Cryptanalysis of ForkAES

Subhadeep Banik[1], Jannis Bossert[2], Amit Jana[3], Eik List[2], Stefan Lucks[2], Willi Meier[4], Mostafizar Rahman[3], Dhiman Saha[5], and Yu Sasaki[6]

[1] EPFL Switzerland, subhadeep.banik@epfl.ch
[2] Bauhaus-Universität Weimar, firstname.lastname(at)uni-weimar.de
[3] CSRU, Indian Statistical Institute,
janaamit001@gmail.com,mrahman454@gmail.com
[4] FHNW, Switzerland, willi.meier@fhnw.ch
[5] IIT Bhilai, dhiman@iitbhilai.ac.in
[6] NTT Secure Platform Laboratories, yu.sasaki.sk@hco.ntt.co.jp

Abstract. Forkciphers are a new kind of primitive proposed recently by Andreeva et al. for efficient encryption and authentication of small messages. They fork the middle state of a cipher and encrypt it twice under two smaller independent permutations. Thus, forkciphers produce two output blocks in one primitive call.
Andreeva et al. proposed ForkAES, a tweakable AES-based forkcipher that splits the state after five out of ten rounds. While their authenticated encrypted schemes were accompanied by proofs, the security discussion for ForkAES was not provided, and founded on existing results on the AES and KIASU-BC. Forkciphers provide a unique interface called reconstruction queries that use one ciphertext block as input and compute the respective other ciphertext block. Thus, they deserve a careful security analysis.
This work fosters the understanding of the security of ForkAES with three contributions: (1) We observe that security in reconstruction queries differs strongly from the existing results on the AES. This allows to attack nine out of ten rounds with differential, impossible-differential and yoyo attacks. (2) We observe that some forkcipher modes may lack the interface of reconstruction queries, so that attackers must use encryption queries. We show that nine rounds can still be attacked with rectangle and impossible-differential attacks. (3) We present forgery attacks on the AE modes proposed by Andreeva et al. with nine-round ForkAES.

Keywords: Symmetric-key cryptography · cryptanalysis · tweakable block cipher · impossible differential · boomerang · yoyo · AE.

1 Introduction

The fast distribution of resource-constrained devices demands efficient encryption and authentication of short messages. Forkciphers are a recent proposal by Andreeva et al. [1] to address this purpose. Like classical (tweakable) block ciphers, they encrypt a plaintext block under a secret key; In contrast, however, forkciphers compute two ciphertext blocks from the same input. To boost the performance, the state in the middle of the computation is forked, and both

ciphertext blocks are computed separately only from the middle. Therefore, the construction can share some computations and has to encrypt only twice over the bottom rounds. Thus, efficient AE schemes can obtain a ciphertext and tag efficiently for messages whose size is at most a block. Owing to this construction, forkciphers provide a new interface called *reconstruction* that takes one of the ciphertext blocks as input and returns the other one.

As instance of particular interest, Andreeva et al. [1] proposed ForkAES, which employs the original key schedule and round function of the AES-128. Moreover, ForkAES is a tweakable block cipher that adopts the concept from KIASU-BC [15]: in every round where the round key is XORed to the state, an additional 64-bit public tweak T is XORed to the topmost two state rows. ForkAES encrypts the plaintext P over the first five rounds exactly as in the KIASU-BC; though, it forks the middle state X and produces from it a ciphertext C_0 exactly as KIASU-BC with the round keys K^5 through K^{10} plus a second ciphertext C_1 under six further round keys K^{11} through K^{16}.

Existing Security Arguments. The adoption of the AES round function and the tweak process from KIASU-BC allowed to profit from existing results, e.g., for the resistance against differential and linear cryptanalysis. Andreeva et al. also considered meet-in-the-middle attacks briefly; concerning further attacks, they stated that: "the security of our forkcipher design can be reduced to the security of the AES and KIASU ciphers for further type of attacks" [1, Sect 3.2]. However, the structure of ForkAES may allow new attack angles, and it appeared to be a highly interesting task for the community to study ForkAES deeply.

Contribution. This work analyzes attack vectors on forkciphers and ForkAES in depth. We generalize it to ForkAES-r_t-r_{b_0}-r_{b_1}, where r_t, r_{b_0} and r_{b_1} denote the number of rounds from P to X, from X to C_0, and from X to C_1, respectively; e.g., ForkAES-5-5-5 means the original ForkAES. While we consider only the case $r_{b_0} = r_{b_1}$, we indicate by ForkAES-$*$-r_{b_0}-r_{b_1} if r_t can be any non-negative integer.

First, we observe that the security of the reconstruction of forkciphers is very different from the encryption and decryption of the conventional AES since the first half of the computation uses the inverse of the round function whereas the second half employs the ordinary round function. We exploit this property by introducing *reflection differential trails* that allow to attack nine rounds (ForkAES-5-4-4) with a low complexity. We also present impossible-differential [4,7,8,9,12,17] and yoyo [3,20] attacks as well as forgery attacks for the AE mode by exploiting the reflection feature.

Second, we consider the restricted case where the reconstruction interface is unavailable. This is natural for some usages. For example, Andreeva et al. [1] suggested to replace the standard CTR mode with forkciphers; two ciphertext blocks of forkciphers can halve the number of primitive calls to generate the same key-stream length. In such settings, reconstruction (and even decryption) queries of forkciphers are not exploitable by adversaries. We show that even in such environments, attacks can reach nine rounds by a rectangle [5,6,10,18,22]

Table 1: Comparison of Attacks. CP and CR denote chosen plaintexts and chosen reconstruction queries, respectively. Due to the limited space, two attacks are omitted and are detailed in the full version of this work [2].

				Time			
Construction	Attack Type	Data		Encs.	MAs	Mem.	Section
Encryption Queries							
ForkAES-∗-4-4	Rectangle	2^{85}	CP	$2^{88.5}$	$2^{92.4}$	$2^{86.4}$	Sect. 6
ForkAES-∗-4-4	Impossible Diff.	$2^{70.2}$	CP	$2^{75.4}$	$2^{110.2}$	2^{100}	Sect. 7
Reconstruction Queries							
ForkAES-∗-4-4	Reflection Diff.	2^{35}	CR	2^{28}	2^{35}	2^{33}	Sect. 3
ForkAES-∗-4-4	Impossible Diff.	$2^{39.4}$	CR	2^{47}	2^{47}	2^{35}	Sect. 4
ForkAES-∗-3-3	Yoyo	$2^{14.5}$	CR	$2^{14.5}$	2^{29}	2^{29}	Sect. 5
ForkAES-∗-4-4	Imp.-diff. Yoyo	$2^{122.83}$	CR	$2^{122.83}$	–	$O(1)$	App. D
Forgery Attacks on AE modes							
PAEF-ForkAES-∗-4-4	Reflection Diff.	2^{92}	CR	2^{92}	–	$O(1)$	App. C

and an impossible-differential attack. Those attacks also exploit the forking step, which produces rectangle quartets from pairs of plaintexts.

Our attacks do not endanger the security of the full ForkAES; however, they contradict some of the designer's claims as they cover one round more than attacks for KIASU-BC [13,21]. More importantly, the forking principle exposes reflection properties in reconstruction queries.

Outline. Next, we briefly revisit the necessary details on the AES, KIASU-BC, and ForkAES. Sections 3 – 5 detail our attacks based on reflection queries and Sections 6 and 7 describe our attacks based on encryption queries. Due to space limitations, those sections contain only a representative description of an attack each; more results are deferred to the supporting material.

2 Preliminaries

General Notation. We assume, the reader is familiar with the concepts of block ciphers and their analysis. Most of the time, we consider bit strings of fixed length. We mostly use uppercase letters (e.g., X) for bit strings, lowercase letters for indices (x), and calligraphic letters for sets ($\mathcal{X}$). For some positive integer n, we interpret bit strings $X \in \{0,1\}^n$ as vector elements of $\mathbb{F}_2^n$, where addition is the bit-wise XOR, denoted by $\oplus$. Moreover, the AES works on byte vectors or byte matrices, i.e., 16-element vectors in $\mathbb{F}_{2^8}$. So, we interpret byte matrices of r rows and c columns as elements of $\mathbb{F}_{2^8}^{r\times c}$.

Forkciphers. Let $\mathcal{B}$, $\mathcal{K}$, and $\mathcal{T}$ be non-empty sets or spaces. A tweakable forkcipher $\widetilde{E}$ is a tuple of three deterministic algorithms: An encryption algorithm $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to (\mathcal{B})^2$; a decryption algorithm $\widetilde{D} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \times \{0,1\} \to \mathcal{B}$;

and a tag-reconstruction algorithm $\widetilde{R} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \times \{0,1\} \to \mathcal{B}$. The encryption produces $\widetilde{E}_K^T(P) = (C_0 \,\|\, C_1)$. We define $\widetilde{E}_K^T(P)[0] = C_0$ and $\widetilde{E}_K^T(P)[1] = C_1$ Decryption and tag reconstruction take a bit b s. t. it holds $\widetilde{D}_K^{T,b}(\widetilde{E}_K^T(P)[b]) = P$, for all $K, T, P, b \in \mathcal{K} \times \mathcal{T} \times \mathcal{B} \times \{0,1\}$. The tag-reconstruction takes K, T, C_b, and b as input, and produces $C_{b\oplus 1}$. The ideal tweakable forked permutation $\widetilde{\Pi}$ encrypts messages P under two independent permutations $\widetilde{\pi}_0, \widetilde{\pi}_1 : \mathcal{T} \times \mathcal{B} \to \mathcal{B}$, and outputs $(C_0 \,\|\, C_1)$ as $C_b \leftarrow \widetilde{\pi}_b(P)$, for $b \in \{0,1\}$.

The AES-128 is a substitution-permutation network over 128-bit inputs, which transforms the input through ten rounds consisting of SubBytes (SB), ShiftRows (SR), MixColumns (MC), and a round-key addition with a round key K^i. At the start, a whitening key K^0 is XORed to the state; the final round omits the MixColumns operation. We write S^i for the state after Round i, and $S^i[j]$ for the j-th byte, for $0 \le i \le 10$ and $0 \le j \le 15$. Further, we use $S^{r,\mathsf{SB}}$, $S^{r,\mathsf{SR}}$, and $S^{r,\mathsf{MC}}$ for the states in the r-th round directly after the SubBytes, ShiftRows, and MixColumns operations, respectively. The byte ordering is given by:

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

We adopt a similar convention for the round keys K^i and their bytes $K^i[j]$, for $0 \le i \le 16$; for both, we also use often a matrix-wise indexing of the bytes from $0,0$ to $3,3$. More details can be found in [11,19].

KIASU-BC [15] is a tweakable block cipher that differs from the AES-128 only in the fact that it XORs a public 64-bit tweak T to the topmost two rows of the state whenever a round key is XORed. We denote the tweak by T and by $T[j]$, $0 \le j \le 7$, the bytes of T. The bytes are ordered as

$$\begin{bmatrix} 0 & 2 & 4 & 6 \\ 1 & 3 & 5 & 7 \end{bmatrix}.$$

ForkAES is a forkcipher based on KIASU-BC. It forks the state after five rounds and transforms it twice to two ciphertexts C_0 and C_1. We denote the states of the first branch by $X^i =^{\text{def}} S^i$, for $5 \le i \le 10$, where $X^5 = S^5$ and $X^{10} = C_0$. Moreover, we denote the states of the second branch by Y^i, for $5 \le i \le 10$, where $Y^5 = S^5$ and $Y^{10} = C_1$. We will also write R for the sequence $\mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$. and KS for an iteration of the AES-128 key schedule. A schematic illustration is given in Fig. 1, and more details can be found in [1]. We will sometimes reorder the linear operations, e.g., swap MixColumns, ShiftRows, and the key addition. We will write $\widetilde{K}^r = \mathsf{MC}^{-1}(K^r)$ and $\widehat{K}^r = \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(K^r))$ for the transformed round keys.

