IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–43. DOI:XXXXXXXX

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

Xiaoyang Dong[3,5,6], Jian Guo[2], Shun Li[1,2,7(✉)] and Phuong Pham[2,4]

[1] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
lishun@ucas.ac.cn
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
guojian@ntu.edu.sg
[3] Tsinghua University, Beijing, P.R.China
xiaoyangdong@tsinghua.edu.cn
[4] Huawei International Pte. Ltd., Singapore
pham.phuong@huawei.com
[5] Zhongguancun Laboratory, Beijing, P.R.China
[6] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, P.R.China
[7] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

Abstract. At CRYPTO 2020, Liu *et al.* demonstrated that many differentials on `Gimli` are, in fact, incompatible. Similar incompatibilities also arise in related-key differentials on `AES`, which are typically addressed in an ad-hoc manner by incorporating additional constraints into the searching models. However, such ad-hoc methods are insufficient to eliminate all incompatibilities and may still produce false positive related-key differentials. At CRYPTO 2022, a novel approach was introduced that combines a Constraint Programming (CP) tool with a triangulation algorithm to search for rebound attacks against `AES`-like hashings. In this paper, we extend and unify these techniques to develop a comprehensive related-key differential search model. Our model not only identifies valid related-key differentials for `AES` and similar ciphers, but also enables immediate verification of the existence of at least one key pair satisfying the differentials. Using this enhanced automatic tool, we discover new related-key differentials for full-round `AES-192`, `AES-256`, `Kiasu-BC`, and for round-reduced `Deoxys-BC`. Based on these findings, we present full-round limited-birthday chosen-key distinguishing attacks on `AES-192`, `AES-256`, and `Kiasu-BC`, as well as the first chosen-key distinguisher on reduced-round `Deoxys-BC`. Furthermore, we identify, for the first time, a limited-birthday distinguisher on 9-round `Kiasu-BC` with practical complexities.

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

1 Introduction

The concept of chosen-key distinguisher, as firstly introduced and discussed in [10], arises in the context of addressing security concerns associated with hash functions constructed from block ciphers. In such scenarios, the key material can be subjected to modification by potential attackers. In the realm of hash functions, it is imperative to ensure that the three fundamental security properties: resistance against collision attacks, preimage attacks, and second preimage attacks. Concerning `AES`, it is worth noting that known-key attacks [44] have already successfully penetrated the full round `AES`-128, as demonstrated

Licensed under Creative Commons License CC-BY 4.0.

in [18, 32, 34]. These attacks leverage statistical non-random properties, which can also be extended to known-key distinguishers applicable to Kiasu-BC [40]. However, it is important to emphasize that while these non-random properties pose a significant threat in the context of block ciphers, but they do not inherently compromise the three essential security properties of the corresponding hash functions.

1.1 Our Contributions

In this paper, we introduce a few chosen-key distinguishers based on related-key (truncated) differentials on AES-like block ciphers. The results are summarized in Table 1.

AES-192/256. Since Biryukov *et al.* [10] 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 have remained open for more than 10 years. At CRYPTO 2013, Fouque *et al.* [28] 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 [79]. In this paper, we close the 10-year gap in research on AES-192 by introducing the first full 12-round limited birthday chosen-key distinguishing attack. In addition, extending the work of Biryukov *et al.*[10], we have successfully distinguished the full 14 rounds of AES-256 once more in the chosen-key setting.

Kiasu-BC. In the open-key setting, the designers stated:

> "*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 introducing chosen-key distinguishing attacks on (practical) 9-round and full 10-round Kiasu-BC. The 10-round attack is based on our newly discovered 10-round related-key differential on Kiasu-BC. Although the 10-round differential cannot lead to valid attacks on full AES-128, we succeed in distinguishing the full 10-round Kiasu-BC with the aid of the additional degrees of freedom of its tweak.

Deoxys-BC. We consider Deoxys-BC against open-key attacks for the first time 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, as claimed by the designers [42]. Note that the full 14-round AES-256 has been distinguished by Biryukov *et al.* [10] and our paper. It is important to note that prior research on Deoxys-BC with more rounds primarily focused on related-tweakey boomerang distinguishers, which differ from our model. While previous studies achieved distinguishers for up to 11 rounds, our proposed model extends this capability, enabling a distinguisher attack on 13 rounds.

1.2 Methodological Advances and Novelty

We introduce a principled, end-to-end methodology that advances the state of the art in automatic cryptanalysis for AES-like ciphers under open-key assumptions:

- **Unified CP-based framework with rebound integration.** We embed rebound degrees of freedom (inbound starting points and outbound feasibility) directly into a single constraint-programming (CP) model. This integration is novel: it natively captures inbound/outbound coupling, key/tweak activity, and input/output truncated

Table 1: Summary of analysis results on AES and others, where KK denotes known-key, CK denotes chosen-key, RK denotes related-key, and MC denotes multicollision.

Target	Settings	Rounds	Time	Memory	Ideal	Ref.
AES-128	MMO Collision	6/10	2^{56}	2^{32}	2^{64}	[33]
		6/10	2^{48}	2^{32}	-	Sect. 4
	KK distinguishing	7/10	2^{56}	-	2^{58}	[44]
		7/10	2^{24}	2^{16}	2^{64}	[52]
		8/10	2^{48}	2^{32}	2^{64}	[33]
		10/10	2^{64}	2^{64}	-	[32]
		10/10	$2^{59.6}$	$2^{58.8}$	-	[18]
		12/10	2^{66}	2^{64}	-	[34]
	CK distinguishing	7/10	2^{8}	2^{8}	2^{64}	[22]
		8/10	$2^{13.4}$	2^{16}	$2^{31.7}$	[39]
		9/10	2^{55}	2^{32}	2^{68}	[28]
AES-192	CK distinguishing	8/12	1	2^{16}	-	[79]
		9/12	1	2^{16}	-	[79]
		12/12	2^{100}	2^{32}	2^{108}	Sect. 4
	RK key-recovery	12/12	2^{124}	$2^{79.8}$	2^{192}	[21]
	RK key-recovery	13/12	2^{170}	$2^{133.5}$	2^{192}	[75]
AES-256	CK distinguishing	9/14	2^{24}	2^{16}	2^{64}	[22]
		9/14	2^{49}	2^{33}	2^{128}	[8]
		14/14	2^{88}	2^{32}	2^{94}	Sect. 4
	MC distinguishing	14/14	$q \cdot 2^{67}$	-	$q \cdot 2^{\frac{q-1}{q+1}128}$	[10]
	RK key-recovery	14/14	$2^{99.5}$	2^{77}	2^{256}	[9]
Kiasu-BC	RK distinguishing	3.5/10	-	-	-	[24]
		4/10	2^{32}	-	-	[23]
		6/10	-	-	-	[24]
	CK distinguishing	9/10	2^{36}	2^{14}	2^{68}	Sect. 5
		10/10	2^{67}	2^{14}	2^{96}	Sect. 5
Deoxys-BC-256	RK distinguishing	9/14	2^{122}	-	2^{128}	[15]
		9/14	$2^{120.4}$	-	2^{128}	[74, 76]
	RK key-recovery	11/14	$2^{249.9}$	$2^{128.2}$	$\geq 2^{253}$	[76]
	CK distinguishing	10/14	2^{69}	-	2^{101}	Sect. 6
Deoxys-BC-384	RK distinguishing	11/16	2^{120}	-	2^{128}	[15]
		11/16	$2^{118.4}$	-	2^{128}	[77]
	RK key-recovery	14/16	$2^{286.2}$	2^{136}	$\geq 2^{287}$	[76]
		15/16	$2^{371.7}$	2^{128}	$\geq 2^{384}$	[70]
	CK distinguishing	13/16	2^{42}	-	2^{62}	Sect. 6

spaces within one optimization, yielding a more expressive and efficient search over compatible trails than prior forward/backward partial extensions or ad-hoc filters.

