Chosen-Key Distinguishing Attacks on Full AES-192, AES-256, Kiasu-BC, and More

Xiaoyang Dong[1,3], Shun Li[2], and Phuong Pham[2]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
xiaoyangdong@tsinghua.edu.cn
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
shun.li@ntu.edu.sg, pham0079@e.ntu.edu.sg
[3] National Financial Cryptography Research Center, Beijing, China

Abstract. At CRYPTO 2020, Liu et al. find that many differentials on `Gimli` are actually incompatible. On the related-key differential of `AES`, the incompatibilities also exist and are handled in different ad-hoc ways by adding respective constraints into the searching models. However, such an ad-hoc method is insufficient to rule out all the incompatibilities and may still output false positive related-key differentials. At CRYPTO 2022, a new approach combining a Constraint Programming (CP) tool and a triangulation algorithm to search for rebound attacks against `AES`-like hashing was proposed. In this paper, we combine and extend these techniques to create a uniform related-key differential search model, which can not only generate the related-key differentials on `AES` and similar ciphers but also immediately verify the existence of at least one key pair fulfilling the differentials. With the innovative automatic tool, we find new related-key differentials on full-round `AES-192`, `AES-256`, `Kiasu-BC`, and round-reduced `Deoxys-BC`. Based on these findings, full-round limited-birthday chosen-key distinguishing attacks on `AES-192`, `AES-256`, and `Kiasu-BC` are presented, as well as the first chosen-key distinguisher on reduced `Deoxys-BC`. Furthermore, a limited-birthday distinguisher on 9-round `Kiasu-BC` with practical complexities is found for the first time.

Keywords: Related-key · Chosen-key · Triangulation algorithm · Constraint Programming · Rebound techniques

1 Introduction

Block ciphers and hash functions play a crucial role as foundational primitives in the field of cryptography. Conventionally, a hash function can be constructed through repetitive iterations of compression functions utilizing the Merkle-Damgård [52,19] domain extender, where the compression function is typically built from a block cipher and PGV hashing modes [60], such as Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), and Miyaguchi-Preneel (MP). When attempting to breach the security of block ciphers, the attacker lacks the knowledge and control over the key material. However, when attacking hash function, particularly its block cipher-based compression function, the attackers holds the advantage of being in control of the key material, which is usually the message block to be hashed. The study of *known-key attacks* on block ciphers was first explored by Knudsen and Rijmen [42], who aimed to demonstrate non-ideal property of these primitives through distinguishers for 7-round AES and some Feistel structures, given that the key information was known to the attacker. This concept was further advanced by Mendel et al. [49] at SAC 2009, who improved the 7-round known-key distinguisher on AES with rebound attacks [50]. At FSE 2010, Gilbert and Peyrin [32] presented a known-key distinguisher on 8-round AES through the innovative application of the Super-Sbox technique in the context of a rebound attack. At ASICRYPT 2014, Gilbert [31] proposed the first known-key distinguisher for the full AES-128. At ACISP 2017, Cui et al. [17] distinguished the full AES-128 again with a statistical integral approach. Recently, Grassi and Rechberger [33] revisited Gilbert's known-key distinguisher and extended it to even 12 rounds on AES-128. The known-key distinguishers are also applied to PRESENT [45,11] and other important primitives [54,63,67,65,58,2] during the last decade. Furthermore, the known-key setting has also been explored from a perspective of provable security [2,15,51,16].

In addition to the known-key setting, at CRYPTO 2009, the *chosen-key setting* was first introduced by Biryukov, Khovratovich, and Nikolic [9], where the focus was on the distinguishing attacks performed with specially selected keys. They distingushed the full AES-256 by constructing q-multicollision in time $q \cdot 2^{67}$, which is much lower than the ideal case $q \cdot 2^{\frac{q-1}{q+1}128}$. At ASIACRYPT 2009 [44], Lamberger et al. built limited-birthday distinguisher on full Whirlpool through connecting multiple inbound phases with chosen key. At INDOCRYPT 2012, Derbez et al. [21] introduced practical chosen-key distinguishers for AES-128 up to 8 rounds and AES-256 with 9 rounds, which were later enhanced through the application of multiple limited-birthday distinguisher [37]. At ASICRYPT 2013, Iwamoto et al. [35] introduced the concept of a limited birthday distinguisher and built several distinguishing attacks on reduced AES in the chosen-key setting. At CRYPTO 2013, Fouque et al. [27] successfully distinguished 9-round AES-128 in the chosen-key setting by proposing a graph-based related-key differential searching algorithm. At ACISP 2019, Zhu et al. [75] proposed practical chosen-key distinguishers on 9-round AES-192.

In general, a block cipher with n-bit block size is a family of permutations that operates on two inputs: a secret key, randomly generated, and an n-bit string message, producing an output string of equal length that appears random. In some implementations, changing a key can be costly, which led to the proposal of *tweakable block ciphers* [68,47] as an alternative. These ciphers have the advantage of being less expensive to change the tweak, compared to changing the key. With a tweakable block cipher, both key and tweak are used to form a permutation. Nowadays, tweakable block ciphers show their flexibility with various applications in cryptographic schemes, such as message-authentication codes

[56], compression functions [26], (authenticated) encryption schemes [59,43], or variable-input-length ciphers [53]. At ASIACRYPT 2014, Jean et al. [39] proposed the `TWEAKEY` framework with the purposes of unifying the vision of key and tweak inputs of a cipher. Based on `TWEAKEY` framework, various tweakable block ciphers are constructed, such as `Deoxys-BC`, `Kiasu-BC`, and `Joltik-BC` and `SKINNY` [5], where `Deoxys-BC` and `SKINNY` have been standarized by ISO [1]. The investigations on these block ciphers are still going on, with a lot of new attacks and techniques have been deployed. The *open-key* settings (i.e., `known-key` and `chosen-key` attacks) are naturally available for tweakable block ciphers but rarely studied by the communities.

1.1 Related-Key Differentials on AES

In our paper, we mainly leverage related-key differentials on `AES` and related ciphers to build our chosen-key distinguishers. The so-called related-key attacks are first introduced by Biham [6] at EUROCRYPT 1993, which allows the attacker to insert differences in both the plaintext and the key. Although the related-key setting is somewhat less relevant for block cipher, it is important and practical when considering block-cipher based hash functions. There have been quite a few papers searching related-key differentials on `AES`. At EUROCRYPT 2010, Biryukov and Nikolic [10] introduced the branch-and-bound method to search the related-key differentials of `AES` and others. At CRYPTO 2013, Fouque et al. [27] proposed the graph-based method to search the related-key differentials and proposed the first 9-round chosen-key distinguisher on `AES-128`. Besides various ad-hoc methods [12,41] incorporating the key of `AES`, there are several tool-based automatic models on searching related-key differentials of `AES`. In 2014, Minier et al. [55] searched the truncated differentials of `AES` with constraint programming (CP). In 2016, Gérault et al. [30] proposed CP-based model to search the chosen-key differential attacks on `AES`. Due the efficiency, CP tools have widely used to search (related-key) differentials [28,69,29] recently.

At CRYPTO 2020, Liu et al. [48] discovered that many differentials on `Gimli` turn out to be incompatible due to inner incompatibilities. The incompatibility also appears in the related-key differentials of `AES`, which has been noted in many works [10,27]. To deal with the incompatibilities, most of the previous works use ad-hoc method [27] or add constraints [29] in the model to bypass the incompatible characteristics. However, it is hard to avoid all the incompatibilities when modelling the search. For example, the 10-round related-key differential on `AES-128` given in [27] turns out to be incompatible, and one can not find a key pair for the differential of the key. To assign compatible key values, Biryukov et al. [41] introduced the triangulating algorithm (TA) to derive collisions of Rijndael-based compression function. At CRYPTO 2022, Dong et al. [24] combined the triangulating algorithm and the rebound attacks to assign compatible key values to bridge multiple inbound phases, which is named as the *triangulating rebound attack*.

1.2 Our Contributions

Our research begin with the observation that the differential trail in AES-like cipher may be incompatible, or unlikely to transform into a limited-birthday distinguisher trivially, primarily due to the lack of degree of freedom in the states, and the unsuitable input/output differentials, the small probabilities of the key and state differentials, etc. To overcome these limitations, we utilize the key and tweak (if exist) to enhance the degree of freedom for generating conforming pairs. To facilitate the search for the limited-birthday distinguisher, we proposed a uniform automatic search model that integrates the key and tweak, as well as every parameter affecting the attack, like input/output differentials, probabilities, etc. In order to validate the attack, it is essential to find at least one key pair for the key differential. To achieve this, we improve a technique that enables immediate verification of the existence of key pairs, using the triangulation algorithm by Biryukov et al. [41] and Leurent-Pernot's new key schedule representation [46]. Therefore, our model can avoid the incompatibilities due to inner incompatibilities of the key differential. We discover a new property of key bridging for `AES` with Leurent and Pernot's new key schedule representation [46], which helps to significantly reduce the time to find the key pair given related-key differentials. A direct application is that we find a new and valid 10-round related-key differential on `Kiasu-BC` with concrete key pairs fulfilling the key differentials (Table 5). Based on our uniform search model, we derive new full-round distinguishing attacks on `AES-192`, `AES-256`, and `Kiasu-BC`, as well as round-reduced attacks on `Deoxys-BC`.

`AES-192` *and* `AES-256`*.* Since Biryukov et al. [9] proposed the full-round chosen-key distinguisher on `AES-256` at CRYPTO 2009, the full-round chosen-key distinguishers on `AES-128` and `AES-192` are opening for more than 10 years. At CRYPTO 2013, Fouque et al. [27] introduced a 9-round chosen-key distinguisher on `AES-128` and left a one-round gap to the full round. For `AES-192`, the longest chosen-key distinguisher only reaches 9 rounds [75]. In this paper, we fill the 10-year gap for `AES-192` by introducing the first full 12-round limited birthday chosen-key distinguishing attack. In addition, the full 14-round `AES-256` is distinguished again in chosen-key setting since Biryukov et al. [9].

`Kiasu-BC` strictly follows `AES`-128 by adding 64-bit tweak XORed to the first two rows of the state after the adding round key action. In the open-key setting, the designers stated that

> "*A possible increment in the number of attacked rounds might happen in the framework of open-key distinguishers (even though we have not been able to improve the known attacks using this extra tweak input).*"

In this paper, we fill the gap by considering the extra tweak and introduce chosen-key distinguishing attacks on (practical) 9-round and full 10-round `Kiasu-BC`. The 10-round attack is based on our new discovered 10-round related-key differential on `Kiasu-BC`. Although the 10-round differential can not lead to valid attacks on full AES-128, we succeed to distinguish full 10-round Kiasu-BC with aids of the additional degrees of freedom of its tweak.

Deoxys-BC has been selected as an ISO standard [1]. We for the first time consider Deoxys-BC against open-key attacks by proposing chosen-key distinguishers on 10-round Deoxys-BC-256 and 13-round Deoxys-BC-384. Our attacks can be another proof that Deoxys-BC is more secure than AES in related-key settings due to the new key schedule, which is claimed by the designers [40]. Note that the full 14-round AES-256 has been distinguished by Biryukov et al. [9] and our paper. Our source code is given at

https://www.dropbox.com/s/ghhqxmx3pmb0kae/experiment.zip?dl=0

Comparing our related-key attacks in chosen-key setting to previous known-key attacks. The known-key and chosen-key settings [42,9] are proposed when considering security issues on hash functions built from block ciphers, where the key material becomes public or changeable by the attackers. For a hash function, the three basic secure properties should be ensured, i.e., the resistance of collision attacks, preimage attacks, and second preimage attacks. For AES, the known-key attacks already reaches full round AES-128 [17,31,33] by expoiting statistical non-random properties, which can also be extended into known-key distinguishers on Kiasu-BC. However, those non-random properties on block ciphers can hardly threat the three basic secure properties of corresponding hash functions.

Comparing with previous related-key attacks In [72], the authors identified an 11-round related-key rectangle distinguisher on Deoxys-BC-384. Building upon it, they extended the distinguisher by adding one round at the beginning and two rounds at the end. This advancement allowed them to execute a key-recovery attack on the 14-round version of Deoxys-BC-384. As far as our knowledge extends, the conversion of the 11-round related-key distinguisher into a chosen-key distinguisher seems straightforward. However, the same cannot be said for the 14-round key-recovery attack on Deoxsys-BC-384, as it does not readily convert into a chosen-key distinguisher. Similar instances can be observed in the full key-recovery attack on AES-192 [8] and the 11/14-round key-recovery attack on Deoxys-BC-256 [72].

In this paper, we introduce a few chosen-key distinguishers based on related-key (truncated) differentials on AES-like block ciphers. Since we exploit the related-key (truncated) differentials, our chosen-key distinguishing attacks are more likely to be converted into real attacks on the hash functions, i.e., collision attacks. The only differences between the related-key (truncated) differentials used in our chosen-key distinguishers and the collision attacks is whether the input and output truncated differential of the underlined block cipher are equal or not. In fact, our tool also finds a practical collision attack on 6-round AES-128, but previous known-key distinguishers can not build such real threat. In this point of view, it is still meaningful to find chosen-key attacks based on related-key (truncated) differentials, although there are already known-key attacks. Therefore, we bring in the first full chosen-key attack on Kiasu-BC based on a novel 10-round related-key (truncated) differentials, although known-key distinguishers on full Kiasu-BC are trivially derived with previous known-key distinguishers on full round AES-128 [17,31,33].

Table 1: A summary of the distinguishing attacks on AES and others, [8] and [72] contain related-key attacks converted from the corresponding distinguishers.

Target	Settings	Rounds	Time	Memory	Ideal	Ref.
AES-128	MMO Collision	6/10	2^{56}	2^{32}	2^{64}	[32]
		6/10	2^{48}	2^{32}	-	Sect. 4
	Known-Key	7/10	2^{56}	-	2^{58}	[42]
		7/10	2^{24}	2^{16}	2^{64}	[49]
		8/10	2^{48}	2^{32}	2^{64}	[32]
		10/10	2^{64}	2^{64}	-	[31]
		10/10	$2^{59.6}$	$2^{58.8}$	-	[17]
		10/10	2^{50}	2^{32}	$2^{65.6}$	[33]
		12/10	2^{66}	2^{64}	-	[33]
	Chosen-Key	7/10	2^{8}	2^{8}	2^{64}	[21]
		8/10	$2^{13.4}$	2^{16}	$2^{31.7}$	[37]
		9/10	2^{55}	2^{32}	2^{68}	[27]
AES-192	Chosen-Key	8/12	1	2^{16}	-	[75]
		9/12	-	-	-	[8]*
		9/12	1	2^{16}	-	[75]
		12/12	2^{100}	2^{32}	2^{108}	Sect. 4
AES-256	Chosen-Key	7/14	2^{8}	2^{8}	2^{64}	[21]
		8/14	2^{8}	2^{8}	2^{64}	[21]
		9/14	2^{24}	2^{16}	2^{64}	[21]
		9/14	2^{49}	2^{33}	2^{128}	[7]
		14/14	2^{119}	-	2^{128}	[9]
		14/14	$q \cdot 2^{67}$	-	$q \cdot 2^{\frac{q-1}{q+1}128}$	[9]
		14/14	2^{88}	2^{32}	2^{94}	Sect. 4
Kiasu-BC	Secret-Key	3.5/10	-	-	-	[23]
		4/10	2^{32}	-	-	[22]
		6/10	-	-	-	[23]
	Chosen-Key	9/10	2^{36}	2^{14}	2^{68}	Sect. 5
		10/10	2^{67}	2^{14}	2^{96}	Sect. 5
Deoxys-BC-256	Secret-Key	9/14	2^{122}	-	2^{128}	[14]
		9/14	$2^{120.4}$	-	2^{128}	[71,72]*
	Chosen-Key	10/14	2^{69}	-	2^{101}	Sect. 6
Deoxys-BC-384	Secret-Key	11/16	2^{120}	-	2^{128}	[14]
		11/16	$2^{118.4}$	-	2^{128}	[73]
	Chosen-Key	13/16	2^{42}	-	2^{58}	Sect. 6

2 Preliminaries

2.1 AES

AES [18] is a 4×4 cell block ciphers that operates on 128-bit state, with three different key sizes of 128, 192, and 256 bits, creating three corresponding versions: AES-128, AES-192, and AES-256. Although the round functions of all versions are the same, the number of execution rounds differ and are 10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256. Each round consists of four major transformations as illustrated in Figure 1:

- SubBytes (SB): applies a non-linear 8-bit substitution-box operation to each cell.
- ShiftRows (SR): shifts the i-th row left by i bytes cyclically.
- MixColumns (MC): mixes every column by multiplying a diffusion matrix over $GF(2^8)$.
- AddRoundKey (AK): adds a 128-bit round key to the internal state.