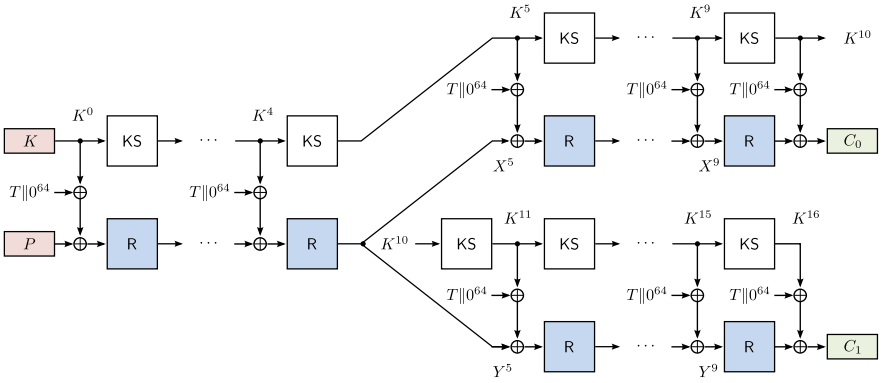

Fig. 1: ForkAES. R is the AES-128 round function; KS a round of its key schedule.

Subspaces of the AES. We adopt the notion of AES subspaces from Grassi et al. [14]. Given a vector space $\mathcal{W}$ and a subspace $\mathcal{V} \subseteq \mathcal{W}$; if a is an element of $\mathcal{W}$, then, a coset $\mathcal{V} \oplus a =^{\text{def}} \{v \oplus a | \forall v \in \mathcal{V}\}$ is a subset of $\mathcal{V}$ in $\mathcal{W}$. We consider vectors and vector spaces over $\mathbb{F}_{2^8}^{4\times 4}$, and denote by $\{e_{0,0}, \ldots, e_{3,3}\}$ the unit vectors of $\mathbb{F}_{2^8}^{4\times 4}$, i.e., $e_{i,j}$ has a single 1 in the i-th row and j-th column. For a vector space $\mathcal{V}$ and a function $F : \mathbb{F}_{2^8}^{4\times 4} \to \mathbb{F}_{2^8}^{4\times 4}$, we let $F(\mathcal{V}) =^{\text{def}} \{F(v) | v \in \mathcal{V}\}$. For a subset $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$ and a subset of vector spaces $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n\}$, we define $\mathcal{V}_{\mathcal{I}} =^{\text{def}} \bigoplus_{i\in\mathcal{I}} \mathcal{V}_i$. We adopt the definitions by Grassi et al. of four families of subspaces for the AES, for $i \in \{0, 1, 2, 3\}$:

- the column spaces $\mathcal{C}_i$ as $\mathcal{C}_i = \langle e_{0,i}, e_{1,i}, e_{2,i}, e_{3,i} \rangle$,
- the diagonal spaces $\mathcal{D}_i$ as $\mathcal{D}_i = \mathsf{SR}^{-1}(\mathcal{C}_i)$,
- the inverse-diagonal spaces $\mathcal{ID}_i$ as $\mathcal{ID}_i = \mathsf{SR}(\mathcal{C}_i)$, and
- the mixed spaces $\mathcal{M}_i$ as $\mathcal{M}_i = \mathsf{MC}(\mathcal{ID}_i)$.

The S-box $\mathsf{S} : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ of the AES has a few well-analyzed properties; here, we briefly recall one that will be relevant in our later attacks.

Property 1. Let $\alpha, \beta \in \mathbb{F}_{2^8} \setminus \{0^8\}$. For $F \in \{\mathsf{S}, \mathsf{S}^{-1}\}$, it holds that $|\{x : \mathsf{F}(x) \oplus \mathsf{F}(x \oplus \alpha) = \beta\}|$ equals four in one, two in 126, and zero in 129 cases. So, for any differential $\alpha \to \beta$, there exists approximately one input x on average that satisfies the differential.

3 Attack on ForkAES-∗-4-4 with Reflection Trails

Our attacks can work for arbitrary value of r_t. Then the round-key indices for two forking parts depend on the value of r_t. To avoid making the analysis unnecessarily complex, we explain our attacks by using the case with $r_t = 5$.

Observations for Reconstruction Queries. Recall that the first half and the last half of the reconstruction is the inverse and the ordinary round function, respectively. This motivates us to consider the reflection property introduced by Kara [16] against the block cipher GOST. The final 16 rounds of GOST consist of an eight-round Feistel network with the round keys in order K^0, K^1, ..., K^7, followed by eight rounds with K^7, K^6, ..., K^0 in this order. Since Feistel networks are involutions, this enables the following so-called *reflection property*.

Proposition 1 (Reflection Property). When an input value V achieves a symmetric state after eight rounds, i.e. left branch value is identical with right branch value, the output of the final eight rounds will be V.

The reflection property is strong, but possesses limitations: there must not exist round constants, the round keys must be ordered inverted in the first and second chunks, and the target function must be an involution.

This paper considers a differential version of the reflection property. To be more general, the same concept applies if we build trails that are invariant w.r.t. XOR. Suppose, a round function F consists of an arbitrary bijective function, an XOR with a round constant c_i, and an XOR with a round key K^i. Consider $2r$ rounds, where the first r rounds apply F and the final r rounds apply F^{-1}. The round keys $K^i, i = 1, 2, \ldots, 2r$ as well as the round constants c_i, $i = 1, 2, \ldots, 2r$ can differ individually. Then, we have the following property.

Proposition 2 (Reflection Differential Trails). If there exists a differential for the r-round transformation F^r that propagates a difference ΔI to ΔO with probability p, there exists a differential for the $2r$-round transformation $(F^{-1})^r \circ F^r$ that propagates a difference ΔI to ΔI with probability at least p^2. This property holds for any choice of round keys and constants in the $2r$ rounds.

Reflection trails can be applied to reconstruction queries of forkciphers where C_1 (resp. C_0) is computed from C_0 (resp. C_1). The first and last halves of a reconstruction query are back- and forward computations of the same round function, and different round keys and round constants do not impact the property.

Reflection trails are particularly useful for the AES, which achieves full diffusion in only two rounds. There, a single active byte propagates to $1 \xrightarrow{F^{-1}} 4 \xrightarrow{F^{-1}} 16$ active bytes. In contrast, it propagates as $1 \xrightarrow{F^{-1}} 4 \xrightarrow{\text{reflect}} 4 \xrightarrow{F} 1$ in the reflection trail, where $\xrightarrow{F}$ and $\xrightarrow{F^{-1}}$ denote the propagation of the number of active S-boxes with F and F^{-1}, respectively, and $\xrightarrow{\text{reflect}}$ denotes the duplication of the state by forkciphers. This idea allows us to build long differential trails.

It is notable that the designers of ForkAES did not expect the existence of reflection trails. In fact, based on the property that the maximum probability of differential characteristics for four-round AES is 2^{-150}, the designers claim as "*Since our ForkAES design uses the* AES *round function, we can easily deduce that our design will provide enough security in this setting after four rounds against differential attacks in the single-key model.*" [1, Sect. 3.2]

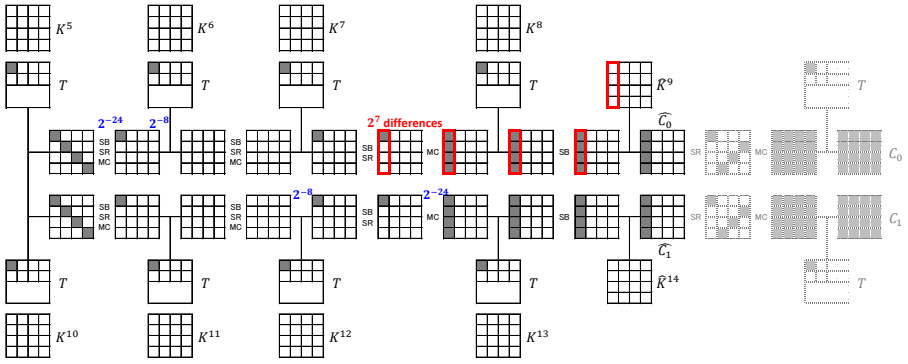

Fig. 2: Truncated differentials for ForkAES-$*$-4-4.

The combination of the reflection trail and a KIASU-like tweak injection yields further efficient differential trails. Tweak difference allows an attacker to create a blank round, and the reflection trail increases the number of blank rounds to 2. Indeed, the reflection trail with $4 \xrightarrow{F^{-1}} 1 \xrightarrow{F^{-1}} 0 \xrightarrow{F^{-1}} 4 \xrightarrow{\text{reflect}} 4 \xrightarrow{F} 0 \xrightarrow{F} 1 \xrightarrow{F} 4$ bytes enables the attacker to build a very efficient trail.

The Differential Trail and Probability. The linear computations in the last round do not affect the security. Hence, instead of ciphertext C_0, C_1 and the last round key K^9, K^{14}, we consider equivalent ciphertext $\widehat{C}_0, \widehat{C}_1$ and equivalent key $\widehat{K}^9, \widehat{K}^{14}$;

$$\widehat{C}_0 \leftarrow \mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}(C_0 \oplus T), \qquad \hat{K}_9 \leftarrow \mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}(K_9),$$
$$\widehat{C}_1 \leftarrow \mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}(C_1 \oplus T), \qquad \hat{K}_{14} \leftarrow \mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}(K_{14}).$$

Refer to Fig. 2 for the differential trail, where we append one round to the above-mentioned trail in reconstruction queries. The attacker queries C_1 and obtains C_0.

The number of active bytes injected by $\widehat{C}_1$ must shrink to one during the inverse of MixColumns and must be canceled by the tweak difference, which occurs with probability 2^{-32}. In Round 6, the four-byte difference in a diagonal position must shrink to one-byte difference and be canceled by the tweak difference. This also occurs with probability 2^{-32}. So, the total probability of this trail is 2^{-64}.

Attack Procedure. During the attack, the tweak difference is fixed.

1. Choose tweaks T and T' with the fixed difference. For each pair T, T', choose 2^{32} distinct values for the first column of $\widehat{C}_1$. Fix the other 12 bytes to arbitrary values and compute the corresponding C_1 offline and query them to obtain the corresponding C_0. Compute the corresponding $\widehat{C}_0$ offline. Hence, we obtain 2^{32} choices of $\widehat{C}_0$ with T and 2^{32} choices of $\widehat{C}_0$ with T'.
2. From 2^{64} pairs of $\widehat{C}_0$ between different tweaks, pick the one with 12 inactive bytes in the Columns 2, 3, and 4 of $\widehat{C}_0$. We expect one right pair.

3. For the right pair, obtain 2^7 key candidates of the first column of $\widehat{K}^9$, which has 1 active byte in the top byte after the inverse of MixColumns and moreover the difference should be one of the 2^7 choices that can be output from the tweak difference after the S-box. This step is colored by red in Fig. 2.
4. Iterate the steps above by shifting the active-byte positions to obtain 2^7 candidates for each column of $\widehat{K}^9$. 2^{28} candidates are then tested exhaustively.

Complexity Evaluation. The data complexity is $4 \cdot (2^{33} + 2^{33}) = 2^{35}$ reconstruction queries. The memory complexity is 2^{33} AES states to store 2^{33} values of $\widehat{C}_0$. The time complexity is 2^{19} memory access to queried data and 2^{28} encryptions for the last exhaustive search. Note that in Step (3), there are 2^7 choices of the input difference to the last SubBytes and the output difference from this SubBytes are fixed to the ciphertext difference of the right pair. For the AES S-box, a randomly chosen pair of input and output differences can be propagated with probability about 2^{-1}, and once they can be propagated, the number of solutions is about 2. Therefore, $2^7 \times (2^{-1})^4$ pairs can be propagated for all the 4 bytes, and the number of total solutions is $2^7 \times (2^{-1})^4 \cdot 2^4 = 2^7$. So, 2^7 candidates of one column of $\widehat{K}^9$ can be obtained with 2^7 computations.

Experimental Verification. In Appendix B, we show an attack on ForkAES-$*$-3-3 which removed the last rounds of the above attack. ForkAES-$*$-3-3 can be attacked with $(Data, Time, Memory) = (2^{19}, 2^{28}, 2^{17})$. We implemented the attack in Java to demonstrate its validity. Refer to Appendix B for the details.

4 Impossible-differential Attack with Reflection Trails

This section describes an impossible-differential distinguisher on ForkAES-$*$-4-4 with reconstruction queries; we will extend it for key recovery.

Distinguisher. The impossible differential distinguisher is as follows.

$$1 \xrightarrow{F^{-1}} 0 \xrightarrow{F^{-1}} 4 \xrightarrow{\text{reflect}} 4 \xrightarrow{F} ? \xrightarrow{F} ? \overset{\text{impossible}}{\longleftrightarrow} ? \xrightarrow{F} ? \xrightarrow{F} 12. \tag{1}$$

The positions of active bytes are illustrated in Fig. 3. The fact that those trails are satisfied with probability zero is explained as follows.