- **Algebraic reparameterization of the AES key schedule for ultra-low probabilities.** Building on triangulation [43] and the Leurent–Pernot key-schedule representation [48], we develop an algebraic view that enables constructive validation of differential trails with probabilities below 2^{-128}. This allows efficient enumeration and filtering of key values satisfying deep key-schedule constraints—capabilities that were not feasible with earlier formulations.

- **Joint automation of key-schedule modeling and differential propagation.** Our toolchain, to the best of our knowledge, is the first to simultaneously model (tweak)key-schedule constraints and state-level differential propagation within one automated search loop. This joint treatment enables the discovery of practical chosen-key distinguishers and collision-style effects in hashing modes that were beyond previous known-key or manual techniques.

Key bridging and applications. Within this unified framework, we identify a new *key bridging* property for AES under the Leurent–Pernot representation, which substantially reduces the time to find conforming key pairs for related-key trails. As a direct application, we obtain a valid 10-round related-key differential on Kiasu-BC together with concrete key pairs (Table 8). Overall, the methodology leads to full-round chosen-key distinguishers for AES-192, AES-256, and Kiasu-BC, and to round-reduced distinguishers on Deoxys-BC.

2 Preliminaries

2.1 AES

AES [19] is a block cipher with a 4×4-byte state that operates on a 128-bit block, and supports three different key sizes: 128, 192, and 256 bits, which correspond to three versions: AES-128, AES-192, and AES-256. Although the round functions of all versions are the same, the number of execution rounds differs and is 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:

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

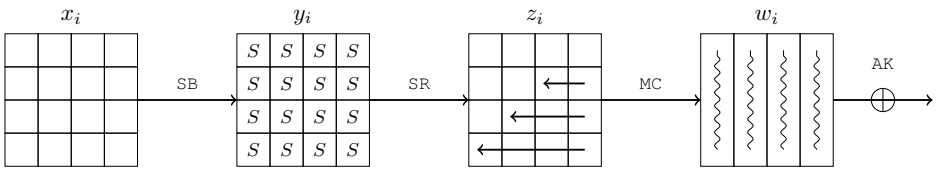


Figure 1: The round function of AES.

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

2.2 Kiasu-BC and Deoxys-BC

In general, a block cipher with n-bit block size is a family of permutations that operates on two inputs: a randomly generated secret key and an n-bit message string, 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* [69, 49] 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 the key and the tweak are used to form a permutation. Nowadays, tweakable block ciphers have demonstrated versatility across a wide range of applications in cryptographic schemes, such as message-authentication codes [58], compression functions [27], (authenticated) encryption schemes [61, 45], or variable-input-length ciphers [55]. At ASIACRYPT 2014, Jean *et al.* [41] proposed the TWEAKEY framework with the purpose of unifying the treatment of the key and the tweak in a cipher. 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). Based on the `TWEAKEY` framework, various tweakable block ciphers are constructed, such as `Deoxys-BC`, `Kiasu-BC`, and `Joltik-BC` and `SKINNY` [6], where `Deoxys-BC` and `SKINNY` have been standardized by ISO [1]. Research on these block ciphers is ongoing, with numerous new attacks and techniques being deployed. The *open-key* settings (i.e., `known-key` and `chosen-key` attacks) are naturally applicable to tweakable block ciphers, but they have been rarely studied in the cryptographic community.

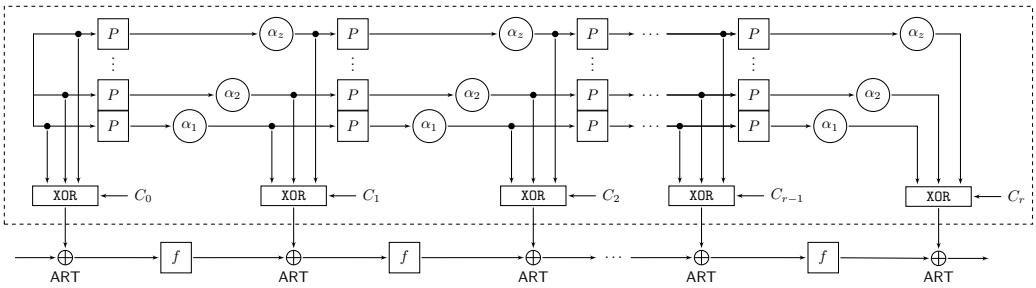


Figure 2: The `STK` construction [38]

`Kiasu-BC` [40] 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 round key addition. As a result, when the tweak is zero, `Kiasu-BC` becomes 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 can now be freely chosen by attackers. This is why a comprehensive investigation into `Kiasu-BC` is necessary to determine any potential negative effects. Previous cryptanalysis of `Kiasu-BC` includes square attack [23], impossible differential attack [24], meet-in-the-middle (MITM) attack [73], and boomerang attack [24, 5]. In the open-key setting, Bao *et al.* [4] 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 improvement was not achieved [40]. In this paper, we fill this gap in the cryptanalysis in open-key setting.

`Deoxys-BC` [42] follows the `STK` construction in Figure 2, using 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 third-party analysis by Cid *et al.* at ToSC 2017 [15]. By using Mixed Integer Linear Programming (MILP) to automatically search for related-key boomerang differential trails by incorporating linear incompatibility, they obtained 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, leveraging the Boomerang Connectivity Table (BCT) technique [14, 74], these distinguishers underwent refinement using an expanded search space that accounts for the generalised switching effect observed in the BCT of the S-box. There have been various key-recovery attacks on `Deoxys-BC`, including boomerang attacks [15, 65] and rectangle attacks [76, 26, 5].

2.3 Limited Birthday Distinguisher

At ASIACRYPT 2013, the limited birthday distinguisher (LBD) was formally introduced by Iwamoto *et al.* [37], although it has been used to distinguish `Whirlpool` [46] and `AES` [33]. At SAC 2013, Jean *et al.* further developed the LBD into a multiple limited birthday distinguisher [39]. In this section, the definition and solution of the limited birthday problem and distinguisher are discussed.