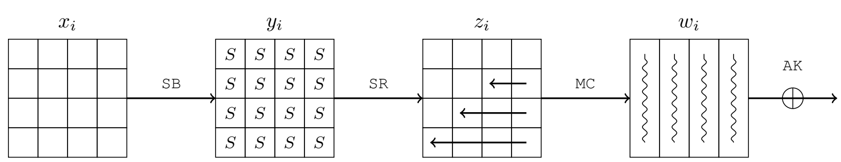


Fig. 1: The round function of AES.

The 128-bit round keys for each round function are generated from the initial key by the KeySchedule (KS) operation. Note that an extra AddRoundKey is add at the beginning of the first round, and a MixColumns operation is removed at the last round.

2.2 Kiasu-BC and Deoxys-BC

The concept of tweakable block ciphers (TBC) was introduced by Liskov et. al. [47] in 2002. At ASIACRYPT 2014, Jean et. al [39] introduced the TWEAKEY framework with the aim of unifying the design of TBCs and enabling the creation of primitives with arbitrary key and tweak sizes. The TWEAKEY framework treats the key and tweak inputs similarly through a tweakey schedule algorithm. To simplify security analysis when the tweakey size is large, Jean *et al.* identified a subclass of TWEAKEY named the STK construction (Figure 2). In this paper, we focus on two important TBCs, i.e., Kiasu-BC [38] and Deoxys-BC [40].

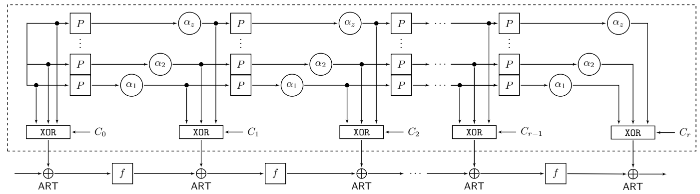


Fig. 2: The STK construction [36]

Kiasu-BC [38] was proposed by Jean et al. in 2014. Its design strictly follows AES-128, with an additional 64-bit tweak XORed to the first two rows of the state after the add round key action. As a result, when the tweak is equal to zero, Kiasu-BC is identical to AES-128. Consequently, the security of Kiasu-BC can be inferred from existing and new analyses of AES-128. However, the addition of the tweak might increase the degrees of freedom for attacks, since the tweak is now freely chosen by attackers. This is why a comprehensive investigation into Kiasu-BC is important to determine any potential negative effects. Previous cryptanalysis of Kiasu-BC includes square attack [22], impossible differential attack [23], meet-in-the-middle attack [70], and boomerang attack [23,4]. In the open-key setting, Bao et al. [3] introduced a MITM preimage attack on 8-round Kiasu-BC in hashing modes. Although the designers claimed that the extra tweak input could improve open-key distinguishers, this was not achieved [38]. In this paper, we fill this gap in the cryptanalysis in open-key setting.

Deoxys-BC [40] has been standardized by ISO [1]. It follows the STK construction in Figure 2, with the use of P to permute each tweakey byte and the replacement of α_i with LFSRs to update each byte. The internal round function f is simply the round function of AES. Deoxys-BC comes in two versions, i.e., Deoxys-BC-256 (14 rounds) and Deoxys-BC-384 (16 rounds). Deoxys-BC has received a lot of attention since its first thirt-party analysis by Cid et al. at ToSC 2017 [14]. By using Mixed Integer Linear Programming (MILP) to automatically search for related-key boomerang differential trails with the aids of incorporating linear incompatibility, they obtained the 8-round and 9-round related-tweakey boomerang distinguishers for Deoxys-BC-256 with a probability 2^{-72} and 2^{-122}, respectively, and 10-round and 11-round related-tweakey boomerang distinguishers for Deoxys-BC-384 with a probability 2^{-84} and 2^{-120}, respectively. Later, based on the Boomerang Connectivity Table (BCT) technique [13,71], these distinguishers were improved. There have been various key-recovery attacks on Deoxys-BC, including boomerang attacks [14,64] and rectangle attacks [72,25,4].

2.3 Limited birthday Distinguisher

At ASICRYPT 2013, the limited birthday distinguisher (LBD) was formally introduced by Iwamoto et al. [35], although it has been used to distinguish Whirlpool [44], AES [32]. At SAC 2013, Jean et al. further developed the LBD into a multiple limited birthday distinguisher [37]. In this section,the definition and solution of the limited birthday problem and distinguisher are discussed.

Definition 1. *(Limited birthday problem). Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a permutation, and let $\Delta_{IN}, \Delta_{OUT} \in \{0,1\}^n$ are two vector spaces of dimensions d_{in} and d_{out} respectively. Find a pair of inputs (x, x') to f such that $x \oplus x' \in \Delta_{IN}$ and $f(x) \oplus f(x') \in \Delta_{OUT}$.*

Note that both f and f^{-1} are accessible, and the complexity of finding a solution will depend on the values of d_{in} and d_{out}. The algorithm proposed in [32] has been proven to match the lower bound complexity for the limited birthday problem in [35] for a black-box function and in [34] for a black-box permutation. We refer to [32] for optimal algorithm and Theorem 1 for its time complexity.

Theorem 1. *There exists an algorithm for finding a limited birthday pair by querying to f and f^{-1} in time:*

$$\max\{2^{\frac{n+1-\max\{d_{in},d_{out}\}}{2}}, 2^{n+1-(d_{in}+d_{out})}\},$$

and using

$$\min\{2^{d_{in}}, 2^{d_{out}}\}$$

classical random access memory.

Definition 2. *(Limited birthday distinguisher in related-key model). Let $E_K : \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher with key $K \in \{0,1\}^m$. Given two truncated difference vector spaces Δ_{IN} of dimension d_{in}, Δ_{OUT} of dimension d_{out} and a differential characteristic for the related-key $K \oplus \Delta_K$, a limited birthday distinguisher against E_K is generated to find a pair of input (x, x') such that $x \oplus x' \in \Delta_{IN}$ and $E_K(x) \oplus E_{K\oplus\Delta_K}(x') \in \Delta_{OUT}$ with an algorithm solved faster than the complexity of the generic algorithm in Theorem 1.*

2.4 The Rebound Attacks

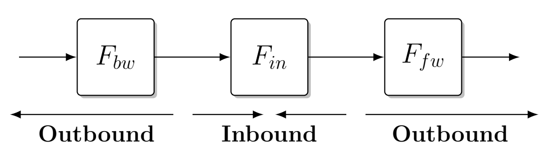


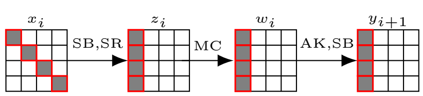
Fig. 3: The Rebound Attack

The rebound attack was first introduced by Mendel et al. in [50]. The attack consists of two phases: an inbound phase and an outbound phase, as illustrated in Figure 3. The internal block cipher or permutation F is split into three subparts: $F = F_{fw} \circ F_{in} \circ F_{bw}$.

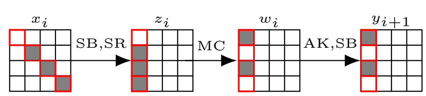
- **Inbound phase**. In the inbound phase, the attackers use the meet-in-the-middle technique to efficiently fulfill the low probability part in the middle of the differential trail. The number of matched pairs in the inbound phase determines the degree of freedom and acts as the starting point for the outbound phase.
- **Outbound phase**. In the outbound phase, the matched values from the inbound phase are computed backward and forward through F_{bw} and F_{fw} to find a pair of values that satisfies the outbound differential trail through a brute-force approach.

Generally speaking, the rebound attack is a technique for efficiently generating a message pair that fulfills the inbound phase by utilizing a truncated differential instead of a single differential characteristic. If the probability of the outbound phase is p, then $1/p$ starting points must be prepared in the inbound phase to expect one pair conforming to the differential trail of the outbound phase. Thus, the degree of freedom should be larger than $1/p$.

The Super-Sbox Technique The Super-Sbox technique was proposed simultaneously by Gilbert and Peyrin [32] and Lamberger et al. [44] with the goal of fulfilling two heavy round S-box layers, as shown in Figure 4 (a), but with low complexity. In [66], Sasaki et al. further reduced the memory complexity by considering non-full-active Super-Sboxes, as shown in Figure 4 (b). As these techniques are applied to hash functions, the master key is fixed and does not provide any degree of freedom. The Super-Sbox technique is then used in the rebound attack to reduce the cost of the two most heavy rounds (inbound part) and to generate sufficient pairs to conform to the probability of the remaining rounds (outbound part).



(a) Super-Sbox



(b) Non-full-active Super-Sbox

Fig. 4: The Two-Round Differential

Super-Sbox Technique. For the jth Super-Sbox $\mathtt{SSB}_j$ and given input difference $\Delta x_i^{(j)}$ ($j = 0$ in Figure 4 (a)), compute all possible $\Delta y_{i+1}^{(j)} = \mathtt{SSB}_j(x \oplus \Delta x_i^{(j)}) \oplus \mathtt{SSB}_j(x)$ for all $x \in \{0,1\}^{32}$. Store the pair $(x, x \oplus \Delta x_i^{(j)})$ in a table $\mathbb{L}^{(j)}[\Delta y_{i+1}^{(j)}]$. With the given $\Delta y_{i+1}^{(j)}$, we find a pair conforming the two-round differential with $(\Delta x_i^{(j)}, \Delta y_{i+1}^{(j)})$ by assessing $\mathbb{L}^{(j)}[\Delta y_{i+1}^{(j)}]$. The memory cost is about 2^{32}.

Non-full-active Super-Sbox. The Property 1 of MDS in $\mathtt{MC}$ is utilised to connect the differences. Take Figure 4 (b) as an example, by guessing the differences of one active byte of Δz_i or Δw_i, we can determine other differences according to Property 1 as $\Delta w_i = \mathtt{MC}(\Delta z_i)$. Then, for a fixed input-output differences $(\Delta x_i^{(j)}, \Delta y_{i+1}^{(j)})$ of $\mathtt{SSB}_j$, we deduce all the input-output differences for the active cells of two S-box layers for each guess and then obtain their cell values by accessing the differential distribution table (DDT) of the S-box. 1 out of 5 active value bytes in z_i and w_i is act as a filter of 2^{-8}, and about 2^5 pairs are found.

Property 1. Suppose MC is an $n \times n$ MDS matrix and $\mathtt{MC} \cdot (z[1], \ldots, z[n])^T = (w[1], \ldots, w[n])^T$, the knowledge of any n out of $2n$ bytes of (z, w) is necessary and sufficient to determine the rest. (z, w) here can be either value or difference.

2.5 Triangulation Algorithm

The Triangulation Algorithm tool is an efficient Gaussian-based algorithm that was introduced in 2009 by Khovratovich et. al. [41]. It is used to solve systems of bijective equations and automatically detect the solution to nonlinear systems. The algorithm models an AES-like block cipher as a system of round function and key schedule equations, where the state bytes and key bytes are considered variables. Initially, all the values of the system are determined by n initial state bytes and k initial key bytes. By fixing m bytes with some constraints, the algorithm returns $n+k-m$ "free variables", which form the basis of the system. The steps of the triangulation algorithm are as follows:

1. Generate a system of equations, where each cell is a variable and the predefined values are fixed as constants.
2. Mark all variables and equations as non-processed.
3. Mark all variables involved in only one non-processed equation as processed, and also mark the equation as processed. If there are no such variables, exit.
4. Repeat step 3 until there are no more non-processed equations.
5. Return all non-processed variables as "free variables".

After the "free variables" are returned, random values can be assigned to them and the other cells can be deduced based on the relationships in the equations.

3 Automatic Tool for finding Differential Distinguishers

In this section, constraint programming is introduced as the main automatic tool for finding truncated differential trails on all targets and differential characteristics in most cases.

Generic solver Constraint Programming (CP) is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X}$ is a finite set of variables; $\mathcal{D}$ refers to the domain, *i.e.*, the set of values each $x_i \in \mathcal{X}$ can have; $\mathcal{C}$ is a set of constraints including relations between variables. Once an objective function is defined, the CSP becomes a Constrained-Optimization Problem (COP). A solution of a COP is an assignment of values to all the variables in $\mathcal{X} = \{x_0, \cdots, x_{n-1}\}$ such that all constraints from $\mathcal{C} =$

$\{c_0, \cdots, c_{m-1}\}$ are satisfied and the objective function achieves its maximum or minimum.

$$
\begin{cases}
\text{/* This model specifically pertains to AES-128, with the variables being \{0,1\} */} \\
\text{1. Objective function: /* to minimize the total number of active Sboxes, } i,j \in [0,\ldots,n-1] \text{ */} \\
\quad obj = \sum \Delta x_r[i,j] + \sum \Delta k_r[i,n-1] \\
\text{2. Constraints on SB in both states and subkeys: /* } SK_r[i] \text{ indicates whether} \\
\text{ the key byte of the last column is active or not in the } r\text{-th round. */} \\
\quad \Delta x_r[i,j] = \Delta y_r[i,j] \\
\quad \Delta SK_r[i] = \Delta k_r[i,n-1] \\
\text{3. Constraints on AK:} \\
\quad \Delta w_r[i,j] + \Delta k_{r+1}[i,j] + \Delta x_{r+1}[i,j] \neq 1 \\
\text{4. Constraints on SR:} \\
\quad \Delta y_r[i,(i+j) \bmod n] = \Delta z_r[i,j] \\
\text{5. Constraints on MC: /* the total number of active bytes before and after} \\
\text{ the MC should fulfill the MDS property */} \\
\quad (\sum_{i=0}^{n-1} \Delta z_r[i,j] + \sum_{i=0}^{n-1} \Delta w_r[i,j]) \in \{0, n+1, n+2, \ldots, 2n\} \\
\text{6. Constraints on KeySchedule:} \\
\quad \Delta k_{r+1}[i,0] + \Delta k_r[i,0] + \Delta SK_r[(i+1) \bmod 4] \neq 1 \\
\quad \Delta k_{r+1}[i,j] + \Delta k_{r+1}[i,j-1] + \Delta k_r[i,j] \neq 1
\end{cases}
\tag{1}
$$

Finding an optimal related-key differential trail is a highly combinatorial problem that hardly scales. To simplify this problem, a usual and efficient way is to divide it into two steps [10,27]. Step 1 searches for all truncated differential trails under a given bound on the number of rounds and active S-Boxes. It may happen that no actual differential characteristic follows the truncated differential trail found in Step 1. Hence, Step 2 examines and decides whether the truncated differential characteristics are valid, and finds the actual differential characteristic that maximizes the probability. Both steps can be approached through CP. Such a CP strategy has been successful in finding related-key differential characteristics for `AES` [30], `Midori` [28], and `SKINNY` [20], in the sense that the truncated differentials match the lower bound on the number of active S-boxes (a.k.a. optimal truncated differentials).