- **Trail from Y^7:** After the tweak injection along with K^6, any number of bytes can be active in the leftmost column. They are moved to different columns by the following ShiftRows operation. After MixColumns, each column is either fully active or fully inactive.
- **Trail from $\widehat{C}_0$:** After the inverse of MixColumns and ShiftRows, at least one inverse diagonal is inactive. Moreover, at least three bytes are active in the state. The subsequent tweak injection (along with K^7), never affects the inactive inverse diagonal. It may cancel one active byte in the state, but does not impact the analysis. In summary, we have the following two properties.
  1. There is at least one inactive byte for each column.

2. The number of active bytes is at least two.

The case that the trail from Y^7 has no active byte is impossible, because the trail from $\widehat{C}_0$ ensures at least three active bytes. The case that the trail from Y^7 has at least 1 fully active column is impossible because the trail from $\widehat{C}_0$ ensures at least one inactive byte for each column. Hence, any trail from Y^7 is impossible to propagate to the difference of $\widehat{C}_0$.

The inactive column position at $\widehat{C}_0$ is the rightmost (4th) column in Fig. 3, but it can also be located in the second or third column position. It cannot be located in the leftmost (first) column because of the tweak difference.

Key Recovery. We append key recovery rounds for the trail in Fig. 3 as depicted in Fig. 4. Suppose, we have a pair of outputs with only a single active column at $\widehat{C}_1$. Then, only five (equivalent-)key bytes must be guessed in those two rounds.

Attack Procedure. During the attack, the tweak difference is fixed.

1. Choose two tweaks T, T' having the fixed difference. For each of T, T', choose 2^{32} distinct values for the active 4-byte values of $\widehat{C}_1$ and fix the other 12 bytes to arbitrary value, say 0. After making 2^{33} reconstruction queries, we obtain 2^{32} choices of $\widehat{C}_0$ associated with T and with T'.
2. From 2^{64} pairs of $\widehat{C}_0$ with different tweaks, pick one with at least one inactive column in Columns 2, 3, or 4 at $\widehat{C}_0$. We expect $3 \cdot 2^{64-32} = 2^{33.58}$ pairs.
3. For each picked pair, derive 2^7 wrong candidates of the top-left byte of $\widehat{K}^{13}$ and the leftmost column of $\widehat{K}^{14}$ by trying 2^7 possible differences in the middle rounds. After evaluating $2^{33.58}$ pairs, we obtain $2^{40.58}$ wrong-key candidates.
4. Iterate the steps above $2^{4.42}$ times by changing the fixed 12 bytes of $\widehat{C}_1$. We obtain $2^{40.58+4.42} = 2^{45}$ wrong candidates of the 5 key bytes. After obtaining 2^N wrong keys, the remaining key space for those five bytes is estimated as

$$2^{40} \cdot (1 - 2^{-40})^{2^N}.$$

   $N = 45$ is sufficient to reduce the remaining key space to 1 since

$$2^{40} \cdot (1 - 2^{-40})^{2^{45}} = 2^{40} \cdot (1 - 2^{-40})^{2^{40} \cdot 2^5} = 2^{40} \cdot e^{-2^5} = 2^{-6.17} < 1.$$

5. Iterate the above steps three times by shifting the active byte positions to recover all bytes of $\widehat{K}^{14}$.

Complexity Evaluation. To recover one column of $\widehat{K}^{14}$, we make $2^{4.42} \cdot (2^{32} + 2^{32}) = 2^{37.42}$ reconstruction queries. The data complexity to recover all bytes of $\widehat{K}^{14}$ is $4 \cdot 2^{37.42} = 2^{39.42}$.

To recover one column of $\widehat{K}^{14}$, we spend 2^{45} encryptions to discard 2^{45} wrong-key candidates. The time complexity to recovery all bytes of $\widehat{K}^{14}$ is $4 \cdot 2^{45} = 2^{47}$.

For the memory complexity, we use the 40-bit counter to record wrong-key candidates, which is equivalent to 2^{33} AES states. To recover 1 column of $\widehat{K}^{14}$, we also need to store 2^{33} $\widehat{C}_0$ and 2^{34} pairs satisfying the differences. Hence, the memory complexity is $2^{33} + 2^{33} + 2^{34} = 2^{35}$ AES states.

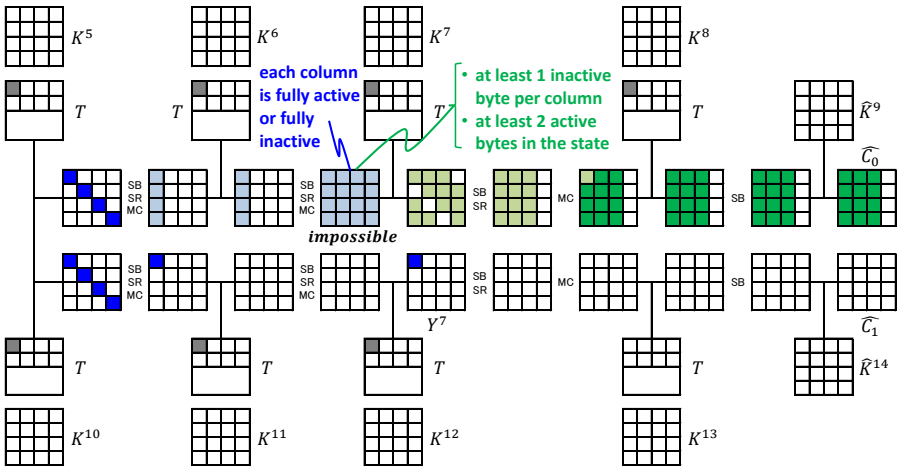

Fig. 3: Impossible-differential distinguisher.

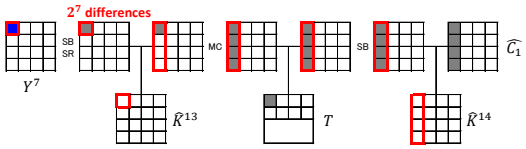

Fig. 4: The appended rounds for key recovery.

5 Yoyo Key-recovery Attack on ForkAES-∗-3-3

The yoyo game was introduced by Biham et al. against Skipjack [3]. Rønjom et al. [20] reported deterministic distinguishers for two generic Substitution-Permutation (SP) rounds. We review existing work in Appendix A. Here, we observe that, during reconstruction queries, two-round decryption and two-round encryption can be computed independently for each column, which we call a *MegaSBox*.

MegaSBox in ForkAES. Refer to Fig. 5 for the MegaSBox construction of ForkAES. Consider any inverse diagonal in $Y^{6,\mathsf{SR}}$. After SR^{-1} and SB^{-1}, the MegaSBox aligns to a column. After MC^{-1}, the column remains independent of the other columns. The inverses of SR and SB align the bytes back into a diagonal. After the reflection, the same operations are applied to these four bytes; after SR, those bytes align to an inverse diagonal in $X^{6,\mathsf{SR}}$. Clearly, the value in this inverse diagonal depends only on the same inverse diagonal in $Y^{6,\mathsf{SR}}$. This can be considered as a MegaSBox with 32-bit input (inverse diagonals). The transition from $Y^{6,\mathsf{SR}}$ to $X^{6,\mathsf{SR}}$ can be depicted in terms of 4 parallel MegaSBoxes. To be explicit, for $x \in \{0,1\}^{32}$, the computation of MegaSbox is defines as $\mathsf{MegaSBox}(x) := \mathsf{SR} \circ \mathsf{SB} \circ \mathsf{ATK} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB} \circ \mathsf{ATK} \circ \mathsf{ATK}^{-1} \circ \mathsf{SB}^{-1} \circ \mathsf{SR}^{-1} \circ$

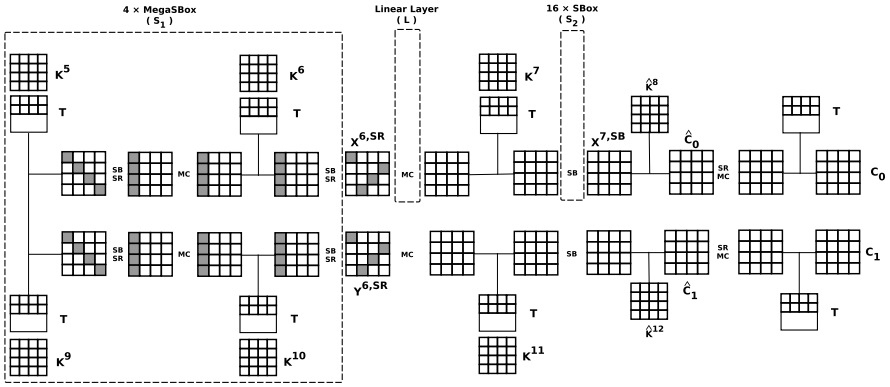

Fig. 5: MegaSBox of ForkAES

$\mathsf{MC}^{-1} \circ \mathsf{ATK}^{-1} \circ \mathsf{SB}^{-1} \circ \mathsf{SR}^{-1}(x)$, where ATK denotes the addition of a round key and a tweak.

Key-recovery Attack. For applying the Yoyo game on ForkAES-$*$-3-3, $S_1 \cdot L \cdot S_2$ needs to be identified. Referring to Fig. 5, following MC and SB of $X^{6,\mathsf{SR}}$ can be regarded as L and S_2 layers respectively. Four MegaSBoxes act as S_1 layer. Thus the operations from $Y^{6,\mathsf{SR}}$ to $X^{7,\mathsf{SB}}$ constitute the $S_1 \cdot L \cdot S_2$ construction. We choose a pair of texts (x_1, x_2) in $Y^{6,\mathsf{SR}}$ and compute $X^{7,\mathsf{SB}}$; bytes are swapped among the texts in $X^{7,\mathsf{SB}}$ and their corresponding values in $Y^{6,\mathsf{SR}}$ are calculated as (x_1', x_2'). Theorem 1 in Appendix A ensures that $\nu(x_1 \oplus x_2) = \nu(x_1' \oplus x_2')$.[7] Refer to Fig. 6 for the attack. It starts with activating one column at $\widehat{C}_1$ for a pair of texts and queries the reconstruction algorithm for a pair of $\widehat{C}_0$. We use the propagation $4 \xrightarrow{\mathsf{MC}^{-1}} 1$ in $Y^{6,\mathsf{SR}}$, which activates a single MegaSBox with probability 2^{-22}. Due to the MegaSBox, only one SuperSbox (inverse diagonal) is active in $X^{6,\mathsf{SR}}$. Out of 4 bytes of the inverse diagonal, at the cost of 2^{-6}, we get one inactive byte. Thus, $\widehat{C}_0$ has one inactive column with probability 2^{-28}.

Attack Procedure.

1. Choose a tweak; choose $2^{14.5}$ distinct random values for the first column of $\widehat{C}_1$. Fix the other 12 bytes to arbitrary value. Obtain the corresponding $\widehat{C}_0$ via reconstruction queries. After this step, we have about 2^{28} pairs of $\widehat{C}_0$.
2. For each of the 2^{28} pairs of $\widehat{C}_0$, check if one column is inactive or not for the pair; we expect one right pair. Once a right pair is obtained, swap the bytes at $\widehat{C}_0$ for applying the yoyo trick and reconstruction algorithm is queried to get a pair which is fully active in $\widehat{C}_1$. We retrieve two such pairs (right pairs).

[7] ν is a so-called zero-differential pattern that denotes the position of inactive words. Refer to Appendix A for more precise definition.

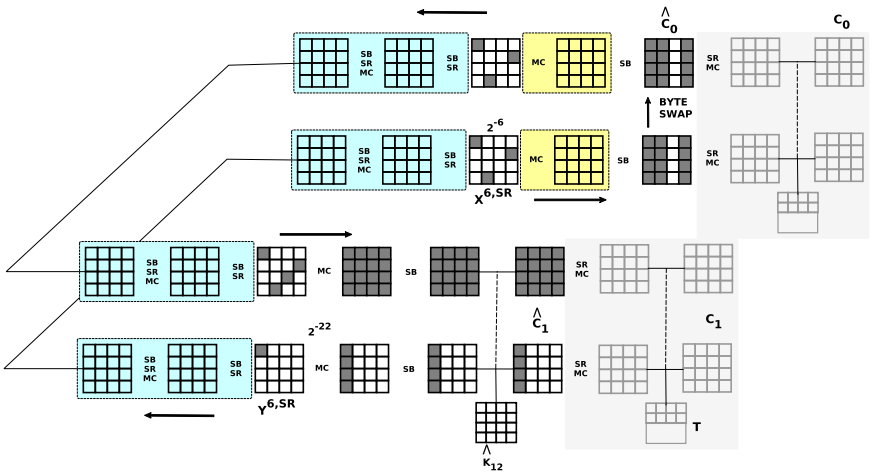

Fig. 6: Yoyo Key Recovery for ForkAES-∗-3-3.