Definition 1. (Limited birthday problem). Let $f : \{0,1\}^n \to \{0,1\}^n$ be a permutation, and let $\Delta_{IN}, \Delta_{OUT} \subset \{0,1\}^n$ be two vector spaces of dimensions d_{in} and d_{out} respectively. Find a pair of inputs (x, x') for which $x \oplus x' \in \Delta_{IN}$ 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 [33] has been proven to match the lower bound complexity for the limited birthday problem in [37] for a black-box function and in [36] for a black-box permutation. We refer to [33] 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 the related-key setting). Let E_K be a block cipher: $\{0,1\}^n \to \{0,1\}^n$, where the 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 aims 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 Open-Key Settings and Related-Key Attacks

In traditional block cipher attacks, the attacker lacks the knowledge or control over the key material. However, when attacking hash function, particularly its block cipher-based compression function, the attacker has 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 [44], 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.* [52] at SAC 2009, who improved the 7-round known-key distinguisher on `AES` with rebound attacks [53]. At FSE 2010, Gilbert and Peyrin [33] 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 ASIACRYPT 2014, Gilbert [32] proposed the first known-key distinguisher for the full `AES-128`. At ACISP 2017, Cui *et al.* [18] also distinguished the full `AES-128` using a statistical integral approach. Recently, Grassi and Rechberger [34] revisited Gilbert's known-key distinguisher and extended it to even 12 rounds on `AES-128`. Over the past decade, known-key distinguishers have also been successfully applied to `PRESENT` [47, 12] and various other important primitives [56, 64, 68, 66, 60, 2]. Furthermore, the known-key setting has also been explored from the perspective of provable security [2, 16, 54, 17].

In addition to the known-key setting, at CRYPTO 2009, the *chosen-key setting* was first introduced by Biryukov, Khovratovich, and Nikolic [10], where the focus was on the

distinguishing attacks performed with specially selected keys. They distinguished the full AES-256 by constructing a q-multicollision in time $q \cdot 2^{67}$, which is significantly lower than the ideal-case complexity of $q \cdot 2^{\frac{q-1}{q+1}128}$. At ASIACRYPT 2009 [46], Lamberger *et al.* built a subspace distinguisher on full Whirlpool, where the distinguishing attack relies on the problem of finding t input pairs to a function f such that their output differences lie within a vector subspace of the image space of f. At INDOCRYPT 2012, Derbez *et al.* [22] introduced practical chosen-key distinguishers for AES-128 up to 8 rounds and AES-256 with 9 rounds, which were later enhanced by applying multiple limited-birthday distinguishers [39]. At ASIACRYPT 2013, Iwamoto *et al.* [37] introduced the concept of a limited birthday distinguisher, which differs from a subspace distinguisher in that it employs fully defined sets IN and OUT. Based on this concept, they built several distinguishing attacks on reduced AES in the chosen-key setting. At CRYPTO 2013, Fouque *et al.* [28] 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.* [79] proposed practical chosen-key distinguishers on 9-round AES-192.

The related-key attacks are first introduced by Biham [7] 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 ciphers, 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 [11] introduced a branch-and-bound method to search for the related-key differentials of AES and others. At CRYPTO 2013, Fouque *et al.* [28] 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 [13, 43] incorporating the key of AES, there are several tool-based automatic models on searching related-key differentials of AES. In 2014, Minier *et al.* [57] searched the truncated differentials of AES with constraint programming (CP). In 2016, Gérault *et al.* [31] proposed CP-based model to search the chosen-key differential attacks on AES. Due to their efficiency, CP tools have been widely used to search (related-key) differentials [29, 72, 30] recently.

At ToSC 2018, Sun *et al.* [71] utilize a SAT tool to improve the accuracy of differential cryptanalysis on LED [35] and Midori [3]. At CRYPTO 2020, Liu *et al.* [50] 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 several works [11, 28]. In simpler terms, these incompatibilities within the differential characteristics are attributable to the propagation of nonlinear layers. For example, with AES, when you consider the input and output of an active S-box, you'll find that the valid values for both the input and output are limited to at most four distinct values. This limitation significantly reduces the degrees of freedom within the corresponding state. In situations where the active S-boxes are densely populated, particularly in instances involving multiple consecutive rounds with over a dozen active S-boxes each, this restriction can render the characteristics invalid. To deal with the incompatibilities, most of the previous works use ad-hoc methods [28], algebraic methods[20, 51], or add constraints [30] to the model to bypass the incompatible characteristics. However, it is hard to avoid all the incompatibilities when modeling the search. Biryukov *et al.* [43] introduced the triangulating algorithm (TA) to efficiently derive collisions in Rijndael-based compression function. At CRYPTO 2022, Dong *et al.* [25] combined the triangulating algorithm and the rebound attacks to assign compatible key values to bridge multiple inbound phases, which is named the *triangulating rebound attack*.

2.5 Triangulation Algorithm

The Triangulation Algorithm, introduced by Khovratovich *et al.* [43], is a Gaussian-elimination-like method for efficiently solving systems of equations over bijective functions.

This algorithm automatically detects solutions for nonlinear systems by treating an AES-like block cipher as a collection of equations encompassing both round functions and key schedule operations.

Algorithm Overview. The algorithm models cipher operations as a system where both state bytes and key bytes are variables. Starting with n initial state bytes and k initial key bytes, imposing constraints on m bytes yields $n + k - m$ degrees of freedom, termed "free variables".

Illustrative Examples. Consider the simple equation $y \oplus F(x) = 0$ where F is bijective. Here, x can be assigned randomly, completely determining y. For more complex systems:

$$\begin{cases} F_1(z) \oplus F_2(y) \oplus F_3(x) = 0 \\ G_1(y) \oplus G_2(x) = 0 \end{cases}, \tag{1}$$

since z appears only in the first equation, either y or x can be chosen freely, determining the remaining variables through the bijective functions.

This principle extends to larger systems. For instance:

$$\begin{cases} E_1(w \oplus z) \oplus E_2(z \oplus y) \oplus E_3(y) \oplus E_4(x) = 0 \\ F_1(z) \oplus F_2(y) \oplus F_3(x) = 0 \\ G_1(y) \oplus G_2(x) = 0 \end{cases}. \tag{2}$$

The algorithm systematically identifies variables appearing in exactly one equation, ensuring deterministic resolution once other variables in that equation are fixed.

Algorithm for AES-like Systems. For AES-like ciphers, the triangulation process follows these steps:

1. **System generation**: Model each cell as a variable with predefined values as constants. For one AES round with cells x_i^r (after AddRoundKey), y_i^r (after MixColumn), and k_i^r (subkey), generate 48 equations of four types:

$$\begin{cases} 2 \cdot S(x_0^r) + 3 \cdot S(x_5^r) + S(x_{10}^r) + S(x_{15}^r) = y_0^r \quad \text{(MixColumn)} \\ y_0^r + k_0^r = x_0^{r+1} \quad \text{(AddRoundKey)} \\ k_0^r + S(k_{13}^r) + \texttt{rcon}_r = k_0^{r+1} \quad \text{(KeySchedule)} \\ k_4^r + k_0^{r+1} = k_4^{r+1} \quad \text{(Key expansion)} \end{cases} \tag{3}$$

   where $\texttt{rcon}_r$ represents predefined round constants.

2. **Initialization**: Mark all variables and equations as "unprocessed".

3. **Variable identification**: Identify variables appearing in exactly one unprocessed equation. Mark these variables and their corresponding equations as "processed". If no such variables exist, terminate.

4. **Iteration**: Repeat Step 3 until no unprocessed equations remain.

5. **Free variable extraction**: Return all remaining unprocessed variables as "free variables".

Solution Construction. Once free variables are identified, random values can be assigned to them, and all other variables are uniquely determined through the equation relationships. This systematic approach ensures efficient solution generation for the constraint system underlying differential characteristic instantiation.

2.6 The Rebound Attacks

The rebound attack was first introduced by Mendel *et al.* in [53]. The attack consists of two phases: an inbound phase and an outbound phase. The internal block cipher or permutation F is split into three sub-parts: $F = F_{fw} \circ F_{in} \circ F_{bw}$.