In 2014, Minier *et al.* [55] proposed to use tools to automate Step 1 for finding the best truncated differentials. In 2020, Gérault *et al.* [29] added more constraints to describe the `KeySchedule` and `MixColumn` more precisely to filter out those truncated differentials without valid differential characteristics following them, which in turn also reduced the search space and improved efficiency of the tool. Within the space of valid truncated differentials, Step 2 follows the same CP program used in Step 1, but considers the exact byte differences for the entire differential characteristics instead of a binary value of 0/1 in truncated differentials. Following these significant developments in [55,29], the search model of related-key differentials on `AES`-like primitives can be described as 1. The

Constraints 1 consists of the 5 steps of the round function in a AES-like cipher of a $n \times n$ bytes state, *i.e.*, SubBytes (SB), AddroundKey (AK), ShiftRow (SR), MixColumn (MC), and KeySchedule, and the objective function to minimize the number of active S-boxes and to maximize the overall differential probability. Hereinafter, x, y, z, w represent the states after AK, SB, SR, and MC respectively, $x_r[i, j]$ (or $x_r[l]$ with $l = i + n \cdot j$) for the byte at row i and column j of round r for $i, j, r = 0, 1, \cdots$, and $\Delta x^r_{i,j} = 1$ if there is a non-zero difference in byte $x^r_{i,j}$ and 0 otherwise. These constraints together describe the underlying cipher in the language of CP, which ensures the differential characteristics found by the program will follow exactly the cipher, as presented in Equation 1.

Technically, the truncated differential search of Step 1 is implemented by MiniZinc [57] language, which is subsequently solved by Picat-SAT [74]. Then the search of actual differential characteristics in Step 2 is defined and solved by Choco solver [61].

3.1 Differences of the Constraints and Extended Techniques

The automatic tools from [29] have been modified. The constraints of the Step 1 searching algorithm are slightly different for AES and Kiasu-BC since we included the following 3 constraints in Equation 2. The value of Len_{Key} is 192, 256, and 192, respectively for AES-192, AES-256, and Kiasu-BC. The constraints of step 2 searching code are changed adaptively with Step 1.

For Deoxys-BC, constraints on KeySchedule should be changed accordingly. The positions where the XOR addition of two or three tweakeys makes a difference in some rounds are designated with *lanes* in Equation 3, where function f is deduced from the tweakey byte permutation h and the *regime* is 2 for Deoxys-BC-256 and 3 for Deoxys-BC-384.

$$
\begin{cases}
1^\star.\ \text{Objective function:} \\
\qquad obj = \sum_{r=4}^{Nr-2} \Delta x_r[i,j] + \text{fixed bytes in round 2} = \text{Outbound Active Bytes} \\
7.\ \text{Constraints on } \mathsf{KeySchedule}: \\
\qquad \sum \Delta k_r[i, n-1] \le Len_{Key} \\
8.\ \text{Constraints of ideal case complexity greater than attacking complexity gene-} \\
\quad \text{rated from limited birthday attack [32]:} \\
\qquad \max(64 - \max(8 \cdot \sum \Delta x_0[i,j], 7 \cdot \sum \Delta x_{Nr-1}[i,j])/2, \\
\qquad\qquad\qquad 129 - 8 \cdot \sum \Delta x_0[i,j] - 7 \cdot \sum \Delta x_{Nr-1}[i,j]) \\
\qquad\qquad\qquad\qquad \ge \sum_{r=4}^{Nr-1} \Delta x_r[i,j] + 7 \cdot \text{fixed bytes in round 2} \\
9.\ \text{Inbound freedom is enough:} \\
\qquad Len_{Key} - \sum \Delta k_r[i, n-1] \\
\qquad\qquad + \text{Starting Points freedom from } \Delta z_1 \\
\qquad\qquad\qquad + \text{Ending Points freedom from } \Delta x_4 \\
\qquad\qquad\qquad\qquad \ge \sum_{r=4}^{Nr-1} \Delta x_r[i,j] + 7 \cdot \text{fixed bytes in round 2}
\end{cases}
\tag{2}
$$

$$\begin{cases} 6^\star.\ \text{Constraints on } \mathsf{TweakeySchedule}: \\ \quad \Delta TK_r[f(r,m) \bmod 4, f(r,m) \text{ div } 4] \le lane_m \\ \quad \sum_{r=0}^{Nr-1} \Delta TK_r[f(r,m) \bmod 4, f(r,m) \text{ div } 4] \ge lane_m \cdot (Nr+1-regime) \end{cases} \tag{3}$$

We conducted a thorough search using the aforementioned constraint model above. Our approach is different from the methodology provided by [27] for 9-round AES-128 LBD distinguisher in which they looked for a 5-round related-key differential characteristic with a higher probability and built 4 rounds backwards. Another strategy used in the search is utilizing more freedom expanded from end points *i.e.*, Δx_4, which is so efficient that just one key pair is required when the same 9-round LBD distinguisher as [27] is used.

3.2 From Differential Characteristics to Conformable Value Pairs with the New AES Key Schedule Representation

The presence of a truncated differential does not necessarily indicate the existence of differential characteristics, and similarly, the presence of differential characteristics does not guarantee the presence of a conforming pair. The first issue can be efficiently validated using CP/MILP automated tools, but the second issue remains an unresolved problem for AES. Recently, techniques based on MILP method are proposed to the experimental verify the difference trails and successfully apply to Speck, Simek and Gimli [48,62]. In this work, we propose an efficient way to generate a pair that conforms to the low-probability differential characteristics of AES-like ciphers using the triangulation algorithm. By following the ideas from [41,24], we can hasten the process of finding a key pair conforming to the key differential trails, regardless of the low probability of the trial. For instance, one key pair of AES-128 key trail in Figure 10 and given in Table 5 can be found in roughly 2^{30} time complexity, despite the original probability being 2^{-126}. This is achieved by fixing the values of 16 active S-boxes in k_6, k_7, k_8, and k_9. By following the steps in Table 2, about 2^{32} subkeys k_9 can be recovered from all possible choices of admissible active S-box values in k_6, k_7, k_8, and k_9, an thus, at least one subkey can pass through the remaining five active S-boxes at k_5, k_2, and k_1.

1. $k_7[8,9,10,11] = k_6[12,13,14,15] \oplus k_7[12,13,14,15]$
2. $k_8[8,9,10,11] = k_7[12,13,14,15] \oplus k_8[12,13,14,15]$
3. $k_8[4,5,6,7] = k_7[8,9,10,11] \oplus k_8[8,9,10,11]$
4. $k_9[8,9,10,11] = k_8[12,13,14,15] \oplus k_9[12,13,14,15]$
5. $k_9[4,5,6,7] = k_8[8,9,10,11] \oplus k_9[8,9,10,11]$
6. $k_9[0,1,2,3] = k_8[4,5,6,7] \oplus k_9[4,5,6,7]$

Table 2: Steps to recover the subkey k_9 from known key bytes

Furthermore, by utilizing the new key schedule representation proposed by Leurent and Pernot in [46], we can further reduce the complexity of finding key pairs to roughly 2^8 by carefully choosing the admissible values for active S-boxes bytes in the key states. In this new representation, the bytes of round key k are transformed into a new basis $s[0], s[1], ..., s[15]$ through these transitions:

$$\begin{aligned}
&s[0] = k[15], && s[1] = k[14] \oplus k[10] \oplus k[6] \oplus k[2], && s[2] = k[13] \oplus k[5], && s[3] = k[12] \oplus k[8],\\
&s[4] = k[14], && s[5] = k[13] \oplus k[9] \oplus k[5] \oplus k[1], && s[6] = k[12] \oplus k[4], && s[7] = k[15] \oplus k[11],\\
&s[8] = k[13], && s[9] = k[12] \oplus k[8] \oplus k[4] \oplus k[0], && s[10] = k[15] \oplus k[7], && s[11] = k[14] \oplus k[10],\\
&s[12] = k[12], && s[13] = k[15] \oplus k[11] \oplus k[7] \oplus k[3], && s[14] = k[14] \oplus k[6], && s[15] = k[13] \oplus k[9].
\end{aligned}$$

Denote k' and s' are the state after one round of key schedule and new representation of key schedule (s'' is the state after two rounds, etc.), then:

$$\begin{aligned}
s'[0] &= k'[15] = k[15] \oplus k[11] \oplus k[7] \oplus k[3] \oplus \mathtt{SB}(k[12]) && = s[13] \oplus \mathtt{SB}(s[12]),\\
s'[1] &= k'[14] \oplus k'[10] \oplus k'[6] \oplus k'[2] = k[14] \oplus k[6] && = s[14],\\
s'[2] &= k'[13] \oplus k'[5] = k[13] \oplus k[9] && = s[15],\\
s'[3] &= k'[12] \oplus k'[8] = k[12] && = s[12],\\
s'[4] &= k'[14] = k[14] \oplus k[10] \oplus k[6] \oplus k[2] \oplus \mathtt{SB}(k[15]) && = s[1] \oplus \mathtt{SB}(s[0]),\\
s'[5] &= k'[13] \oplus k'[9] \oplus k'[5] \oplus k'[1] = k[13] \oplus k[5] && = s[2],\\
s'[6] &= k'[12] \oplus k'[4] = k[12] \oplus k[8] && = s[3],\\
s'[7] &= k'[15] \oplus k'[11] = k[15] && = s[0],\\
s'[8] &= k'[13] \oplus k[13] \oplus k[9] \oplus k[5] \oplus k[1] \oplus \mathtt{SB}(k[14]) && = s[5] \oplus \mathtt{SB}(s[4]),\\
s'[9] &= k'[12] \oplus k'[8] \oplus k'[4] \oplus k'[0] = k[12] \oplus k[4] && = s[6],\\
s'[10] &= k'[15] \oplus k'[7] = k[15] \oplus k[11] && = s[7],\\
s'[11] &= k'[14] \oplus k'[10] = k[14] && = s[4],\\
s'[12] &= k'[12] = k[12] \oplus k[8] \oplus k[4] \oplus k[0] \oplus \mathtt{SB}(k[13]) \oplus c_i && = s[9] \oplus \mathtt{SB}(s[8]) \oplus c_i,\\
s'[13] &= k'[15] \oplus k'[11] \oplus k'_7 \oplus k'[3] = k[15] \oplus k[7] && = s[10],\\
s'[14] &= k'[14] \oplus k'[6] = k[14] \oplus k[10] && = s[11],\\
s'[15] &= k'[13] \oplus k'_9 = k[13] && = s[8].
\end{aligned}$$

Following the new key-schedule representation, the key bytes $k[12], k[13], k[14]$, and $k[15]$, where the **SubBytes** operator acts on, will not be affected after the transformation since they are converted to $s[12], s[8], s[4]$ and $s[0]$, respectively. Furthermore, the following relationships exist between these bytes:

$$\begin{aligned}
s[8] &= s'[15] = s''[2] = s'''[5] = \mathtt{SB}(s'''[4]) \oplus s''''[8],\\
s[0] &= s'[7] = s''[10] = s'''[13] = \mathtt{SB}(s'''[12]) \oplus s''''[0],\\
s[4] &= s'[11] = s''[14] = s'''[1] = \mathtt{SB}(s'''[0]) \oplus s''''[4],\\
s[12] &= s'[3] = s''[6] = s'''[9] = \mathtt{SB}(s'''[8]) \oplus s''''[12] \oplus c_i.
\end{aligned}$$

These relations result in the following observation on AES-128 key schedule.

Proportion 1 (Key bridging). *By the new representation of key schedule of AES-128, the knowledge of the fourth columns of the subkeys at round 4 k_4 and round 5 k_5 allows to deduce the fourth columns of the subkey at round 1 k_1.*

Returning to the preceding example, the admissible active S-box values of k_6, k_7, k_8, and k_9 are chosen sequentially so that four active S-boxes at k_2 and k_1 are fulfilled in roughly 2^8 time complexity by the constraints:

$$\begin{aligned}
k_5[13] &= s_5[8] = \mathtt{SB}(s_8[4]) \oplus s_9[8] = \mathtt{SB}(k_8[14]) \oplus k_9[13],\\
k_2[13] &= s_2[8] = \mathtt{SB}(s_5[4]) \oplus s_6[8] = \mathtt{SB}(\mathtt{SB}(s_8[0]) \oplus s_9[4]) \oplus k_6[13]\\
&= \mathtt{SB}(\mathtt{SB}(k_8[15]) \oplus k_9[14]) \oplus k_6[13],\\
k_2[14] &= s_2[4] = \mathtt{SB}(s_5[0]) \oplus s_6[4] = \mathtt{SB}(\mathtt{SB}(s_8[12] \oplus s_9[0]) \oplus k_6[14]\\
&= \mathtt{SB}(\mathtt{SB}(k_8[12]) \oplus k_9[15]) \oplus k_6[14],\\
k_2[15] &= s_2[0] = \mathtt{SB}(s_5[12]) \oplus s_6[0] = \mathtt{SB}(\mathtt{SB}(s_8[8]) \oplus s_9[12] \oplus c_i) \oplus k_6[15]\\
&= \mathtt{SB}(\mathtt{SB}(k_8[13]) \oplus k_9[12] \oplus c_i) \oplus k_6[15],\\
k_1[13] &= s_1[8] = \mathtt{SB}(s_4[4]) \oplus s_5[8] = \mathtt{SB}(\mathtt{SB}(s_7[0]) \oplus s_8[4]) \oplus k_5[13]\\
&= \mathtt{SB}(\mathtt{SB}(k_7[15]) \oplus k_8[14]) \oplus k_5[13].
\end{aligned}$$

The new key bridging observation helps reduce the complexity of finding key pairs by tracking the validity of the relationship between distant active S-boxes. With these properties, the complexity of finding differential characteristics can even be reduced by filtering out trails that do not meet these above constraints.

4 Improved attacks on AES

This section presents several attacks on AES and AES-like hash functions. To begin, we introduce a new collision attack on 6-round AES-128-MMO/MP with a time complexity of only 2^{48}, achieved through the use of a new truncated differential trail. Compared to the best concurrent collision attack on 6-round AES-128, as presented in [32], our new attack has reduced the complexity by 2^8 times. Additionally, two new distinguisher attacks on full-round AES-192 and AES-256 have been obtained through the improvement of an automatic tool search specialized in limited birthday distinguisher.