3. For both right pairs, obtain $\widehat{K}^{12}$ that have only 1 active byte in the first column, e.g., by exhaustively guessing a single-byte difference before MixColumns and propagate them through MixColumns. Each right pair suggests 2^{10} key candidates. By analyzing 2 right pairs, the key will be uniquely fixed.
4. Step 3 is iterated for the remaining columns.

Complexity Evaluation and Experimental Verification. The attack needs $2^{14.5}$ reconstruction queries; its time complexity is $2^{14.5}$ memory accesses, and the memory complexity is $2^{14.5}$ AES states for $2^{14.5}$ values $\widehat{C}_0$.

We verified the attack on ForkAES-∗-3-3 by implementing it in Java. The attack started with initializing an oracle that randomly chooses a key, before the steps in the attack procedure above were followed. In the key-recovery phase, two right pairs were used to retrieve candidates for each column of $\widehat{K}^{12}$. Using the first right pair yielded 976, 1296, 1008, and 976 candidates for Column 0, 1, 2, and 3, respectively. The second right pair reduced the candidates to 1, 1, 2, and 1, respectively. Hence, we obtained two key candidates.

6 Rectangle Attack with Encryption Queries

This section describes a rectangle attack on ForkAES-∗-4-4; for concreteness, we exemplify it for five top rounds. Briefly spoken, boomerangs and rectangles are types of differential cryptanalysis where a given cipher E is split into sub-ciphers $E = E_2 \circ E_m \circ E_1$ such that there exist a differential $\alpha \to \beta$ with probability p over E_1, a middle trail $\beta \to \gamma$ with probability r, and a differential $\gamma \to \delta$ with probability q over E_2. Note that, we approximate the middle part E_m to be

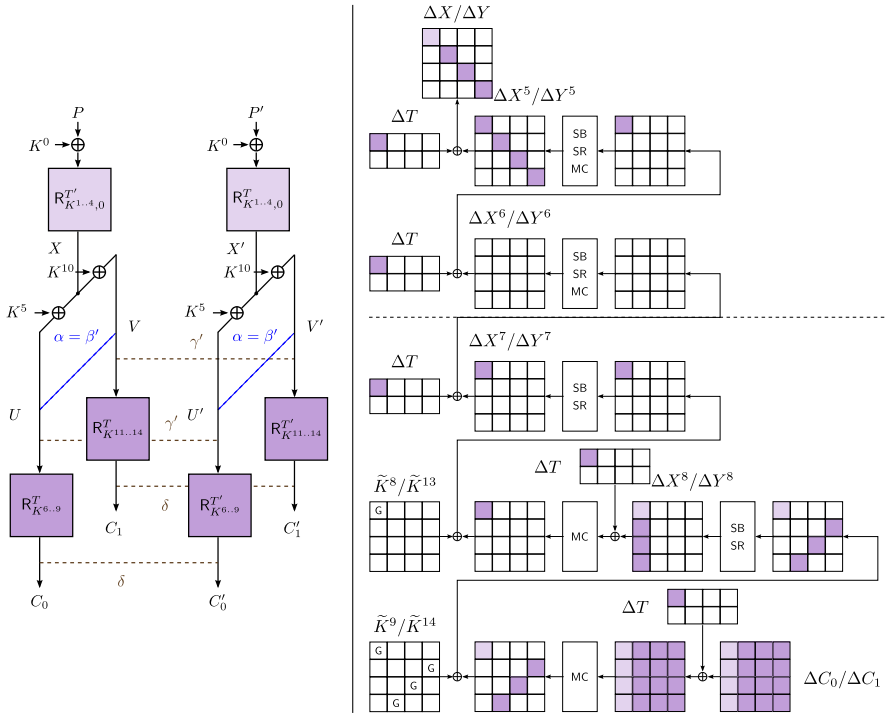

Fig. 7: Overview (**left**) and bottom trail (**right**) of our rectangle attack. The key recovery covers the parts below the dashed horizontal line and guesses the bytes with G.

empty for our attack. The differentials are often referred to as upper and lower differentials or trails. The probability of a correct quartet is often approximated by $r(pq)^2$ since the trails must hold for both pairs.

We consider two tuples (P, T) and (P', T') that are encrypted to (C_0, C_1) and (C'_0, C'_1), respectively. We denote by $\Delta X^r = X^r \oplus X'^r$ their differences between the states after Round r that lead to C_0, and by $\Delta Y^r = Y^r \oplus Y'^r$ the differences in the states that lead to C_1. For clarity, we define that the fork from X to C_0 employs the round keys K^5 through K^9, and the fork from Y to C_1 uses K^{10} through K^{14}. An overview is depicted on the left side of Fig. 7. There, $\mathsf{R}^T_{K^{i..j}}$ means the round sequence $\mathsf{R}^T_{K^j} \circ \cdots \circ \mathsf{R}^T_{K^i}$. We construct 2^8 sets $\mathcal{S}^i$ of 2^s plaintext-tweak tuples. The sets differ in $T[0]$; all plaintexts in a set share the same tweak. So, we can combine 2^s texts (tuples of C_0, C_1) of Set $\mathcal{S}^i$ with 2^s texts of Set $\mathcal{S}^j$, for $i \neq j$, or $2^s \cdot \binom{2^8}{2} \simeq 2^{2s+15}$ pairs (quartets of C_0, C_1, C'_0, C'_1).

The Top Differential. In contrast to the pure AES or to KIASU-BC, the forking step guarantees that the difference between the inputs to Rounds 6 and 10 is equal for each plaintext. So, the top differential reduces to the key addition, that is, the XOR with K^5 for the branch that encrypts from X to C_0, and to the XOR with K^{10} for the branch that encrypts from Y to C_1. So, $\alpha = \beta = K^5 \oplus K^{10}$ holds with probability one for each pair. The adversary collects pairs and waits

that the difference at the beginning of the bottom trail occurs, whose probability can be approximated by 2^{-128}. From approximately 2^{2s+15} pairs, we expect 2^{2s-113} to have a specific difference γ at the forking step.

For the Middle Phase and the Bottom Differential, we use two simplifying assumptions: (1) all differences after five rounds are equally possible; (2) all four-byte values of the keys $K^5[0,5,10,15]$ and $K^{10}[0,5,10,15]$ are equally possible. The bottom trail is shown on the right side of Fig. 7. There are four active S-boxes at the start of Round 6. We consider only text pairs with a non-zero tweak difference $\Delta T[0]$. To estimate the probability, we iterate over all possible values of $X^{6,\mathsf{SB}}[0,5,10,15] = (\bar{x}_0, \bar{x}_1, \bar{x}_2, \bar{x}_3)$, all differences $K^5[0,\ 5,\ 10,\ 15] \oplus K^{10}[0,\ 5,\ 10,\ 15] = (\beta_0,\ \beta_1,\ \beta_2,\ \beta_3)$ and all non-zero 255 tweak differences $\Delta T[0] \neq 0$; $\Delta T[0]$ maps uniquely through MC^{-1} to the differences in $X^{6,\mathsf{SB}}[0,\ 5,\ 10,\ 15] \oplus X'^{6,\mathsf{SB}}[0,\ 5,\ 10,\ 15]$; the same difference must hold between the terms $Y^{6,\mathsf{SB}}[0,5,10,15] \oplus Y'^{6,\mathsf{SB}}[0,\ 5,\ 10,\ 15]$. We define $\mathsf{MC}^{-1}((\Delta T[0],0,0,0)) = (\zeta_0,\ \zeta_1,\ \zeta_2,\ \zeta_3)$. Note that ζ_0 defines ζ_1, ζ_2, and ζ_3 uniquely. Moreover, $(\bar{x}_0,\ \bar{x}_1,\ \bar{x}_2,\ \bar{x}_3,\ \zeta_0,\ \beta_0,\ \beta_1,\ \beta_2,\ \beta_3)$ are mutually independent. This is the setting as in the Boomerang-connectivity Table [10] whose entries contain the number of values x_i for a pair (ζ_i, β_i) that satisfy the boomerang switch for a byte. So, the BCT values already sum over all values x_i. Over all choices of the values $\bar{x}_i$, all non-zero differences ζ_i, and non-zero differences β_i, we obtain a probability of

$$\frac{1}{255\cdot(256)^8}\sum_{\zeta_0\neq 0}\sum_{\beta_0}(\Pr[\zeta_0]\cdot\Pr[\beta_0]\cdot\mathsf{BCT}(\beta_0,\zeta_0))\cdot\sum_{\beta_1}(\Pr[\beta_1]\cdot\mathsf{BCT}(\beta_1,\zeta_1))\cdot$$
$$\sum_{\beta_2}(\Pr[\beta_2]\cdot\mathsf{BCT}(\beta_2,\zeta_2))\cdot\sum_{\beta_3}(\Pr[\beta_3]\cdot\mathsf{BCT}(\beta_3,\zeta_3)) = \frac{(520)^4}{255\cdot 256^8} \simeq 2^{-35.905}.$$

Here, we use the fact that each row and column of the BCT sums to 520 for the AES S-box. So, the probability for the switch can be approximated by $2^{-36}\cdot 2^{-128}$ for hitting our difference between two queries. The remainder in the bottom trail holds with probability 1. Thus, we can expect about 2^{2s-149} correct pairs.

Offline Preparations. We define a linear map $F : \mathbb{F}_{2^8}^{4\times 4} \to \mathbb{F}_{2^8}^{12}$ that returns the value of the 12 inactive bytes in $\Delta X^{9,\mathsf{SR}}$. So, we can identify pairs (C_i, C_i') with our desired difference from collisions between $F(\mathsf{MC}^{-1}(T \oplus C_b)) = F(\mathsf{MC}^{-1}(T' \oplus C_b'))$ with two evaluations of F per text instead of comparing all differences.

We can perform another step for saving efforts later. We define $\widetilde{X}^{r,\mathsf{SR}} =^{\text{def}} \mathsf{SR}(\mathsf{SB}(X^{r-1})) \oplus \widetilde{K}^r$, and $\widetilde{Y}^{r,\mathsf{SR}}$, $\widetilde{X'}^{r,\mathsf{SR}}$, and $\widetilde{Y'}^{r,\mathsf{SR}}$ analogously. Let $x = \widetilde{X}^{9,\mathsf{SR}}[0,\ 7,\ 10,\ 13]$, $x' = \widetilde{X'}^{9,\mathsf{SR}}[0,7,10,13]$, $k^8 = \widetilde{K}^8[0]$, and $k^9 = \widetilde{K}^9[0,7,10,13]$ be short forms. We construct a hash map $\mathcal{H} : \mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \times \mathbb{F}_{2^8}^4 \times \mathbb{F}_{2^8}^4 \to \left(\mathbb{F}_{2^8}^5\right)^*$ such that for all inputs $(T[0], T'[0], x, x')$, $\mathcal{H}$ returns exactly those keys (k^8, k^9) that map x and x' to a zero difference at $\Delta X^{7,\mathsf{MC}}$. The trail contains 32 bit conditions that have to be fulfilled; thus, $\mathcal{H}$ maps to approximately 2^8 suggestions of 40 key bits on average. $\mathcal{H}$ can be used also to obtain suggestions for $\widetilde{K}^{13}[0]$ and $\widetilde{K}^{14}[0,7,10,13]$ from inputs $Y^{9,\mathsf{SR}}[0,7,10,13]$, $Y'^{9,\mathsf{SR}}[0,7,10,13]$, $T[0]$, and $T'[0]$.