- **Inbound phase**. In the inbound phase, the attackers use the MITM 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 $1/p$.

2.7 Rebound as a Technique vs. Cryptanalytic Objectives

Rebound is a construction technique (or analysis strategy), not an end in itself. It can be utilized under different attack settings (secret-key, related-key, known-key, chosen-key; with or without tweak) and can support multiple *objectives*:

- **Distinguishers**: show that a primitive deviates from an ideal random permutation/-function (e.g., structural, differential, or limited-birthday distinguishers).
- **Key-recovery attacks**: recover (parts of) the secret key under some setting (e.g., related-(tweak)key).
- **Collision/near-collision finding**: especially for block-cipher–based compression functions in standard hashing modes (DM/MMO/MP).
- **Other outcomes**: forgeries/structural properties in authenticated-encryption or tweakable settings.

What rebound *is*. A way to construct conforming pairs by trading low-probability middle constraints for meet-in-the-middle feasibility (inbound), then extending along differential trails (outbound). It is orthogonal to the attack goal and to the adversarial setting.

What rebound *is not*. It is not synonymous with a distinguisher, nor does it automatically imply key recovery. The same technique can underpin either a distinguisher, a collision attack on a hashing mode, or a key-recovery attack—depending on the setting, degrees of freedom, and how inbound/outbound are arranged.

Where distinguishers fit. A *distinguisher*—including limited-birthday distinguishers—only aims to separate the primitive from an ideal random permutation/function. It does not, by itself, recover keys. In open-key contexts relevant to hashing or tweakable block ciphers, chosen-key (or related-(tweak)key) distinguishers are natural and meaningful; rebound is one effective method to realize them by generating enough compatible pairs.

Implications for this work. Our use of rebound (combined with automatic search and algebraic key-schedule handling) is a means to construct trails and conforming pairs. The resulting *chosen-key distinguishing attacks* are precisely distinguishers in the open-key setting: they demonstrate non-random behaviour of AES-like ciphers (`AES`, `Kiasu-BC`, `Deoxys-BC`) under adversarial control of key/tweak material. In some settings, similar constructions can be adapted to yield collision-style results in hashing modes; in others, they remain distinguishers by design. Both outcomes are valid, and the chosen-key distinguisher should be interpreted strictly as a separability result, not as a claim of key recovery.

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.

The 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, and $\mathcal{C}$ is a set of constraints including relations between variables. Once an objective function is defined, the CSP becomes a Constraint Optimization Problem (COP).

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 [11, 28]. Step 1 searches for all truncated differential trails under a given bound on the number of rounds and active S-Boxes. Step 2 examines whether the truncated differential characteristics are valid, and finds the actual differential characteristic that maximizes the probability. Both steps can be approached through CP.

3.1 Two-Step Search Methodology

Our constraint programming approach employs a two-step architecture that addresses the scalability challenges of finding optimal related-key differential trails:

- **Step 1 (Truncated Search)**: Uses binary activity indicators $\Delta x_r[i,j] \in \{0,1\}$ to find structurally valid truncated differential trails. Implemented in MiniZinc with Picat-SAT solver.

- **Step 2 (Instantiation)**: Upgrades to exact byte differences $\Delta x_r[i,j] \in \{0,1,\ldots,255\}$ and finds concrete differential characteristics with DDT compatibility. Implemented using Choco constraint solver.

The two-step decomposition enables efficient search space pruning: Step 1 operates on binary patterns (reducing domain size from 2^8 to $\{0,1\}$ per byte), while Step 2 handles exact probability computation and value assignment.

Following developments in [57, 30], the search model can be described as Model (4), consisting of the round function components (`SB`, `AK`, `SR`, `MC`) and `KeySchedule`, with objective to minimize active S-boxes and maximize differential probability.

$$\begin{cases} 1.\ \text{Objective function: minimize total active Sboxes} \\ \quad obj = \sum \Delta x_r[i,j] + \sum \Delta k_r[i,n-1] \\ 2.\ \text{Constraints on } \texttt{SB}\text{:} \\ \quad \Delta x_r[i,j] = \Delta y_r[i,j], \quad \Delta SK_r[i] = \Delta k_r[i,n-1] \\ 3.\ \text{Constraints on } \texttt{AK}\text{:} \\ \quad \Delta w_r[i,j] + \Delta k_{r+1}[i,j] + \Delta x_{r+1}[i,j] \neq 1 \\ 4.\ \text{Constraints on } \texttt{SR}\text{:} \\ \quad \Delta y_r[i,(i+j) \bmod n] = \Delta z_r[i,j] \\ 5.\ \text{Constraints on } \texttt{MC}\text{: MDS property} \\ \quad (\sum_{i=0}^{n-1} \Delta z_r[i,j] + \sum_{i=0}^{n-1} \Delta w_r[i,j]) \in \{0, n+1, n+2, \ldots, 2n\} \\ 6.\ \text{Constraints on } \texttt{KeySchedule}\text{:} \\ \quad \Delta k_{r+1}[i,0] + \Delta k_r[i,0] + \Delta SK_r[(i+1) \bmod n] \neq 1 \\ \quad \Delta k_{r+1}[i,j] + \Delta k_{r+1}[i,j-1] + \Delta k_r[i,j] \neq 1 \end{cases} \tag{4}$$

Model (4) provides the structural backbone for both steps: Step 1 uses binary activity variables and feasibility filters, while Step 2 upgrades to byte-wise differences and probability scoring.

3.1.1 Step 1: Truncated Trail Search.

Step 1 enforces the following constraints at the truncated level:

- **Structural constraints**: SubBytes abstraction ($\Delta_{\text{out}} = \Delta_{\text{in}}$), ShiftRows/MixColumns propagation with MDS bounds
- **Activity minimization**: Sliding window optimization over consecutive rounds
- **Complexity guards**: Ideal-vs-dedicated complexity bounds, inbound freedom adequacy
- **Tweakey constraints** (Deoxys): Lane coverage via selectors ensuring valid tweakey schedule

We modified the tools from [30] with additional constraints in Model (5) for `AES` and `Kiasu-BC`, where Len_{Key} is set to 192, 256, and 192 bits respectively.

$$\begin{cases} \text{Objective: } obj = \sum_{r=4}^{Nr-2} \Delta x_r[i,j] + \text{fixed bytes in round 2} \\ \text{Key budget: } \sum \Delta k_r[i,n-1] \leq Len_{Key} \\ \text{Ideal complexity bound: idealcase} \geq \text{attack complexity} \\ \text{Inbound freedom: } Len_{Key} - \sum \Delta k_r[i,n-1] + \text{freedom} \geq \text{outbound cost} \end{cases} \tag{5}$$

Remark 1. Model (5) adopts a $2+2+x$ round partitioning strategy for the rebound, driven by specific imbalances that restrict the feasible configurations to either $2+2+x$ or $x+2+2$ rounds. While we also evaluated the $x+2+2$ configuration, it did not yield superior results. In the $2+2+x$ layout, the middle 2 rounds are dedicated to the Super-SBox technique.

