Impossible-Differential and Boomerang Cryptanalysis of Round-Reduced Kiasu-BC

Christoph Dobraunig[1]*, Eik List[2]

[1] Graz University of Technology, Austria
christoph.dobraunig@iaik.tugraz.at
[2] Bauhaus-Universität Weimar, Germany
eik.list@uni-weimar.de

Abstract. Kiasu-BC is a tweakable block cipher proposed by Jean et al. at ASIACRYPT 2014 alongside their TWEAKEY framework. The cipher is almost identical to the AES-128 except for the tweak, which renders it an attractive primitive for various modes of operation and applications requiring tweakable block ciphers. Therefore, studying how the additional tweak input affects security compared to that of the AES is highly valuable to gain trust in future instantiations.
This work proposes impossible-differential and boomerang attacks on eight rounds of Kiasu-BC in the single-key model, using the core idea that the tweak input allows to construct local collisions. While our results do not threat the security of the full-round version, they help concretize the security of Kiasu-BC in the single-key model.

Keywords: Symmetric-key cryptography · cryptanalysis · tweakable block cipher

1 Introduction

TWEAKEY. At ASIACRYPT 2014, Jean et al. [18] proposed the TWEAKEY framework together with three software-efficient tweakable block ciphers based on the AES round function Deoxys-BC, Joltik-BC, and Kiasu-BC. Such tweakable block ciphers process, in addition to key and plaintext, an additional public input, called the *tweak*. While the first construction that followed this concept was the AES candidate by Schroeppel and Orman [33], the formal foundations have been laid by Liskov, Rivest, and Wagner [24]. Nowadays, tweakable block ciphers possess various applications in cryptographic schemes, such as compression functions (e. g. [16]), variable-input-length ciphers (e.g. [27]), message-authentication codes (e.g. [29]), or (authenticated) encryption schemes (e.g., [23,31]).

* This work has been supported in part by the Austrian Science Fund (project P26494-N15).

Table 1: Existing attacks on KIASU-BC and comparison with attacks proposed in this work. ACC = chosen plaintexts and adaptive chosen ciphertexts; CP = #chosen plaintexts; (A)CC = #(adaptive) chosen ciphertexts; E = #Encryption equivalents; MA = #Memory accesses.

		Computations				
Rds.	Attack Type	E	MA	Data (CP)	Memory	Ref.
7	Integral	2^{82}		2^{40}	2^{41}	[15]
7	Integral	$2^{48.5}$		$2^{43.6}$	$2^{41.7}$	[15]
8	MitM	2^{116}		2^{116}	2^{86}	[34]
8	Imposs. Diff.	2^{118}	$2^{125.2}$	$2^{117.6}$	$2^{101.6}$	[1]
7	Rectangle	2^{79}	2^{80}	2^{79}	2^{78}	App. A
7	Boomerang	2^{65}	$2^{66.6}$	2^{65} ACC	2^{60}	App. B
8	Imposs. Diff.	$2^{116.1}$	$2^{120.2}$	2^{118}	2^{102}	Sec. 3
8	Boomerang	$2^{103.1}$	2^{103}	2^{103} ACC	2^{60}	Sec. 5
10	Bicliques	$2^{125.34}$		2^{88} CC	2^{62}	Sec. 6

Kiasu-BC*.* While DEOXYS-BC and JOLTIK-BC use a new linear tweak and key schedule, and in the case of JOLTIK-BC a round function different from AES [30] working on 64-bit blocks, the design of KIASU-BC strictly follows AES-128. KIASU-BC uses exactly the key schedule, round function, and number of rounds of the AES-128. The only difference is an additional 64-bit tweak that is XORed to the topmost two rows of the state after every round. So, KIASU-BC is identical to the AES-128 if the tweak is set to all-zeroes. Therefore, KIASU-BC may appear attractive as primitive for instantiating ciphers, AE schemes, or MACs based on tweakable block ciphers, for it can reuse existing components of AES implementations. In addition, all the existing and newly done analysis for AES-128 is directly applicable to KIASU-BC. However, the additional tweak input enhances the freedom in attacks. Thus, a comprehensive cryptanalysis of KIASU-BC is necessary to determine possible negative effects.
The designers' analysis in [17] concentrates on differential and meet-in-the-middle attacks. They stress that the size of the tweak and the position where it is XORed to the state has been the result of a careful security analysis and *"the current choice in* KIASU-BC *assures that no such high probability characteristics exist on more than 6 rounds and no boomerang characteristics on more than 7 rounds"* [19]. Concluding from an automated search, the designers argue that the minimum number of active S-boxes for seven-round KIASU-BC is 22, corresponding to an upper bound of the probability of differential characteristics of $(2^{-6})^{22} = 2^{-132}$. Since the bound is not tight, they conclude in [17, Sec. 4.1] that *"in the framework of related-key related-tweak differential attacks [*KIASU-BC*] has only at most one round security loss compared to* AES*"*.

Regarding Meet-in-the-Middle attacks, the designers [17, Sec. 4.2] conclude that *"the same [MitM] attacks existing for* AES-*128 appl[y] to* Kiasu-BC*"*. Concerning further attacks in the single-key model, [17, Sec. 4.3] states that *"the security level of* Kiasu-BC *against the remaining types of attacks stays the same"*. Recently, integral attacks showed that the latter claim does at least not hold in general [15]. The additional degrees of freedom from the choice of the tweak leads to improved attacks on Kiasu-BC compared to the AES-128.

Contribution. This work complements the analysis by [15] with differential-based attacks on Kiasu-BC on eight rounds of Kiasu-BC. Our attacks share the observation that a chosen non-zero tweak difference allows to cancel a difference in the state at the beginning of some round. We propose impossible-differential, and boomerang analysis of Kiasu-BC with lower time complexities and/or higher number of covered rounds than comparable attacks on the AES-128. Our detailed results are summarized in Table 1 and compared with existing results on Kiasu-BC. We would like to mention that while this work was under review, independent work of Abdelkhalek, Tolba, and Youssef [1] lead to an submission of an impossible-differential attack on Kiasu-BC similar to ours (based on different trails) to a journal and also independently meet-int-the-middle attacks on Kiasu-BC [34] have been published.

Outline. In the following, Section 2 briefly recalls the basics of Kiasu-BC. Section 3 presents our impossible-differential attack. Section 4 continues with a brief introduction of boomerang and rectangle attacks, which is followed by the description of a boomerang attack on eight rounds in Section 5. We give a description of a biclique-based accelearated exhaustive search in Section 6. For the interested reader, we provide three further boomerang attacks in Appendices A and B. Section 7 concludes this work.

2 Brief Overview of Kiasu-BC

Kiasu-BC [18] is a tweakable block cipher that adopts the state size (128 bits), key size (128 bits), round function – consisting of SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddKey (AK) – as well as number of rounds (10), and key schedule from the AES-128. We assume the reader is familiar with the structure of the AES; otherwise, we refer to e. g. [13,30] for details.
Kiasu-BC adds to the AES-128 only an additional public 64-bit tweak $T = (T[0] \parallel T[1] \parallel \ldots \parallel T[7])$. During the en-/decryption, the same tweak is XORed to the topmost two state rows at every occurrence of AddKey in every round, as illustrated in Figure 1. In the following, we denote by

- S^i the state after Round i, for $0 \leq i \leq 10$.
- K^i the round key of Round i, for $0 \leq i \leq 10$.
- $S^i[j]$ the j-th byte of the state S^i, for $0 \leq j \leq 15$, indexed column-wise as usual for the AES, and as illustrated in State S^{i-1} in Figure 1.
- $K^i[j]$ the j-th byte of the round key K^i, for $0 \leq j \leq 15$.

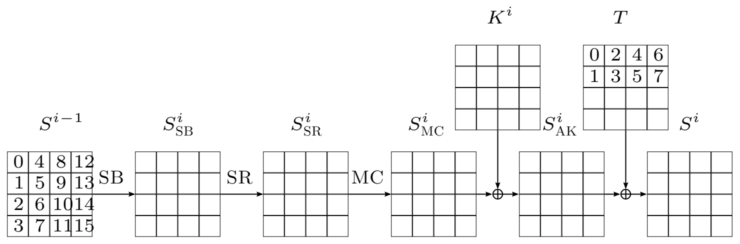


Fig. 1: Round Function of Kiasu-BC.

- $T[j]$ the j-th byte of the tweak T, for $0 \leq j \leq 7$. Note that these bytes are enumerated differently than states and keys, as illustrated in Figure 1.
- S^i_{SB}, S^i_{SR}, S^i_{MC}, and S^i_{AK} the intermediate states in Round i directly after SubBytes, ShiftRows, MixColumns, and AddKey, respectively.

For brevity, we also use the notions $\overline{\mathrm{SB}} = \mathrm{SB}^{-1}$, $\overline{\mathrm{SR}} = \mathrm{SR}^{-1}$, and $\overline{\mathrm{MC}} = \mathrm{MC}^{-1}$. Note that the order of some operations can be swapped without affecting the result to simplify the description of attacks. For instance, the order of the MixColumns and AddKey operations with a round key K^i can be swapped if the key addition with the equivalent round key $\widehat{K}^i = \mathrm{MC}^{-1}(K^i)$ is performed instead. This means, for all $x, K^i \in \{0,1\}128$, it holds that $K^i \oplus \mathrm{MC}(x) = \mathrm{MC}(\widehat{K}^i \oplus x)$. The same argument holds for decryption.

3 Impossible-Differential Attack on 8-Round Kiasu-BC

This section describes an impossible-differential attack on 8-round Kiasu-BC. For this attack, we modify existing differential trails for the AES-128 by introducing differences in the tweak, so that they can be used in attacks. Those introduced differences allow us to extend the key-recovery phase. First, we briefly recall the concept of such attacks and explain how we can reuse existing differential trails for the AES-128. Then, we provide the impossible differential, followed by a detailed description of the attack.

3.1 Impossible-Differential Attacks

In this section, we briefly recall the main concept of impossible-differential attacks. Assume, E is a given cryptographic transform; in the following, we consider E to be a tweakable block cipher or a reduced version thereof: $E : \mathbb{F}_2^k \times \mathbb{F}_2^t \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, for $k, t, n \geq 1$. The adversary partitions E into parts $E = E_3 \circ E_2 \circ E_1$, as illustrated in Figure 2. Then, she searches for a differential $\Delta_X \not\rightarrow \Delta_Y$ which has probability 0 over E_2. The differential trail is extended in forwards and backwards direction over the parts E_1 and E_3, respectively. Next, the adversary chooses pairs of plaintexts with a desired input difference Δ_{in} and requests their corresponding ciphertexts. For each plaintext-ciphertext pair (P^i, C^i) with a desired ciphertext difference Δ_{out}, the adversary partially encrypts it over E_1, and

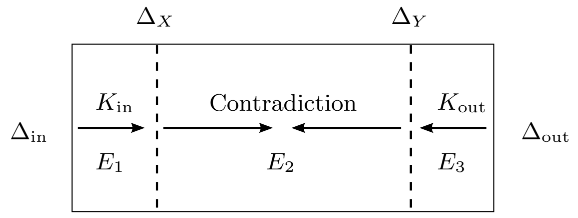


Fig. 2: Schematic illustration of an impossible-differential attack.

partially decrypts it over E_3^{-1} to reconstruct the differences Δ_X^i, Δ_Y^i at the beginning and at the end of the trail. The key material that is required for partial encryption over E_1 and decryption over E_3 is guessed. If the differences Δ_X^i and Δ_Y^i conform to the start and end differences of the impossible differential trail Δ_X and Δ_Y for any of the available pairs, the current key guess must be invalid and can be discarded. Clearly, the involved differences can also be truncated. Moreover, early-abort techniques can speed up the search; e.g., if guessing a few key bits in E_1 allows to conclude that Δ_X cannot be fulfilled for a certain pair, it need not be considered further for the current key guess.

3.2 Impossible Differentials on Kiasu-BC

Influence of the Tweak. The tweak input provides the adversary with additional freedom that can be used for instance to extend the number of covered rounds. In Kiasu-BC, the tweak can be used to cancel a difference in the trail, which allows to pass one round for free. Moreover, since the tweak is not modified by a tweak-schedule over the rounds, the subsequent tweak addition will produce exactly the same difference that occurred before the *free* round.

A Concrete Impossible Differential. There exist various impossible differentials – e.g. [4,25,26] on round-reduced AES – which can serve as base of our analysis of Kiasu-BC. Though, we have to ensure that the influence of the tweak difference preserves the impossibility of the differential in the middle. Figure 3 shows an impossible differential over 3.5 rounds based on the trail used in [4]. In addition, we use a tweak difference ΔT with a single active byte $T[0]$. In forward direction, the single active byte in the state S^i from the beginning of the trail always activates at least three bytes in the first column of S^{i+1} – depending on whether the tweak difference cancels the difference in $S^{i+1}[0]$ or not. In both cases, the three rightmost columns which correspond to $\overrightarrow{S}^{i+2}[4,\ldots,15]$ are always active. In backward direction, the fact that only three bytes are active in S^{i+3} and the diffusion in the third inverse round will ensure a zero difference in $\overleftarrow{S}^{i+2}[1,6,11,12]$, which contradicts with $\overrightarrow{S}^{i+2}[4,\ldots,15]$, independent from whether the tweak difference cancels out the diagonal $\overleftarrow{S}^{i+2}[0,5,10,15]$ or not.

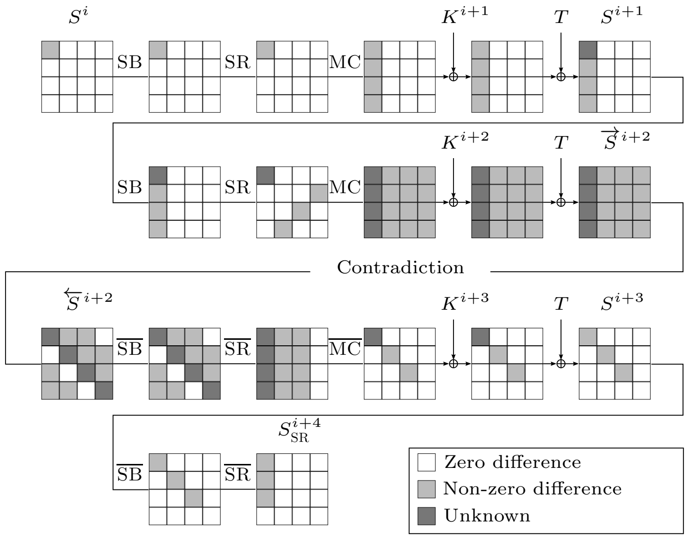


Fig. 3: Impossible differential for 3.5 rounds of Kiasu-BC.

3.3 Attack Procedure

We can extend our impossible differential by two rounds at the beginning and the end each to a key-recovery attack over Rounds 3 through 10 of Kiasu-BC, using the fact that the final round omits MixColumns. Figure 4 shows our differential trail. The following describes the individual steps.