4.1 Collision attacks on 6-round AES-128-MMO/MP

As depicted in Figure 5, focusing on the first Super-Sbox $\mathtt{SSB}^{(0)}$ marked by red box, for the fixed chosen difference Δw_3 and a given k_3 computed by the initial value for the input key, we derive $\mathtt{SSB}^{(0)}$ in Δy_3 and apply the Super-Sbox technique following these steps.

1. Brute-force all the 2^{32} values of $y_3^{(0)}$, compute and store $(x_3^{(0)}, \Delta x_3^{(0)})$ in the list L_0.

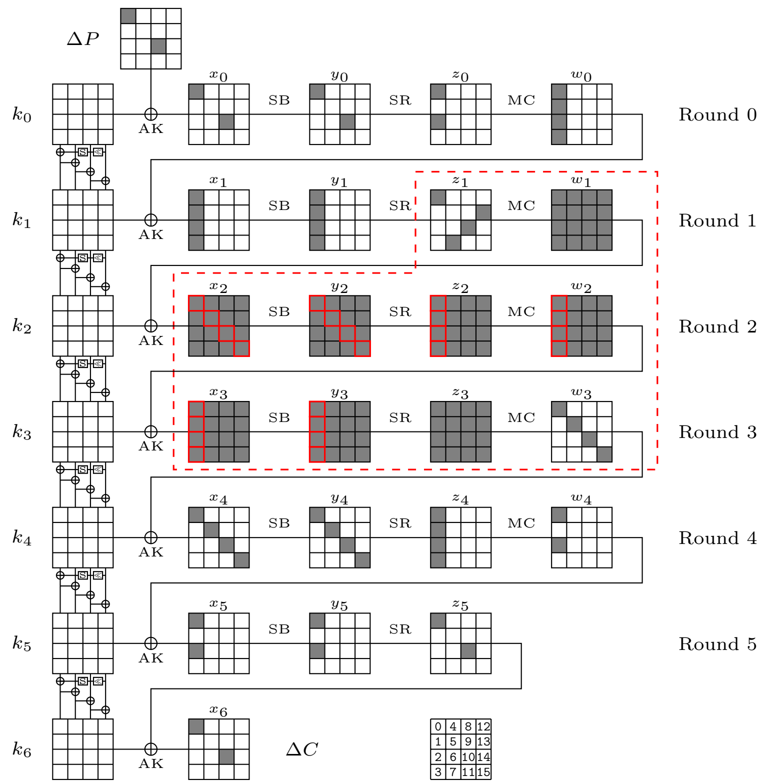


Fig. 5: Collision attack on 6-round AES-128.

2. Store the value of $(x_2^{(0)}, \Delta x_2^{(0)})$ with the corresponding entry $(x_3^{(0)}, \Delta x_3^{(0)})$ after passing through AK, MC, SR, and SB operators.

3. For each difference Δz_1, compute the difference $x_2^{(0)}$ marked by red box and find the corresponding entry $\Delta x_2^{(0)}$ and the value $x_2^{(0)}$ in the list L_0.

The Super-Sboxes can be computed in parallel with complexity 2^{32} with the expectation that one pair is obtained for each difference pair $(\Delta w_3, \Delta z_1)$. Therefore, we could construct enough 2^{48} pairs in the inbound phase and compute backward and forward with a total 2^{48} time complexity because the outbound probability is 2^{48}.

4.2 Distinguisher on Full round AES-192 and AES-256

We investigated the distinguishing characteristics of limited birthday distinguishers on full round AES-192 and AES-256. The search is specifically focused on the 12 round AES-192 (resp. the 14 round AES-256), where the objective function of the truncated trail search on Step 1 is number of active bytes in round $4, \ldots, R-2$ (R is 12 for AES-192 and 14 for AES-256, round counter starts from 0). The experiment's results indicate that AES-192 has at least 11 active bytes in its state before SubByte operation from round 4 to round 10, and AES-256 has at least 10 active bytes in its state prior to SubByte operation from round 4 to round 12. Both of these truncated trails with minimum active bytes can be instantiated to be differential characteristics in the search of Step 2. Step 2 is determined with an objective function of total time complexity, which includes the forward and backward outbound phases through the remaining active S-boxes under the following constraints:

– keyschedule differential is solvable, *i.e.*, keyschedule part has probability higher than 2^{-192} for AES-192 (resp. 2^{-256} for AES-256),
– inbound freedom is enough, *i.e.*, available key pairs and starting points are enough for the pair of value for outbound phase,
– ideal case is higher than attack complexity, *i.e.*, the critical part, ideal case time complexity calculating from the input and output differential space must be higher than the total attack time complexity of inbound phase and outbound phase.

These 3 criteria are both coded into the constraints in Step 1 and Step 2 searches. Finally, using rebound techniques, a full round LBD on AES-192 with an attacking complexity of 2^{100} was obtained in Figure 7, while a full round LBD on AES-256 with attacking complexity of 2^{88} was obtained in Figure 12. The inbound phase covers y_1 to x_4.

The LBD Attack on Full AES-192. The attack procedures for AES-192 are described following:

1. Find a key pair $(k, k \oplus \Delta_K)$ conforming to the key differential characteristic.

2. Random assign a compatible difference for $\Delta x_4[8]$ and compute $\Delta y_3 = \mathtt{SR}^{-1}(\mathtt{MC}^{-1}(\Delta x_4 \oplus \Delta k_4))$.

3. Follow the Super-Sbox technique in Section 2.4, we construct the table $\mathbb{L}^{(i)}[\Delta y_3^{(i)}]$ that corresponds to the difference $\Delta y_3^{(i)}$ by iterating 2^{32} values of $y_3^{(i)}$ and propagate the values backwards until $w_1^{(i)}$.

4. Random choose a compatible difference for Δy_1 and compute $\Delta w_1 = \mathtt{MC}(\mathtt{SR}(\Delta y_1))$. Deduce the values of w_1 by assessing the precomputation table in Step 3.

5. After obtaining all the cell values of w_1 and Δw_1, the pairs are computed backward and forward to filter the one passing through all the remaining active S-boxes in the rest differential characteristic given in Table 4.

6. Return to Step 1 if no pair is obtained.

As shown in Table 4, the key differential (i.e., Key differences in Table 4) is of probability 2^{-96}, which contains 16 active S-boxes (each with a probability of 2^{-6}). Given a key pair satisfying the key differential, the probability of the outbound phase in the internal states (i.e., State differences in Table 4) is 2^{-100}, which consists of two parts

- the probability 2^{-72} to pass the active S-boxes from Round 4 to Round 10 in Figure 7,
- the probability 2^{-28} to pass the active S-boxes in Round 1 in Figure 7.

With the help the triangulation algorithm, it is possible to derive a key pair with a time complexity of 2^{24} although the probability of the key differential trail is 2^{-96}.

Computing the conforming key pair with the triangulation algorithm. The key differential characteristic has 16 active S-boxes, each having a probability of 2^{-6}. Out of these, 12 active S-boxes (highlighted in blue in Figure 6) of K_0, K_2, and K_4 have been fixed to their admissible values individually. We input the system of equations of all subkeys to the triangulation algorithm with the 12 known bytes, and receive the output of 12 free bytes marked by dots in K_2, K_3, and K_4 in Figure 6. We then choose random values for these free bytes, and subsequently, all the remaining bytes of K_4 have been computed using Table 3. However, since the key values still need to pass through the 4 remaining active S-boxes, each with a probability of 2^{-6}, it takes a total of 2^{24} time to uncover a single conforming key pair.

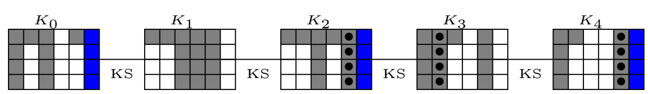


Fig. 6: The selection of fixed and free bytes in the AES-192 subkeys.

1. $K_3[20, 21, 22, 23] = K_4[16, 17, 18, 19] \oplus K_4[20, 21, 22, 23]$
2. $K_3[16, 17, 18, 19] = K_2[20, 21, 22, 23] \oplus K_3[20, 21, 22, 23]$
3. $K_3[12, 13, 14, 15] = K_2[16, 17, 18, 19] \oplus K_3[16, 17, 18, 19]$
4. $K_1[20, 21, 22, 23] = K_2[16, 17, 18, 19] \oplus K_2[20, 21, 22, 23]$
5. $K_1[16, 17, 18, 19] = K_0[20, 21, 22, 23] \oplus K_1[20, 21, 22, 23]$
6. $K_2[12, 13, 14, 15] = K_1[16, 17, 18, 19] \oplus K_2[16, 17, 18, 19]$
7. $K_3[8, 9, 10, 11] = K_2[12, 13, 14, 15] \oplus K_3[12, 13, 14, 15]$
8. $K_4[12, 13, 14, 15] = K_3[16, 17, 18, 19] \oplus K_4[16, 17, 18, 19]$
9. $K_4[8, 9, 10, 11] = K_3[12, 13, 14, 15] \oplus K_4[12, 13, 14, 15]$
10. $K_4[4, 5, 6, 7] = K_3[8, 9, 10, 11] \oplus K_4[8, 9, 10, 11]$
11. $K_4[0, 1, 2, 3] = K_3[4, 5, 6, 7] \oplus K_4[4, 5, 6, 7]$

Table 3: Steps to recover the subkey K_4 from known key bytes

Analysis of the Inbound Phase: Step 2 and Step 4 contribute 2^7 and 2^{32} degrees of freedom, respectively, resulting in $2^{7+32} = 2^{39}$ starting points. In order to obtain one pair passing through the outbound phase, whose probability is 2^{-100}, 2^{100-39} key pairs need to be prepared in Step 1 by changing the 12 free bytes marked by dots in Figure 6. The time complexity of Step 1 is $2^{61+24} = 2^{85}$. Totally, $2^{61+7+32} = 2^{100}$ starting points are generated.

The total time complexity of the attack is approximately $2^{85} + 2^{100} \approx 2^{100}$ with a memory requirement of 2^{32} to store the Super-Sboxes. In the ideal case, with three unknown bytes in Δ_{IN} and Δ_{OUT} deduced from three known difference active S-boxes, the attacker can expect to find a solution that verifies the required property in a time equivalent to $\max\{2^{\frac{128+1-\max\{7,14\}}{2}}, 2^{128+1-(7+14)}\}$, which gives a time complexity equivalent to 2^{108} encryption queries.

Table 4: Differential characteristic used in the distinguisher of 12 rounds of `AES`-192. The two lines of state differences are the respectively the state difference after `AddRoundKey` and after `MixColumn`. The last line state difference is the state difference after `ShiftRow`.

Round	State differences	Key differences
Plaintext	2D3D06BB ??000000 3D9D9DBC 00000000	
0	00000000 ??000000 00000000 00000000 00000000 219D9DBC 00000000 00000000	2D3D06BB 0C000000 3D9D9DBC 00000000
1	0C000000 0C000000 10000000 1C000000 ???????? ???????? ???????? ????????	0C000000 2D9D9DBC 10000000 1C000000
2	???????? ???????? ???????? ???????? ??000000 ??000000 ??9D9DBC ??000000	219D9DBC 219D9DBC 2D9D9DBC 00000000
3	??000000 ??000000 ??000000 ??000000 219D9DBC 219D9DBC 219D9DBC 219D9DBC	10000000 0C000000 2D9D9DBC 0C000000
4	00000000 00000000 0C000000 00000000 00000000 00000000 219D9DBC 00000000	219D9DBC 219D9DBC 2D9D9DBC 219D9DBC
5	0C000000 00000000 00000000 00000000 219D9DBC 00000000 00000000 00000000	0C000000 00000000 219D9DBC 00000000
6	0C000000 0C000000 00000000 00000000 219D9DBC 219D9DBC 00000000 00000000	2D9D9DBC 0C000000 00000000 00000000
7	00000000 00000000 0C000000 00000000 00000000 00000000 219D9D9D 00000000	219D9DBC 219D9DBC 0C000000 00000000
8	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 219D9DBC 00000000
9	0C000000 0C000000 0C000000 0C000000 219D9DBC 219D9DBC 219D9DBC 219D9DBC	0C000000 0C000000 0C000000 0C000000
10	0C000000 0C000000 00000000 0C000000 219D9DBC 219D9DBC 00000000 219D9DBC	2D9D9DBC 2D9D9DBC 219D9DBC 2D9D9DBC
11	00000000 0C000000 00000000 0C000000 00000000 ??000000 00000000 ??000000	219D9DBC 2D9D9DBC 00000000 2D9D9DBC
Ciphertext	0C000000 ??9D9DBC 00000000 ??9D9DBC	0C000000 219D9DBC 00000000 2D9D9DBC

Analysis of the `AES`-256 distinguisher. The procedure of `AES`-256 distinguisher attack is similar to `AES`-192, using the concrete differential characteristic specified in Table 6. The degrees of freedom are calculated from the number of possible key pairs and state pairs conforming the truncated round Inbound

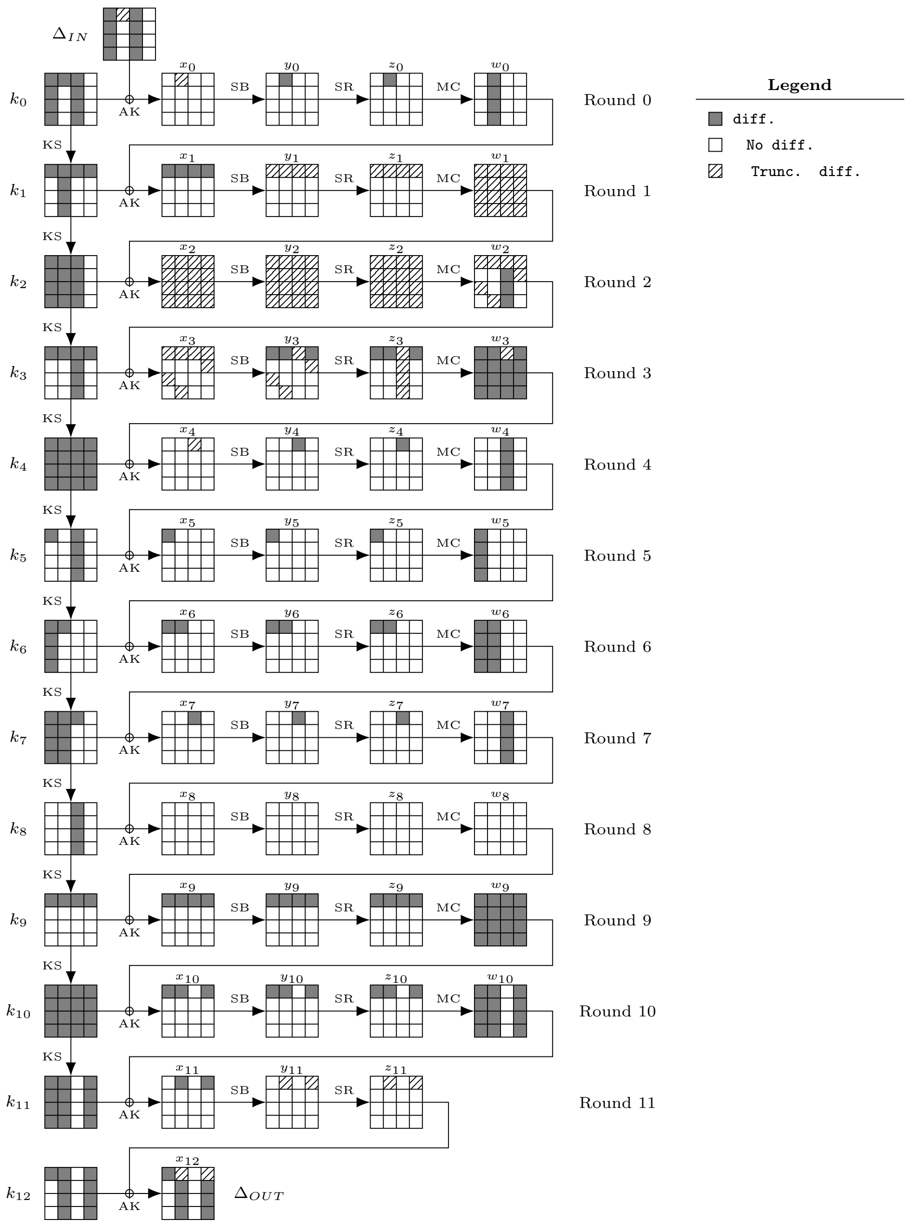


Fig. 7: Differential characteristic of 12-round AES-192 used in the distinguisher.

part. It is estimated that 2^{32} starting points are generated for each key pair, and around 2^{56} key pairs need to be found. The time to find one key pair is about 2^8 with triangulation algorithm. The total complexity for finding one key pair and message pairs that satisfy the entire differential trail is $2^{56+8} + 2^{88} \approx 2^{88}$, while it is 2^{94} in the ideal case.