Beginning from these middle 2 rounds, the forward extension (the trailing $+x$ rounds) primarily facilitates *exact* differential characteristic propagation, which scales better as more rounds are appended. In contrast, the two-round *backward* extension is intentionally kept short and uses a mix of *truncated* and *definite* difference propagation. This asymmetric design maintains a distinguisher that still outperforms the ideal bound. Concretely,

attempting to push the backward part to 3 rounds from z_1 typically invalidates the LBD (the two successive `MixColumn` layers inflate a single truncated input into a fully active Δ_{IN}), whereas restricting the backward side to 2 rounds preserves a valid distinguisher with about a 2^8 advantage (as illustrated in Figure 7). This observation is consistent with the ideal-case lower bound in Theorem 1: once Δ_{IN} becomes fully active, the ideal complexity quickly dominates and any dedicated LBD becomes ineffective.

Therefore, the $2 + 2 + x$ split is not only theoretically motivated by linear-diffusion growth, but also serves as a practical heuristic to search for the *longest* valid distinguishers: two rounds backward (mixed truncated/definite) to preserve the advantage, and x rounds forward (exact propagation) to extend the trail.

For `Deoxys-BC`, tweakey constraints use lane selectors:

$$\begin{cases} \Delta TK_r[f(r,m) \bmod 4, f(r,m) \operatorname{div} 4] \leq lane_m \\ \sum_{r=0}^{Nr-1} \Delta TK_r[f(r,m) \bmod 4, f(r,m) \operatorname{div} 4] \geq lane_m \cdot (Nr + 1 - regime) \end{cases} \tag{6}$$

3.1.2 Step 2: Characteristic Instantiation.

Step 2 upgrades binary variables to byte-wise differences and adds:

- **DDT constraints**: S-box input/output compatibility
- **Exact arithmetic**: Finite field operations for MixColumns
- **Probability optimization**: Minimize total attack complexity
- **Key bridging**: Integration of [48] technique for efficient key pair generation

More details could be found in Supplementary Material D and E. The key bridging technique transforms round keys into a new basis representation, enabling detection of distant S-box relationships and reducing key pair complexity to approximately 2^8 operations.

3.2 Implementation Workflow

The complete attack implementation follows this sequence:

1. **Cipher modeling**: Implement round function and key schedule in both MiniZinc and Java/Choco
2. **Truncated search**: Execute Step 1 with MiniZinc/Picat-SAT pipeline
3. **Instantiation**: Process Step 1 results with Choco solver for Step 2
4. **Pair generation**: Apply triangulation algorithm with key schedule representation
5. **Validation**: Verify generated pairs experimentally

Our implementation bridges the two steps through solution parsing, constraint translation, and incremental search capabilities. The complete source code is available at https://github.com/Lish10/Chosen_key_distinguishers.

Applicability. Our techniques apply to `AES`-like SPN designs with small S-boxes and complex key schedules. The chosen-key setting captures scenarios in block-cipher-based hashing and tweakable ciphers where adversaries control key material. Our distinguishers use explicit related-key differentials rather than statistical irregularities, making them applicable to hashing modes while providing realistic security assessment for open-key deployments.

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 that achieves a time complexity of only 2^{48} through the use of a new truncated differential trail. In contrast to the best collision attack on 6-round AES-128, as presented in [33], our new attack reduces the time complexity by a factor of 2^8. Additionally, two new distinguishing attacks on full-round AES-192 and AES-256 have been obtained through the improvement of an automatic search specialized in limited birthday distinguisher.

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

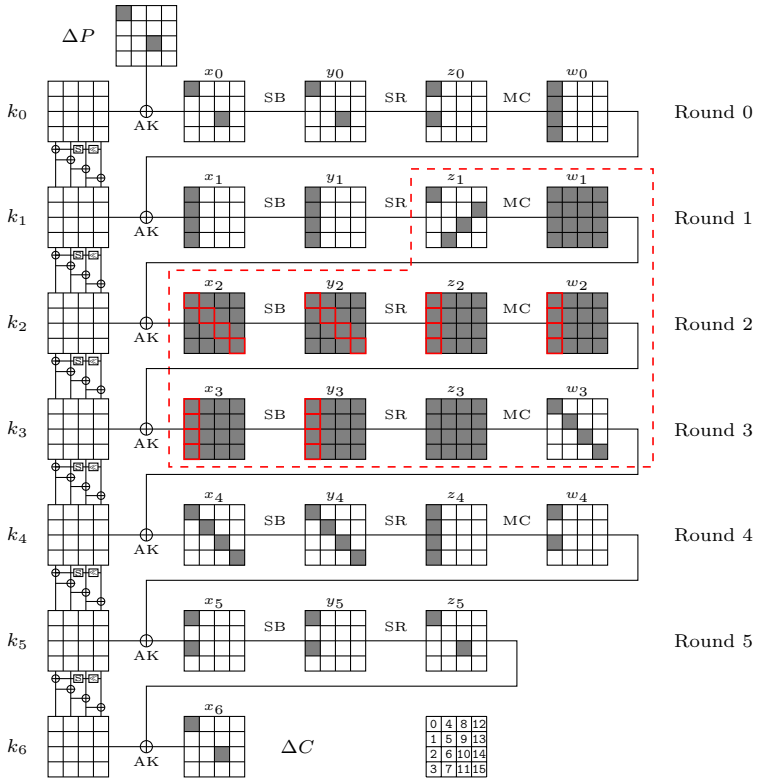


Figure 3: Collision attack on 6-round AES-128-MMO/MP.

The Super-Sbox Technique. The Super-Sbox technique was proposed simultaneously by Gilbert and Peyrin [33] and Lamberger *et al.* [46] with the goal of fulfilling two heavy round S-box layers, as shown in Figure 4 (a), but with low complexity. In [67], 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 phase) and to generate sufficient pairs to conform to the probability of the remaining rounds (outbound phase).

For the jth Super-Sbox $\texttt{SSB}_j$ and given input difference $\Delta x_i^{(j)}$ ($j = 0$ in Figure 4 (a)), compute all possible $\Delta y_{i+1}^{(j)} = \texttt{SSB}_j(x \oplus \Delta x_i^{(j)}) \oplus \texttt{SSB}_j(x)$ for all $x \in \{0,1\}^{32}$. Store the pair



(a) Super-Sbox (b) Non-full-active Super-Sbox

Figure 4: The Two-Round Differential

$(x, x \oplus \Delta x_i^{(j)})$ in a table $\mathbb{L}^{(j)}[\Delta y_{i+1}^{(j)}]$. With the given $\Delta y_{i+1}^{(j)}$, we find a pair conforming to 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 MC is utilized 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 $\Delta w_i = \texttt{MC}(\Delta z_i)$. Then, for a fixed input-output differences $(\Delta x_i^{(j)}, \Delta y_{i+1}^{(j)})$ of $\texttt{SSB}_j$, 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 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 $\texttt{MC} \cdot (z[1], \ldots, z[n])^T = (w[1], \ldots, w[n])^T$, 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.

As depicted in Figure 3, we focus on the first Super-Sbox $\texttt{SSB}^{(0)}$ 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 $\texttt{SSB}^{(0)}$ 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.

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 $\Delta x_2^{(0)}$ marked by red box and find the corresponding entry $\Delta x_3^{(0)}$ and the value $x_3^{(0)}$ in the list L_0.