Precomputation. Initially, we precompute a hash table $\mathcal{H}_{\text{precomp}}$ which maps pairs $(S^1_{\text{AK}},\ {S'}^1_{\text{AK}}) \leftarrow (S^1_{\text{MC}}, {S'}^1_{\text{MC}})$. For all possible pairs $S^1_{\text{MC}}[0,1,2,3]$ and ${S'}^1_{\text{MC}}[0,1,2,3]$ which differ only in Byte 0, compute

$$S^1_{\text{AK}}[0,5,10,15] = \text{SB}^{-1}(\text{SR}^{-1}(\text{MC}^{-1}(S^1_{\text{MC}}[0,1,2,3])))$$

and ${S'}^1_{\text{AK}}[0,5,10,15]$ accordingly. Define $\Delta S^1_{\text{AK}}[0,5,10,15] = S^1_{\text{AK}}[0,5,10,15] \oplus {S'}^1_{\text{AK}}[0,5,10,15]$. Compute $\Delta S^1_{\text{MC}}[0] = S^1_{\text{MC}}[0] \oplus {S'}^1_{\text{MC}}[0]$ and store the pairs as tuples $(S^1_{\text{AK}}[0,5,10,15],\ {S'}^1_{\text{AK}}[0,5,10,15])$ in a hash table $\mathcal{H}_{\text{precomp}}$ indexed by $(\Delta S^1_{\text{AK}} \parallel \Delta S^1_{\text{MC}}[0])$. Since there are 2^{24} values for $S^1_{\text{MC}}[1,2,3]$ and $2^8 \cdot (2^8 - 1) \approx 2^{16}$ pairs for $S^1_{\text{MC}}[0]$, there exist about 2^{40} possible pairs. So, $\mathcal{H}_{\text{precomp}}$ has 2^{40} buckets and one element in each on average.

Structures. We will consider sets and structures of plaintexts. A set $\mathcal{S}$ consists of 2^{32} plaintexts P_i which all share equal values in bytes $P_i[1,2,3,4,6,7,8,9,11,12,13,14]$, and are pair-wise distinct in the bytes $P_i[0,5,10,15]$. Assigned to each set is a concrete tweak T. A structure $\mathcal{L}$ consists of 2^8 sets, where each set in $\mathcal{L}$ differs only in the tweak byte $T[0]$. We can build pairs of plaintext-tweak inputs (P_i, T_i) and (P_j, T_j) only inside the same structure. Though, since we want that

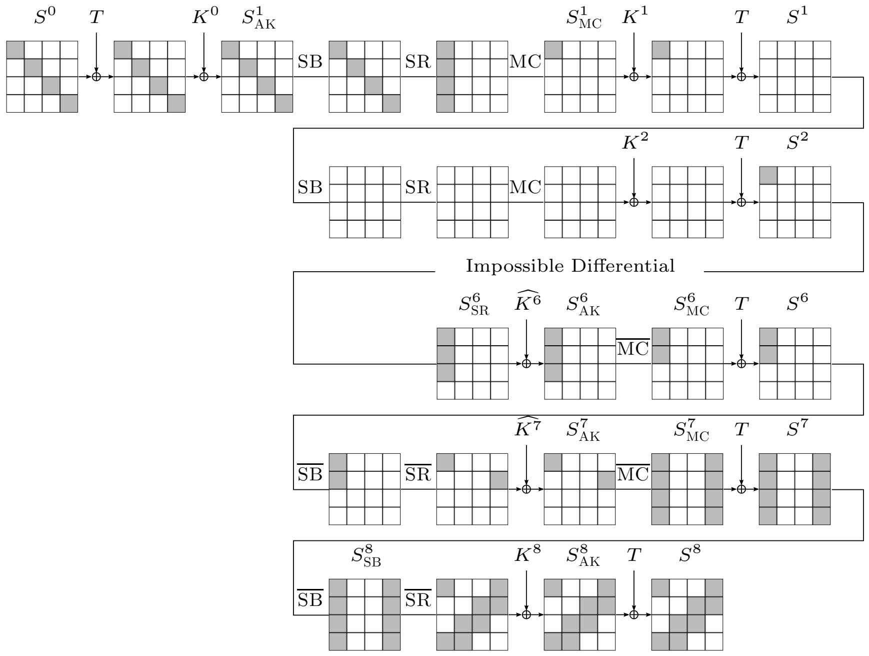


Fig. 4: 8-round impossible-differential attack trail.

pairs differ in their tweaks, we have to build pairs across different sets in a structure. Moreover, their bytes 0, 5, 10, and 15 after the initial tweak addition must differ, i.e. $(P_i[0] \oplus T_i[0]) \neq (P_j[0] \oplus T_j[0])$. Given two distinct sets $\mathcal{S}$ and $\mathcal{S}'$, we obtain $2^{32} \cdot (2^8 - 1)^4 \approx 2^{63.98} \approx 2^{64}$ pairs. Since there are 2^8 sets in a structure, we can build in total $\binom{2^8}{2} \cdot 2^{63.98} \approx 2^{78.97} \approx 2^{79}$ pairs per structure.

Step 1) Choose 2^n structures, which yields about 2^{n+79} possible pairs. For each structure, do the following steps:

1. Ask for the corresponding ciphertexts $C_i \leftarrow E_K^{T_i}(P_i)$ of the structure.
2. Invert the final tweak XOR and insert all states S^8_{AK} into a hash table $\mathcal{L}$, indexed by bytes $S^8_{\text{AK}}[1, 2, 4, 5, 8, 11, 14, 15]$.
3. For each bucket in the hash table that contains more than one entry, consider every combination of pairs therein. We can expect $2^{n+79} \cdot 2^{-8 \cdot 8} = 2^{n+15}$ such pairs that are equal in bytes $S^8_{\text{AK}}[1, 2, 4, 5, 8, 11, 14, 15]$.

Step 2) The straight-forward approach would be to guess 32 bits of $K^8[3, 6, 9, 12]$ and partially decrypt these bytes for the remaining pairs to obtain $S^7_{\text{MC}}[12, 13, 14, 15]$. Though, we can use an improvement by Lu et al. [25] to speed up the search. The improvement is based on the following observation: given a random

pair of differences $\Delta X, \Delta Y \in \mathbb{F}_2^8$ over the AES S-box, there is on average one pair of $X, X' \in \mathbb{F}_2^8$ with $X \oplus X' = \Delta X$ such that $S(X) \oplus S(X') = \Delta Y$.[3]
For any pair $(S_{\text{AK}}^8[3,6,9,12],\ {S'}_{\text{AK}}^8[3,6,9,12])$, their difference $\Delta S_{\text{SB}}^8[12,13,14,15]$ is known. Hence, the knowledge of $\Delta S_{\text{AK}}^7[12,13,14,15]$ can be used to derive the values of $S_{\text{SB}}^8[12,13,14,15]$ and $S_{\text{SB}}'^8[12,13,14,15]$ and thus to derive $K^8[3,6,9,12]$. There exist only $2^8 - 1$ possible values of $\Delta S_{\text{AK}}^7[12,13,14,15]$ with exactly one active Byte 13. So, one can perform this step as follows:

1. Initialize 2^{32} empty lists, one for each guess of $K^8[3,6,9,12]$.
2. For each pair $(S_{\text{AK}}^8[3,6,9,12],\ {S'}_{\text{AK}}^8[3,6,9,12])$, and for each of the 255 possible differences $\Delta S_{\text{AK}}^7[12,13,14,15] = (0,*,0,0)$, derive $K^8[3,6,9,12]$ that leads this pair to $\Delta S_{\text{AK}}^7[12,13,14,15]$ and add this pair to the list corresponding to that key guess.

For each pair and difference guess, we expect one key suggested on average due to the S-box property observed above. These $2^{n+15} \cdot 255 \approx 2^{n+23}$ suggestions are distributed over the 2^{32} possible keys. So, we expect about 2^{n-9} pairs for each guess of $K^8[3,6,9,12]$.

Step 3) In this step, one could guess 32 bits of $K^8[0,7,10,13]$ and partially decrypt these bytes for the remaining pairs to obtain $S_{\text{AK}}^7[0,1,2,3]$. Though, this step can be improved in a similar fashion as Step 2):

1. Initialize 2^{32} empty lists, one for each guess of $K^8[0,7,10,13]$.
2. For each pair $(S_{\text{AK}}^8[0,7,10,13],\ {S'}_{\text{AK}}^8[0,7,10,13])$ and for each of the 255 possible differences $\Delta S_{\text{AK}}^7[0,1,2,3] = (*,0,0,0)$, derive the key $K^8[0,7,10,13]$ that leads this pair to $\Delta S_{\text{AK}}^7[0,1,2,3]$ and add this pair to the list corresponding to that key guess.

Again, we expect one key suggested on average for each pair and each difference guess. These $2^{n-9} \cdot 255 \approx 2^{n-1}$ suggestions are distributed over the 2^{32} possible keys. So, we expect about 2^{n-33} pairs for each guess of $K^8[0,7,10,13]$.

Step 4) The goal of the adversary in this step is to check for all remaining pairs and for the current guess of $K^8[0,3,6,7,9,10,12,13]$ if the difference $\Delta S_{\text{AK}}^6[0,1,2,3]$ is zero in exactly one byte, and if the zero byte is Byte 1, 2, or 3. Note that we do not want a zero difference in $S_{\text{AK}}^6[0]$ since it could render the impossible differential possible. The straight-forward approach would be to guess the bytes $\widehat{K}^7[0,13]$ and decrypt the states $S^7[0,13]$ of all remaining pairs to obtain $S^6[0,1]$. Again, we use the improvement by Lu et al. [25] instead.
There are $3 \cdot 255^3$ possible differences $\Delta S_{\text{AK}}^6[0,1,2,3]$ with exactly three active bytes such that the zero-difference byte is not Byte 0. Among those, $3 \cdot 255$ differences map to a difference $\Delta S_{\text{MC}}^6[0,1,2,3]$ where only Bytes 0 and 1 are active. So, the adversary has to check for each pair and each guess of $\widehat{K}^7[0,13]$ whether

[3] More precisely, 129 out of 256 trails $\Delta X \rightarrow \Delta Y$ are impossible, about half (126) propose two solutions, and 1 trail proposes four solutions.

$\Delta S^6_{\mathrm{MC}}[0,1,2,3]$ belongs to these $3 \cdot 255$ differences. Again, given the input/output differences of the SubBytes operation, i.e., $\Delta S^6_{\mathrm{MC}}[0,1]$ and $\Delta S^7_{\mathrm{AK}}[0,13]$, one can efficiently determine the values $S^6_{\mathrm{MC}}[0,1]$ and $S'^6_{\mathrm{MC}}[0,1]$ and therefore determine the value of $\widehat{K}^7[0,13]$.
The 2^{n-33} pairs and the $3 \cdot 255$ differences yield $3 \cdot 2^{n-25}$ candidates $\widehat{K}^7[0,13]$ distributed over the 2^{16} possible values. Thus, we expect for a given guess of $\widehat{K}^7[0,13]$ about $3 \cdot 2^{n-25}/2^{16} = 3 \cdot 2^{n-41}$ pairs which yield the input difference to the impossible differential for each guess of the considered bytes in K^8 and $\widehat{K}^7$.

Step 5) This step eliminates wrong values of $K^0[0,5,10,15]$ using the precomputed hash table $\mathcal{H}_{\mathrm{precomp}}$. For this purpose, initialize a list $\mathcal{K}$ for the 2^{32} values of $K^0[0,5,10,15]$. For each of the remaining $3 \cdot 2^{n-41}$ pairs (P_i, P_j):

1. Compute $\Delta_{i,j}[0,5,10,15] = (P_i[0,5,10,15] \oplus T_i) \oplus (P_j[0,5,10,15] \oplus T_j)$ and $\Delta T_{i,j}[0] = T_i[0] \oplus T_j[0]$.
2. Access the bucket indexed by $\Delta_{i,j}[0,5,10,15] \,\|\, \Delta T_{i,j}[0]$ in $\mathcal{H}_{\mathrm{precomp}}$. For each tuple $(S^1_{\mathrm{AK}}[0,5,10,15],\ S'^1_{\mathrm{AK}}[0,5,10,15],\ \Delta S^1_{\mathrm{MC}}[0])$ in that bucket, remove from $\mathcal{K}$ the key entry $K^0[0,5,10,15] = P_i[0,5,10,15] \oplus (T_i[0,1],0,0) \oplus S^1_{\mathrm{AK}}[0,5,10,15]$.

Finally, if $\mathcal{K}$ is not empty, output the remaining value(s) in $\mathcal{K}$ along with the current key guess of $\widehat{K}^7[0,13]$ and $K^8[0,3,6,7,9,10,12,13]$.

Wrong-Key Elimination. We can determine the data complexity D such that the following inequality is fulfilled:

$$\left(1 - 2^{-(c_{\mathrm{in}}+c_{\mathrm{out}})}\right)^D < \frac{1}{2^{|k_{\mathrm{in}} \cup k_{\mathrm{out}}|}},$$

where c_{in} and c_{out} denote the number of bit conditions to be fulfilled at the top (in) and bottom (out) parts of the cipher that wrap the impossible differential. $k_{\mathrm{in}} \cup k_{\mathrm{out}}$ denote the number of combined top and bottom key bits that are guessed. Consider that the probability to filter wrong key is 2^{-32} for $K^0[0,5,10,15]$, 2^{-48} for $K^8[0,3,6,7,9,10,12,13]$, and $3 \cdot 2^{-8}$ for $K^7[0,13]$. Hence, we have $c_{\mathrm{in}} + c_{\mathrm{out}} = \log_2(2^{-32-48} \cdot 3 \cdot 2^{-8}) \approx 86$ bit conditions. So, the probability that a wrong key passes is about $(1 - 2^{-86})$ per tested pair. The guessed key material sums up to $k_{\mathrm{in}} \cup k_{\mathrm{out}} = 32 + 64 + 16 = 112$ bits. So, we need

$$\left(1 - 2^{-86}\right)^D \leq 2^{-112}$$

which is true for $D \geq 2^{93}$ pairs. Since we can expect about 2^{n+15} pairs from 2^n structures, this method yields also that 2^{78} structures, i.e., 2^{118} chosen plaintexts, are required for the attack.

Complexity Analysis. The time complexity is composed of the following steps:

0. The **precomputation** requires $\approx 2 \cdot 2^{40} \cdot 4/16 = 2^{36}$ single-round decryptions, which is equivalent to $2^{36}/8 = 2^{33}$ eight-round decryptions.