5 Distinguisher on Full round `Kiasu-BC`

The tweakable block cipher `Kiasu-BC` was introduced by Jean et al. [38] as a candidate in the CAESAR competition for authenticated encryption. The design of `Kiasu-BC` is crucially similar to `AES` cipher, except for the appearance of a 64-bit tweak value which is XORed to the two first rows of the state in each round after adding the round-key. Thus, it can be seen as the simplest instance of the `TWEAKEY` framework [39], where an identical tweak is used for all round functions. The tweak T is described in Figure 8.

$$T = \begin{array}{|c|c|c|c|} \hline T_0 & T_2 & T_4 & T_6 \\ \hline T_1 & T_3 & T_5 & T_7 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Fig. 8: 64-bit tweak in `Kiasu-BC`: $T = T_0 \| T_1 ... \| T_7$.

5.1 Practical 9-round Distinguisher

In this section, by using the degree of freedom from the tweak, we enhance the 9-round distinguisher of `AES`-128 in [27] to be consistent with `Kiasu-BC` distinguisher and reduce the time complexity of the distinguisher from 2^{55} to 2^{36}. The same characteristic in Figure 9 and the key pair found in [27] are utilised, and three S-boxes which include 2 S-boxes in Round 4 and 1 S-box in Round 1 are fulfilled by the tweak cells. One pair found is shown in Figure 13 in Supplementary Material.

The Inbound Phase: As shown in Figure 9, the Inbound phase marked with dash lines starts from z_1 to z_3 and now extends to x_1 to x_4 so that the S-box in x_1 and 2 S-boxes in $x_4[1]$ and $x_4[13]$ are covered. To generate data pairs and a tweak conforming a given difference $(\Delta z_1, \Delta z_3)$ and the truncated differential in Figure 9, the following steps are performed:

1. Compute $\Delta x_2 = \mathtt{MC}(\Delta z_1) \oplus \Delta k_2$ and $\Delta y_3 = \mathtt{SR}^{-1}(\Delta z_3)$.

2. Assign compatible differences for $(\Delta x_3[0], \Delta x_3[3])$ and compute $(\Delta w_2[0], \Delta w_2[3]) = (\Delta x_3[0], \Delta x_3[3]) \oplus (\Delta k_3[0], \Delta k_3[3])$, $\Delta z_2^{(0)} = \mathtt{MC}^{-1}(\Delta w_2^{(0)})$, and check whether the difference of $\Delta z_2^{(0)}$ is compatible with the Super-Sboxes tuple difference $(\Delta x_2[0], \Delta x_2[5], \Delta x_2[10], \Delta x_2[15])$. If yes, deduce the value of these cells by assessing DDT, and compute $w_2^{(0)}$. Build the list L_0 to store the tuples $(w_2^{(0)}, \Delta w_2^{(0)})$ satisfied the condition: $\mathtt{SB}(w_2[3] \oplus k_3[3]) \oplus \mathtt{SB}(w_2[3] \oplus \Delta w_2[3] \oplus k'_3[3]) = \Delta y_3[3]$.

3. Similar to step 2, build the lists L_1, L_2, and L_3 to store the tuples $(w_2^{(1)}, \Delta w_2^{(1)})$, $(w_2^{(2)}, \Delta w_2^{(2)})$, and $(w_2^{(3)}, \Delta w_2^{(3)})$ by the corresponding differences $\Delta x_3[4,5]$, $\Delta x_3[9,10]$, $\Delta x_3[14,15]$ and the Super-Sboxes tuples $\Delta x_2[4,9,14,3]$, $\Delta x_2[8,13,2,7]$, and $\Delta x_2[12,1,6,11]$.

4. For all possible tuples in L_0, L_1, L_2, L_3:
   - Deduce $x_3[0,4,5,9]$, $y_3[0,4,5,9]$ by assessing the DDT, and $tw[0,2,3,5] = w_2[0,4,5,9] \oplus k_3[0,4,5,9] \oplus x_3[0,4,5,9]$. Compute bytes $x_3[3,10,14,15] = w_2[3,10,14,15] \oplus k_3[3,10,14,15]$ and the corresponding y_3 cells. Compute $w_3[0,1,2,3] = \mathtt{MC}(\mathtt{SR}(y_3[0,5,10,15]))$ and $tw[1] = w_3[1] \oplus k_4[1] \oplus x_4[1]$ for $x_4[1]$ deduced from DDT.
   - With full state of w_2 are known, compute backward to x_2. Compute $w_1[9] = x_2[9] \oplus k_2[9] \oplus tw[5]$ and $w_1[10,11] = x_2[10,11] \oplus k_2[10,11]$. With $z_1[8]$ deduced from DDT, $w_1[8] = \mathtt{MC}^{-1}(z_1[8], w_1[9,10,11])$ and $tw[4] = w_1[8] \oplus k_2[8] \oplus x_2[8]$.
   - Assign random value to $x_3[12]$ and deduce $tw[6] = w_2[12] \oplus k_3[12] \oplus x_3[12]$. Compute $x_3[1] = w_2[1] \oplus k_3[1] \oplus tw[1], x_3[6,11] = w_2[6,11] \oplus k_3[6,11]$ and the corresponding $w_3[12,13,14,15] = \mathtt{MC}(\mathtt{SR}(\mathtt{SB}([x_3[12,1,6,11]))))$. Obtain $tw[7] = w_3[13] \oplus x_4[13] \oplus k_4[13]$ with $x_4[13]$ deduces from DDT.
   - After generating enough 2^{36} pairs of full states (w_2, w'_2) and the tweak tw for the starting points, stop.

Analysis of the Inbound Phase: In Step 2, 2^{14} pairs of difference $(\Delta x_3[0], \Delta x_3[3])$ are sampled. Only the condition for $w_2[3]$ acts as a filter of probability 2^{-7} for one byte to hit the difference $\Delta y_3[3]$, even though the matching difference for $\Delta z_2^{(0)}$ is 2^{-4} but also leads to 2^4 assembled values passing through S-boxes. Therefore, about 2^7 tuples are stored in list L_0. Similar evaluations are applied for L_1, L_2, and L_3, resulting in corresponding 2^{14}, 2^7, and 2 tuples being counted. Since $x_3[12]$ are chosen randomly in step 4 and at least 2 assembled values are found for each $x_4[1]$ and $x_4[13]$, we expect about $2^{7+14+7+1-1+8+2} = 2^{38}$ (-1 for the same pair with different sequences) starting points are found for a given difference $(\Delta z_1, \Delta z_3)$, which leads to an ample number of starting points for the outbound phase despite using only one random difference of z_1. The memory complexity has been diminished to 2^{14}, as opposed to the 2^{32} memory storage required by the Super-Sboxes technique [32].

The Outbound Phase: Since the probability of the forward differential characteristic is 2^{36} for the remaining 6 active S-boxes at $x_4[9], x_5, x_6$, and x_7, and no active S-box left in the backward trail, we expect to find a pair satisfying the whole 9-round trail in 2^{36} time complexity.

Generic Time Complexity: The differential trail has 5 unknown difference bytes in the Δ_{IN} and 3 unknown difference bytes in the Δ_{OUT}, giving space vectors of $d_{in} = 39$ and $d_{out} = 21$. Following Theorem 1, a generic algorithm takes around 2^{68} querying time to find a pair conforming to the input and output differences of an ideal permutation on $n = 128$, and uses 2^{21} of memory storage.

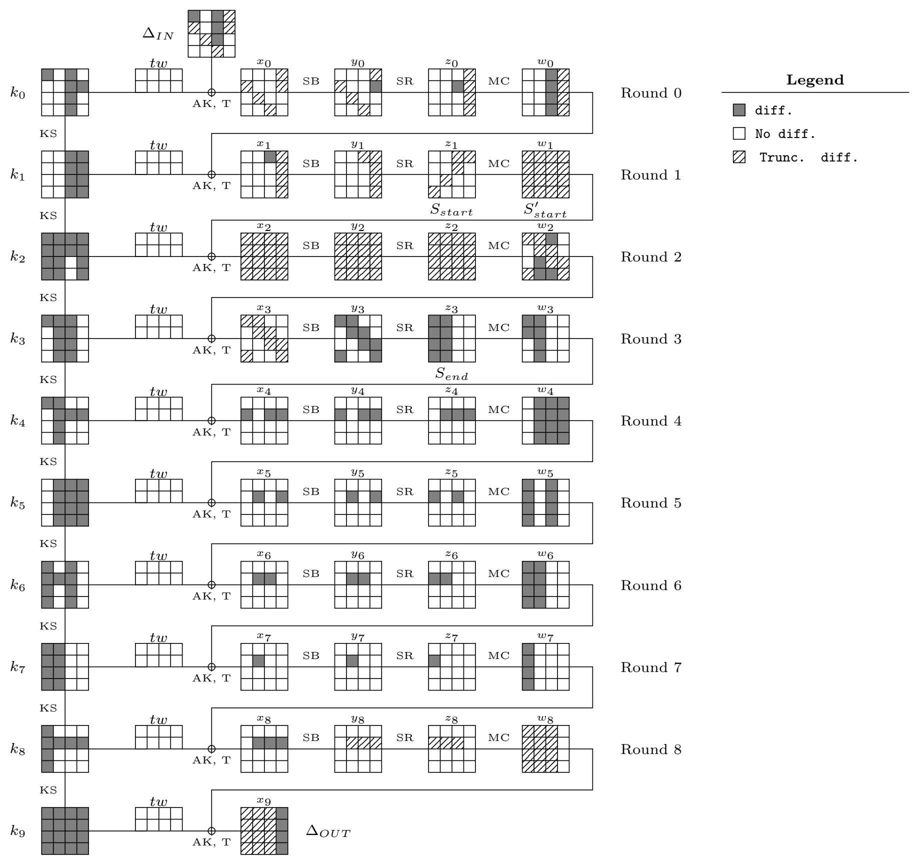


Fig. 9: Differential characteristic of 9-round `Kiasu-BC` used in the distinguisher.

5.2 The First 10-round Distinguisher

The 7-round AES-128 Related-key Differential Trail. As shown in Figure 10, the new 7-round differential trail is marked with gray color from round 3 to round 9, which consists of 11 active S-boxes in states with probability 2^{-67} and active S-boxes in subkeys. We note that while this characteristic trail may not be the best fit for 7-round related-key AES-128, it aligns better with the degrees of freedom from the tweak, thus providing a more advantageous limited birthday distinguisher for `Kiasu-BC`. By extending 3 rounds backward, a 10-round trail is formed with 3 rounds in the Inbound part and 2^{-67} probability in the Outbound part.

The Inbound phase: Compared to the previous 9-round distinguisher, an additional round has been added to the Inbound part of the 10-round attack, encompassing rounds 1 to 3. However, the attack still starts at z_1, which is marked as S_{start}, and uses the degrees of freedom from the tweak to fulfill the active S-boxes in x_1. Given a difference Δz_1 compatible with the fixed differences in x_1, we find the pairs of states and a tweak conforming to the Inbound trail with fixed differences $(\Delta z_1, \Delta y_3)$ by the following steps:

1. Deduce $\Delta x_2 = \texttt{MC}(\Delta z_1) \oplus \Delta k_2$.

2. Assign compatible differences to $\Delta x_3[0,1,3]$ and compute backward $\Delta w_2^{(0)} = \Delta x_3^{(0)} \oplus k_3^{(0)}$ and $\Delta z_2^{(0)} = \texttt{MC}^{-1}(\Delta w_2^{(0)})$. Next, check the compatibility of $\Delta x_2[0,5,10,15]$ and $\Delta z_2^{(0)}$ and deduce the corresponding cells if all differences are compatible. After that, $w_2^{(0)}$ is computed and store the pair $(w_2^{(0)}, \Delta w_2^{(0)})$ in to list L_0 if the condition $\texttt{SB}(w_2[3] \oplus k_3[3]) \oplus \texttt{SB}(w_2[3] \oplus \Delta w_2[3] \oplus k_3'[3]) = \Delta y_3[3]$ is passed.

3. Similar to step 2, build the lists L_1, L_2, and L_3 to store the pairs $(w_2^{(1)}, \Delta w_2^{(1)}), (w_2^{(2)}, \Delta w_2^{(2)})$, and $(w_2^{(3)}, \Delta w_2^{(3)})$ by the corresponding differences $\Delta x_3[4,5,6], \Delta x_3[9,11]$, and $\Delta x_3[12,14,15]$.

4. Deduce $z_1[0,5,7,9]$ by assessing the `DDT`.

5. For all possible tuples in L_0, L_1, L_2, L_3:
   - Deduce $x_3[0,9,12]$ by assessing the `DDT`, and compute $tw[0,5,6] = w_2[0,9,12] \oplus k_3[0,9,12] \oplus x_3[0,9,12]$.
   - Since the full state of w_2 are known, we compute backward to obtain full state of x_2, and the bottom two rows of w_1 by XORing x_2 with k_2 in the respected cells. Extra cells $w_1[0,9,12] = x_2[0,9,12] \oplus k_2[0,9,12] \oplus tw[0,5,6]$ are also computed. Obtain $w_1[1] = \texttt{MC}^{-1}(z_1[0], w_1[0,2,3]), w_1[4,5] = \texttt{MC}^{-1}(z_1[5,7], w_1[6,7]), w_1[8] = \texttt{MC}^{-1}(z_1[9], w_1[9,10,11])$ and $tw[1,2,3,4] = w_1[1,4,5,8] \oplus x_2[1,4,5,8] \oplus k_2[1,4,5,8]$.
   - Filter the tuples satisfied the following conditions:
     (a) $\texttt{SB}(w_2[1] \oplus k_3[1] \oplus tw[1]) \oplus \texttt{SB}(w_2[1] \oplus \Delta w_2[1] \oplus k_3'[1] \oplus tw[1]) = \Delta y_3[1]$.
     (b) $\texttt{SB}(w_2[4] \oplus k_3[4] \oplus tw[2]) \oplus \texttt{SB}(w_2[4] \oplus \Delta w_2[4] \oplus k_3'[4] \oplus tw[2]) = \Delta y_3[4]$.
     (c) $\texttt{SB}(w_2[5] \oplus k_3[5] \oplus tw[3]) \oplus \texttt{SB}(w_2[5] \oplus \Delta w_2[5] \oplus k_3'[5] \oplus tw[3]) = \Delta y_3[5]$.
6. Randomly choose $tw[7]$ and form the starting points for the Outbound phase.