The Super-Sboxes can be precomputed in parallel with a time complexity of 2^{32}. On average, one conforming pair is obtained for each difference pair $(\Delta w_3, \Delta z_1)$. Consequently, we iterate over 2^{16} possible values of Δw_3, and for each such value, try 2^{32} distinct values of Δz_1, yielding a total of 2^{32} conforming pairs per Δw_3. 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}.

Remark 2. While our current analysis focuses on AES-128, the methodology naturally extends to Rijndael-256-256, which features a $256 + 256$ internal state (compared to AES-128's $128 + 128$ state). This extension is particularly significant given NIST's announced plans to standardize Rijndael-256-256, potentially addressing some security limitations of AES-based hash constructions. Our CP-based search framework can be directly adapted to Rijndael-256-256 by extending the state variables from 4×4 to 4×8 bytes and adjusting the round function constraints accordingly.

However, the extension introduces new challenges: the larger state increases the computational complexity of the Super-Sbox technique, and the key schedule becomes more

complex with additional rounds and state dependencies. Our triangulation-based key pair validation techniques will require adaptation to handle the expanded key schedule structure. Nevertheless, the fundamental principles of our two-step search methodology—truncated trail optimization followed by concrete instantiation—remain applicable and may reveal new distinguisher patterns that were not apparent in the smaller AES variants.

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

We investigated the distinguishing characteristics of limited birthday distinguishers on the full rounds of 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 the 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 guided by an objective function minimizing the total time complexity, which includes the forward and backward outbound phases through the remaining active S-boxes under the following constraints:

- key schedule 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,
- the ideal case is higher than the attack complexity, *i.e.*, the critical part, the 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 6, while a full round LBD on AES-256 with attacking complexity of 2^{88} was obtained in Figure 11. The inbound phase covers y_1 to x_4.

4.2.1 The LBD Attack on Full AES-192.

The attack procedures for AES-192 are described as follows:

1. Find a key pair $(k, k \oplus \Delta_K)$ conforming to the key differential characteristic.
2. Loop over compatible differences for $\Delta x_4[8]$ and compute $\Delta y_3 = \mathtt{SR}^{-1}(\mathtt{MC}^{-1}(\Delta x_4 \oplus \Delta k_4))$.
3. Following the Super-Sbox technique in Section 4.1, 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. Randomly 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, we compute the pairs backward and forward to filter the one passing through all the remaining active S-boxes in the rest differential characteristic given in Table 3.

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

As shown in Table 3, the key differential (i.e., key differences in Table 3) is of probability 2^{-135}, which contains 20 active S-boxes (5 with a probability of 2^{-6} and 15 with a probability of 2^{-7}). Given a key pair satisfying the key differential, the probability of the outbound phase in the internal states (i.e., State differences in Table 3) 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 6,
- the probability 2^{-28} to pass the active S-boxes in Round 1 in Figure 6.

With the help of the triangulation algorithm, it is possible to derive a key pair with a time complexity of 2^{28} although the probability of the key differential trail is 2^{-135}.

Computing the conforming key pair with the triangulation algorithm. The key differential characteristic has 20 active S-boxes. Out of these, 12 active S-boxes (highlighted in blue in Figure 5) 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 5. We proceeded to fix an additional set of four active S-boxes within $K_6[20, 21, 22, 23]$ to their permissible values. This enabled us to deduce the relationships involving the bytes of $K_2[16, 17, 18, 19]$, $K_3[4, 5, 6, 7]$, and $K_4[16, 17, 18, 19]$, which we derived using the information presented in Table 2. Specifically, we could derive a constraint: $K_2[16, 17, 18, 19] \oplus K_3[4, 5, 6, 7] = c_0$, where c_0 represents a constant deduced from 12th and 13th equations in Table 2. Nevertheless, considering that the key values must still pass through the four remaining active S-boxes in $K_7[20, 21, 22, 23]$, each with a probability of 2^{-7}, it would require a total of 2^{28} attempts to unveil a single conforming key pair. One key pair is found and presented in Table 4.

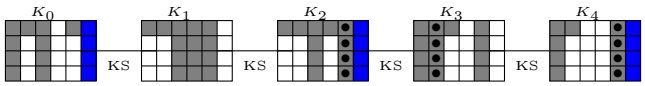


Figure 5: The selection of fixed and free bytes in the AES-192 subkeys.

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 5. The time complexity of Step 1 is $2^{61+28} = 2^{89}$. In total, $2^{61+7+32} = 2^{100}$ starting points are generated.

The total time complexity of the attack is approximately $2^{89} + 2^{100} \approx 2^{100}$ 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 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.

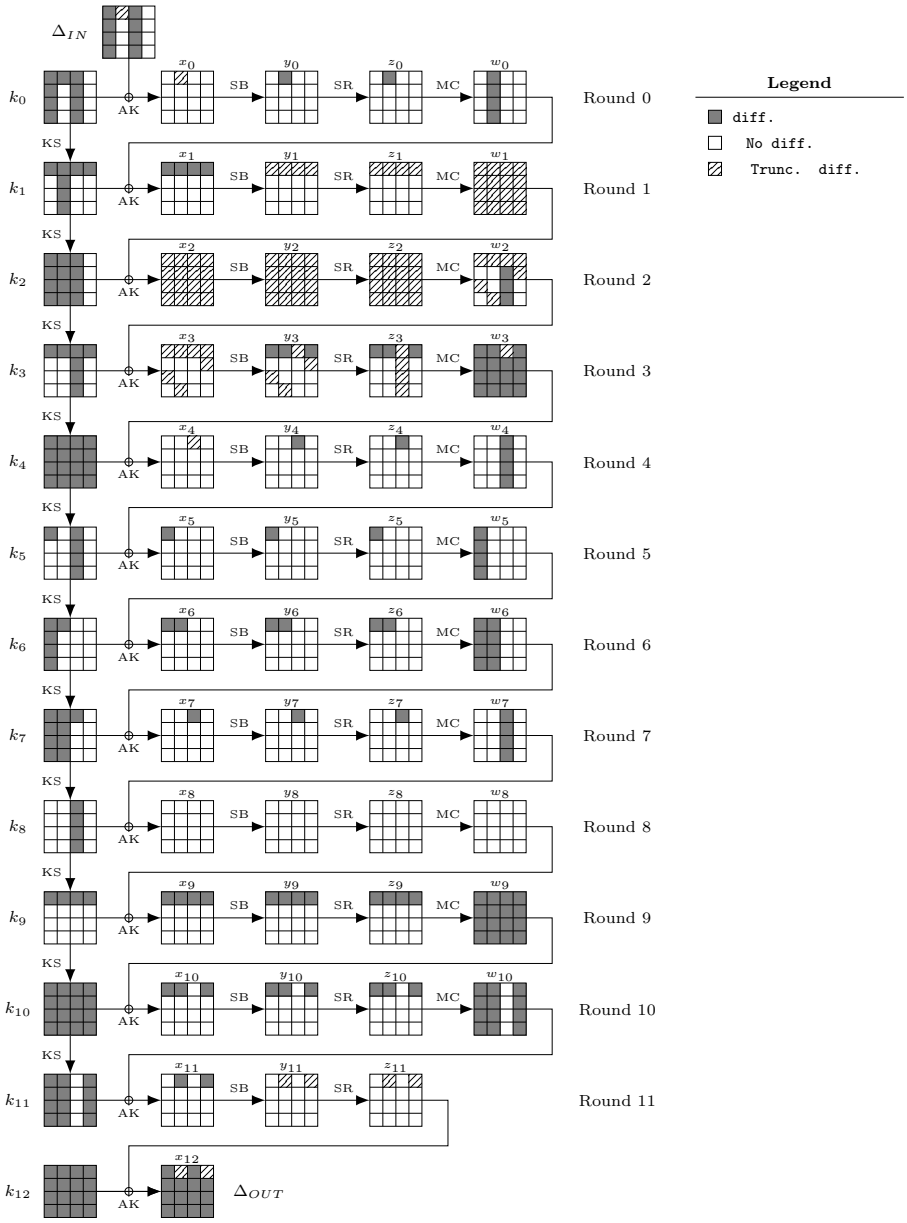


Figure 6: Differential characteristic of 12-round AES-192 used in the distinguisher.