Attack Steps. The steps are as follows:

1. Initialize an empty list $\mathcal{Q}$. Initialize two zeroed lists of byte counters for 40 key bits each: $\mathcal{K}$ for $(\widetilde{K}^8[0], \widetilde{K}^9[0,7,10,13])$, and $\mathcal{L}$ for $(\widetilde{K}^{13}[0], \widetilde{K}^{14}[0,7,10,13])$.
2. Precompute $\mathcal{H}$.
3. Choose an arbitrary base tweak $T \in \mathbb{F}_{2^8}^{2\times 4}$. Construct 2^8 sets $\mathcal{S}^i$. For each set, choose 2^s plaintexts P such that all texts in a set use the same tweak value T. Ask for their 2^{s+8} encryptions (T, C_0, C_1), invert the final tweak addition, and the final $\mathsf{MixColumns}$ operation for each output tuple (C_0, C_1).
4. We define $Q_b = F(\mathsf{MC}^{-1}(T \oplus C_b))$, for $b \in \{0,1\}$. For all ciphertexts, compute Q_0 and Q_1 from C_0 and C_1 and store (T, C_0, C_1, Q_0, Q_1) into buckets of $\mathcal{Q}$.
5. Focus on pairs of tuples (T, C_0, C_1, Q_0, Q_1) and $(T', C_0', C_1', Q_0', Q_1')$ if $T[0] \neq T'[0]$, $C_0 = C_0'$ and $C_1 = C_1'$. We call such pairs of tuples with our desired property **correct pairs**. Discard all tuples that do not form correct pairs.
6. For each correct pair, lookup in $\mathcal{H}$ suggestions for the 40 key bits $\widetilde{K}^8[0]$ and $\widetilde{K}^9[0, 7, 10, 13]$ from $T[0]$, $T'[0]$, $\widetilde{X}^{9,\mathsf{SR}}[0, 7, 10, 13]$, and $\widetilde{X'}^{9,\mathsf{SR}}[0, 7, 10, 13]$. We expect 2^8 suggestions on average. For each suggested key candidate, increment its corresponding counter in $\mathcal{K}$.
7. Similarly, for each correct pair, lookup in $\mathcal{H}$ the suggestions for the 40 key bits $\widetilde{K}^{13}[0]$ and $\widetilde{K}^{14}[0, 7, 10, 13]$. We expect 2^8 suggestions on average. For each suggestion, increment the corresponding counter in $\mathcal{L}$.
8. Output the keys in $\mathcal{K}$ and $\mathcal{L}$ in descending order of their counters.
9. While the adversary has 80 key bits, the key schedule may render it more performant to start from the 40 bits of either $\widetilde{K}^8[0]$, $\widetilde{K}^9[0, 7, 10, 13]$ or $\widetilde{K}^{13}[0]$, $\widetilde{K}^{14}[0, 7, 10, 13]$ and search the 88 remaining key bits with the given data.

Complexity. From 2^8 sets of 2^s texts each, we expect 2^{2s-149} correct pairs; $s = 77$ is expected to yield about 2^5 correct pairs on average, and needs 2^{85} plaintext-tweak tuples. The time complexity consists of the following terms:

- $\mathcal{H}$ can be precomputed in Step (2) by decrypting one column over two rounds 2^{80} times, which yields at most $2/13 \cdot 1/4 \cdot 2^{80} \simeq 2^{75.3}$ encryption equivalents.
- Step (3) needs 2^{s+8} encryptions of 13 AES rounds each.
- Step (4) employs $2 \cdot 2^{s+8}$ evaluations of F and $2 \cdot 2^{s+8} \cdot (s+8)$ memory accesses (MAs). This step yields $2^{2s+15} \cdot 2^{-192} = 2^{2s-177}$ wrong pairs plus 2^{2s-149} correct pairs on average.
- Step (6) does not need $\mathcal{H}$, but can test the keys on-the-fly, for $2 \cdot 2^5$ states of 2^{40} keys, of 1/4 of the state through two out of 13 rounds. Each surviving pair requires $2 \cdot 2^8$ MAs to $\mathcal{H}$ plus $2 \cdot 2^8$ MAs to $\mathcal{K}$ and $\mathcal{L}$ on average. We expect an average sum of all counters of $2^8 \cdot 2^{2s-149} = 2^{13}$ in each of both lists, distributed normally over the keys. For $s = 77$, we expect $(2^{-23} \cdot 2^8) + 2^5 \cdot 2^8 \simeq 2^{13}$ counters over the 40 key bits on average.

We can expect that the correct keys have a significantly higher number of counts. So, we obtain about $2^{75.3} + 2 \cdot 2^5 \cdot 2^{40} \cdot \frac{1}{4} \cdot \frac{2}{13} + 2^{s+8} + 2 \cdot 2^{s+8} + 2^{88} \simeq 2^{88.5}$ Encryptions and $2 \cdot 2^{s+8} \cdot (s+8) + 2 \cdot 2^{2s-177} \cdot 2 \cdot 2^8 + 2 \cdot 2^5 \cdot 2^8 \simeq 2^{92.4}$ MAs. The attack needs 2^{80} byte counters for the keys; $\mathcal{Q}$ needs $2^{s+8} \cdot (2 \cdot 16 + 8) < 2^{s+13.33} \simeq 2^{90.4}$ bytes of memory, or $2^{86.4}$ states, which dominates the memory complexity.

7 Impossible-differential Attack with Encryption Queries

Impossible Differentials. This section outlines an impossible-differential attack on ForkAES-$*$-4-4. Again, we describe it for five top rounds. The high-level idea is straight-forward: The adversary queries plaintexts under tweaks that differ only in $T[0]$ and waits for tuples $(C_{i,0}, T_i)$ and $(C_{j,0}, T_j)$. It inverts the final MC^{-1} operation and tweak addition, and uses the ciphertexts only if their difference $\Delta\widetilde{C}_0$ (before MC) activates only the inverse diagonal $\mathcal{ID}_0$, as given in the left side of Fig. 8. It deduces those key bytes $\widetilde{K}^9[0, 7, 10, 13]$ and $\widetilde{K}^8[0]$ that lead to a zero difference in $\Delta X^{7,\mathsf{MC}}$, i.e., that cancel after the tweak XOR at the end of Round 7. Then, there is a zero difference through the inverse Round 7, which leads to a single active byte in $\Delta X^{6,\mathsf{MC}}$, and to a single active diagonal at the start of Round 6. Again, see the left side of Fig. 8. The second trail decrypts ΔC_1 backwards to $\Delta Y = \Delta X$. So, at least one of the following cases must hold:

(1) ΔY^7 has at least one fully active column: $\Delta Y^7 \in \mathcal{C}_i$.
(2) Bytes $\Delta Y^7[1, 2, 3]$ are active.
(3) $\Delta C_1 \in \mathcal{M}_0$, i.e., is in the mixed space, generated by $\Delta Y^{9,\mathsf{SR}} \in \mathcal{ID}_0$.

In Case (3), the ΔY trail is similar to the ΔX trail. So, we have a distinguisher similar to the rectangle distinguisher described in Section 6. However, this section tries to exploit a different distinguisher with lower data complexity and does not have to wait for such an event. In the Cases (1) and (2), the Columns 1 to 3 of ΔY^7 are either completely active or completely inactive. Thus, the adversary can guess eight bytes of $\widetilde{K}^{14}$ that are mapped to one of those columns and can filter out all key guesses where one of those columns would become partially active.

Offline Preparations. Again, we can define a linear map F of rank 96 such that $F(\mathsf{MC}^{-1}(\Delta C_0 \oplus \Delta T)) = 0$ so that we can identify pairs with our desired difference from collisions in $\Delta\widetilde{X}^{9,\mathsf{SR}}$. We construct a hash map $\mathcal{H}_0 : \mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \times \mathbb{F}_{2^8}^4 \times \mathbb{F}_{2^8}^4 \to (\mathbb{F}_{2^8}^5)^*$ that maps $(T[0], T'[0], \widetilde{X}^{9,\mathsf{SR}}[0, 7, 10, 13], \widetilde{X'}^{9,\mathsf{SR}}[0, 7, 10, 13])$ to all five-byte keys that yield $\Delta X^{7,\mathsf{MC}} = 0$.

We construct a second hash map $\mathcal{H}_1 : \mathbb{F}_{2^8}^8 \times \mathbb{F}_{2^8}^8 \to (\mathbb{F}_{2^8}^8)^*$. For all inputs $x = (\widetilde{Y}^{9,\mathsf{SR}}[2, 3, 5, 6, 8, 9, 12, 15], \widetilde{Y'}^{9,\mathsf{SR}}[2, 3, 5, 6, 8, 9, 12, 15])$, $\mathcal{H}_1(x)$ returns exactly the keys $\widetilde{K}^{14}[2, 3, 5, 6, 8, 9, 12, 15]$ that yield one of the impossible differentials in $\Delta Y^{8,\mathsf{SR}}$. $\mathcal{H}_1$ does not need the tweak as input since the final tweak addition, MixColumns, and ShiftRows can be inverted before the lookup in $\mathcal{H}_1$; the tweak addition at the end of Round 8 does not affect the difference in $\Delta Y^{8,\mathsf{SR}}$. Note that $\mathcal{H}_1$ can be built more efficiently from several smaller lookup tables since the columns can be computed independently from each other.

There exist four combinations of bytes $\Delta Y^{8,\mathsf{SR}}[i, j]$ with $(i, j) \in \{(8, 15), (9, 12), (10, 13), (11, 14)\}$ and two options if Byte i or Byte j is active. Among 2^{32} difference inputs to MC^{-1}, 2^{24} are mapped to an output difference with a zero-difference byte at a fixed index. On the other hand, $2^{32} - 2^{24}$ inputs yield a non-zero difference at a given byte index. Thus, given an input $\widetilde{Y}^{9,\mathsf{SR}}$, $\mathcal{H}_1$ returns $4 \cdot 2$ combinations of $2^{24} \cdot (2^{32} - 2^4) \simeq 2^{56}$ keys that yield the

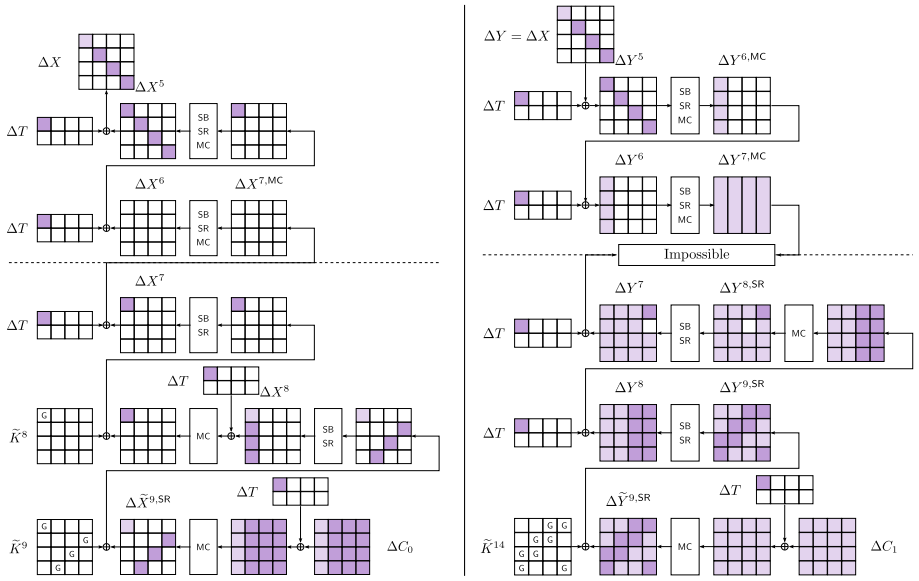

Fig. 8: Left: The trail $\Delta C_0 \to \Delta X$. **Right:** One variant of an impossible trail $\Delta C_1 \not\leftarrow \Delta Y$. White bytes are inactive, light-blue bytes possibly active, and dark-blue bytes are active. Parts below the dashed horizontal lines are considered in the on-line phase. The columns of $\Delta Y^{7,\mathsf{MC}}$ are either fully active or fully inactive.

impossible differential. This can be evaluated with $4 \cdot 2$ calls to two 32-bit tables each, or 16 tables that map 32 state bits to 2^{32} or 2^{24} keys. So, $\mathcal{H}_1$ needs $8 \cdot 2^{32} \cdot 2^{32} \cdot 4$ bytes $+ 8 \cdot 2^{32} \cdot 2^{24} \cdot 4$ bytes $\simeq 2^{72}$ bytes of memory. The tables can be computed with at most $16 \cdot 2^{32} \cdot 2^{32}$ quarter-rounds of the AES, which is at most $16/13 \cdot 2^{64} \simeq 2^{64.3}$ equivalents of ForkAES-5-4-4.