Analysis of the Inbound Phase: The list L_0 has approximately 2^{14} elements since 2^{21} differences of $x_3[0, 1, 3]$ are sampled and about 2^{14} tuples are passed through a filter of 2^{-7} in Step 2. Estimating with a similar computation, we anticipate that 2^{14} tuples are stored in list L_1, 2^7 tuples in list L_2, and an equal number, 2^7 tuples, in list L_3. Among the 2^{42} list combinations formed in Step 5, about 2^{21} tuples are passed through 3 conditions. Since the $tw[7]$ is freely chosen, we can form approximately 2^{29} solutions for each choice of difference Δz_1. Consequently, in order to acquire the necessary 2^{67} pairs for the Outbound phase, 2^{38} iterations of Δz_1 differences are performed, resulting in a total time complexity of 2^{67} and memory complexity of 2^{14} for the entire attack. While the generic algorithm incurs a time complexity of 2^{96}, given the 4 free difference cells in Δ_{IN} and no differences in Δ_{OUT}.

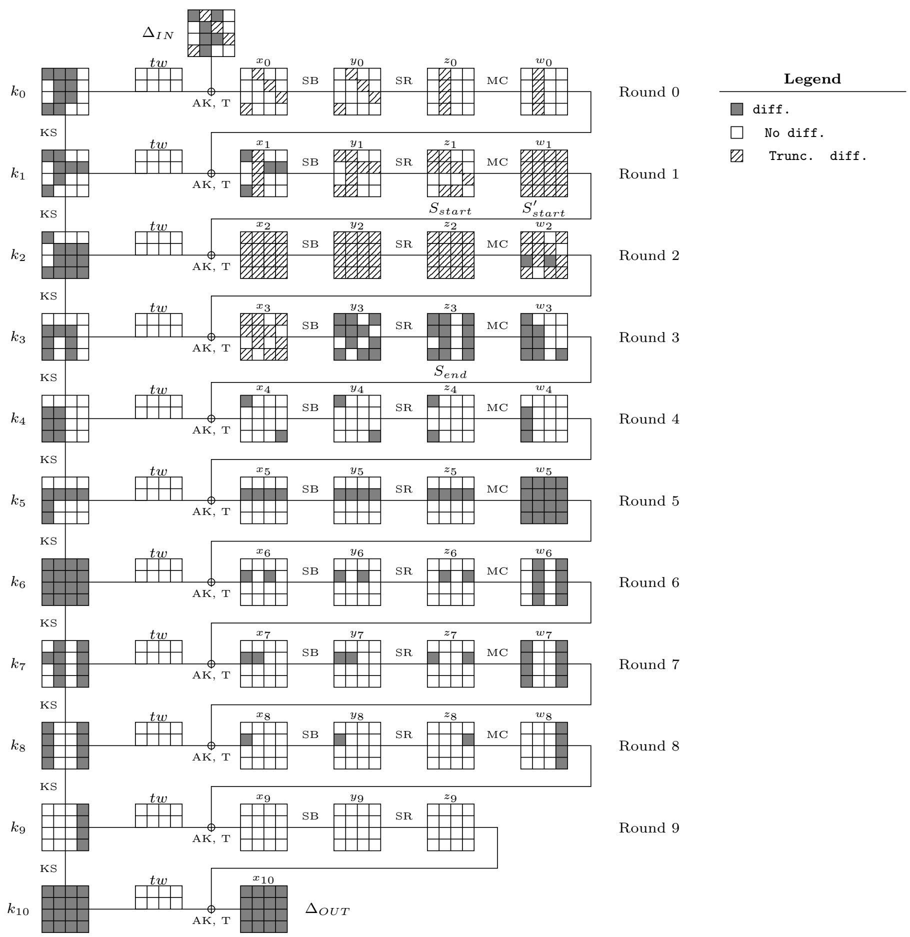


Fig. 10: Differential characteristic of 10-round `Kiasu-BC` used in the distinguisher.

Table 5: The key pair conforming to differential characteristic used in the 10-round `Kiasu-BC` distinguisher.

Round	k	k'	$k \oplus k'$
0	1C709357 CDF3E375 6A486781 D3D9E8D5	8E7093B7 64280395 51CD8781 D3D9E8D5	920000E0 A9DBE0E0 3B85E000 00000000
1	28EB9031 E5187344 8F5014C5 5C89FC10	BAEB90D1 DEC39344 8F0E14C5 5CD7FC10	920000E0 3BDBE000 005E0000 005E0000
2	8D5B5A7B 6843293F E7133DFA BB9AC1EA	B65B5A9B 6898C9DF E796DD1A BB41210A	3B0000E0 00DBE0E0 0085E0E0 00DBE0E0
3	3123DD91 5960F4AE BE73C954 05E908BE	31A63D71 593EF4AE BEA829B4 05E908BE	0085E0E0 005E0000 00DBE0E0 00000000
4	271373FA 7E738754 C0004E00 C5E946BE	2796931A 7EA867B4 C0004E00 C5E946BE	0085E0E0 00DBE0E0 00000000 00000000
5	2949DD5C 573A5A08 973A1408 52D352B6	29CC3DBC 57645A08 97641408 528D52B6	0085E0E0 005E0000 005E0000 005E0000
6	6F49935C 3873C954 AF49DD5C FD9A8FEA	54CC73BC 03A829B4 94CC3DBC C6416F0A	3B85E0E0 3BDBE0E0 3B85E0E0 3BDBE0E0
7	973A1408 AF49DD5C 00000000 FD9A8FEA	97641408 94CC3DBC 00000000 C6416F0A	005E0000 3B85E0E0 00000000 3BDBE0E0
8	AF49935C 00004E00 00004E00 FD9AC1EA	94CC73BC 00004E00 00004E00 C641210A	3B85E0E0 00000000 00000000 3BDBE0E0
9	0C311408 0C315A08 0C311408 F1ABD5E2	0C311408 0C315A08 0C311408 CA703502	00000000 00000000 00000000 3BDBE0E0
10	58328CA9 5403D6A1 5832C2A9 A999174B	6BA7637C 67963974 6BA72D7C A1D7187E	3395EFD5 3395EFD5 3395EFD5 084E0F35

6 Distinguisher on `Deoxys-BC`

`Deoxys-BC` adopts the same round function of `AES`. The distinction lies in the `KeySchedule` part, where the subtweakeys STK_i are generated from both the key and the tweak, replacing the subkeys that were previously derived solely from the master key. These subtweakeys are then XORed with the state in the `AddRoundTweakey` operation.

Description of the `Tweakey Schedule`. In the tweakey schedule of `Deoxys-BC`, the key K and the tweak T are concatenated as $KT = K \| T$ to form the subtweakey state. Depending on the size of the tweak, KT is divided into two 128-bit words TK_0^1, TK_0^2 or three 128-bit words TK_0^1, TK_0^2, and TK_0^3 corresponding to `Deoxys-BC`-256 or `Deoxys-BC`-384. We denote STK_i $(i \geq 0)$ as the 128-bit subtweakey added to the state at round i during the `AddRoundTweakey` operation. Then the subtweakey is constructed as $STK_i = TK_i^1 \oplus TK_i^2 \oplus RC_i$ for `Deoxys-BC`-256 and $STK_i = TK_i^1 \oplus TK_i^2 \oplus TK_i^3 \oplus RC_i$ for `Deoxys-BC`-384, where RC_i is the round constant. The values of TK_i^1, TK_i^2, TK_i^3 are defined by the following linear transformation:

$$TK_{i+1}^1 = h(TK_i^1),\ TK_{i+1}^2 = h(LFSR_2(TK_i^2)),\ TK_{i+1}^3 = h(LFSR_3(TK_i^3))$$

where h is a linear byte permutation on 16 bytes defined by:

$$h(x_0 \| x_1 \| ... \| x_{15}) = x_1 \| x_6 \| x_{11} \| x_{12} \| x_5 \| x_{10} \| x_{15} \| x_0 \| x_9 \| x_{14} \| x_3 \| x_4 \| x_{13} \| x_2 \| x_7 \| x_8.$$

6.1 New `Deoxys-BC` Differential Characteristics

We have searched for the differential characteristics of limited birthday distinguishers on 10-round `Deoxys-BC`-256 and 13-round `Deoxys-BC`-384 with the CP automatic tool. Specifically, the Step 1 search is similar to the truncated differential search on full-round `AES`-192/256 in Section 4.2, but the varying linear tweakey results in varying constraints on the tweakey part. For Step 1 of the truncated trail search, we add the constraints of the linear incompatibility [14] between differential propagations in the tweakey and the state. We also impose additional requirements that the freedom from tweakey differences must be greater than 1, allowing us to utilize these freedoms to search for better differential characteristics in Step 2. For Step 2 search, we do not use the automatic tool like Choco-solver used in AES LBD search. With greater freedom from the tweakey, we can directly search for the rear partial differential characteristic and go backward 4 rounds using the same method as outlined in [27]. We found a 10-round LBD on Deoxys-BC-256 in Figure 11 and an 13-round LBD on Deoxys-BC-384 in Figure 14 with a probability of 2^{-69} and 2^{-42} respectively.

6.2 Deoxys-BC Limited Birthday Distinguisher Attacks

In Figure 11, attacking complexity is forward outbound complexity times backward outbound complexity. The forward outbound phase has 10 active bytes from rounds 5 to 9 with a probability of 2^{-69}, and the backward outbound phase has no remaining active byte, resulting in a total attacking complexity of 2^{69}. In the ideal case, the time complexity, as calculated by Theorem 1, would be $2^{129-d_{in}-d_{out}} = 2^{129-0-28} = 2^{101}$.

In Figure 14, the backward outbound phase has 6 active bytes from round 1 to round 6 with a probability of 2^{-42} and no active bytes in the forward rounds, hence total attacking complexity is also 2^{42}. While the time complexity, as determined by Theorem 1, would be $2^{129-d_{in}-d_{out}} = 2^{129-32-39} = 2^{58}$ in the ideal scenario. We take advantage of the fact that the tweakeys are freely chosen by the attacker in both Deoxys-BC-256 and Deoxys-BC-384 to connect the active S-boxes in the heavy Inbound phases. For example, STK_2 and STK_3 are carefully chosen to connect the active bytes between round 2 and round 3, and between round 3 and round 4, respectively. In this way, the first distinguishing attack on 10-round Deoxys-BC-256 and 13-round Deoxys-BC-384 can be done with a time complexity of 2^{69} and 2^{42} respectively. We only outline the procedure in the distinguisher of 10-round Deoxys-BC-256, and the procedure for 13-round Deoxys-BC-384 is given in Supplementary Material C. The Inbound part, starting from Round 2 to Round 4, executes the following steps to produce a pair that conforms to the given differences $(\Delta z_2, \Delta x_4)$:

1. Deduce all the active bytes in x_2, x_3, and x_4 by assessing DDT. The corresponding active bytes in z_2, z_3, and z_4 are all acquired.

2. Randomly assign values to $STK_2[4,5,6]$ and $STK_2[12]$. Compute $w_2[4,5,6,12] = x_3[4,5,6,12] \oplus STK_2[4,5,6,12]$. From the Property 1, obtain $w_2[7] = \mathtt{MC}^{-1}(z_2[6], w_2[4,5,6])$, $w_2[13,14,15] = \mathtt{MC}^{-1}(z_2[12,13,15], w_2[12])$ and the corresponding tweakey cells $STK_2[7] = w_2[7] \oplus x_3[7]$, $STK_2[13,14,15] = w_2[13,14,15] \oplus x_3[13,14,15]$.

3. Compute $w_3[4,5,6,7] = \mathtt{MC}(z_3[4,5,6,7])$ and deduce $STK_3[4,6] = w_3[4,6] \oplus x_4[4,6]$. Randomly choose $STK_3[1,9]$ and deduce $STK_3[3,11]$ by Property 1.

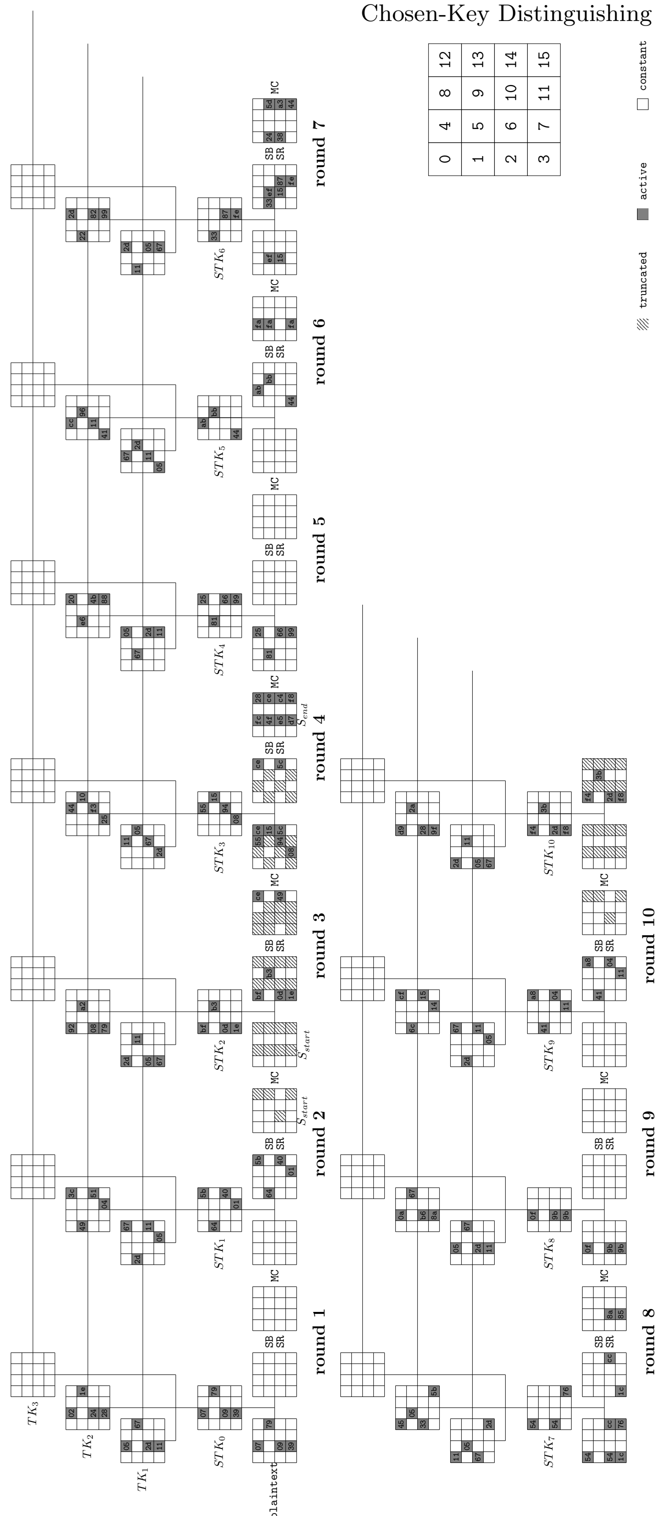


Fig. 11: LBD attack on 10-round Deoxys-BC-256

4. Solve the system of linear equations of the STK_2 and STK_3. 14 out of 32 bytes of two tweakeys are fixed, 18 bytes in tweakeys can be freely chosen, which contributes enough degrees of freedom for needed starting point pairs.