1. **Step 1** requires 2^{n+40} encryptions.
2. **Step 2** can be implemented by a look-up table, as suggested by Lu et al. [25]. By storing the results efficiently, one can fetch a key in one access even if several keys are suggested. Lu et al. state that most queries fail, whereas about 1/16 (on average) of the queries return 16 options of 32-bit keys each, and a smaller fraction can return more options. In total, this step requires $255 \cdot 2^{n+15} \approx 2^{n+23}$ memory accesses (MA).
3. **Step 3** requires $255 \cdot 255 \cdot 2^{n+15} \approx 2^{n+31}$ memory accesses, since for all 255 differences for the first 32-bit guesses of $K^8[3, 6, 9, 12]$, we consider another 255 differences for the second 32 guessed bits $K^8[0, 7, 10, 13]$.
4. **Step 4** requires $2^{n-33} \cdot 3 \cdot 255 \approx 3 \cdot 2^{n-25}$ memory accesses in a lookup table to determine from the differences $\Delta S^6_{\text{MC}}[0, 1, 2, 3]$ the guess for $\widehat{K}^7[0, 13]$. Since we have to perform this step for each of the 2^{64} guesses of $K^8[0, 3, 6, 7, 9, 10, 12, 13]$, this step requires in total $2^{64} \cdot 2^{n-33} \cdot 3 \cdot 255 \approx 3 \cdot 2^{n+39}$ MA.
5. **Step 5** analyzes $3 \cdot 2^{n-41}$ remaining pairs, leading in average to one memory access to $\mathcal{H}_{\text{precomp}}$ plus one memory access to $\mathcal{K}$. This step is repeated 2^{80} times (for each guess of $K^8[0, 3, 6, 7, 9, 10, 12, 13]$ and $\widehat{K}^7[0, 13]$). So, the time complexity of this step is $3 \cdot 2^{n-41} \cdot 2 \cdot 2^{80} = 3 \cdot 2^{n+40}$ memory accesses.
6. In an **exhaustive step**, we can identify the remaining key bytes. This step requires negligible time regarding the total computational complexity.

So, for $n = 78$, the overall time complexity of the attack results from

$$\begin{aligned} T &\approx (2^{n+40} + 2^{33}) \text{ Enc} + (2^{n+23} + 2^{n+31} + 3 \cdot 2^{n+39} + 3 \cdot 2^{n+40}) \text{ MA} \\ &\approx 2^{118} \text{ Enc} + 2^{120.2} \text{ MA}. \end{aligned}$$

The precomputation table requires $2 \cdot 2^{40} \cdot (4 + 4 + 1) < 2^{45}$ bytes to store the values $S^1_{\text{AK}}[0, 5, 10, 15]$, $S'^1_{\text{AK}}[0, 5, 10, 15]$, and the difference $\Delta S^1_{\text{MC}}[0]$ for each entry. The simple approach would further use $2^{8 \cdot (4+2+8)} = 2^{112}$ bytes of memory for storing the deleted values of the four bytes of K^0, the two bytes of $\widehat{K}^7$, and the eight bytes of K^8. Instead, Lu et al. [25] proposed to perform the attack separately for each key guess, and to immediately append an exhaustive search for the remaining bytes of each guess that is not discarded. So, we have to store instead the about $2^{n+23} = 2^{101}$ suggestions which remain after Step 2, which consist of two plaintexts and two ciphertexts of 16 bytes each. So, the memory complexity of the attack requires $2^{45} + 2^{106} \approx 2^{106}$ bytes of memory, which is equivalent to 2^{102} states.
Several optimizations seem possible to further reduce the attack complexities. For instance, Boura et al. [11] propose to use multiple impossible differentials in order to reduce the data complexity. There are $\binom{4}{2} = 6$ options which two bytes can be chosen to be active in the difference $\Delta S^6[0, 1, 2, 3]$. Each option requires to consider a different set of guessed bytes in $\widehat{K}^7$ and K^8, and a different set of output differences. Moreover, the attack could be executed also with shifted versions of the state differential and tweak difference, or several times in parallel. We omit their description for simplicity.

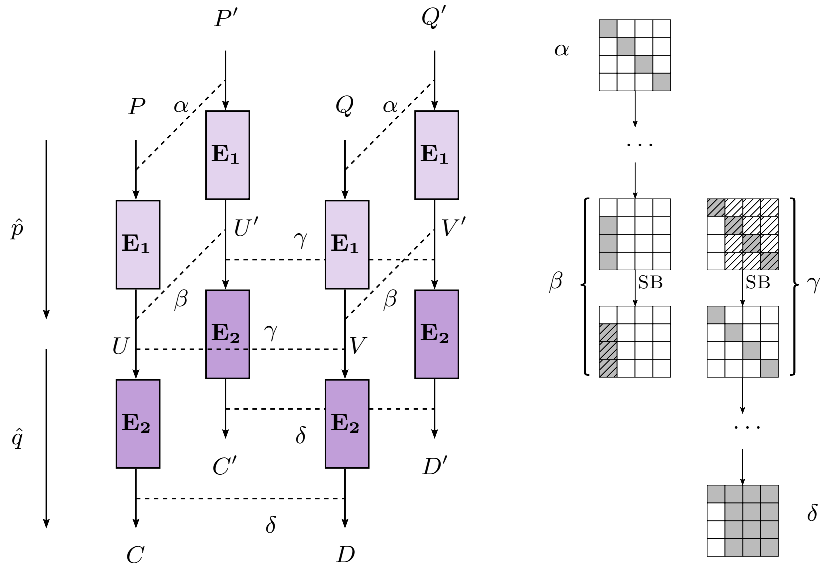


Fig. 5: Left: Schematic illustration of a rectangle quartet. **Right:** Example trails in a ladder switch. Highlighted cells represent active bytes. Hatched bytes are *not* traced through the SubBytes operation at the transition between upper and lower trail.

4 Boomerang and Rectangle Attacks

This section briefly recalls the concepts of boomerang and rectangle attacks.

The Boomerang Attack. Boomerang attacks [35] allow to analyze a given cryptographic transform that lacks long differentials with sufficient probability, but for which short differentials with high probabilities exist. Say, $E : \{0,1\}k \times \{0,1\}n \to \{0,1\}n$ is a cipher that can be 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. These differentials are usually called *upper* and *lower trail*, respectively. A boomerang distinguisher follows the procedure:

1. Choose a plaintext pair (P, P'), with $P' = P \oplus \alpha$, and ask for its corresponding ciphertext (C, C') through E.
2. Compute $D = C \oplus \delta$ and $D' = C' \oplus \delta$ to obtain the ciphertext pair (D, D'), and ask for its corresponding plaintext (Q, Q'). This is also called a δ-*shift.*
3. If it holds that $Q \oplus Q' = \alpha$, then (P, P', Q, Q') forms a *correct quartet.*

Proposition 1. *For a correct quartet (P, P', Q, Q'), with $P' = P \oplus \alpha$, $Q' = Q \oplus \alpha$, $U' = U \oplus \beta$, $V' = V \oplus \beta$, and $V = U \oplus \gamma$, it follows that $V' = V \oplus \beta = (U \oplus \gamma) \oplus \beta = (U \oplus \beta) \oplus \gamma = U' \oplus \gamma$.*

Correct quartets are found with probability $(pq)^2$ since the trails must hold for both pairs. The 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 it holds that $\beta \neq \gamma$. So, the probability of a correct quartet increases to $(\hat{p}\hat{q})^2$ with

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

It must hold that $\hat{p}\hat{q} \gg 2^{-n/2}$ for the attack to work. So, for N plaintext pairs, one expects about $N \cdot (\hat{p}\hat{q})^2$ correct quartets in an attack, but only $N \cdot 2^{-n}$ correct quartets for an ideal primitive.

The Rectangle Attack. In boomerang attacks, the adversary needs to query its oracles with chosen plaintexts and adaptively with chosen ciphertexts. The *rectangle* attack [5] developed from the *amplified boomerang* [20], both of which transform the boomerang into a pure chosen-plaintext attack. The main idea is to encrypt many pairs (P, P') with difference $P' \oplus P = \alpha$ in the 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 part of Figure 5 illustrates a quartet schematically. Biham et al. presented further improvements to the technique in [6]. The disadvantages of rectangle compared to boomerang attacks are the large data complexity and the large number of potential quartets.

Ladder Switch. The ladder switch [7] exploits that the transition between upper and lower trail can be positioned not only between arbitrary operations inside rounds, but also at different steps for different parts of the state, when the parts are processed independently. The right part of Figure 5 illustrates an example for the SubBytes operation of the AES/Kiasu-BC. Since SubBytes processes each byte independently of the others, we can choose to trace only bytes which are inactive in SubBytes through the upper trail with probability one. The other bytes, which are inactive in the lower trail in our example, are processed through SubBytes in the lower trail, again with probability one. So, the probability of the trail transitions does not decrease by the active S-boxes.

5 Boomerang Attack on 8-Round Kiasu-BC

In the following, we describe a boomerang attack on the final eight rounds of Kiasu-BC, which is an extension of a seven-round rectangle attack and a seven-round boomerang attack. For interested readers, we provide those in the Appendices A and B. The upper and lower trails are depicted in Figure 6. We append a key- guessing phase over the final round. Again, we use the fact that the final round omits the MixColumns operation so that only four key bytes have to be considered. Figure 7 shows the steps that wrap the upper and lower trails.

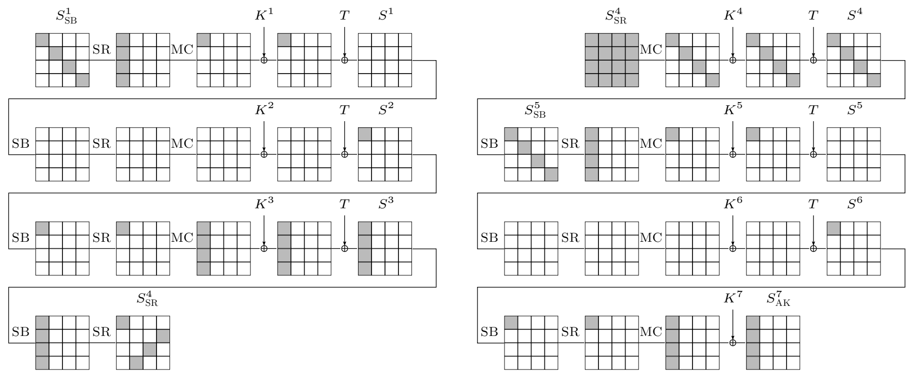


Fig. 6: Upper (**left**) and lower (**right**) trails of our boomerang attack on Kiasu-BC.

5.1 Attack Procedure

Differentials. Both trails share the same idea: start from a state directly after the SubBytes operation with a difference ΔS^i_{SB} wherein only the bytes on the main diagonal are active, and choose it in a way such that the differences in the state ΔS^i_{MC} and in the tweak ΔT will cancel each other after the first round, i.e., $\Delta S^i = 0$. Then, it follows that $\Delta S^{i+1} = \Delta T$ contains only one and ΔS^{i+2} only four active bytes. In the upper trail, only Bytes $S^{i+2}[0, 1, 2, 3]$ are traced through the final SubBytes operation after Round 3. The lower trail then adds the rest of Round 4, i.e. it starts from a fully active state difference ΔS^4_{SR} such that it yields a difference only in the main diagonal after Round 4 with probability one. The remaining Rounds 5-7 follow the same trail as the first three rounds.
Assuming a correct quartet, both its pairs pass the S-box in Round 7 with probability $(2^{-6})^2$. The pairs need not have a specific difference after the final S-box, but only the same difference γ' in the middle. So, the four S-boxes in Round 5 are passed with probability about $(2^{-3.5 \cdot 4})^2 = 2^{-28}$. Concerning the four S-boxes at the bottom of the upper trail, the second pair has a probability of about $(2^{-7})^4 = 2^{-28}$ to pass them with the same trail as the first pair. The final S-box at the beginning of Round 3 is then passed by the second pair with probability 2^{-7}. So, for the correct guess of $K^8[0, 7, 10, 13]$, the probability of a correct pair to follow our trails is about $2^{-12} \cdot 2^{-28} \cdot 2^{-28} \cdot 2^{-7} = 2^{-75}$. Each structure yields $\binom{2^{40}}{2} \cdot 2^{-32} \approx 2^{47}$ pairs which collide after the first round. Hence, for 2^{31} structures, we can expect $2^{31} \cdot 2^{47} \cdot 2^{-75} = 2^3$ correct quartets, and recover 32 bits of K^8, $K^8[0, 7, 10, 13]$ and/or 32 bits of K^0, $K^0[0, 5, 10, 15]$. The remaining 96 bits of either round key can be found e.g. by exhaustive search.

Structures and Sets. We build in total 2^n structures consisting of 2^{40} plaintexts each. For every structure, we choose an arbitrary base plaintext $\widehat{P}$ and derive 2^{32} plaintexts from it by iterating over all values of Bytes 0, 5, 10, and 15, and leaving all other bytes constant. We use the same 2^{32} plaintexts in each of 2^8 sets $\mathcal{T}^i$ in the structure, for $0 \leq i < 2^8$, where the sets differ only in the

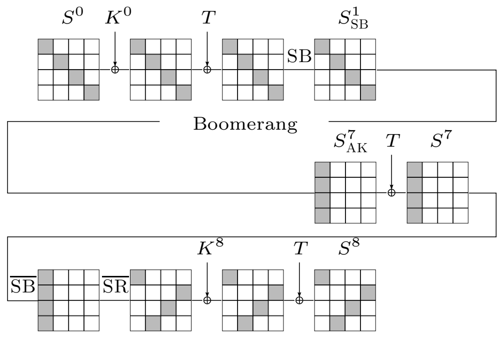


Fig. 7: 8-round boomerang attack trail.

first tweak byte. This means, we derive 2^8 tweaks $T^i = (\langle i \rangle, \widehat{T}[1], \ldots, \widehat{T}[7])$, for $0 \leq i < 2^8$, from $\widehat{T}$ and assign T^i to all texts in $\mathcal{T}^i$.