Attack Procedure. The steps in the attack are as follows:

1. Initialize two empty lists $\mathcal{Q}$ and $\mathcal{K}$; the latter will hold all 13-byte keys $\widetilde{K}^8[0]$, $\widetilde{K}^9[0, 7, 10, 13]$, and $\widetilde{K}^{14}[2, 3, 5, 6, 8, 9, 12, 15]$.
2. Choose an arbitrary base tweak $T \in \mathbb{F}_{2^8}^{2\times 4}$. Construct 2^8 sets $\mathcal{S}^i$ from iterating over $T[0]$. For each set, choose 2^s plaintexts P. All texts in a set use the same tweak T^i with $T^i[0] = i$. Ask for their 2^{s+8} encryptions (T, C_0, C_1).
3. For each ciphertext, invert the final tweak addition, the final MC operation, and process all ciphertexts by F: $Q_b = F(\mathsf{MC}^{-1}(C_b \oplus T))$, for $b \in \{0, 1\}$. Store (T, C_0, C_1, Q_0, Q_1) into buckets of $\mathcal{Q}$.
4. Only consider pairs of tuples (T, C_0, C_1, Q_0, Q_1) and $(T', C_0', C_1', Q_0', Q_1')$ if $T \neq T'$ and $Q_0 = Q_0'$. Discard all other tuples. We call pairs of tuples with our desired property **correct pairs**.
5. For each correct pair, derive from $\mathcal{H}_0$ the key candidates $\widetilde{K}^8[0]$ and $\widetilde{K}^9[0, 7, 10, 13]$ that yield a zero difference in $\Delta X^{7,\mathsf{MC}}$. With $\widetilde{Y}^{9,\mathsf{SR}}$ and $\widetilde{Y'}^{9,\mathsf{SR}}$, further

derive from $\mathcal{H}_1$ all key candidates for $\widetilde{K}^{14}[2,3,5,6,8,9,12,15]$ that yield one of the impossible differentials. Remove those candidates from $\mathcal{K}$.
6. Output the 13-byte key candidates remaining in $\mathcal{K}$.

Conditions and Complexities. The adversary queries 2^8 sets of 2^s texts each and guesses 13 key bytes in total: $\widetilde{K}^9[0,7,10,13]$, $\widetilde{K}^8[0]$, and $\widetilde{K}^{14}[2, 3, 5, 6, 8, 9, 12, 15]$, i.e., 104 key bits. The attack requires pairs with $\Delta C_0 \in \mathcal{M}_0$, which occurs with probability of approximately $p \simeq 2^{-96}$. We can assume that $(\Delta C_0, \Delta C_1) \in \mathcal{M}_0 \times \mathcal{M}_0$ never occurs by accident; while it could theoretically still occur and could be exploited, we consider a different distinguisher here.

The probability that a key $\widetilde{K}^9[0,7,10,13]$ reduces the four active bytes in $\Delta X^{9,\mathsf{SR}}$ to a single active byte in $\Delta X^{8,\mathsf{MC}}[0]$ is 2^{-24}, and its difference is $\Delta T[0]$ in ΔX^7 with probability 2^{-8}. So, a key in the ΔX trail yields our desired differential with a probability of about 2^{-32}. There are four options which columns in ΔY^7 become partially active, and two options for the order which of the two known bytes in this column are active/inactive. The probability for one inactive byte is $(2^{-8}-1)\cdot(1-2^{-8}) \simeq 2^{-8}$; so, a key yields the impossible differential in $\Delta Y^{7,\mathsf{MC}}$ with probability approximately $2^{-32}\cdot 2^{-8}\cdot 4\cdot 2 \simeq 2^{-37}$.

In the framework by Boura et al. [9], this can be represented as 37 bit conditions that have to be fulfilled to filter a key from a given correct pair. The probability for a wrong key to survive is $p_{\mathsf{survive}} = \left(1-2^{-37}\right)^N$, where N is the number of correct pairs. For 2^{104} keys, $p_{\mathsf{survive}} \leq 2^{-104}$ would allow us to filter all keys to only the correct key, plus at most a few more false positives. For this purpose, we need $N \geq 2^{43.2}$ pairs with 12 inactive bytes in $\Delta\widetilde{X}^{9,\mathsf{SR}}$, which yields $2^{43.2}\cdot 2^{12\cdot 8} = 2^{139.2}$ necessary pairs. From 2^s structures, we can construct about 2^{2s+15} pairs, which gives $s = 62.1$ or $C_N = 2^{s+8} = 2^{70.1}$ queries. The computational complexity is composed of the following terms:

- Precompute $\mathcal{H}_0$ with 2^{80} times twice a quarter round of the AES, which can be approximated by $2^{80}\cdot 2/13\cdot 1/4 \simeq 2^{75.3}$ encryption equivalents.
- Precompute $\mathcal{H}_1$ with at most $2^{64.3}$ encryption equivalents.
- Encrypt 2^{s+8} plaintext-tweak tuples.
- Invert $2^{s+8}\cdot 2$ times the final tweak addition, MixColumns, and ShiftRows operation, which can be overestimated by $2^{70.1}\cdot 2\cdot 1/13 \approx 2^{67.5}$ encryptions.
- Apply F to all states C_0, which is at most 2^{s+8} ForkAES computations, or $2^{70.1}\cdot 2 \simeq 2^{70.1}$ encryptions. Moreover, we need $2\cdot 2^{s+8}\cdot(s+8) = 2\cdot 70.1\cdot 2^{70.1} \simeq 2^{77.3}$ MAs on average with an efficient data structure. We obtain about $2^{2s+15-96} \simeq 2^{2s-81} = 2^{43.2}$ remaining pairs.
- For each of the $2^{43.2}$ pairs allows to filter keys. Since we have 37 bit conditions, each pair allows to filter $2^{104-37} = 2^{67}$ keys on average from $\mathcal{H}_0$ and $\mathcal{H}_1$ with two MAs each and remove them from $\mathcal{K}$.
- Our attack aims at recovering 104 bits of $\widetilde{K}^9$ and $\widetilde{K}^{14}$. So, the final term for recovering 64 remaining key bits of $\widetilde{K}^{14}$ can be estimated by 2^{64} encryptions.

The time complexity can be bounded by about $2^{75.3}+2^{64.3}+2^{70.1}+2^{67.5}+2^{70.1}+2^{64} \simeq 2^{75.4}$ encryptions and $2\cdot 2^{70.1}+2^{77.3}+2^{43.2}\cdot 2+2^{43.2}\cdot 2^{67} \simeq 2^{110.2}$ MAs.

The attack needs $2^{80} \cdot 2^8 \cdot 40$ bits for $\mathcal{H}_0$, at most 2^{72} bytes for the components of $\mathcal{H}_1$, $2^{s+8} = 2^{70.2} \cdot (2 \cdot 16 + 8) < 2^{s+14} = 2^{76.2}$ bytes for $\mathcal{Q}$, and 2^{104} byte counters or (2^{100} states) for $\mathcal{K}$; the latter term dominates the memory complexity.

Acknowledgments. Parts of this work have been initiated during the group sessions of the 8th Asian Workshop on Symmetric Cryptography (ASK 2018) held at the Indian Statistical Institute in Kolkata. We would also like to thank the anonymous reviewers and the designers of ForkAES for their helpful comments. Subhadeep Banik is supported by the Ambizione Grant `PZ00P2_179921`, awarded by the Swiss National Science Foundation.

References

1. Elena Andreeva, Reza Reyhanitabar, Kerem Varici, and Damian Vizár. Forking a Blockcipher for Authenticated Encryption of Very Short Messages. IACR Archive, 2018. `https://eprint.iacr.org/2018/916`, Version: 20180926:123554.
2. Subhadeep Banik, Jannis Bossert, Amit Jana, Eik List, Stefan Lucks, Willi Meier, Mostafizar Rahman, Dhiman Saha, and Yu Sasaki. Cryptanalysis of ForkAES. Cryptology ePrint Archive, Report 2019/289, 2019. `https://eprint.iacr.org/2019/289`.
3. Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In Stafford E. Tavares and Henk Meijer, editors, *SAC*, volume 1556 of *LNCS*, pages 362–376. Springer, 1998.
4. Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.
5. Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001.
6. Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *LNCS*, pages 1–16. Springer, 2002.
7. Céline Blondeau. Accurate Estimate of the Advantage of Impossible Differential Attacks. *IACR Trans. Symmetric Cryptol.*, 2017(3):169–191, 2017.
8. Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the Impossible Possible. *J. Cryptology*, 31(1):101–133, 2018.
9. 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 (1)*, volume 8873 of *LNCS*, pages 179–199. Springer, 2014.
10. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT II*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018.
11. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
12. Patrick Derbez. Note on Impossible Differential Attacks. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 416–427. Springer, 2016.

13. Christoph Dobraunig and Eik List. Impossible-Differential and Boomerang Cryptanalysis of Round-Reduced KIASU-BC. In Helena Handschuh, editor, *CT-RSA*, volume 10159 of *LNCS*, pages 207–222. Springer, 2017.
14. Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace Trail Cryptanalysis and its Applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2):192–225, 2016.
15. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT (2)*, volume 8874 of *LNCS*, pages 274–288, 2014.
16. Orhun Kara. Reflection Cryptanalysis of Some Ciphers. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *LNCS*, pages 294–307. Springer, 2008.
17. Lars Knudsen. DEAL – A 128-bit block cipher. *Complexity*, 258(2):216, 1998.
18. Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Trans. Information Theory*, 57(4):2517–2521, 2011.
19. National Institute of Standards and Technology. FIPS 197. *National Institute of Standards and Technology, November*, pages 1–51, 2001.
20. Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT I*, volume 10624 of *LNCS*, pages 217–243. Springer, 2017.
21. Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. A Meet in the Middle Attack on Reduced Round Kiasu-BC. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E99-A(10):21–34, Oct 2016.
22. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

Appendices

A Methods

This section briefly recalls the necessary details on (1) boomerang and rectangle attacks, (2) impossible-differential attacks, and (3) yoyo attacks.

A.1 Boomerang and Rectangle Attacks

Boomerangs Attacks [22] are a form of advanced differential cryptanalysis that allows to compose two short high-probability differentials in cases where long differentials with sufficient probability are lacking. Given a cryptographic transform $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$, it is decomposed into parts $E = E_2 \circ E_1$ such that there exist a differential $\alpha \to \beta$ with probability p over E_1 and a differential $\gamma \to \delta$ with probability q over E_2. Often, the differentials are referred to as upper and lower differentials or trails. A boomerang distinguisher then chooses plaintext pairs (P, P'), with $P' = P \oplus \alpha$, and asks for the corresponding ciphertexts (C, C') through E. It derives $D = C \oplus \delta$ and $D' = C' \oplus \delta$ to obtain the ciphertext pair (D, D'), and asks for the corresponding plaintext tuples (Q, Q'). If $Q \oplus Q' = \alpha$, then (P, P', Q, Q') forms a *correct quartet*. The probability of a correct quartet is often approximated by $(pq)^2$ since the trails must hold for both pairs. The

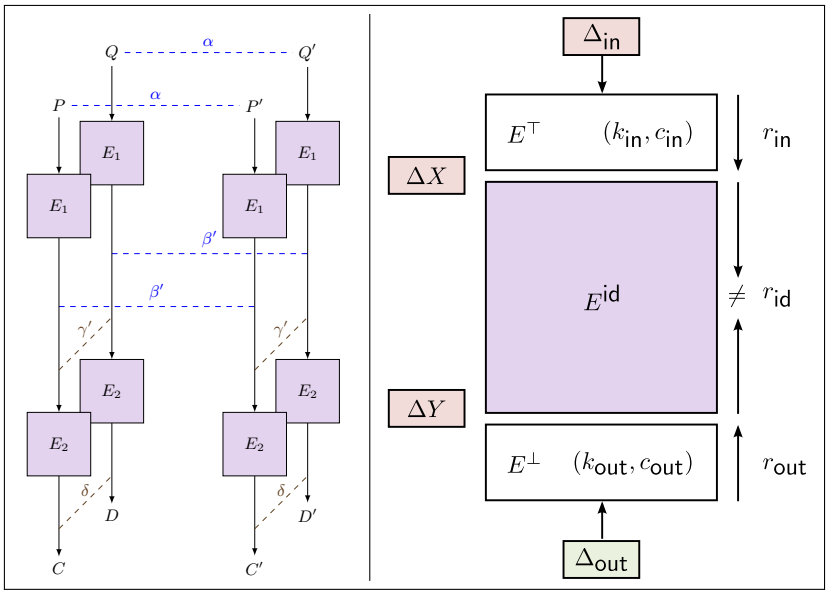

Fig. 9: Schematic illustrations of a related-tweakey rectangle (**left**) and an impossible-differential attack (**right**).

probability can be increased by considering all possible internal trails $\alpha \to \beta'$ and $\gamma' \to \delta$ as long as *both* pairs in the quartet have differences β' and γ' in the middle and $\beta \neq \gamma$. Hence, the simplified probability of a correct quartet increases to $(\widehat{p}\widehat{q})^2$ with

$$\widehat{p} = \sqrt{\sum_{\beta'} \Pr^2\left[\alpha \to \beta'\right]} \quad \text{and} \quad \widehat{q} = \sqrt{\sum_{\gamma'} \Pr^2\left[\gamma' \to \delta\right]}.$$

Clearly, the attack demands that $\widehat{p}\widehat{q} \gg 2^{-n/2}$ for the differential to be probable. Given N plaintext pairs, one expects about $N \cdot (\widehat{p}\widehat{q})^2$ correct quartets in an attack, but only $N \cdot 2^{-n}$ correct quartets for an ideal primitive, which yields a distinguishing event.