Table 2: Represent subkey $K_6[20, 21, 22, 23]$ using fixed and free key bytes

1. $K_3[20, 21, 22, 23] = K_4[16, 17, 18, 19] \oplus K_4[20, 21, 22, 23]$
2. $K_3[16, 17, 18, 19] = K_2[20, 21, 22, 23] \oplus K_3[20, 21, 22, 23]$
$= K_2[20, 21, 22, 23] \oplus K_4[16, 17, 18, 19] \oplus K_4[20, 21, 22, 23]$
3. $K_3[12, 13, 14, 15] = K_2[16, 17, 18, 19] \oplus K_3[16, 17, 18, 19]$
$= K_2[16, 17, 18, 19] \oplus K_2[20, 21, 22, 23] \oplus K_4[16, 17, 18, 19] \oplus K_4[20, 21, 22, 23]$
4. $K_1[20, 21, 22, 23] = K_2[16, 17, 18, 19] \oplus K_2[20, 21, 22, 23]$
5. $K_1[16, 17, 18, 19] = K_0[20, 21, 22, 23] \oplus K_1[20, 21, 22, 23]$
$= K_0[20, 21, 22, 23] \oplus K_2[16, 17, 18, 19] \oplus K_2[20, 21, 22, 23]$
6. $K_2[12, 13, 14, 15] = K_1[16, 17, 18, 19] \oplus K_2[16, 17, 18, 19]$
$= K_0[20, 21, 22, 23] \oplus K_2[20, 21, 22, 23]$
7. $K_3[8, 9, 10, 11] = K_2[12, 13, 14, 15] \oplus K_3[12, 13, 14, 15]$
$= K_0[20, 21, 22, 23] \oplus K_2[16, 17, 18, 19] \oplus K_4[16, 17, 18, 19] \oplus K_4[20, 21, 22, 23]$
8. $K_4[12, 13, 14, 15] = K_3[16, 17, 18, 19] \oplus K_4[16, 17, 18, 19]$
$= K_2[20, 21, 22, 23] \oplus K_4[20, 21, 22, 23]$
9. $K_4[8, 9, 10, 11] = K_3[12, 13, 14, 15] \oplus K_4[12, 13, 14, 15]$
$= K_2[16, 17, 18, 19] \oplus K_4[16, 17, 18, 19]$
10. $K_4[4, 5, 6, 7] = K_3[8, 9, 10, 11] \oplus K_4[8, 9, 10, 11]$
$= K_0[20, 21, 22, 23] \oplus K_4[20, 21, 22, 23]$
11. $K_4[0, 1, 2, 3] = K_3[4, 5, 6, 7] \oplus K_4[4, 5, 6, 7]$
$= K_0[20, 21, 22, 23] \oplus K_3[4, 5, 6, 7] \oplus K_4[20, 21, 22, 23]$
12. $K_5[20, 21, 22, 23] = \texttt{rcon}_4 \oplus K_4[20, 21, 22, 23] \oplus K_4[16, 17, 18, 19] \oplus K_4[12, 13, 14, 15]$
$\oplus K_4[8, 9, 10, 11] \oplus K_4[4, 5, 6, 7] \oplus K_4[0, 1, 2, 3] \oplus \texttt{SubWord}(\texttt{RotWord}(K_4[20, 21, 22, 23]))$
$= \texttt{rcon}_4 \oplus K_2[16, 17, 18, 19] \oplus K_2[20, 21, 22, 23] \oplus K_3[4, 5, 6, 7]$
$\oplus \texttt{SubWord}(\texttt{RotWord}(K_4[20, 21, 22, 23]))$
13. $K_6[20, 21, 22, 23] = \texttt{rcon}_5 \oplus K_4[20, 21, 22, 23] \oplus K_4[12, 13, 14, 15] \oplus K_4[4, 5, 6, 7]$
$\oplus \texttt{SubWord}(\texttt{RotWord}(K_5[20, 21, 22, 23]))$
$= \texttt{rcon}_5 \oplus K_0[20, 21, 22, 23] \oplus K_2[20, 21, 22, 23] \oplus K_4[20, 21, 22, 23]$
$\oplus \texttt{SubWord}(\texttt{RotWord}(K_5[20, 21, 22, 23]))$

Table 3: 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 219D9DBC 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	4CE00998 ???D9424 40E00998 ???D9424	4CE00998 617D9424 40E00998 6D7D9424

4.2.2 Analysis of the AES-256 distinguisher.

The procedure for the AES-256 distinguisher attack is similar to that of AES-192, using the concrete differential characteristic specified in Table 5. The degrees of freedom are calculated from the number of possible key pairs and state pairs conforming to the truncated round Inbound 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.* [40] as a candidate in the CAESAR competition for authenticated encryption. The design of Kiasu-BC closely resembles that of the AES cipher, with the notable exception of the inclusion of a 64-bit tweak value. This tweak value is subjected to a bitwise XOR operation with the first two rows of the state in each round following the addition of the round-key. Thus, it can be seen as the simplest instance of the TWEAKEY framework [41], where an identical tweak is used for all round functions.

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 [28] 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 7 and the key pair found in [28] are utilized. Three S-boxes—two in Round 4 and one in Round 1—are fulfilled using the tweak cells. One pair found is shown in Figure 12 in Supplementary Material.

The Inbound Phase: As shown in Figure 7, the Inbound phase 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 7, the following steps are performed:

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

2. Assign compatible differences for $(\Delta x_3[0], \Delta x_3[3])$ and compute $(\Delta w_2[0], \Delta w_2[3]) = (\Delta x_3[0], \Delta x_3[3]) \oplus (\Delta k_3[0], \Delta k_3[3])$, $\Delta z_2^{(0)} = \texttt{MC}^{-1}(\Delta w_2^{(0)})$, and check whether the difference of $\Delta z_2^{(0)}$ is compatible with the tuple difference $(\Delta x_2[0], \Delta x_2[5], \Delta x_2[10], \Delta x_2[15])$ of the Super-Sboxes. 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 $(w_2^{(0)}, \Delta w_2^{(0)})$ satisfying the condition: $\texttt{SB}(w_2[3] \oplus k_3[3]) \oplus \texttt{SB}(w_2[3] \oplus \Delta w_2[3] \oplus k_3'[3]) = \Delta y_3[3]$.

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:

(a) 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 $w_3[0,1,2,3] = \texttt{MC}(\texttt{SR}(y_3[0,5,10,15]))$ and $tw[1] = w_3[1] \oplus k_4[1] \oplus x_4[1]$ for $x_4[1]$ deduced from DDT.