Steps. Choose $\delta' \in \{0,1\}8$ so that there exists a differential `0x01` $\to \delta'$ with probability 2^{-6} through the S-box. Derive $\delta = \text{MC}((\delta',\ 0,\ 0,\ 0))$. Then:

1. Choose 2^n structures of 2^{40} plaintext-tweak tuples (P, T) each. Ask for their ciphertexts C.
2. Initialize a list for all possible values of $K^0[0,5,10,15]$ and $K^8[0,7,10,13]$.
3. For each of 2^n structures and for each key guess $K^8[0,7,10,13]$:
   a) For each ciphertext and corresponding tweak (C, T_C), derive a tweak T_D with $T_D[1,\ldots,7] = T_C[1,\ldots,7]$ and $T_D[0] = T_C[0] \oplus$ `0x1`. Then, partially decrypt C under T_D through the inverse final round to obtain $S^7_{\text{AK}}[0,1,2,3]$. Compute ${S'}^7_{\text{AK}}[0,1,2,3] = S^7_{\text{AK}}[0,1,2,3] \oplus \delta[0,1,2,3]$ and determine $D[0,7,10,13]$ by reencrypting ${S'}^7_{\text{AK}}[0,1,2,3]$ over the final round, again under T_D. Copy the 12 bytes $D[1,2,3,4,5,6,8,9,11,12,14,15]$ from the corresponding bytes of C.
   b) Ask for the plaintexts Q of all 2^{n+40} shifted ciphertexts (D, T_D).
   c) Sort the 2^{40} plaintexts Q together with their tweaks T_Q (we define $T_Q = T_D$), corresponding ciphertext D, and the original plaintext P from which D was derived as tuples $(Q,\ T_Q,\ D,\ P)$ into 2^{96} buckets indexed by $Q[1,2,3,4,6,7,8,9,11,12,13,14]$. Since we search for pairs (Q, Q') with $Q \oplus Q' = \alpha$ and $T_Q \neq T_{Q'}$ from the same structure, we expect in average $2^n \cdot \binom{2^{40}}{2} \cdot 2^{-96} \approx 2^{n-17}$ false-positive colliding pairs (Q, Q') for each candidate K^8. Since $\Pr[T_Q \neq T_{Q'}] = 255/256$, we still have about $2^{n-17} \cdot 255/256 \approx 2^{n-17}$ colliding pairs per key.
   d) For each potential quartet (P, P', Q, Q'), we want to identify the key values $K^0[0,5,10,15]$ which could create the state collisions after the first round. Let S^1_P, $S^1_{P'}$, S^1_Q, $S^1_{Q'}$ denote their corresponding states after the first round. For a correct quartet, it holds that $S^1_P = S^1_{P'}$ and analogously, $S^1_Q = S^1_{Q'}$. There is a unique mapping from the tweak difference $S^1_{\text{SR},P} \oplus S^1_{\text{SR},P'}$ backwards to $S^1_{\text{SB},P} \oplus S^1_{\text{SB},P'}$. Since the S-box has two solutions

on average, there are on average two key values per byte $K^0[0,5,10,15]$ such that P and P' are mapped to the correct difference ΔS^1_{SB} which yields $S^1_P \oplus S^1_{P'} = 0$. So, we derive those values for $K^0[0,5,10,15]$ for P and P'. We derive those key values analogously for Q and Q'. For each key byte, we have $2 \cdot 2$ combinations on average, i.e., four pairs among which any pair matches with probability about $4 \cdot 2^{-8} = 2^{-6}$. So, an invalid quartet survives this four-byte filter with probability $(2^{-6})^4 = 2^{-24}$. For each surviving pair, we increment the counter for the values of the current $K^0[0,5,10,15]$ and $K^7[0,7,10,13]$. Over all 2^{32} keys K^8, we expect $2^{32} \cdot 2^{n-17} \cdot 2^{-24} = 2^{n-9}$ (false-positive) quartets on average.

e) For $n = 31$, we expect 2^3 correct quartets, and 2^{22} false positives. While 2^3 can be expected on average, at least three correct quartets occur with significant probability. Correct quartets suggest the same 64 bits $K^8[0,7,10,13]$ and $K^0[0,5,10,15]$. Assuming that the key suggestions from the 2^{22} false-positive quartets are uniformly distributed, we can expect only the correct 64 bits of $K^0[0,5,10,15]$ be suggested at least four times. So, we output the candidate(s) with the highest counters.

5.2 Complexity.

The time complexity of the attack consists of the following:

– **Step 1** requires 2^{n+40} full encryptions.
– **Step 3a** requires $2^{32} \cdot 2^{n+40} \cdot 4/16 = 2^{n+70}$ single-round decryptions of four bytes for each ciphertext and key candidate to derive S^7_{AK} and the same amount of single-round encryptions to derive the shifted ciphertexts D, which is equivalent to $2 \cdot 2^{n+70} \cdot 1/8 = 2^{n+68}$ eight-round encryptions.
– **Step 3b** requires $2^{32} \cdot 2^{n+40} = 2^{n+72}$ full decryptions.
– **Step 3c** requires $2^{32} \cdot 2^{n+40} = 2^{n+72}$ MAs with an efficient data structure.
– **Step 3d** requires $2^{32} \cdot 4 \cdot 2^{n-17} \cdot 4/16 = 2^{n+15}$ single-round decryptions (equivalent to 2^{n+12} eight-round encryptions) of one column backwards through the first round for each text in each potential quartet (P, P', Q, Q') and each key. It requires $2^{n-9} + 2^{n+3}$ memory accesses for the false-positive and the correct quartets, which are negligible in the total computational complexity.
– An **exhaustive** search step requires 2^{96} full encryptions.

The time complexity is given by approximately

$$\begin{aligned}&(2^{n+40} + 2^{n+68} + 2^{n+72} + 2^{n+12} + 2^{96}) \text{ Enc} + 2^{n+72} \text{ MA}\\ \approx\ & 2^{103.1} \text{ Enc} + 2^{103} \text{ MA}.\end{aligned}$$

The data complexity consists of $2^{n+40} = 2^{71}$ chosen plaintexts and $2^{32} \cdot 2^{71} = 2^{103}$ adaptively chosen ciphertexts. The attack can be run with memory for 2^{40} states plus 2^{64} single-byte key counters, which is equivalent to about 2^{60} states.

Table 2: Probabilities of correct quartets from experiments with our 6- and 7-round boomerang distinguishers on Kiasu-BC and Mini-Kiasu-BC with 100 random keys per experiment. Probability deviations after subtracting/adding the standard deviation to the average #correct quartets are given in square brackets. Rds. = #rounds; str. = #structure(s).

	6-round distinguisher		7-round distinguisher
	Kiasu-BC	Mini-Kiasu-BC	Mini-Kiasu-BC
Rds.	2^{16} texts, 2^{8} sets, 1 str.	2^{16} texts, 2^{4} sets, 1 str.	2^{16} texts, 2^{4} sets, 2^{10} str.
3	$2^{-12.00}$ $[2^{\pm 0.01}]$	$2^{-4.00}$ $[2^{\pm 0.01}]$	–
4	$2^{-19.00}$ $[2^{\pm 0.03}]$	$2^{-5.88}$ $[2^{\pm 0.01}]$	$2^{-14.61}$ $[2^{\pm 0.01}]$
5	$2^{-25.42}$ $[2^{-0.22}, 2^{0.19}]$	$2^{-7.14}$ $[2^{\pm 0.01}]$	$2^{-26.94}$ $[2^{-0.17}, 2^{0.16}]$
6	–	$2^{-7.14}$ $[2^{\pm 0.01}]$	$2^{-30.65}$ $[2^{-0.83}, 2^{0.53}]$
7	–	–	$2^{-30.12}$ $[2^{-0.71}, 2^{0.47}]$

5.3 Experimental Verification

Murphy [28] showed that boomerangs and rectangles base on oversimplified conditional systems. He stressed that the techniques need a revised foundation, which is clearly beyond the scope of this work. While the complexity of our attacks prohibits to implement full versions of them at the moment, we implemented variants of our boomerang distinguishers to support our analysis:

1. the 6-rd. distinguisher under our 7-rd. boomerang attack with Kiasu-BC,
2. the 6-rd. boomerang distinguisher with a downscaled version of Kiasu-BC, called Mini-Kiasu-BC hereafter,
3. the 7-rd. distinguisher under our 8-rd. attack, again with Mini-Kiasu-BC.
4. the 7-rd. boomerang key-recovery attack with Mini-Kiasu-BC.

We defined Mini-Kiasu-BC as a nibblewise-operating variant of Kiasu-BC that employs the same high-level structure as Kiasu-BC in downscaled manner, i.e., the same number and order of operations, equal number of rounds and key schedule, the same ShiftRows, AddKey, and AddTweak operations, the same MDS matrix, though, with multiplications in $\mathbb{GF}(2^4)$ under the irreducible polynomial $\mathtt{x}^4 + \mathtt{x} + \mathtt{1}$, operating on nibbles instead of bytes, and with the Piccolo S-box instead of that of the AES. Note that the Piccolo S-box has a maximal probability of 2^{-2} for differential trails. Moreover, for differences $\delta, \gamma' \in \{0,1\}4$, it holds for differential trails $\delta \to \gamma'$ through the inverse Piccolo S-box that $\sqrt{\sum_{\gamma' \in \{0,1\}^4} \Pr^2[\delta \xrightarrow{S^{-1}} \gamma']}$ is 2^{-1} for two values of δ, $2^{-1.5}$ for five, and about $2^{-1.21}$ for the remaining eight non-zero values of δ.
To verify the intermediate probabilities of our trails, we studied several round-reduced versions of the distinguishers.[4] Since building structures was unnecessary

[4] The source code is freely available at `https://github.com/medsec/kiasubc`.

for this purpose, we omitted the first round for those, and started directly from the state S^1 after the first round. Thereupon, we (1) chose a random base plaintext, (2) created plaintexts by iterating over the values of the first column and all possible values of the first tweak byte, (3) encrypted the resulting plaintext-tweak pairs over the remaining rounds, (4) applied the δ-shift, and (5) decrypted 3, 4, 5, or 6 rounds. For each of our experiments, we chose the base plaintexts of our structures and 100 keys randomly from `/dev/urandom`. The results of our experiments are summarized in Table 2.

Distinguishers. Concerning the 6-round distinguisher for Mini-Kiasu-BC, the active S-box in Round 6 of the lower trail is passed with probability $\approx (2^{-2})^2$ for both pairs. The S-box in Round 4 is passed with probability about 2^{-2}, and that at the beginning of Round 3 is passed with probability about $2^{-1.21}$, both for the second pair only. Concerning the 6-round trail for Kiasu-BC, the active S-box in Round 6 of the lower trail is passed with probability $\approx (2^{-6})^2$ for both pairs; the active S-box at the end of the upper trail with probability about 2^{-7}; the S-box at the beginning of Round 3 multiplies a factor of $2^{-6.5}$. From our intuition, this factor results from the matter that some quartets pass the S-box with probability 2^{-6} and some with 2^{-7}. So, the obtained differential probabilities are slightly higher than expected.
Concerning the 7-round trail for Mini-Kiasu-BC, the five active S-boxes in the lower trail are passed with a probability about $(2^{-2})^2 \cdot (2^{-1.21 \cdot 4})^2$. The four active S-boxes at the bottom of the upper trail in Round 4 are passed with slightly lower probability than expected, i.e., $2^{-12.3} \approx (2^{-3})^4$. Again, we anticipate this to result from quartets that pass the lower trail with lower probability. The final S-box in Round 3 is passed with probability between 2^{-2} and 2^{-3} for the second pair. So, while we do not have figures for Kiasu-BC over the 7-round distinguisher yet, the analysis with Mini-Kiasu-BC provides us with at least a good indication that we can expect for Kiasu-BC that the corresponding probabilities are close to $(2^{-6})^2$ for the S-box in Round 7, about $(2^{-3.5 \cdot 4})^2$ for those in Round 5, $(2^{-7})^4$ for those in Round 4, and about 2^{-7} for that in Round 3, as in our theoretical analysis.

Key-Recovery. In addition, we implemented the 7-round boomerang attack with the key-recovery stage for Mini-Kiasu-BC, which yielded practical complexity. For this purpose, we created a structure of 2^{16} sets of 2^4 texts each by choosing a random base plaintext P and base tweak T_P, and iterated over all values of $P[0, 5, 10, 15]$ and $T[0]$. We collected the corresponding ciphertexts C. For all 2^{20} ciphertexts and for all 2^{16} candidates $K^7[0, 7, 10, 13]$, we derived D from the δ-shift and the corresponding shifted tweak $T_D = T_P[0] \oplus \delta$, obtained the corresponding plaintexts Q in a sorted list, and searched for matching quartets. For the correct key, we obtained always more than $55,000 \approx 2^{2 \cdot 16} \cdot \binom{2^4}{2} \cdot 2^{-16} \cdot 2^{-7.14} \approx 2^{15.77}$ quartets. We sorted the list lexicographically and used the first 16 quartets in order for subkey recovery. In total, we tested 100 randomly chosen keys with independently random base plaintext and base tweak. Each run identified the single correct key candidate with more than $55,000$ quartets, whereas the second highest candidate was suggested by only about 1/4 of that amount. So, we consider our experiments to show that $K^0[0, 5, 10, 15]$ and $K^7[0, 7, 10, 13]$ can reliably be recovered for Mini-Kiasu-BC and similar results can be expected for the full Kiasu-BC.

6 Biclique-Based Accelerated Exhaustive Search

This section describes a biclique-based accelerated exhaustive search on full Kiasu-BC, similar to the attacks in [10] on the full AES-128. Prior, we briefly recall the concept of such attacks before we explain how to tweak and extend the existing bicliques for the AES-128. Thereupon, we describe our attack in detail and give a complexity analysis for the individual steps.

6.1 Biclique-Based Accelerated Exhaustive Search

From Splice-and-Cut to Bicliques. Biclique cryptanalysis [22] originally developed as a generalization of initial structures [3,32] in the splice-and-cut meet-in-the-middle (MitM) attack framework by Aoki and Sasaki [2]. While the first applications of bicliques targeted hash functions, Bogdanov et al. demonstrated that they can also serve as a generic tool for accelerating the exhaustive search on keyed primitives [10], including but not limited to the full AES. Since then, bicliques have seen numerous applications on block ciphers. While being not directly a differential technique, they can be efficiently constructed from related-key differential trails under a few restrictions, which renders it senseful to consider them in a series of differential-based attacks.

Bicliques. A biclique is a bipartite graph between two sets of states $\mathcal{X}$ and $\mathcal{Y}$ such that every node $x_i \in \mathcal{X}$ is connected with every node $y_j \in \mathcal{Y}$. A biclique is called *balanced* if $|\mathcal{X}| = |\mathcal{Y}|$, and possesses a *dimension* d iff $|\mathcal{X}| = |\mathcal{Y}| = 2^d$. In the context of cryptanalysis, the sets $\mathcal{X}$ and $\mathcal{Y}$ represent start and end states of a part in a given primitive, respectively. In the following, assume we are given a cipher $E : \{0,1\}k \times \{0,1\}n \to \{0,1\}n$, which can be split into parts $E = \mathcal{B} \circ E_2 \circ E_1$, where we will construct a biclique over part $\mathcal{B}$. In an attack, the adversary divides the k-bit key space $\mathcal{K}$ into 2^{k-2d} distinct groups of 2^{2d} distinct elements each. Then, a biclique can efficiently test all keys of one group.
Among the techniques for constructing bicliques, *independent bicliques* [10] represent the most intuitive approach. Therein, the adversary searches for related-key differential trails Δ_i^K and ∇_j^K in forward and backward direction, respectively. Assume a starting state x_0. In each group of keys, the adversary chooses a base key $K_{0,0}$ and computes $y_0 = \mathcal{B}(K_{0,0}, x_0)$. Then, she uses the $2^d - 1$ forward key differences Δ_i^K, derives the forward keys $K_{i,0} = K_{0,0} \oplus \Delta_i^K$, and computes $y_i = \mathcal{B}(K_{i,0}, x_0)$, for $1 \leq i < 2^d$. In decryption direction, she uses the $2^d - 1$ backward key differences ∇_j^K, derives $K_{0,j} = K_{0,0} \oplus \nabla_j^K$, and computes $x_j = \mathcal{B}^{-1}(K_{0,j}, y_0)$, for $1 \leq j < 2^d$.

