above machine and have not found the impossible differential with an inactive SuperSBox. For random permutation, we have found this differential in 187840320 ($\approx 2^{27.48}$) iterations.

A.1 Success Probability

For experimental verification of success probabilities, a blackbox is considered which can act as a cipher $\mathcal{C}$ (we consider $\mathcal{C}$ =AESQ/AES) or as a random permutation $\mathcal{R}$. To calculate the success probability, consider the following confusion matrix in Table 5.

Table 5: Confusion Matrix of $\mathcal{C}$ and $\mathcal{R}$

Actual \ Observed	$\mathcal{C}$	$\mathcal{R}$
$\mathcal{C}$	$o_c - n_{FP}$	n_{FN}
$\mathcal{R}$	n_{FP}	$o_r - n_{FN}$

The experiment is performed considering $\mathcal{C}$ in the blackbox a_c times and $\mathcal{R}$ in the blackbox a_r times. Based on the output, $\mathcal{C}$ is decided o_c times and $\mathcal{R}$ is decided o_r times. The numbers of false positives and false negatives are denoted by n_{FP} and n_{FN} respectively. Then the success probability is given by

$$\begin{aligned} Pr[Success] &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{o_c + o_r} \\ &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{a_c + a_r}. \end{aligned}$$

Table 6 shows the experimental results of various distinguishers and their corresponding success probabilities. The success probabilities calculated by experimentation is close enough to their respective theoretically estimated values.

Table 6: Experimental Verification of Success Probability

Distinguisher	$\#n$	Blackbox	Detected as AESQ/AES	Detected as $\mathcal{R}$	Experimental Success Probability	Estimated Success Probability
$\texttt{AESQ}_{1\to9}$	1000	$\texttt{AESQ}_{1\to9}$	826	174	0.75	0.70
		$\mathcal{R}$	324	676		
$\texttt{AESQ}_{2\to10}$	1000	$\texttt{AESQ}_{2\to10}$	645	355	0.70	0.82
		$\mathcal{R}$	258	742		
$\texttt{AESQ}_{2\to11}$	100	$\texttt{AESQ}_{2\to11}$	100	0	0.86	0.77
		$\mathcal{R}$	29	71		
$\texttt{AES}_{1\to6}$	100	$\texttt{AES}_{1\to6}$	100	0	0.81	0.83
		$\mathcal{R}$	39	61		
$\texttt{AES}_{1\to8}$	100	$\texttt{AES}_{1\to8}$	100	0	0.82	0.83
		$\mathcal{R}$	37	63		