Rectangle Attacks. Rectangle attacks [5] are a chosen-plaintext form of the boomerang concept. The core difference of rectangles is to encrypt many plaintext (P, P') with difference α and simply hope that some of those will form a quartet with the desired differences in the middle. Given N plaintext pairs, the number of correct quartets is reduced to $N^2 \cdot 2^{-n} \cdot (\hat{p}\hat{q})^2$ by a birthday argument. The left side of Fig. 9 illustrates a related-tweakey rectangle. Biham et al. presented further technical improvements to the technique in [6]. The major disadvantages of rectangle compared to boomerang attacks are the higher data complexity and the large number of potential quartets that have to be handled.

VERIFICATION. Boomerangs and rectangles represent oversimplified equation systems, as was first stressed by Murphy [18]. Cid et al. [10] proposed the concept of a Boomerang-connectivity Table for S-box-based ciphers as a tool that allows to identify if boomerangs can really hold. We consider their approach later in this work.

A.2 Impossible-differential Attacks

Impossible-differential attacks have been proposed by Knudsen [17] and Biham et al. [4]. In this work, we follow the generic framework by Boura et al. [9]; the interested reader is referred to the refinements by Blondeau [7].

The search of an impossible-differential attack usually consists of two steps: first, one searches for a differential $\Delta X \not\to \Delta Y$ with probability zero through a sub-cipher; thereupon, one propagates the start difference ΔX backwards through r_{in} rounds to an input difference Δ_{in}, and the end difference ΔY forwards through r_{out} rounds to an output difference Δ_{out}. The complexity can be reduced by considering Δ_{in} and Δ_{out} as vector spaces that can contain multiple allowed start and end differences.

The adversary queries plaintexts P_i and constructs pairs (P_i, P_j) s. t. their difference lies in Δ_{in}; it asks for their corresponding ciphertexts C_i, and considers only ciphertext pairs (C_i, C_j) whose differences fall into the space spanned by Δ_{out}. Next, it guesses key bits through the r_{in} inner rounds and the r_{out} outer rounds. All key candidates that yield ΔX and ΔY for any pair in the middle must be wrong and can be discarded. Boura et al. denoted by c_{in} the number of bit conditions that have to be fulfilled for a pair with input difference in Δ_{in} to yield ΔX. Similarly, they denoted the number of bit conditions that must be fulfilled to get from Δ_{out} to ΔY by c_{out}. We denote the key sets by $\mathcal{K}_{\mathsf{in}}$ and $\mathcal{K}_{\mathsf{out}}$. The number of key bits involved is denoted by k_{in} and k_{out} respectively. The right image of Fig. 9 comprises the notations of an impossible-differential attack.

The number of pairs needed to filter is chosen such that the probability of a key to survive is low. Following [7,17], let T_K be the random variable that counts the number of pairs that allow discard key candidate K. Knudsen [17] assumed that T_K follows a binomial distribution with parameters $(N, p = 2^{-(c_{\mathsf{in}}+c_{\mathsf{out}})})$ that a pair leads to the impossible differential for a given key, the probability that this key candidate survives all pairs can be approximated by

$$p_{\mathsf{survive}} = \Pr\left[T_K = 0\right] = \binom{N}{0} \cdot p^0 \cdot (1-p)^N = \left(1 - 2^{-(c_{\mathsf{in}}+c_{\mathsf{out}})}\right)^N .$$

This probability can be approximated by e^{-pN}. The complexities are given by:

– Data: $C_N \cdot N$. For obtaining the necessary number of pairs N, Boura et al. [9] formulated the following complexity, based on the limited birthday problem:

$$C_N = \max\left\{ \min_{\Delta \in \{\Delta_{\mathsf{in}}, \Delta_{\mathsf{out}}\}} \left\{ \sqrt{N \cdot 2^{n+1-|\Delta|}} \right\}, N \cdot 2^{n+1-|\Delta_{\mathsf{in}}|-|\Delta_{\mathsf{out}}|} \right\}. \quad (2)$$

– Memory: N pairs;
– Time:

$$C_N \cdot C_E + \left(1 + \frac{2^{|k_{\mathsf{in}} \cup k_{\mathsf{out}}|}}{2^{c_{\mathsf{in}}+c_{\mathsf{out}}}}\right) \cdot N \cdot C_{E'} + 2^{k-\alpha} \cdot C_E. \tag{3}$$

The number of plaintext pairs is chosen s. t. N, the expected number of the ciphertext pairs with difference in Δ_{out} fulfills that $p_{\mathsf{survive}} \leq 2^{-\alpha}$; so, the attack reduces the key spaces by α bits on average. In the most conservative fashion, N is chosen to be smaller than $2^{-|k_{\mathsf{in}} \cup k_{\mathsf{out}}|}$, so that only the correct key is expected to survive. The term C_N simply refers to the complexity of finding N pairs with ciphertext difference in Δ_{out}. C_E is the cost for evaluating the primitive, k denotes the key size, and $C_{E'}$ the costs of the partial encryption to detect impossible differentials.

One can consider multiple impossible differentials to employ the same data for multiple impossible trails. Boura et al. point out that this strategy affects only the first term, C_N, but not the further terms of the memory complexity.

Boura et al. aimed at providing generic formulae for the complexities of impossible-differential attacks. At FSE 2016, Derbez pointed out cases where the generic time-complexity calculation was not correct [12]; those counter-examples considered optimized attacks wherein the key was recovered part by part. In their follow-up work [8], Boura et al. addressed Derbez' findings and emphasized that also given their generic formulae, the exact complexity of each attack needs to be carefully computed. Later in this work, we will refrain from employing those optimizations. Therefore, the complexity calculations for our attacks should be sub-optimal, but circumvent the pitfalls pointed out by Derbez.

A.3 Yoyo Game

The yoyo game was introduced by Biham et al. for the cryptanalysis of Skipjack [3]. Recently, Rønjom et al. [20] reported a deterministic distinguisher for two generic Substitution-Permutation (SP) rounds. This result has been applied to eight-round ForkAES to perform key recovery attack. Before discussing the distinguisher for two generic SP rounds, some notations and definitions need to be reused originally defined in [20].

Let, $F : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be a generic permutation where $q = 2^k$. Then, F is given by:

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

Here, S is considered as a concatenation of n SBoxes operating in parallel on individual *words* from $\mathbb{F}_q$ and L denotes the linear layer over $\mathbb{F}_q^n$. A vector of words $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_{n-1}) \in \mathbb{F}_q^n$ forms the *states*. The Zero-difference Pattern (ZDP) is defined by [20] as below:

Definition 1 (Zero-difference Pattern [20]). 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.

The weight of the zero-difference Pattern of α refers to the number of inactive words in α and is denoted by $wt(\nu(\alpha))$

The Yoyo game depends on the swapping of words among the texts. The following definition describes the swapping mechanism.

Definition 2 (Word Swapping [20]). 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

$$\rho^v(\alpha, \beta)_i = \begin{cases} \alpha_i, & \text{if } v_i = 1; \\ \beta_i, & \text{if } v_i = 0. \end{cases} \tag{4}$$

A.4 Yoyo Distinguisher for two generic SP Rounds

Two generic SP rounds is denoted as $G_2 = L \cdot S \cdot L \cdot S$ where the final L layer can be omitted as it has no effect in terms of security. Also, the two substitution layers does not need to be same. After modification, $G_2 = S_1 \cdot L \cdot S_2$. The deterministic distinguisher for two generic SP rounds is described by the following theorem.

Theorem 1 (The Yoyo Game [20]). 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

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

B Attack on ForkAES-∗-3-3 with Reflection Trails

Refer to Fig. 10 for the differential propagation in reconstruction queries. We use the following propagation of the number of active S-boxes for 2 rounds

$$4 \xrightarrow{F^{-1}} 1 \xrightarrow{F^{-1}} 0 \xrightarrow{\text{reflect}} 0 \xrightarrow{F} 1 \xrightarrow{F} 4,$$

and append 1 round to the end for the key recovery and the structure technique.

We only consider a fixed difference for the tweak, which can be fixed to any value, say 0 x01. The only probabilistic propagation is reduction of the number of active bytes during the inverse of MixColumns and cancellation of state difference with tweak difference. Those occur with probability 2^{-32} in total. The attack procedure is as follows.

1. Choose two tweaks T, T' having the fixed difference. For each of T, T', choose 2^{16} distinct values for the first column of $\widehat{C}_1$. Fix the other 12 bytes to arbitrary value, say 0. Obtain the corresponding $\widehat{C}_0$ via reconstruction queries.
2. From 2^{32} pairs of $\widehat{C}_0$, pick up the one that has 12 inactive bytes in the 2nd, 3rd, and 4th columns of $\widehat{C}_0$. We expect only 1 right pair.
3. By using the right pair, obtain 2^7 key candidates of the first column of $\widehat{K}^8$ as demonstrated for the attack against ForkAES-∗-4-4.

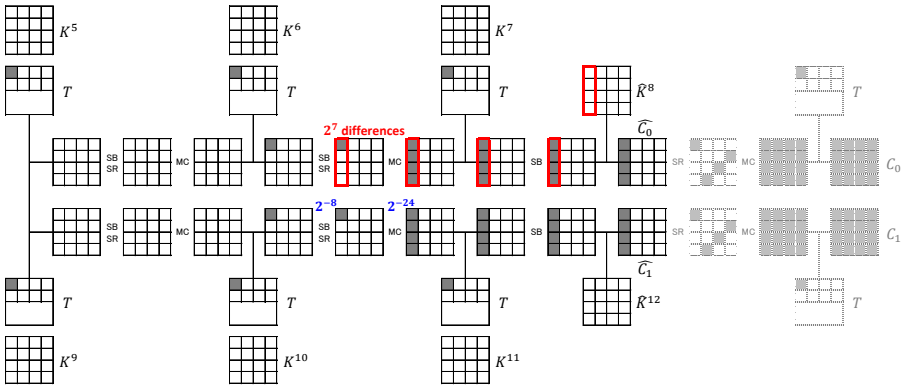

Fig. 10: Truncated Differentials of 8-Round Attack.

4. Iterate the above steps by shifting the active-byte positions.

The attack requires 2^{19} reconstruction queries, 2^{19} memory access to operate queries, and a memory to store 2^{17} AES states.

We implemented the 8-round attack by Java. We first fixed the 128-bit key value to a certain value and ran the procedure above. According to the theory, we should be able to obtain 2^7 key candidates for each column of $\widehat{K}^8$. In our experiment, the number of key candidates for column 0, 1, 2, and 3 are 304, 144, 96 and 208, respectively. Hence, the complexity for the key recovery is $304 \times 144 \times 96 \times 206 \approx 2^{29.7}$. We successfully recovered the correct key by exhaustively trying those $2^{29.7}$ key candidates.

C Forgery Attacks on AE Schemes with ForkAES-∗-4-4

PAEF. The forkcipher designers proposed several dedicated nonce-based authentication modes, among them, PAEF that provides 128-bit security is depicted in Fig. 11. To attack nonce-based AE modes, the difficulty lies in the single use of each nonce, which can prevent many advanced differential based attacks. Indeed, our attacks based on encryption queries assume the repeated use of nonces in those modes. In contrast, our attack here is a (first-order) differential attack that can directly be used to mount forgery attacks.

Description. Let N, A, tw, C, T denote a nonce, an associated data, a tweak, a ciphertext and a tag, respectively. Overall, the attacker first observes a valid tuple of (N, A, tw, C, T). Then she generates (N, A, tw', C', T') that passes the verification procedure. In forkcipher-based modes, one of the two output blocks is the ciphertext block and the other output block is used to update the tag. Hence, the reflection trails for reconstruction queries are suitable for this scenario.