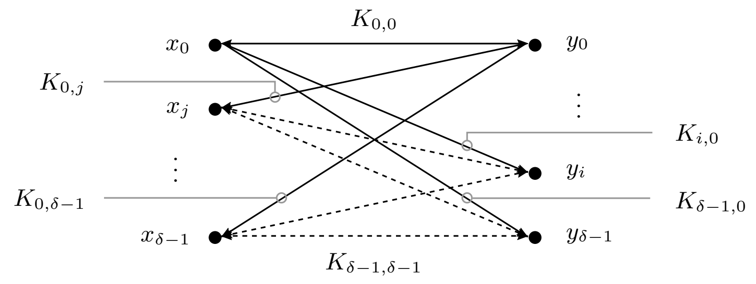


Fig. 8: Schematic illustration of a d-dimensional biclique. $\delta = 2^d - 1$.

Let denote $K_{i,j} = K_{0,0} \oplus \Delta_i^K \oplus \nabla_j^K$. Bogdanov et al. [10] showed that, if the forward and backward differential trails over $\mathcal{B}$ share *no* active non-linear components, then one can derive 2^{2d} trails with only the $2 \cdot 2^d$ computations above by combining any start state x_i with any end state y_j under key $K_{i,j}$:

$$x_j \xleftrightarrow[\mathcal{B}]{K_{i,j}} y_i, \qquad \text{for } 0 \leq i, j < 2^d.$$

Figure 8 depicts a d-dimensional biclique schematically. Alternatively, one can start the computations from y_0. Moreover, bicliques do not have to be balanced.

Matching. In splice-and-cut attacks, the remaining parts $E_2 \circ E_1$ are covered by a usual MitM matching procedure, whereas in a biclique-based accelerated exhaustive search, the biclique is combined with a partial matching over the remaining parts of the cipher wherein every key is tested.
Assume the biclique has been constructed over the last part of the cipher, here $\mathcal{B}$. The adversary obtains 2^d ciphertexts C_i (the states y_i above) and asks a decryption oracle for their corresponding plaintexts P_i. From each P_i, the adversary *precomputes* 2^d forward matchings states $\overrightarrow{v}_{i,0} = E_1(K_{i,0}, P_i)$. For all further keys $K_{i,j} = K_{i,0} \oplus \nabla_j^K$, she computes $\overrightarrow{v}_{i,j} = E_1(K_{i,j}, P_i)$, where only those parts over E_1 have to be *recomputed*, where the key differences ∇_j^K affect the state. Similarly, from each state S_j (the states x_j above), the adversary *precomputes* 2^d matching states $\overleftarrow{v}_{0,j} = E_2^{-1}(K_{0,j}, S_j)$. For all further keys $K_{i,j} = K_{0,j} \oplus \Delta_i^K$, she computes $\overleftarrow{v}_{i,j} = E_2^{-1}(K_{i,j}, S_j)$ similarly, where again only those parts over E_2 have to be recomputed, where the key differences Δ_i^K affect the states. The setting is shown in Figure 9. Evaluating only a part of the states $\overrightarrow{v}_{i,j}$ and $\overleftarrow{v}_{i,j}$ suffices for matching; if such a pair matches, the corresponding key candidate $K_{i,j}$ is verified, e.g. by the encryption of a different plaintext-ciphertext pair.

Complexity Calculation. The computational complexity comprises (1) the costs for constructing the biclique with $2 \cdot 2^d \cdot C_{\mathcal{B}}$ computations, C_{biclique}, (2) the costs for the $2^d \cdot C_{E_2 \circ E_1}$ precomputations, C_{precomp}, (3) the costs for $2^{2d} - 2^d$ recomputations, C_{recomp}, and (4) the costs for checking false-positive keys, C_{falsepos}, for each of the 2^{k-2d} key groups:

$$C = 2^{k-2d} \cdot \left(C_{\text{biclique}} + C_{\text{precomp}} + C_{\text{recomp}} + C_{\text{falsepos}} \right).$$

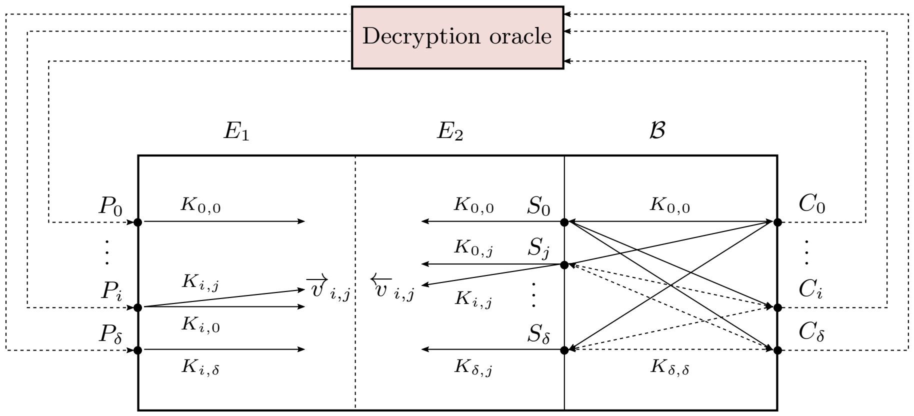


Fig. 9: Biclique-based attack setting with a biclique constructed over $\mathcal{B}$. $\delta = 2^d - 1$.

The attack needs about 2^d states stored in memory at a time. For the AES, the number of S-boxes that have to be recomputed per key has established as the standard metric [9,10]; in total, the AES-128 computes 200 S-boxes: $10 \cdot 16$ S-boxes in its message schedule and $10 \cdot 4$ S-boxes in its key schedule.

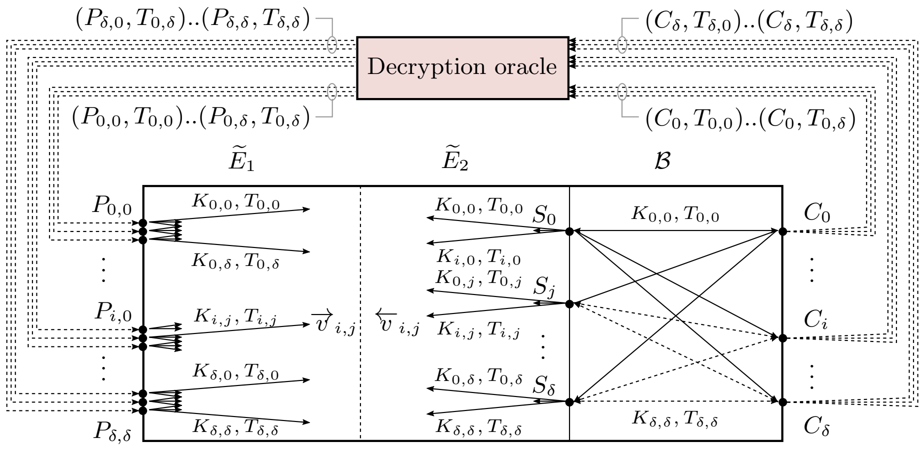
6.2 Tweaked Bicliques

For the analysis of a given tweakable primitive $\widetilde{E} : \{0,1\}k \times \{0,1\}t \times \{0,1\}n \rightarrow \{0,1\}n$, the tweak provides a second parameter to the computations. Similarly as in the previous section, we assume that $\widetilde{E}$ can be decomposed into parts $\widetilde{E} = \mathcal{B} \circ \widetilde{E}_2 \circ \widetilde{E}_1$, and for illustration, we construct a biclique over $\mathcal{B}$. [5]
For each group of keys, the adversary chooses a base key $K_{0,0}$ and an additional base tweak $T_{0,0}$ to compute $y_0 = \mathcal{B}(K_{0,0}, T_{0,0}, x_0)$. For each further forward computation, she can choose a tweak difference Δ_i^T, derive $T_{i,0} = T_{0,0} \oplus \Delta_i^T$, and compute $y_i = \mathcal{B}(K_{i,0}, T_{i,0}, x_0)$, for $1 \le i < 2^d$. Similarly, she can choose $2^d - 1$ further tweak differences ∇_j^T and use $T_{0,j} = T_{0,0} \oplus \nabla_j^T$ as tweaks for the backward computations $x_j = \mathcal{B}^{-1}(K_{0,j}, T_{0,j}, y_0)$, for $1 \le j < 2^d$. Like the keys, we compose the tweaks by XORing their forward and backward differences $T_{i,j} = T_{0,0} \oplus \Delta_i^T \oplus \nabla_j^T$. Again, if the differential paths of forward and backward computations share *no* active non-linear components, then it holds that

$$y_i = \mathcal{B}(K_{i,j}, T_{i,j}, x_j) \quad \text{and} \quad x_j = \mathcal{B}^{-1}(K_{i,j}, T_{i,j}, y_i), \quad \text{for } 0 \le i, j < 2^d.$$

Matching. When considering tweaks as an additional parameter in the biclique, the matching procedure becomes more involved. When the biclique is constructed at the end of $\widetilde{E}$ as in our imagined setting, each ciphertext $C_i = y_i$ in the group must be requested from the decryption oracle for every tweak $T_{i,j}$. So, one has 2^{2d}

[5] Note that using the tweak as parameter in biclique-based attacks has already been used in the attacks on Skein/ThreeFish in [21] to obtain local collisions.



$(P_{\delta,0}, T_{0,\delta})..(P_{\delta,\delta}, T_{\delta,\delta})$
$(C_\delta, T_{\delta,0})..(C_\delta, T_{\delta,\delta})$
Decryption oracle
$(P_{0,0}, T_{0,0})..(P_{0,\delta}, T_{0,\delta})$
$(C_0, T_{0,0})..(C_0, T_{0,\delta})$
$\widetilde{E}_1$
$\widetilde{E}_2$
$\mathcal{B}$
$P_{0,0}$
$P_{i,0}$
$P_{\delta,\delta}$
$K_{0,0}, T_{0,0}$
$K_{0,\delta}, T_{0,\delta}$
$K_{i,j}, T_{i,j}$
$K_{\delta,0}, T_{\delta,0}$
$K_{\delta,\delta}, T_{\delta,\delta}$
$\overrightarrow{v}_{i,j}$
$\overleftarrow{v}_{i,j}$
$K_{i,0}, T_{i,0}$
$K_{0,j}, T_{0,j}$
$K_{0,\delta}, T_{0,\delta}$
S_0
S_j
S_δ
C_0
C_i
C_δ

Fig. 10: Attack setting with a tweaked biclique constructed over $\mathcal{B}$. $\delta = 2^d - 1$.

ciphertexts $C_{i,j}$ and their related plaintexts $P_{i,j}$ for one key group, as illustrated in simplified manner in Figure 10. By fixing C_0 and $T_{0,0}$ over all key groups, one can manage that the total data complexity increases by *only* a factor of 2^d compared to the tweakless setting.

In the following, the adversary computes 2^{2d} forward matchings states $\overrightarrow{v}_{i,j} = \widetilde{E}_1(K_{i,j}, T_{i,j}, P_{i,j})$. So, if E is not terribly weak, the precomputation approach is most likely inapplicable for the forward computations of the matching phase. In the backward phase, the adversary can precompute 2^d matching states $\overleftarrow{v}_{0,j} = \widetilde{E}_2^{-1}(K_{0,j}, T_{0,j}, S_j)$. For all further computations $\overleftarrow{v}_{i,j} = \widetilde{E}_2^{-1}(K_{i,j}, T_{i,j}, S_j)$, she still needs to recompute only those parts over $\widetilde{E}_2$, where the key differences Δ_i^K and tweak differences Δ_i^T affect the internal states.

Optimizations. The calculation of the computation costs is similar to that in the tweakless setting. Though, if the bicliques are constructed over the final part of the cipher as in our example, it is useful to employ only tweak differences Δ_i^T and fix all tweak differences ∇_j^T to 0, for $0 \leq j < 2^d$. Then, the data complexity remains the same as in the tweakless setting, and the precomputation approach can be applied over the full matching parts. Yet, the effect of the tweak differences on the state and the number of recomputed parts has to be taken into account for the backward recomputations.

Similarly, if the bicliques are constructed at the beginning part of the cipher, it is useful to employ only tweak differences ∇_j^T and fix all tweak differences Δ_i^T to zero, for $0 \leq j < 2^d$. Then, the tweak differences ∇_j^T have to be considered only in the forward computations of the matching phase.

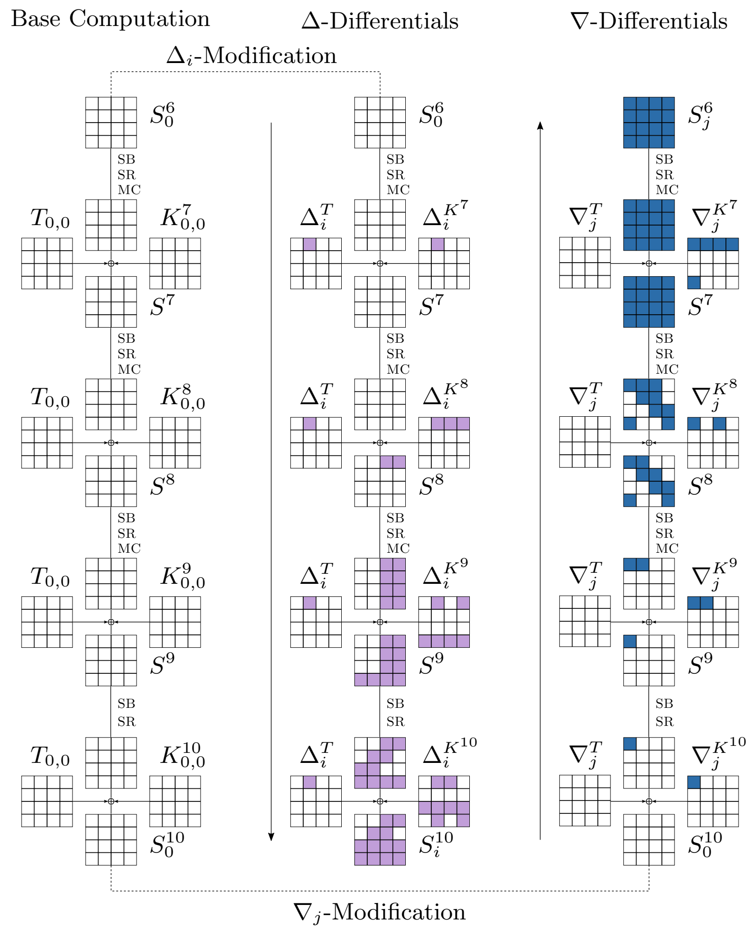


Fig. 11: Independent biclique over the final four rounds of KIASU-BC. Active bytes in the differences are highlighted.