7 Discussion and Conclusion

7.1 Possible Increment in the Number of Attacked Rounds

The full-round partial truncated differential characteristics of AES-192 and AES-256 are successfully obtained by the Constraint Programming automatic search tool. However, no 10-round trail for AES-128 limited birthday distinguisher has been found. The 10-round trail illustrated in Figure 10 works for the Kiasu-BC attack, however, it is unsuitable for AES-128 as there are not enough degrees of freedom for generating starting point pairs for the Outbound phases. Therefore, searching for a full-round distinguishing attack on AES-128 remains a potential future work.

The attacks on full-round Deoxys-BC-256 and Deoxys-BC-384 have not been deployed. Actually, we only use two and three subtweakeys to connect several rounds in both 10-round Deoxys-BC-256 and 13-round Deoxys-BC-384 attacks, respectively, despite the large degrees of freedom in the tweakeys. Therefore, a similar investigation could be done on more rounds by extending the Inbound part to 3 rounds for Deoxys-BC-256 and to 5 rounds for Deoxys-BC-384.

7.2 Conclusion

In this paper, we investigate the security of AES and two tweakable block ciphers Kiasu-BC and Deoxys-BC against related-key distinguishing attacks. With the aid of Constraint Programming automatic search tool, we successfully found the full-round related-key differential characteristics for AES-192, AES-256, Kiasu-BC, a 10-round related-key differential characteristic for Deoxys-BC-256, and an 13-round related-key differential characteristic for Deoxys-BC-384, with acceptable probabilities for limited birthday attacks. For tweakable block ciphers, the inbound phase is extended to 4 rounds by taking advantage of the available degrees of freedom from the tweaks. These tweaks' free bytes also contribute to reduce both the time and memory complexities. As a consequence, some practical distinguishing attacks on Kiasu-BC were presented. Applying our method to other tweakable block ciphers is a potential future avenue of exploration.

References

1. ISO/IEC 18033-7:2022(en) Information security — Encryption algorithms — Part 7: Tweakable block ciphers. https://www.iso.org/obp/ui/#iso:std:iso-iec:18033:-7:ed-1:v1:en.
2. Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In *FSE 2013*, volume 8424, pages 348–366.

3. Zhenzhen Bao, Lin Ding, Jian Guo, Haoyang Wang, and Wenying Zhang. Improved meet-in-the-middle preimage attacks against AES hashing modes. *IACR Trans. Symmetric Cryptol.*, 2019(4):318–347, 2019.
4. Augustin Bariant and Gaëtan Leurent. Truncated boomerang attacks and application to aes-based ciphers. *IACR Cryptol. ePrint Arch.*, page 701, 2022.
5. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016, Proceedings, Part II*, pages 123–153. Springer, 2016.
6. Eli Biham. New types of cryptanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *EUROCRYPT '93, Proceedings*, volume 765, pages 398–409.
7. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *EUROCRYPT 2010, Proceedings*, volume 6110, pages 299–319.
8. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009, Proceedings*, volume 5912, pages 1–18.
9. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In *CRYPTO 2009, Proceedings*, volume 5677, pages 231–249.
10. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In *EUROCRYPT 2010, Proceedings*, volume 6110, pages 322–344.
11. Céline Blondeau, Thomas Peyrin, and Lei Wang. Known-key distinguisher on full PRESENT. In *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 455–474.
12. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In Phillip Rogaway, editor, *CRYPTO 2011, Proceedings*, volume 6841, pages 169–187.
13. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In *EUROCRYPT 2018, Proceedings, Part II*, volume 10821, pages 683–714.
14. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A security analysis of Deoxys and its internal tweakable block ciphers. *IACR Transactions on Symmetric Cryptology*, pages 73–107, 2017.
15. Benoit Cogliati and Yannick Seurin. On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In *EUROCRYPT 2015, Proceedings, Part I*, volume 9056, pages 584–613.
16. Benoît Cogliati and Yannick Seurin. Strengthening the known-key security notion for block ciphers. In *FSE 2016*, volume 9783, pages 494–513.
17. Tingting Cui, Ling Sun, Huaifeng Chen, and Meiqin Wang. Statistical integral distinguisher with multi-structure and its application on AES. In *ACISP 2017, Proceedings, Part I*, volume 10342, pages 402–420.
18. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
19. Ivan Damgård. A design principle for hash functions. In *CRYPTO '89, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.
20. Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020.

21. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Faster chosen-key distinguishers on reduced-round AES. In *INDOCRYPT 2012*, volume 7668, pages 225–243.
22. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Square attack on 7-round Kiasu-BC. In *ACNS 2016, Proceedings*, volume 9696, pages 500–517.
23. Christoph Dobraunig and Eik List. Impossible-differential and boomerang cryptanalysis of round-reduced Kiasu-BC. In *CT-RSA 2017, Proceedings*, volume 10159, pages 207–222.
24. Xiaoyang Dong, Jian Guo, Shun Li, and Phuong Pham. Triangulating rebound attack on aes-like hashing. *IACR Cryptol. ePrint Arch.*, page 731, 2022.
25. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In *EUROCRYPT 2022, Proceedings, Part III*, volume 13277, pages 3–33.
26. 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)*, 7(7.5):3, 2010.
27. Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. Structural evaluation of aes and chosen-key distinguisher of 9-round aes-128. In *CRYPTO 2013*, pages 183–203. Springer, 2013.
28. David Gérault and Pascal Lafourcade. Related-key cryptanalysis of midori. In Orr Dunkelman and Somitra Kumar Sanadhya, editors, *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, volume 10095 of *Lecture Notes in Computer Science*, pages 287–304, 2016.
29. David Gerault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Computing aes related-key differential characteristics with constraint programming. *Artificial intelligence*, 278:103183, 2020.
30. David Gérault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 584–601. Springer, 2016.
31. Henri Gilbert. A simplified representation of AES. In *ASIACRYPT 2014, Proceedings, Part I*, volume 8873, pages 200–222.
32. Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for aes-like permutations. In *FSE 2010, Seoul, Korea, February 7-10, 2010*, pages 365–383, 2010.
33. Lorenzo Grassi and Christian Rechberger. Revisiting gilbert's known-key distinguisher. *Des. Codes Cryptogr.*, 88(7):1401–1445, 2020.
34. Akinori Hosoyamada, María Naya-Plasencia, and Yu Sasaki. Improved attacks on sLiSCP permutation and tight bound of limited birthday distinguishers. *IACR Trans. Symmetric Cryptol.*, 2020(4):147–172, 2020.
35. Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-birthday distinguishers for hash functions. In *ASIACRYPT 2013, Proceedings, Part II*, volume 8270, pages 504–523.
36. Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.
37. Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple limited-birthday distinguishers and applications. In *SAC 2013*, pages 533–550. Springer, 2013.

38. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Kiasu v1. *Submitted to the CAESAR competition*, 2014.
39. Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Tweaks and keys for block ciphers: the tweakey framework. In *ASIACRYPT 2014*, pages 274–288. Springer, 2014.
40. Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. Deoxys v1. 41. *Submitted to CAESAR*, 124, 2016.
41. Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic. Speeding up collision search for byte-oriented hash functions. In *CT-RSA 2009*, pages 164–181.
42. Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *ASIACRYPT 2007, Proceedings*, volume 4833, pages 315–324.
43. Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *FSE 2011*, pages 306–327. Springer, 2011.
44. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full whirlpool compression function. In *ASIACRYPT 2009, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 126–143, 2009.
45. Martin M. Lauridsen and Christian Rechberger. Linear distinguishers in the key-less setting: Application to PRESENT. In *FSE 2015*, volume 9054, pages 217–240.
46. Gaëtan Leurent and Clara Pernot. New representations of the AES key schedule. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Proceedings, Part I*, volume 12696, pages 54–84. Springer, 2021.
47. Moses Liskov, Ronald L Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO 2002*, pages 31–46. Springer, 2002.
48. Fukang Liu, Takanori Isobe, and Willi Meier. Automatic verification of differential characteristics: Application to reduced gimli. In *CRYPTO 2020, Proceedings, Part III*, volume 12172, pages 219–248.
49. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. In *SAC 2009, Revised Selected Papers*, volume 5867, pages 16–35.
50. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced whirlpool and grøstl. In *FSE 2009, Leuven, Belgium, February 22-25, 2009*, pages 260–276, 2009.
51. Bart Mennink and Bart Preneel. On the impact of known-key attacks on hash functions. In *ASIACRYPT 2015, Proceedings, Part II*, volume 9453, pages 59–84.
52. Ralph C. Merkle. A certified digital signature. In *CRYPTO '89, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 218–238, 1989.
53. Kazuhiko Minematsu. Building blockcipher from small-block tweakable blockcipher. *Designs, Codes and Cryptography*, 74(3):645–663, 2015.
54. Marine Minier, Raphael C.-W. Phan, and Benjamin Pousse. Distinguishers for ciphers and known key attack against rijndael with large blocks. In *AFRICACRYPT 2009, Proceedings*, volume 5580, pages 60–76.
55. Marine Minier, Christine Solnon, and Julia Reboul. Solving a symmetric key cryptographic problem with constraint programming. In *ModRef 2014, Workshop of the CP 2014 Conference*, page 13, 2014.
56. Yusuke Naito. Full prf-secure message authentication code based on tweakable block cipher. In *International Conference on Provable Security*, pages 167–182. Springer, 2015.
57. Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In

International Conference on Principles and Practice of Constraint Programming, pages 529–543. Springer, 2007.
58. Ivica Nikolic, Josef Pieprzyk, Przemyslaw Sokolowski, and Ron Steinfeld. Known and chosen key differential distinguishers for block ciphers. In *ICISC 2010*, volume 6829, pages 29–48.
59. Thomas Peyrin and Yannick Seurin. Counter-in-tweak: authenticated encryption modes for tweakable block ciphers. In *CRYPTO 2016*, pages 33–63. Springer, 2016.
60. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93, Proceedings*, volume 773, pages 368–378.
61. Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. Choco solver documentation. *TASC, INRIA Rennes, LINA CNRS UMR*, 6241:13–42, 2016.
62. Sadegh Sadeghi, Vincent Rijmen, and Nasour Bagheri. Proposing an milp-based method for the experimental verification of difference-based trails: application to speck, SIMECK. *Des. Codes Cryptogr.*, 89(9):2113–2155, 2021.
63. Yu Sasaki. Known-key attacks on rijndael with large blocks and strengthening *ShiftRow* parameter. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC 2010*, volume 6434, pages 301–315.
64. Yu Sasaki. Improved related-tweakey boomerang attacks on Deoxys-BC. In *Africacrypt 2018*, pages 87–106. Springer, 2018.
65. Yu Sasaki, Sareh Emami, Deukjo Hong, and Ashish Kumar. Improved known-key distinguishers on feistel-sp ciphers and application to camellia. In *ACISP 2012*, volume 7372, pages 87–100.
66. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox analysis: Applications to ECHO and grøstl. In *ASIACRYPT 2010, Singapore, December 5-9, 2010. Proceedings*, pages 38–55, 2010.
67. Yu Sasaki and Kan Yasuda. Known-key distinguishers on 11-round feistel and collision attacks on its hashing modes. In *FSE 2011*, volume 6733, pages 397–415.
68. Rich Schroeppel and Hilarie Orman. The hasty pudding cipher. *AES candidate submitted to NIST*, page M1, 1998.
69. Siwei Sun, David Gérault, Pascal Lafourcade, Qianqian Yang, Yosuke Todo, Kexin Qiao, and Lei Hu. Analysis of aes, skinny, and others with constraint programming. *IACR Trans. Symmetric Cryptol.*, 2017(1):281–306, 2017.
70. Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. A meet in the middle attack on reduced round Kiasu-BC. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 99-A(10):1888–1890, 2016.
71. Haoyang Wang and Thomas Peyrin. Boomerang switch in multiple rounds. application to AES variants and Deoxys. *IACR Transactions on Symmetric Cryptology*, pages 142–169, 2019.
72. Boxin Zhao, Xiaoyang Dong, and Keting Jia. New related-tweakey boomerang and rectangle attacks on Deoxys-BC including bdt effect. *IACR Transactions on Symmetric Cryptology*, pages 121–151, 2019.
73. Boxin Zhao, Xiaoyang Dong, Keting Jia, and Willi Meier. Improved related-tweakey rectangle attacks on reduced-round Deoxys-BC-384 and deoxys-i-256-128. In *INDOCRYPT 2019*, volume 11898, pages 139–159.
74. Neng-Fa Zhou, Håkan Kjellerstrand, and Jonathan Fruhman. *Constraint solving and planning with Picat.* Springer, 2015.
75. Chunbo Zhu, Gaoli Wang, and Boyu Zhu. Fast chosen-key distinguish attacks on round-reduced AES-192. In *ACISP 2019*, volume 11547, pages 573–587.

Supplementary Material

A `AES` Differential Characteristics

Table 6: Differential characteristic used in the distinguisher of 14 rounds of `AES`-256.