In this attack, we specify exact difference value between tweak cancellation and the last output in both branches as shown in Fig. 12. Because each branch

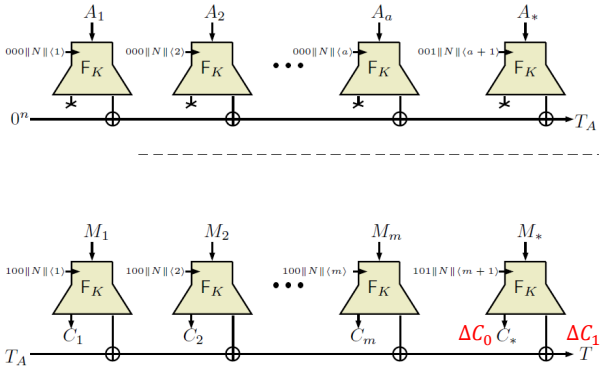

Fig. 11: Nonce-respecting forgery for the PAEF authenticated-encryption mode.

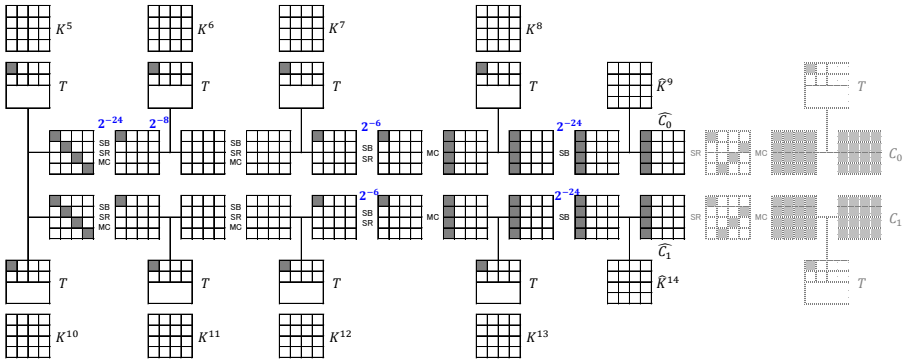

Fig. 12: Differential Tail of 9-Round Forgery in AE Modes.

involves 5 active S-boxes, the probability of the propagation is 2^{-30} in each branch. For the reflection part, we truncate the differences, thus the probability is 2^{-32}. Note that the difference of $\Delta\widehat{C}_0$ and $\Delta\widehat{C}_1$ must be linearly converted to ΔC_0 and ΔC_1 offline. In the end, we know that by modifying the last ciphertext block C_* to $C' = C_* \oplus \Delta C_0$, the tag will be $T' = T \oplus \Delta C_1$ with probability 2^{-92}.

As a result, the attacker can generate valid (N, A, tw', C', T') after 2^{92} distinct queries of (N, A, tw, C, T) with good probability or after 1 query with probability 2^{-92}, which is faster than the claimed 128-bit security.

Remarks. Note that the S-box inverse, key addition, and S-box in the reflection part can be viewed as an application of a key-dependent S-box under a fixed key. Because both of the input and output differences of each key-dependent S-box is fixed, the probability of this propagation can be 0 for some choices of ΔT. To avoid this situation, the attacker should make $2^{92}/255$ queries for each of 255 possible ΔT. Although the probability of being good ΔT is 2^{-4}, the probability

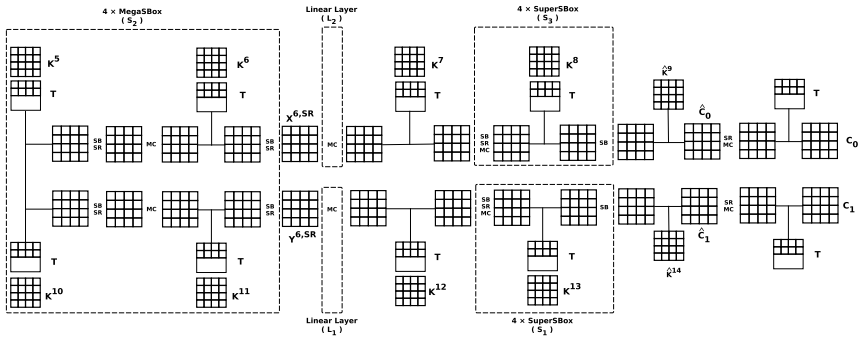

Fig. 13: Construction of $S_1 \cdot L_1 \cdot S_2 \cdot L_2 \cdot S_3$ for ForkAES-$*$-4-4

of cancellation for a good ΔT is 2^{-28}. Therefore, this does not impact to the complexity evaluation.

The designers of forking ciphers also proposed SAEF and fGCM. The same attack can also be applied to SAEF and fGCM because the process of the last message block is identical.

D Impossible-differential Yoyo Distinguisher for ForkAES-$*$-4-4

The computation of the reconstruction queries in ForkAES-$*$-4-4 can be considered as $S_1 \cdot L_1 \cdot S_2 \cdot L_2 \cdot S_3$ (3 generic SP-rounds), where S_1 and S_2 correspond to SuperSbox and MegaSBox respectively. Fig. 13 reveals this construction. Unlike 2 generic SP-rounds, there is no generic distinguisher for 3 generic SP-rounds. Instead, properties specific to AES can be exploited.

The AES states consists of four 32-bit words and the minimum number of active words over $L_1 \cdot S_1$ or $L_2 \cdot S_3$ layer is 5 for AES. Equivalently, the number of maximum inactive words over $L_1 \cdot S_1$ or $L_2 \cdot S_3$ layer is 3 or less for AES. Namely, if $wt(\nu(L_1 \cdot S_1(p^1) \oplus L_1 \cdot S_1(p^2))) = t$, then it is impossible that $wt(\nu(p^1 \oplus p^2)) \geq (4-t)$. Also, $wt(\nu(L_1 \cdot S_1(p^1) \oplus L_1 \cdot S_1(p^2))) = wt(\nu(S_2 \cdot L_1 \cdot S_1(p^1) \oplus S_2 \cdot L_1 \cdot S_1(p^2))) = t$. If p^1 and p^2 encrypt to c^1 and c^2 respectively, then it is also impossible to have $wt(\nu(c^1 \oplus c^2)) \geq (4-t)$.

For any random pair of texts,

$$Pr\big[wt(\nu(L_1 \cdot S_1(p^1) \oplus L_1 \cdot S_1(p^2))) = t\big] = \tbinom{4}{t} \times \tfrac{(2^{32}-1)^{(4-t)}}{2^{32\times 4}} \approx \tbinom{4}{t} \times 2^{32\times -t}.$$

For any random pair of texts also we have

$$Pr\big[wt(\nu(p^1 \oplus p^2)) = (4-t)\big] = Pr\big[wt(\nu(c^1 \oplus c^2)) = (4-t)\big] = \tbinom{4}{4-t} \times \tfrac{(2^{32}-1)^{(t)}}{2^{32\times 4}} \approx \tbinom{4}{4-t} \times 2^{32\times (t-4)}.$$

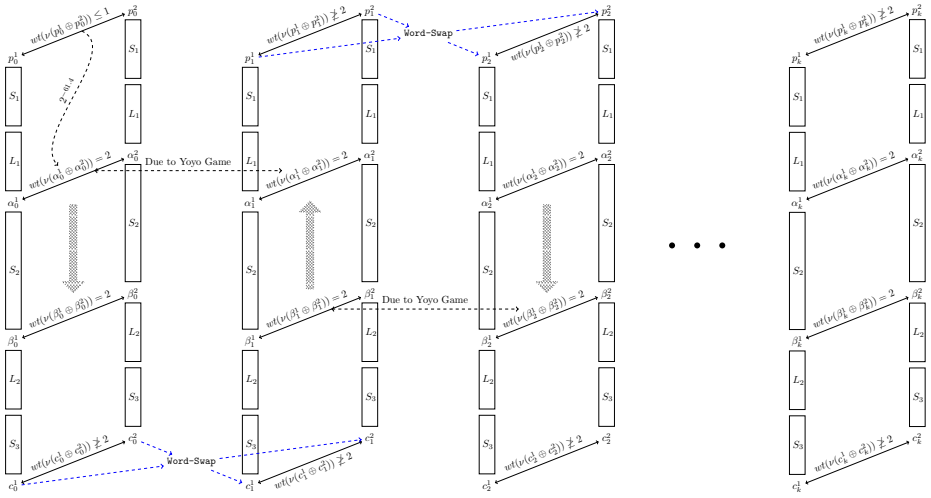

Fig. 14: For ForkAES-X-4-4, at least **one** pair out of $2^{61.4}$ pairs is expected for which $wt(\nu(\alpha_0^1 \oplus \alpha_0^2)) = 2$. By Yoyo Game $wt(\nu(\alpha_1^1 \oplus \alpha_1^2)) = 2$ which implies that $wt(\nu(p_k^1 \oplus p_k^2)) \not\geq 2$ and $wt(\nu(c_k^1 \oplus c_k^2)) \not\geq 2$, for $0 \leq k \leq 2^{61.4}$

This event is impossible for ForkAES-$*$-4-4 if $wt(\nu(L_1 \cdot S_1(p^1) \oplus L_1 \cdot S_1(p^2))) = t$. So, the attack proceeds as taking a random pair of plaintexts at $\widehat{C}_1$ and obtain the corresponding $\widehat{C}_0$ by querying the reconstruction algorithm and check whether $(4-t)$ words are inactive or not. If $(4-t)$ words are inactive, then the pair is dropped as it is a wrong pair; otherwise, words are swapped between them and again the reconstruction algorithm is queried for obtaining $\widehat{C}_1$ and same procedure follows. For each pair, zero difference in $\widehat{C}_1$ and $\widehat{C}_0$ needs to be checked $\frac{1}{2\times\binom{4}{4-t}\times(2^{32})^{(t-4)}}$ times each to match the random case. For a right pair, zero difference in $\widehat{C}_0$ and $\widehat{C}_1$ should never be $(4-t)$. The data complexity of the attack is

$$\frac{1}{\binom{4}{t}\times 2^{32\times -t}} \times \frac{1}{\binom{4}{4-t}\times 2^{32\times(t-4)}}$$

which is minimum at $t = 2$ with value $2^{122.83}$.

Attack Procedure.

1. Take any two texts p_0^1, p_0^2 at $\widehat{C}_1$ s.t. $wt(\nu(p_0^1 \oplus p_0^2)) \leq 1$ and compute their corresponding C_1 offline and query them to obtain the corresponding C_0. Compute the corresponding $\widehat{C}_0$ offline. Let, the new texts are c_0^1, c_0^2.
2. Check whether $wt(\nu(c_0^1 \oplus c_0^2)) \geq 2$. If the weight is greater than equal to 2, then discard the pair and start from step 1. Otherwise, swap words between c_0^1 and c_0^2 to obtain c_1^1, c_1^2. Use these new pairs to query the reconstruction algorithm to obtain p_1^1, p_1^2 and discard the pair if $wt(\nu(p_1^1 \oplus p_1^2)) \geq 2$.
   In general, by applying the yoyo game $p_k^1, p_k^2, c_k^1, c_k^2$ is obtained by using

$p^1_{k-1}, p^2_{k-1}, c^1_{k-1}, c^2_{k-1}$. The initial pair p^1_0, p^2_0 is dropped if $wt(\nu(p^1_k \oplus p^2_k)) \geq 2$ or $wt(\nu(c^1_k \oplus c^2_k)) \geq 2$ for $0 \leq k \leq 2^{61.4}$ and a new pair of texts p^1_0, p^2_0 is chosen and the process is repeated. For ForkAES-$*$-4-4, at least one pair (right pair) is expected s.t., $wt(\nu(p^1_k \oplus p^2_k)) \not\geq 2$ or $wt(\nu(c^1_k \oplus c^2_k)) \not\geq 2$ for $0 \leq k \leq 2^{61.4}$. If such a pair is found, we distinguish ForkAES-$*$-4-4.
In ForkAES-$*$-4-4, at $\widehat{C}_0$ and $\widehat{C}_1$ each word corresponds to each column. So, in this case swapping of words refers to the exchange of columns between two texts.

3. If step 1 is repeated $2^{61.4}$ times and the right pair is not found then it is not ForkAES-$*$-4-4.

Complexity Evaluation. The attack has a data complexity of $2^{122.83}$ reconstruction queries, requires the time for their encryption by the oracle plus $2^{122.83}$ XOR operations and word swapping between texts. The attack requires negligible storage requirements.