6.3 Biclique-Based Attack on Full Kiasu-BC

We construct bicliques of dimension $d = 8$ over the Rounds 7 through 10 of KIASU-BC, where we fix C_0 over all bicliques. We partition the key space into $2^{k-2d} = 2^{112}$ groups, where we choose the base keys $K_{0,0}$ of the groups from K^{10}. Concerning the forward key differentials, we derive the key differences Δ_i^K by iterating over the 2^8 values of $K^7[4]$ and deriving the corresponding differences in K^{10}. For the backward key differentials, we derive the key differences ∇_j^K directly from K^{10} by iterating over the 2^8 values of $K^{10}[0]$.
Additionally, we choose a base tweak $T_{0,0}$ arbitrarily as long as it is identical for all key groups. We derive tweak differences Δ_i^T by iterating over the 2^8 values of $T[2]$ (which represents Byte 4 in states and keys) such that $\Delta_i^T[2] = \Delta_i^{K^r}[4]$ holds, for $0 \le i < 2^8$, for $r \in \{7, 8, 9, 10\}$, i.e. the round keys K^7, K^8, K^9, and

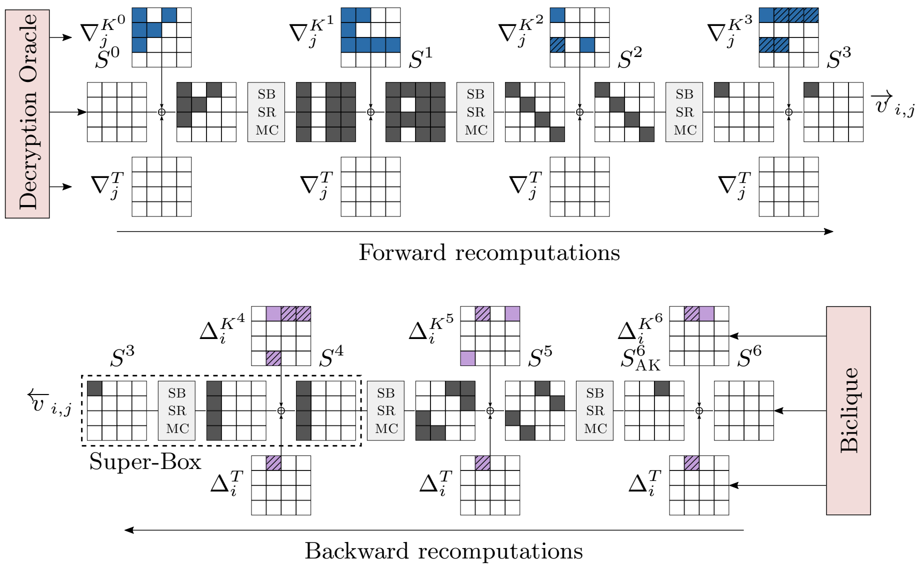


Fig. 12: Matching part of the accelerated-exhaustive-search attack on Kiasu-BC. Hatched active bytes need not be recomputed.

K^{10}. Note that this property depends in non-trivial manner on the difference propagation through the AES key schedule. Our choice is motivated by the goal to have key and tweak difference cancel each other in the XOR at the end of Round 7, so that the forward differentials in our bicliques have no active bytes in the states in Round 8. We do not introduce tweak differences ∇_j^T in the backward trails of the biclique, i.e. $\nabla_j^T = 0$ for $0 \leq j < 2^d$. The resulting four-round biclique trails are shown in Figure 11. Due to the choice of key and tweak differences, we could increase the length of the biclique by one round compared to the independent bicliques on the AES-128 [10]. Note that only 12 bytes are active in the ciphertexts. Moreover, the difference in bytes $\Delta S_i^{10}[10] = \Delta S_i^{10}[14]$ is always equal due to the key schedule. So, by fixing C_0 over all key groups, we obtain at most 2^{88} different ciphertexts.

Matching. The matching covers the Rounds 1 through 6. We match in Byte 0 of the state S^3. For each group, the 2^8 ciphertext-tweak pairs $(C_i, T_{i,0})$ are given to a decryption oracle to obtain the corresponding plaintexts P_i. Since we fixed all tweak differences ∇_j^T to zero, these requests suffice and we can apply the recomputation approach in the forward *and* backward steps of the matching procedure. In the following, one precomputes and stores the states $\overrightarrow{v}_{i,0} = \widetilde{E}_1(K_{i,0}, T_{i,0}, P_i)$. For each further key $K_{i,j} = K_{i,0} \oplus \nabla_j^K$, one partially computes $\overrightarrow{v}_{i,j} = \widetilde{E}_1(K_{i,j}, T_{i,0}, P_i)$ where only those parts have to be recomputed where the key differences ∇_j^K affect the states. In backward direction, one precomputes the states $\overleftarrow{v}_{0,j} = \widetilde{E}_2^{-1}(K_{0,j}, T_{0,0}, S_j)$; for each further key

$K_{i,j} = K_{0,j} \oplus \Delta_i^K$ and tweak $T_{i,0} = T_{0,0} \oplus \Delta_i^T$, one partially computes the states $\overleftarrow{v}_{i,j} = \widetilde{E}_2^{-1}(K_{i,j}, T_{i,0}, S_j)$ where only those parts have to be recomputed where the key differences Δ_i^K and tweak differences Δ_i^T affect the states.
The bytes that have to be recomputed in forward and backward direction are highlighted in Figure 12. Hatched active bytes in the keys and tweaks need not be recomputed for our partial matching in $S^3[0]$. In forward direction, we have to recompute only five S-boxes in the first, 14 in the second and four S-boxes in the third round, in addition to one S-box in the key schedule, $\nabla K_j^1[14]$.[6] In backward direction, we have to recompute one S-box in Round 4, four S-boxes in Round 5, and four S-boxes in Round 6, in addition to one S-box in the key schedule, $\Delta K_i^5[12]$. Note that the choice of $\Delta_i^T[2] = \Delta_i^K[4]$ implies that it also holds that $\Delta_i^T[2] = \Delta K_i^5[4] = \Delta K_i^6[4]$. As a result, the differences in these bytes cancel, which saves one active byte in $S_{\text{AK}}^6[4]$. Thus, we do not have to recompute the four S-boxes $S^5[3, 4, 9, 14]$. In total, the number of S-boxes that we have to recompute sums up to $5 + 14 + 4 + 1 + 1 + 4 + 4 + 1 = 34$.

Attack Complexity. The costs for constructing a biclique are upper bounded by 2^8 backward computations of four rounds each and $2^8 - 1$ forward computations of two rounds each, or $2^{7.26}$ full encryptions of Kiasu-BC. The precomputation costs are given by at most $2 \cdot 2^8$ computations of three rounds or $2^9 \cdot 3/10 \approx 2^{7.26}$ full encryptions. The recomputation costs are given by at most $2^{16} \cdot 34/200 \approx 2^{13.44}$ encryptions. Since we match in a single byte, we can expect $2^{16} \cdot 2^{-8} = 2^8$ false positive keys per key group on average. The encryption costs for 2^{88} ciphertexts by the decryption oracle is negligible. Therefore, the computational complexity is about

$$C \approx 2^{112} \cdot \left(2^{7.26} + 2^{7.26} + 2^{13.44} + 2^8\right) + 2^{64} \approx 2^{125.51} \text{ encryptions.}$$

The memory complexity is upper bounded by storing 2^8 states at a time.

Improvement from the Sieve-in-the-Middle Approach. One can apply the sieve-in-the-middle approach by Canteaut et al. [12] to further reduce the computational complexity. Concerning the backward recomputations over the fourth round, we are interested only in the first column. The column is similar to an AES Super-box [14], mapping a 32-bit input $S^4[0, 1, 2, 3]$ together with a 32-bit (twea)key $K^4[0, 1, 2, 3] \oplus (T[0], T[1], 0, 0)$ to a 32-bit output $S^3[0, 5, 10, 15]$. In an offline phase before the attack, one can construct 2^{32} lookup tables (one table per value of $K^4[0, 1, 2, 3]$), which require $2^{32} \cdot 2^{32} \cdot 4 = 2^{66}$ bytes of memory in total. In the recomputation phase, one can then lookup for each value $S^4[0, 1, 2, 3]$, if there exists a valid transition from $S^4[0, 1, 2, 3]$ to $S^3[0]$ in the table for the current value $K_{i,j}^4[0, 1, 2, 3]$. This approach allows to save the costs of the five S-boxes in Rounds 4 and 5 of the backward recomputation part for a single lookup, which reduces the recomputation costs from 34 S-boxes to $29 + 1$ lookups compared to 200 S-boxes in the full Kiasu-BC. So, in contrast to [8,12],

[6] The AES key schedule uses the rightmost column of the round keys K^0 through K^9 as inputs to S-boxes.

we include the table lookup in the recomputation costs. Consequently, the computational complexity becomes about

$$C \approx 2^{112} \cdot \left(2^{7.26} + 2^{7.26} + 2^{13.26} + 2^{8}\right) + 2^{64} \approx 2^{125.34} \text{ encryptions.}$$

The required memory increases to $2^8 \cdot 16 + 2^{66} \approx 2^{66}$ bytes, or 2^{62} state equivalents. Yet, as mentioned in [12], this improvement is beneficial only on platforms on which a lookup in a table of 2^{32} elements is faster than recomputing five S-boxes.

7 Conclusion

This work proposed differential-based attacks on eight rounds of Kiasu-BC, which share the idea that the tweak input allows to construct a local collision. While the designers already indicated that there exist boomerangs on at most seven rounds, they had to consider attacks in the single- as well as in the related-key setting. Our described boomerang and rectangle attacks do not violate their claims, but concretize the security threats in the single-key model and illustrate that Kiasu-BC possesses one round less security than the AES-128. Moreover, the claim that the bounds of existing attacks on the AES for other attacks than boomerangs, conventional differentials, and meet-in-the-middle can be translated without modification to Kiasu-BC does not hold in general, which was already observed by [15] and was confirmed by our impossible-differential attack.

Acknowledgments. The authors thank Ralph Ankele, Christof Beierle, and Maria Eichlseder for the fruitful discussions at the DISC workshop in March 2016 at Bochum, and the reviewers for their helpful comments.

References

1. Ahmed Abdelkhalek, Mohamed Tolba, and Amr M. Youssef. Impossible Differential Cryptanalysis of 8-round Kiasu-BC, 2016. To appear.
2. Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography'08*, volume 5381 of *LNCS*, pages 103–119. Springer, 2008.
3. Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO'09*, volume 5677 of *LNCS*, pages 70–89. Springer, 2009.
4. Behnam Bahrak and Mohammad Reza Aref. Impossible differential attack on seven-round AES-128. *IET Information Security*, 2(2):28–32, 2008.
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. Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 1–18. Springer, 2009.
8. Andrey Bogdanov, Donghoon Chang, Mohona Ghosh, and Somitra Kumar Sanadhya. Bicliques with Minimal Data and Time Complexity for AES. In Jooyoung Lee and Jongsung Kim, editors, *ICISC*, volume 8949 of *LNCS*, pages 160–174. Springer, 2014.
9. Andrey Bogdanov, Elif Kavun, Christof Paar, Christian Rechberger, and Tolga Yalcin. Better than Brute-Force - Optimized Hardware Architecture for Efficient Biclique Attacks on AES-128. In *ECRYPT Workshop, SHARCS - Special Purpose Hardware for Attacking Cryptographic Systems*, 2012.
10. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. Cryptology ePrint Archive, Report 2011/449, 2011. `http://eprint.iacr.org/`.
11. 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.
12. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-Middle: Improved MITM Attacks. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *LNCS*, pages 222–240. Springer, 2013.
13. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
14. Joan Daemen and Vincent Rijmen. Understanding Two-Round Differentials in AES. In *SCN*, volume 4116 of *LNCS*, pages 78–94, 2006.
15. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Square Attack on 7-Round Kiasu-BC. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS*, volume 9696 of *LNCS*, pages 500–517. Springer, 2016.
16. Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. Submission to NIST (Round 3), 2010.
17. Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. KIASU v1.1, 2014. First-round submission to the CAESAR competition, `http://competitions.cr.yp.to/-caesar-submissions.html`.
18. 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.
19. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: the TWEAKEY framework. Cryptology ePrint Archive, Report 2014/831, 2014. `http://eprint.iacr.org/`.
20. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *FSE*, volume 1978 of *LNCS*, pages 75–93. Springer, 2000.
21. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. Cryptology ePrint Archive, Report 2011/286, 2011. `http://eprint.iacr.org/`.
22. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In Anne Canteaut, editor, *FSE*, volume 7549 of *LNCS*, pages 244–263. Springer, 2012.

23. Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011.
24. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002.
25. Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New Impossible Differential Attacks on AES. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *LNCS*, pages 279–293. Springer, 2008.
26. Hamid Mala, Mohammed Dakhilalian, Vincent Rijmen, and Mahmoud Modarres-Hashemi. Improved Impossible Differential Cryptanalysis of 7-Round AES-128. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT'10*, volume 6498 of *LNCS*, pages 282–291. Springer, 2010.
27. Kazuhiko Minematsu. Building blockcipher from small-block tweakable blockcipher. *Designs, Code and Cryptography*, 74(3):645–663, 2015.
28. Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Trans. Information Theory*, 57(4):2517–2521, 2011.
29. Yusuke Naito. Full PRF-Secure Message Authentication Code Based on Tweakable Block Cipher. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec*, volume 9451 of *LNCS*, pages 167–182. Springer, 2015.
30. National Institute of Standards and Technology. FIPS 197. *National Institute of Standards and Technology, November*, pages 1–51, 2001.
31. Thomas Peyrin and Yannick Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO I*, volume 9814 of *LNCS*, pages 33–63. Springer, 2016.
32. Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 134–152. 2009.
33. Richard Schroeppel and Hilarie Orman. The Hasty Pudding Cipher. *AES candidate submitted to NIST*, 1998.
34. 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.
35. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

Supplementary Material

A Rectangle Attack on 7-Round Kiasu-BC

This section describes a rectangle attack on the final seven rounds of Kiasu-BC.