Round	State differences	Key differences
Plaintext	`8E474700 00000000 8E474700 00??0000`	
0	`00000000 00000000 00000000 00??0000` `00000000 00000000 A6C46262 00000000`	`8E474700 0C000000 8E474700 00000000`
1	`00C46262 A6000000 00000000 00000000` `00000000 ???????? ???????? ????????`	`00C46262 A6000000 A6C46262 00000000`
2	`???????? ???????? ???????? ????????` `??C462?? ????6262 00????00 0000????`	`8E474700 8E474700 00000000 00000000`
3	`??0000?? ????0000 00????00 0000????` `8E474700 00090000 00000000 00000000`	`00C46262 A6C46262 00000000 00000000`
4	`00000000 00090000 00000000 00000000` `A6C46262 00000000 00000000 00000000`	`8E474700 00000000 00000000 00000000`
5	`A6000000 A6000000 A6000000 A6000000` `8E4747C9 8E4747C9 8E4747C9 8E4747C9`	`00C46262 A6000000 A6000000 A6000000`
6	`00000000 00000000 00000000 00000000` `00000000 00000000 00000000 00000000`	`8E4747C9 8E4747C9 8E4747C9 8E4747C9`
7	`A6000000 00000000 A6000000 00000000` `8E4747C9 00000000 8E4747C9 00000000`	`A6000000 00000000 A6000000 00000000`
8	`00000000 00000000 00000000 00000000` `00000000 00000000 00000000 00000000`	`8E4747C9 00000000 8E4747C9 00000000`
9	`A6000000 A6000000 00000000 00000000` `8E4747C9 8E4747C9 00000000 00000000`	`A6000000 A6000000 00000000 00000000`
10	`00000000 00000000 00000000 00000000` `00000000 00000000 00000000 00000000`	`8E4747C9 8E4747C9 00000000 00000000`
11	`A6000000 00000000 00000000 00000000` `8E4747C9 00000000 00000000 00000000`	`A6000000 00000000 00000000 00000000`
12	`00000000 00000000 00000000 00000000` `00000000 00000000 00000000 00000000`	`8E4747C9 00000000 00000000 00000000`
13	`A6000000 A6000000 A6000000 A6000000` `??000000 ??000000 ??000000 ??000000`	`A6000000 A6000000 A6000000 A6000000`
Ciphertext	`??4747C9 ??000000 ??000000 ??000000`	`C94747C9 00000000 00000000 00000000`

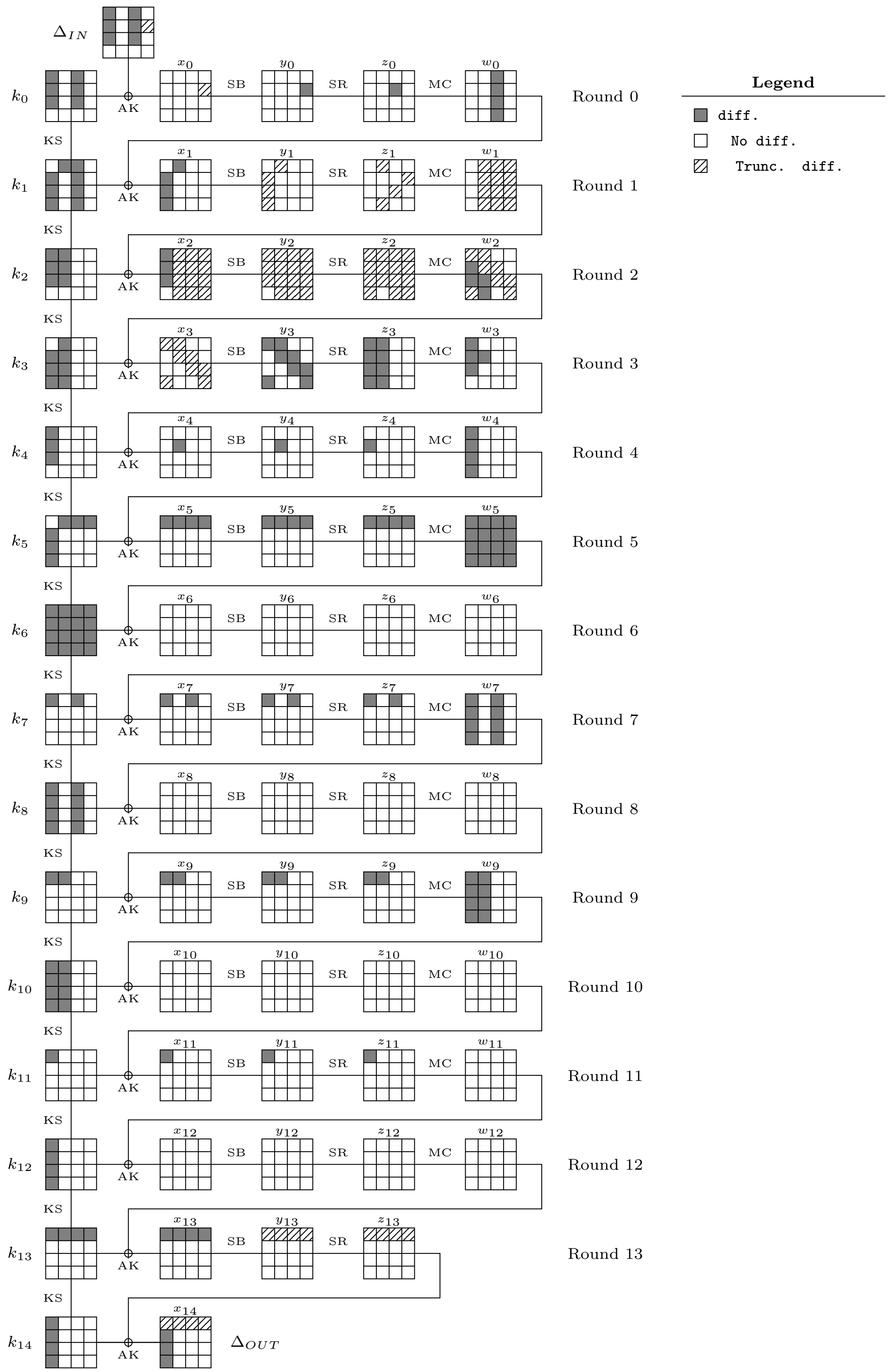


Fig. 12: Differential characteristic of 14-round AES-256 used in the distinguisher.

B Kiasu-BC Differential Characteristics

Table 7: Differential characteristics used in the distinguisher of 9 rounds of Kiasu-BC [27].

Round	State differences	Key differences
Plaintext	B3??0000 0000??00 28F47A?? ????0000	
0	00??0000 0000??00 000000?? ????0000 00000000 00000000 8EF47A7A ????????	B3000000 00000000 A6F47A7A 008E0000
1	00000000 00000000 28000000 ???????? ???????? ???????? ???????? ????????	00000000 00000000 A6F47A7A A67A7A7A
2	???????? ???????? ???????? ???????? ??0000?? ????7A7A 8E????7A 0000????	8E7A7A7A 8E7A7A7A 288E0000 8EF47A7A
3	??0000?? ????0000 00????00 0000???? 288E0000 8E7A7A7A 00000000 00000000	28000000 A67A7A7A 8EF47A7A 00000000
4	008E0000 00000000 008E0000 008E0000 00000000 8EF47A7A 8EF47A7A 8EF47A7A	28000000 8E7A7A7A 008E0000 008E0000
5	00000000 008E0000 00000000 008E0000 8EF47A7A 00000000 8EF47A7A 00000000	00000000 8E7A7A7A 8EF47A7A 8E7A7A7A
6	00000000 008E0000 008E0000 00000000 8EF47A7A 8EF47A7A 00000000 00000000	8EF47A7A 008E0000 8E7A7A7A 00000000
7	00000000 008E0000 00000000 00000000 8EF47A7A 00000000 00000000 00000000	8EF47A7A 8E7A7A7A 00000000 00000000
8	00000000 008E0000 008E0000 008E0000 ???????? ???????? ???????? 00000000	38EF47A7A 008E0000 008E0000 008E0000
Ciphertext	???????? ???????? ???????? 787A7A7A	78F47A7A 787A7A7A 78F47A7A 787A7A7A

Table 8: The key pair conforming to differential characteristic used in the 9-round Kiasu-BC distinguisher Kiasu-BC [27].

Round	k	k'	$k \oplus k'$
0	BD219F91 37EBDD3C 623F76DB 34AD0BBB	0E219F91 37EBDD3C C4CB0CA1 34230BBB	B3000000 00000000 A6F47A7A 008E0000
1	290A7589 1EE1A8B5 7CDEDE6E 4873D5D5	290A7589 1EE1A8B5 DA2AA414 EE09AFAF	00000000 00000000 A6F47A7A A67A7A7A
2	A40976DB BAE8DE6E C6360000 8E45D5D5	2A730CA1 3492A414 EEB80000 00B1AFAF	8E7A7A7A 8E7A7A7A 288E0000 8EF47A7A
3	CE0A75C2 74E2ABAC B2D4ABAC 3C917E79	E60A75C2 D298D1D6 3C20D1D6 3C917E79	28000000 A67A7A7A 8EF47A7A 00000000
4	47F9C329 331B6885 81CFC329 BD5EBD50	6FF9C329 BD6112FF 8141C329 BDD0BD50	28000000 8E7A7A7A 008E0000 008E0000
5	0F839053 3C98F8D6 BD573BFF 000986AF	0F839053 B2E282AC 33A34185 8E73FCD5	00000000 8E7A7A7A 8EF47A7A 8E7A7A7A
6	2EC7E930 125F11E6 AF082A19 AF01ACB6	A033934A 12D111E6 21725063 AF01ACB6	8EF47A7A 008E0000 8E7A7A7A 00000000
7	1256A749 0009B6AF AF019CB6 00003000	9CA2DD33 8E73CCD5 AF019CB6 00003000	8EF47A7A 8E7A7A7A 00000000 00000000
8	F152C42A F15B7285 5E5AEE33 5E5ADE33	7FA6BE50 F1D57285 5ED4EE33 5ED4DE33	8EF47A7A 008E0000 008E0000 008E0000
9	544F0772 A51475F7 FB4E9BC4 A51445F7	2CBB7D08 DD6E0F8D 83BAE1BE DD6E3F8D	78F47A7A 787A7A7A 78F47A7A 787A7A7A

Fig. 13: A pair conforming to the 9-round Kiasu-BC distinguisher.

Table 9: Differential characteristics used in the distinguisher of 10 rounds of `Kiasu-BC`. The two lines of state differences are the respectively the state difference after `AddRoundKey` and after `MixColumn`.

Round	**State differences**	**Key differences**
Plaintext	920000?? ??DBE0E0 3B??E000 0000??00	
0	000000?? ??000000 00??0000 0000??00 00000000 ???????? 00000000 00000000	920000E0 A9DBE0E0 3B85E000 00000000
1	920000E0 ???????? 005E0000 005E0000 ???????? ???????? ???????? ????????	920000E0 3BDBE000 005E0000 005E0000
2	???????? ???????? ???????? ???????? ????E0?? ??????00 00??E0?? ??00????	3B0000E0 00DBE0E0 0085E0E0 00DBE0E0
3	????E0?? ??????00 00??00?? ??00???? 1C85E0E0 00DBE0E0 00000000 00000032	0085E0E0 005E0000 00DBE0E0 00000000
4	1C000000 00000000 00000000 00000032 0060E0E0 00000000 00000000 00000000	0085E0E0 00DBE0E0 00000000 00000000
5	00E50000 005E0000 005E0000 005E0000 3BDBE0E0 3BDBE0E0 3BDBE0E0 3BDBE0E0	0085E0E0 005E0000 005E0000 005E0000
6	005E0000 00000000 005E0000 00000000 00000000 3BDBE0E0 00000000 3BDBE0E0	3B85E0E0 3BDBE0E0 3B85E0E0 3BDBE0E0
7	005E0000 005E0000 00000000 00000000 3BDBE0E0 00000000 00000000 3BDBE0E0	005E0000 3B85E0E0 00000000 3BDBE0E0
8	005E0000 00000000 00000000 00000000 00000000 00000000 00000000 3BDBE0E0	3B85E0E0 00000000 00000000 3BDBE0E0
9	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00000000 3BDBE0E0
Plaintext	3395EFD5 3395EFD5 3395EFD5 084E0F35	3395EFD5 3395EFD5 3395EFD5 084E0F35

C 13-round `Deoxys-BC`-384 LBD Attack

The Inbound part starts from Round 7 to Round 10 of Figure 14 that executes the following steps:

1. Deduce all the values of active bytes in x_7, x_8, x_9, and x_{10} by assessing `DDT`. Also acquire the active bytes in z_7, z_8, z_9, and z_{10}.

2. Assign a random value to $x_9[6]$ and compute all bytes of w_9. Obtain $STK_9[3,4,9,14] = w_9[3,4,9,14] \oplus x_{10}[3,4,9,14]$.

3. Obtain $STK_7[1] = w_7[1] \oplus x_8[1]$. Assign random values to $STK_7[4,14]$ and obtain the corresponding $STK_7[7,15]$.

4. Randomly choose $STK_8[1,2,3,4,5,9,10,11,13,14,15]$ and deduce the values of $STK_8[0,6,7,8,12]$ respectively.

5. Deduce the remaining key bytes STK_7 after full w_7 and x_8 are computed.
6. Solve the system of linear equations of STK_7, STK_8, and STK_9. With 36 out of 48 bytes are known, TA can find the 12 free bytes from the system, which gives enough degrees of freedom for the outbound phase.

D `Deoxys-BC` Differential Characteristics

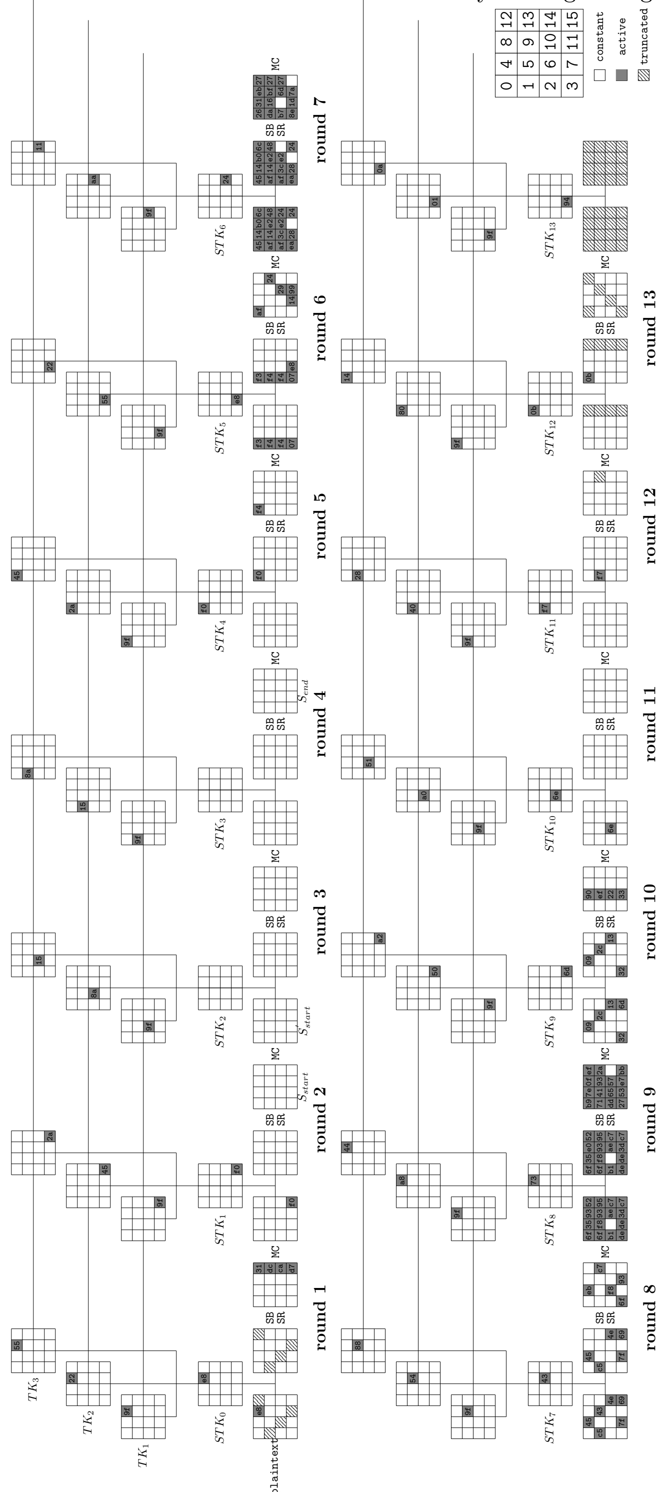


Fig. 14: LBD attack on 13-round Deoxys-BC-384