Differentials. The upper and lower trails used in our attack are illustrated in Figure 13. Both trails share the same idea: start from a state directly after the SubBytes operation with a difference ΔS^i_{SB} wherein only the bytes on the main diagonal are active, and choose it in a way such that the differences in the state ΔS^i_{MC} and in the tweak ΔT will cancel each other after the first round, i.e., $\Delta S^i = 0$. Then, it follows that $\Delta S^{i+1} = \Delta T$ contains only one and ΔS^{i+2}

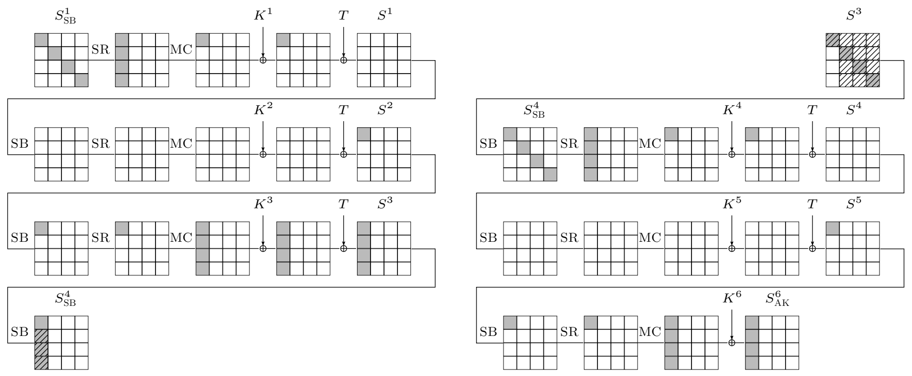


Fig. 13: Upper (**left**) and lower (**right**) trails of our boomerang attacks on 7-round Kiasu-BC. Hatched bytes are not traced through the middle SubBytes operation.

only four active bytes. We employ a *ladder switch* in the transition phase: in the upper trail, only Bytes $S^{i+2}[0, 4, \ldots, 15]$ are traced through the final SubBytes operation after Round 3, of which only Byte 0 is active. So, only the inactive Bytes $S^{i+2}[1, 2, 3]$ are traced through SubBytes in Round 4.

Attack Procedure. We append a key-guessing phase over the final round. We use the fact that the final round omits the MixColumns operation so that only four key bytes have to be considered. Figure 14 depicts the steps that wrap the upper and lower trails. In the following, we detail the individual steps of the attack.

Upper Trail. The texts in each set $\mathcal{T}^i$ in a structure cover all values in Bytes $S^0[0, 5, 10, 15]$. Since T^i is equal for all texts in a set, and since AddTweak, AddKey, SubBytes, and ShiftRows are permutations, the texts in the set cover all values in the first column also at $S^1_{\text{SR}}[0, 1, 2, 3]$ (see Figure 13). Since our upper trails need pairs with non-zero tweak difference $\Delta\ T[0]$, we cannot form pairs $(P^i_j, P^i_{j'})$ from the same set, but have to form pairs across sets instead. Consider two distinct sets $\mathcal{T}^i$ and $\mathcal{T}^j$ in the same structure. Their tweak difference in Byte 0 is given by $\Delta\ T_{i,j}[0] = \langle i \rangle \oplus \langle j \rangle \neq 0$. Since MixColumns is linear, there exists a unique difference $\Delta S^1_{\text{SR}}[0, 1, 2, 3]$ that is mapped by MixColumns^{-1} to $(\Delta T_{i,j}[0], 0, 0, 0)$. So, a pair that shall follow our upper trail must have that difference $\Delta S^1_{\text{SR}}[0, 1, 2, 3]$.
Consider two texts P^i and P^j from two distinct sets $\mathcal{T}^i$ and $\mathcal{T}^j$, and let $S^1_{\text{SR},i}$ and $S^1_{\text{SR},j}$ denote their corresponding states after ShiftRows in the first round. For each state $S^1_{\text{SR},i} \in \mathcal{T}^i$, there is exactly one text $S^1_{\text{SR},j} \in \mathcal{T}^j$ such that $S^1_{\text{SR},i}[0, 1, 2, 3] \oplus S^1_{\text{SR},j}[0, 1, 2, 3] = \Delta S^1_{\text{SR}}[0, 1, 2, 3]$. So, we have 2^{32} valid pairs between any two distinct sets $\mathcal{T}^i, \mathcal{T}^j$ in a structure, with $0 \leq i < j < 2^8$. Since there are $\binom{2^8}{2} \approx 2^{15}$ pairs of sets, we obtain approximately $2^{15} \cdot 2^{32} = 2^{47}$ pairs

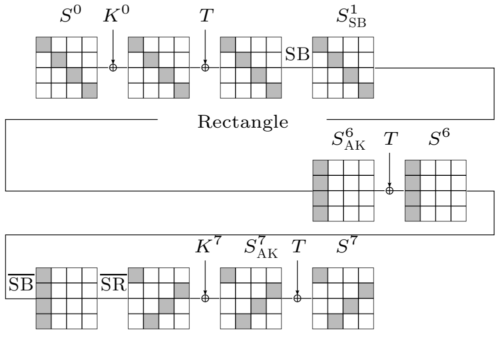


Fig. 14: The trail in our 7-round rectangle attack.

per structure which collide at State S^1. We call them *valid pairs*, hereafter. So, we obtain 2^{n+47} valid pairs from 2^n structures.
After the SubBytes operation in Round 3, all valid pairs will have only Byte 0 active in their difference ΔS^3_{SB}. So, at most 255 possible differences can occur in the first column $\Delta S^3_{\text{AK}}[0, 1, 2, 3]$. From the tweak addition in Round 3, Byte 0 can be affected by any of 256 different tweak values $T[0]$. So, a valid pair can have at most $255 \cdot 256 \approx 2^{16}$ differences after E_1. From one structure, we obtain 2^{47} valid pairs, each of which have one of the 2^{16} possible differences after E_1.

Transition Phase. For two pairs (P, P') and (Q, Q'), we denote their corresponding states after the upper trail by (U, U') and (V, V'), respectively. Moreover, we denote their corresponding tweaks as $(T_U, T_{U'})$ and $(T_V, T_{V'})$, respectively. We search for pairs (U, U') and (V, V') that form quartets (U, U', V, V') such that it holds $U \oplus V = U' \oplus V' = \gamma$.
At the beginning of the lower trail, we trace only inactive bytes through the SubBytes operation in Round 4. For a quartet, we need to have exactly the difference $\Delta S^4_{\text{AK}}[0] = T_U[0] \oplus T_V[0]$ at the end of Round 4. Since all considered operations in Round 4 are linear, there is a unique mapping from $\Delta S^4_{\text{AK}}[0, 1, 2, 3]$ backwards to the diagonal difference $\Delta S^4_{\text{SB}}[0, 5, 10, 15]$. Thus, it must hold for a quartet that $(U \oplus V) = \Delta S^4_{\text{SB}}$ for $\Delta T[0] = (T_U[0] \oplus T_V[0])$. From Proposition 1, it follows then also that $(U' \oplus V') = \Delta S^4_{\text{SB}}$. However, the second pair (U', V') will follow our lower trail only if their tweaks share the same difference, i.e. only if $\Delta T = (T_U \oplus T_V) = (T_{U'} \oplus T_{V'})$.
So, the 2^{n+47} pairs (U, U') after E_1 distribute over at most 2^{24} possible *buckets*, concerning the 2^{16} possible differences ΔS^4_{SB} and the 2^8 possible tweak differences $\Delta T_{U,U'}[0] = T_U[0] \oplus T_{U'}[0]$. So, we can expect about 2^{n+23} pairs per bucket.[7] Pairs from the same bucket can be combined with each other to 2^{2n+45} quartets. Though, since we need a non-zero tweak difference in the lower trails, we require that $T_U[0] \neq T_V[0]$ and $T_{U'}[0] \neq T_{V'}[0]$; we still have about $2^{2n+45} \cdot 255/256 \approx 2^{2n+45}$ quartets per bucket. From 2^{24} buckets, there are in total $2^{24} \cdot 2^{2n+45} = 2^{2n+69}$ quartets. Since a *matching* quartet has the differ-

[7] Note that any other than the uniform distribution allows to build more quartets.

ence γ after E_1 with probability 2^{-128}, we expect about $2^{2n+69} \cdot 2^{-128} = 2^{2n-59}$ matching quartets.

Lower Trail. All matching quartets have a local collision after Round 4 in both pairs; since we required that $(T_U \oplus T_V) = (T_{U'} \oplus T_{V'})$, matching quartets share the same difference ΔS^5 in both their pairs. The 255 possible differences ΔS^5 imply at most 255 possible differences ΔS^6_{AK} where only the first column is active. We restrict the lower trails to quartets whose pairs produce a difference δ that is among the 255 possible ones, fixed over all quartets. Since the AES S-box can produce only $127 \approx 2^7$ input-output trails, the probability that both pairs in one quartet have the same difference ΔS^6_{SB} is about $(2^{-7})^2$. Since all further operations in the lower trail are linear, both pairs in a matching quartet also possess the same difference $\Delta S^6_{\mathrm{AK}} = \delta$. We call those *correct quartets*, hereafter. Thus, we expect $2^{2n-59} \cdot 2^{-14} = 2^{2n-73}$ correct quartets with difference δ.

Steps. The attack consists of the following steps:

0. Precompute a hash table $\mathcal{H}_{\mathrm{precomp}}$ containing all possible 4-byte transitions of values x and $x' = x \oplus \delta[0,1,2,3] \oplus (\Delta T[0], 0, 0, 0)$ through SubBytes and ShiftRows and their output values y and y' after SubBytes and ShiftRows. Index the table by the output difference $y \oplus y' = \Delta S^7_{\mathrm{SR}}[0,7,10,13]$ and $(\Delta T[0], 0, 0, 0)$. There are 2^{32} values x and 2^8 tweak differences $\Delta T[0]$; so, $\mathcal{H}_{\mathrm{precomp}}$ will contain $2^8 \cdot 2^{32} = 2^{40}$ buckets, each with 127^4 valid transitions.
1. Choose 2^n structures of 2^{40} plaintext-tweak tuples (P, T) each and ask for their ciphertexts C. Invert the final tweak addition.
2. Insert the 2^{n+40} ciphertexts C together with the index of their corresponding structure into a hash table with 2^{96} buckets indexed by the ciphertext Bytes 1-6, 8, 9, 11, 12, 14, and 15.
3. For each ciphertext pair (C, C') which fell into the same bucket, denote their corresponding structures by S and S', respectively. Attach to C also the index of S', and attach to C' also the index of S. There are about $(2^{n+40})^2/2 \approx 2^{2n+79}$ pairs distributed over 2^{96} buckets. So, we expect about $2^{2n+79} \cdot 2^{-96} = 2^{2n-17}$ collisions from pairs that fell into equal buckets.
4. For each colliding pair, lookup in $\mathcal{H}_{\mathrm{precomp}}$ if there exists a valid transition through the inverse Round 7 to δ. Since there are four active S-boxes, and 127 possible transitions, 127^4 of them are valid among the 256^4 possible ones. This gives approximately a four-bit filter which reduces the number of collisions to about 2^{2n-21}.
5. Initialize a list for all possible values of $K^0[0,5,10,15]$ and $K^7[0,7,10,13]$.
6. For each of the remaining 2^{2n-21} colliding pairs (C, C'), use $(C[0,7,10,13] \oplus C'[0,7,10,13], \Delta T[0])$ to lookup in $\mathcal{H}_{\mathrm{precomp}}$ the possible values $y = S^6_{\mathrm{SR}}[0,7,10,13]$ and $y' = {S'}^6_{\mathrm{SR}}[0,7,10,13]$ that could map C and C' to δ. We expect one suggestion one average (i.e. two in 126 out of 255 cases, four in 1 out of 255 cases, and none in 129 cases); derive the value of $K^7[0,7,10,13] = C[0,7,10,13] \oplus S^6_{\mathrm{SR}}[0,7,10,13]$. This yields a 32-bit filter that reduces the number of remaining pairs from 2^{2n-17} to 2^{2n-49}.

7. To identify correct quartets, we search for pairs of pairs (C, D) and (C', D'), where the corresponding structure indices $S_C = S_{C'}$ and $S_D = S_{D'}$ match, where we denote by S_C the structure index of C, by $S_{C'}$ that of C' and so on. Otherwise, their corresponding plaintexts (P, P') and (Q, Q') cannot have difference α. We can model the search for matching structures by a probabilistic experiment from graph theory: given an undirected graph with $\widehat{n} = 2^n$ nodes (corresponding to the 2^n structures), we add uniformly at random $\widehat{m} = 2^{2n-49}$ edges. The end points of an edge correspond to the structure indices of a pair S_C and S_D. It is a well-known fact that when throwing M balls into N bins, one expects $\binom{M}{2}/N$ collisions on average. Since our graph has $\widehat{n}^2$ possible edges (C and D can be part of the same structure, i.e. the same node in the graph), this yields for $M = \widehat{m} = 2^{2n-49}$ and $N = \hat{n}^2 = (2^n)^2$ an expected number of
$$\frac{\binom{2^{2n-49}}{2}}{(2^n)^2} = \frac{2^{4n-98}}{2 \cdot 2^{2n}} = 2^{2n-99}$$
collisions of two edges in our graph model. Since each edge corresponds to a pair (C, D), we obtain about 2^{2n-99} quartets whose structures indices $S_C = S_{C'}$ and $S_D = S_{D'}$ match per key candidate for K^7.
8. For each potential quartet (C, C', D, D'), we consider their corresponding plaintexts (P, P', Q, Q') next. Let S^1_P, $S^1_{P'}$, S^1_Q, $S^1_{Q'}$ denote their corresponding states after the first round. For a correct quartet, it holds that $S^1_P = S^1_{P'}$ and analogously, $S^1_Q = S^1_{Q'}$. There is a unique mapping from the tweak difference $S^1_{\text{SR},P} \oplus S^1_{\text{SR},P'}$ backwards to $S^1_{\text{SB},P} \oplus S^1_{\text{SB},P'}$. Since the S-box has two solutions on average, there are on average two key values per byte $K^0[0, 5, 10, 15]$ such that P and P' are mapped to the correct difference ΔS^1_{SB} which yields $S^1_P \oplus S^1_{P'} = 0$. So, we derive those values for $K^0[0, 5, 10, 15]$ for P and P'. We derive those key values analogously for Q and Q'. For each key byte, we have $2 \cdot 2$ combinations on average, i.e., four pairs among which any pair matches with probability about $4 \cdot 2^{-8} = 2^{-6}$. So, an invalid quartet survives this four-byte filter with probability $(2^{-6})^4 = 2^{-24}$. For each surviving pair, increment the counter for the current $K^0[0, 5, 10, 15]$ and $K^7[0, 7, 10, 13]$. Over all 2^{32} keys K^7, we expect $2^{32} \cdot 2^{2n-99} \cdot 2^{-24} = 2^{2n-91}$ quartets.
9. For $n = 38$, we expect $2^{2n-73} = 2^3$ correct quartets. So, the probability is significant to obtain at least two correct quartets and the counter for the correct subkey will be at least two. The probability for a false-positive count is about $2^{2n-91} = 2^{-15}$ over all 2^{32} keys on average. So, we output the candidate(s) with the highest counters.

The above attack recovers 32 bits of K^0, $K^0[0, 5, 10, 15]$, and 32 bits of K^7, $K^7[0, 7, 10, 13]$. We propose to conduct an following, similar attack with shifted differential trails consisting basically of the same steps. The shifted differential trails are given in Supporting Material C for the sake of completeness. The attack requires again 2^n structures of 2^{40} texts, iterating over plaintext bytes $P[3, 4, 9, 14]$ and tweak byte $T[2]$, and can recover another 32 bits of K^0,

$K^0[3,4,9,14]$, and 32 bits of K^7, $K^7[1,4,11,14]$ with the same probability as above. The remaining 64 bits of K^0 and/or K^7 can be found e.g. exhaustively.

Complexity Analysis. The total computational complexity of the attack results the following step complexities:

– The precomputations require about $2^{40} \cdot 4/16 \approx 2^{38}$ single-round decryptions.
– **Step 1** requires 2^{n+40} full encryptions.
– **Step 2** requires 2^{n+40} memory accesses (MAs) with an efficient data structure.
– **Step 3** requires 2^{n+40} MAs to the buckets and $2 \cdot 2^{2n-17}$ MAs to attach the structure indices.
– **Step 4** requires 2^{2n-17} MAs.
– **Step 6** requires 2^{2n-21} MAs into $\mathcal{H}_{\text{precomp}}$.
– **Step 7** requires $2^{32} \cdot 2 \cdot 2^{2n-49} = 2^{2n-16}$ MAs of the remaining pairs into a hash table according to their structure indices.
– **Step 8** requires $2^{32} \cdot 4 \cdot 2^{2n-99} \cdot 4/16 = 2^{2n-67}$ single-round decryptions of the first round for each remaining quartet (P, P', Q, Q'). Moreover, it requires $2^{32} \cdot 2^{2n-99} \cdot 2^{-24} + 2^3 = 2^{2n-91} + 2^3$ MAs to insert key candidates.

So, for $n = 38$, the overall complexity of one execution of the attack is about

$$\begin{aligned} T &\approx 2^{n+40} + \left(2^{38} + 2^{2n-67}\right) \cdot \frac{1}{7} \text{ Enc } + \\ &\quad 2^{n+40} + 2^{n+40} + 2^{2n-16} + 2^{2n-17} + 2^{2n-21} + 2^{2n-16} + 2^{2n-91} + 2^3 \text{ MA} \\ &\approx 2^{78} \text{ Enc } + 2^{79} \text{ MA}. \end{aligned}$$

Since we propose to execute the attack once again in shifted form, and to subsequently recover 64 keys bits with exhaustive search, we obtain approximately 2^{79} encryptions and 2^{80} memory accesses. The data complexity is $2 \cdot 2^{78} = 2^{79}$ chosen plaintexts. The attack requires memory for 2^{n+40} states for the ciphertexts, 2^{40} eight-byte values in $\mathcal{H}_{\text{precomp}}$, $2^{2n-21} \cdot 2$ pairs for the remaining pairs after Step 4, and 2^{64} single-byte key counters. All remaining steps require less memory. The second execution of the attack can reuse existing memory. So, the attack needs memory for at most

$$M \approx \left(2^{78} \cdot 16\right) + \left(2^{40} \cdot 8\right) + \left(2^{55} \cdot 16 \cdot 2\right) + 2^{64} \approx 2^{82} \text{ bytes or } 2^{78} \text{ states.}$$

B Boomerang Attack on 7-Round Kiasu-BC

We can transform the rectangle attack from Supporting Material A into a boomerang attack with lower computational complexity. For this purpose, we reuse the same differentials and final round, but change the attack procedure.

Differentials. Similar as before, we choose 2^n structures, with 2^8 tweak-dependent sets of plaintexts each; for efficiency reasons, let us consider not $2^8 \times 2^{32}$ texts, but let denote the number of distinct texts in a structure by 2^p and assume uniform distribution over the tweaks. From our texts, we obtain $2^n \cdot \binom{2^p}{2} \cdot 2^{-32} = 2^{n+2p-33}$ pairs (P, P') which collide after the first round. The remaining steps in the upper trail are passed with probability one. Next, we trace the pairs (C, D) and (C', D') with their associated tweaks (T_C, T_D) and $(T_{C'}, T_{D'})$ backwards through the lower trail. The probability that the active S-box in Round 6 yields the correct tweak difference $\Delta S^5 = T_C \oplus T_D$ is 2^{-6} for each pair. In contrast to our rectangle attack, we can choose the tweak difference in our ciphertexts here. So, we have a probability of 2^{-12} for the lower trail. The four active S-boxes at the transition phase are not traced through SubBytes in the lower trail.
Next, we trace both pairs backwards through the upper trail. The S-box at S^4_{SB} needs to have only the same difference for both pairs. The second pair matches the difference of the first one with probability about 2^{-7}. For equal differences ΔS^3_{AK} in both pairs, we need $T_P[0] \oplus T_{P'}[0] = T_Q[0] \oplus T_{Q'}[0]$. We XOR a fixed tweak difference of $T_P[0] \oplus T_Q[0] = T_{P'}[0] \oplus T_{Q'}[0] =$ `0x01` to all ciphertext tweaks to ensure that this event holds with probability one. Here, we only have to care about the active S-box in Round 3. The probability that it maps ΔS^3_{SB} through the S-box to $T_Q \oplus T_{Q'}$ is about 2^{-7}. Then, the second pair (Q, Q') follows the remaining Rounds 2 and 1 to difference α with probability one. Note that we can consider any α where only the main diagonal is active. Summing up, the lower trail has probability $(2^{-6})^2 = 2^{-12}$, and the upper trail a probability of approximately $2^{-7} \cdot 2^{-7} = 2^{-14}$. Given $2^{n+2p-33}$ pairs per structure, we expect $2^{n+2p-33} \cdot 2^{-26} = 2^{n+2p-59}$ correct quartets on average. So, a single structure with $p = 2^{32}$ texts suffices to obtain 2^5 correct quartet on average.

Steps. Choose the difference $\delta' \in \{0,1\}8$ such that there exists a differential `0x01` $\to \delta'$ with probability 2^{-6} through the S-box. Derive the difference $\delta = \text{MC}((\delta', 0, 0, 0))$. The further steps are as follows:

0. Precompute a hash table $\mathcal{H}_{\text{precomp}}$, which performs the δ shift, i.e. derives the shifted ciphertext bytes $D[0, 7, 10, 13]$ for given $C[0, 7, 10, 13]$, $K^7[0, 7, 10, 13]$, and tweak $(T[0], 0, 0, T[7])$. For the creation of $\mathcal{H}_{\text{precomp}}$, take $S^7_{\text{AK}}[0, 7, 10, 13]$ together with its corresponding tweak $T_C[0]$ and partially decrypt the final round to obtain $S^6_{\text{AK}}[0, 1, 2, 3]$. Derive a tweak T_D with $T_D[1, \ldots, 7] = T_C[1, \ldots, 7]$ and $T_D[0] = T_C[0] \oplus$ `0x1`. Compute ${S'}^6_{\text{AK}}[0, 1, 2, 3] = S^6_{\text{AK}}[0, 1, 2, 3] \oplus \delta[0, 1, 2, 3]$ and determine $D[0, 7, 10, 13]$ by reencrypting ${S'}^6_{\text{AK}}[0, 1, 2, 3]$ over the final round using T_D.
1. Choose 2^n structures of 2^p plaintext-tweak tuples (P, T) each, following the procedure of our rectangle attack. Ask for their ciphertexts C.
2. Initialize a list for all possible values of $K^0[0, 5, 10, 15]$ and $K^7[0, 7, 10, 13]$.
3. For each key guess $K^7[0, 7, 10, 13]$:
    a) For each structure and ciphertext C with its corresponding tweak T_C, apply the δ-shift by computing $S^7_{\text{AK}}[0, 7, 10, 13]$ and look up the value $D[0, 7, 10, 13]$ it is mapped (together with $T[0]$) to by $\mathcal{H}_{\text{precomp}}$. Copy

the 12 bytes $D[1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15]$ from C. Derive the tweak T_D with $T_D[1, \ldots, 7] = T_C[1, \ldots, 7]$ and $T_D[0] = T_C[0] \oplus \texttt{0x1}$.

b) Ask for the plaintexts Q of all 2^{n+p} shifted ciphertexts (D, T_D).

c) Sort the 2^{n+p} obtained plaintexts Q (we define $T_Q = T_D$) together with their tweaks T_Q, their associated structure index S_Q, their corresponding ciphertext D, and the original plaintext P from which D was derived as tuples $(Q,\ T_Q,\ S_Q,\ D,\ P)$ into 2^{96} buckets according to the value of $Q[1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14]$. Since we search for pairs (Q, Q') with $Q \oplus Q' = \alpha$, $T_Q \neq T_{Q'}$, and $S_Q = S_{Q'}$, we expect in average $2^n \cdot \binom{2^p}{2} \cdot 2^{-96} \approx 2^{n+2p-97}$ false-positive colliding pairs (Q, Q') for each key candidate K^7. Since $\Pr[T_Q \neq T_{Q'}] = 255/256$, we still have about $2^{n+2p-97} \cdot 255/256 \approx 2^{n+2p-97}$ colliding pairs per key.

d) We could discard all pairs (Q, Q') whose structure indices S_Q and $S_{Q'}$ differ since these cannot have difference α. Since a single structure suffices, we skip this step and note it only for the sake of consistency.

e) For each potential quartet (P, P', Q, Q'), we want to identify the key values $K^0[0, 5, 10, 15]$ which could create the state collisions after the first round. So, for a correct quartet, their states after the first round match for both pairs, i.e., $S^1_P = S^1_{P'}$ and $S^1_Q = S^1_{Q'}$. Using the same arguments as in our rectangle attack, each byte in a pair gives four key-byte suggestions which hold for both pairs with probability 2^{-8}. Hence, an invalid quartet survives this filtering step with probability at most $(2^{-6})^4 = 2^{-24}$ for all four bytes of $K^0[0, 5, 10, 15]$. For each surviving pair, increment the counter for the current $K^0[0, 5, 10, 15]$ and $K^7[0, 7, 10, 13]$. Over all 2^{32} keys K^7, we expect $2^{32} \cdot 2^{n+2p-97} \cdot 2^{-24} = 2^{n+2p-89}$ quartets.

f) For $n = 0$ and $p = 32$, i.e. a single structure with 2^8 sets of 2^{24} texts each, we expect 2^5 correct quartets, and have a small probability of $2^{n+2p-89} = 2^{-25}$ for false positives. So, we output the key candidate(s) with the highest counters.

The attack recovers $K^0[0, 5, 10, 15]$ and $K^7[0, 7, 10, 13]$. Again, one can repeat the attack with shifted differentials (see Supporting Material C), which requires again 2^{40} texts, iterating over plaintext bytes $P[3, 4, 9, 14]$ and tweak byte $T[2]$, and can recover another 32 bits of K^0, $K^0[3, 4, 9, 14]$, and 32 bits of K^7, $K^7[1, 4, 11, 14]$ with the same probability as the attack above. The remaining 64 bits of K^0 and/or K^7 can be found e.g. by exhaustive search.

Complexity Analysis. The computational complexity results from:

– The **precomputations** require about $4 \cdot 2^8 \cdot 2^8 \cdot 2^8 = 2^{26}$ single-round decryptions and single-round re-encryptions (we have one ciphertext byte, one tweak byte, and one key byte as parameters) since we can create four independent tables, one for each byte of $D[0, 7, 10, 13]$.
– **Step 1** requires 2^{n+p} full encryptions.
– **Step 3a** requires $2 \cdot 2^{32} \cdot 2^{n+p}$ memory accesses (MA) for each ciphertext pair and key candidate to lookup the shifted pairs D.
– **Step 3b** requires $2^{32} \cdot 2^{n+p}$ full decryptions.

– **Step 3c** requires $2^{32} \cdot 2^{n+p}$ memory accesses with an efficient data structure.
– **Step 3e** requires $2^{32} \cdot 4 \cdot 2^{n+2p-97} \cdot 4/16 = 2^{n+2p-65}$ single-round decryptions of one column backwards through the first round for each text in each potential quartet (P, P', Q, Q') and each key. It requires a few memory accesses for false-positive and 2^5 memory accesses for correct quartets, which are negligible in the total effort.

For $n = 0$ and $p = 32$, the complexity of one execution of the attack is

$$\begin{aligned} T &\approx 2^{26}/7 + 2^{n+p} + 2^{n+p+32} + 2^{n+2p-65} \text{ Enc } + 2^{n+p+33} + 2^{n+p+32} \text{ MA} \\ &\approx 2^{64} \text{ Enc } + 2^{65.6} \text{ MA}. \end{aligned}$$

Since we propose to execute the attack once again in shifted form, and to subsequently recover 64 keys bits with exhaustive search, we obtain $T \approx 2^{65}$ Enc $+2^{66.6}$ MA. The data complexity is $2 \cdot 2^{32} = 2^{33}$ chosen plaintexts and $2 \cdot 2^{32} \cdot 2^{32} = 2^{65}$ adaptively chosen ciphertexts. The attack requires memory for 2^{26} bytes for $\mathcal{H}_{\text{precomp}}$, 2^{32} states for the ciphertexts, a few remaining pairs, and 2^{64} single- byte key counters. All remaining steps require less memory. The second execution can reuse existing memory. So, the attack can be implemented with memory for at most 2^{64} bytes or 2^{60} states.

C Shifted Differentials for the 7-Round Rectangle Attack

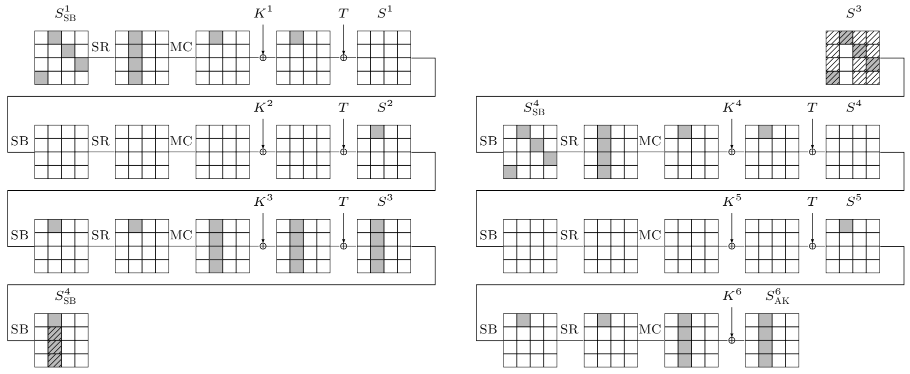


Fig. 15: Shifted upper (**left**) and lower (**right**) trails of our boomerang attacks on 7-round KIASU-BC. Hatched bytes are not traced through SUBBYTES in Round 4.

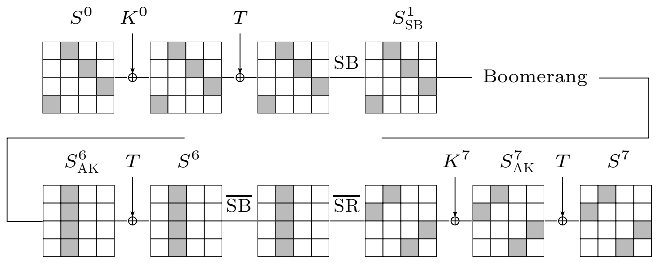


Fig. 16: 7-round boomerang attack trail.