(b) With full state of w_2 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, $w_1[8] = \texttt{MC}^{-1}(z_1[8], w_1[9,10,11])$ and $tw[4] = w_1[8] \oplus k_2[8] \oplus x_2[8]$.

(c) 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 $w_3[12,13,14,15] = \texttt{MC}(\texttt{SR}(\texttt{SB}([x_3[12,1,6,11]])))$. Obtain $tw[7] = w_3[13] \oplus x_4[13] \oplus k_4[13]$ with $x_4[13]$ deduces from DDT.

(d) 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. This arises from the fact that each difference $\Delta z_2[i]$ can correspond to $\Delta x_2[5i]$ with a probability of 2^{-1}. However, it is essential to note that the matching value encompasses two possibilities, $z_2[i]$ and $z_2[i] \oplus \Delta z_2[i]$. 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 $x_3[12]$ is 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 [33].

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 there is 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.

5.2 The First 10-round Distinguisher

The 7-round AES-128 Related-key Differential Trail. As shown in Figure 8, 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 phase and 2^{-67} probability in the Outbound part.

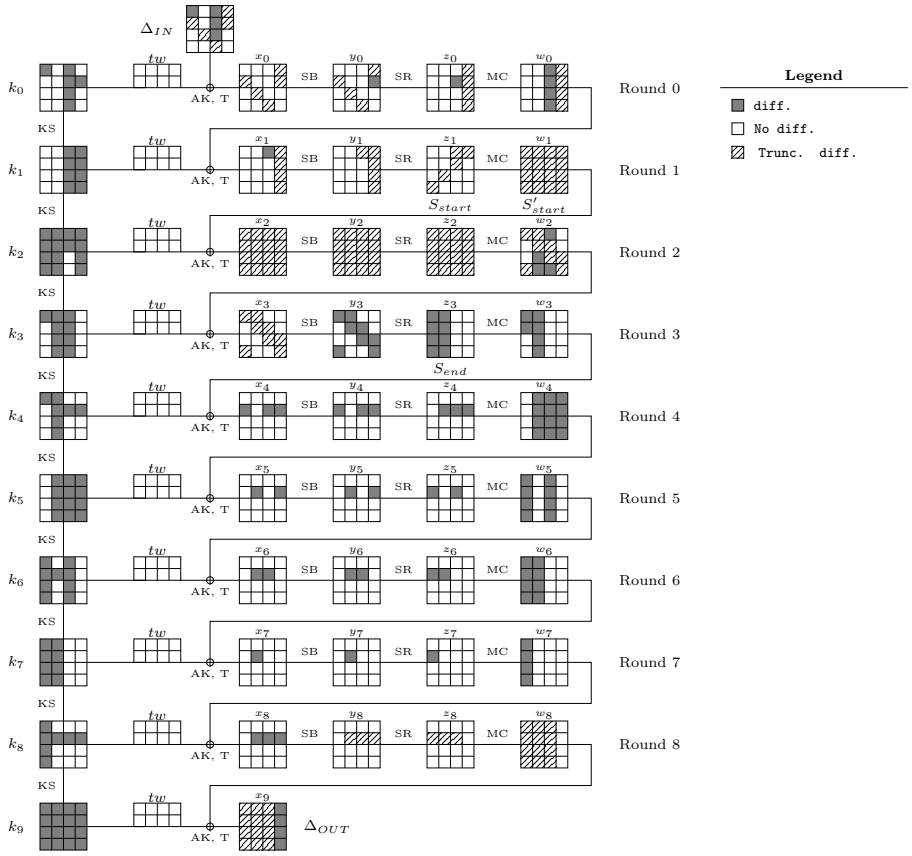


Figure 7: Differential characteristic of 9-round `Kiasu-BC` used in the distinguisher.

The Inbound Phase: Compared to the previous 9-round distinguisher, an additional round has been added to the Inbound phase 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 pairs of states and a tweak that conform 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`.

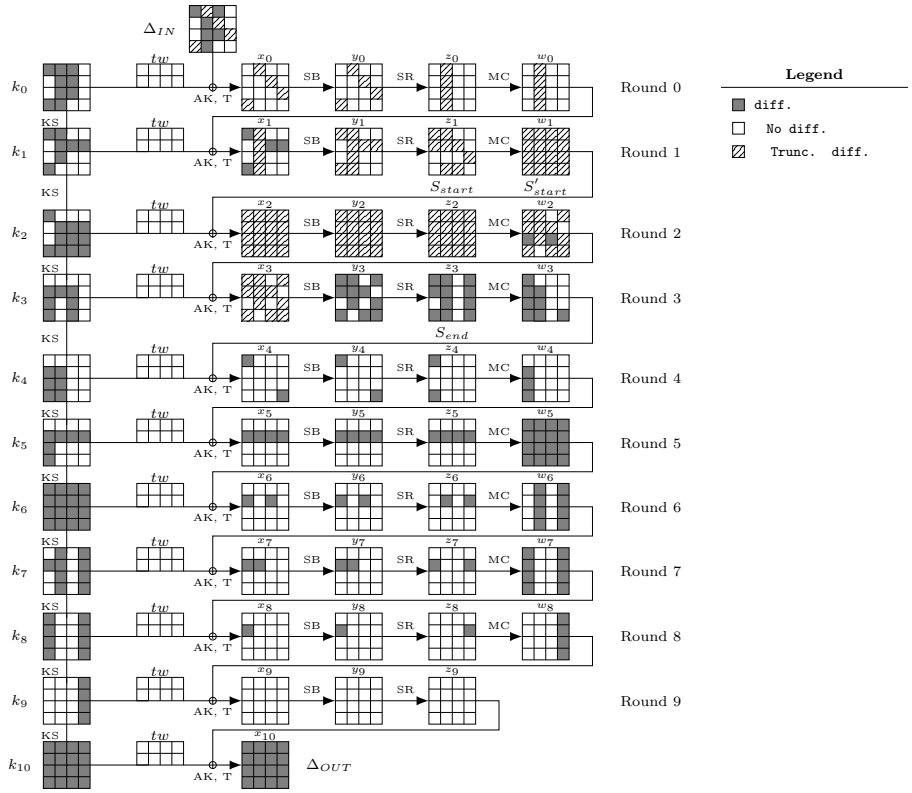


Figure 8: Differential characteristic of 10-round `Kiasu-BC` used in the distinguisher.

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 satisfying 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 Rebound Phases: 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, and 2^7 tuples each in list L_2 and 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. One key pair is found and presented in Table 8. 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}.

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.

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 changing linear tweakey introduces different constraints on the tweakey part. For Step 1 of the truncated trail search, we add the constraints of the linear incompatibility [15] between differential propagations in the tweakey and the state. We also impose the additional requirements that the freedom in 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 Chocosolver used in AES LBD search. With greater freedom from the tweakey, we can directly search for the rear partial differential characteristic and then go backward 4 rounds using the method outlined in [28]. We found a 10-round LBD on Deoxys-BC-256 in Figure 9 and an 13-round LBD on Deoxys-BC-384 in Figure 10 with a probability of 2^{-69} and 2^{-42} respectively.

6.2 Deoxys-BC Limited Birthday Distinguisher Attacks

In Figure 9, the attacking complexity is the product of the forward and backward outbound complexities. 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 10, 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

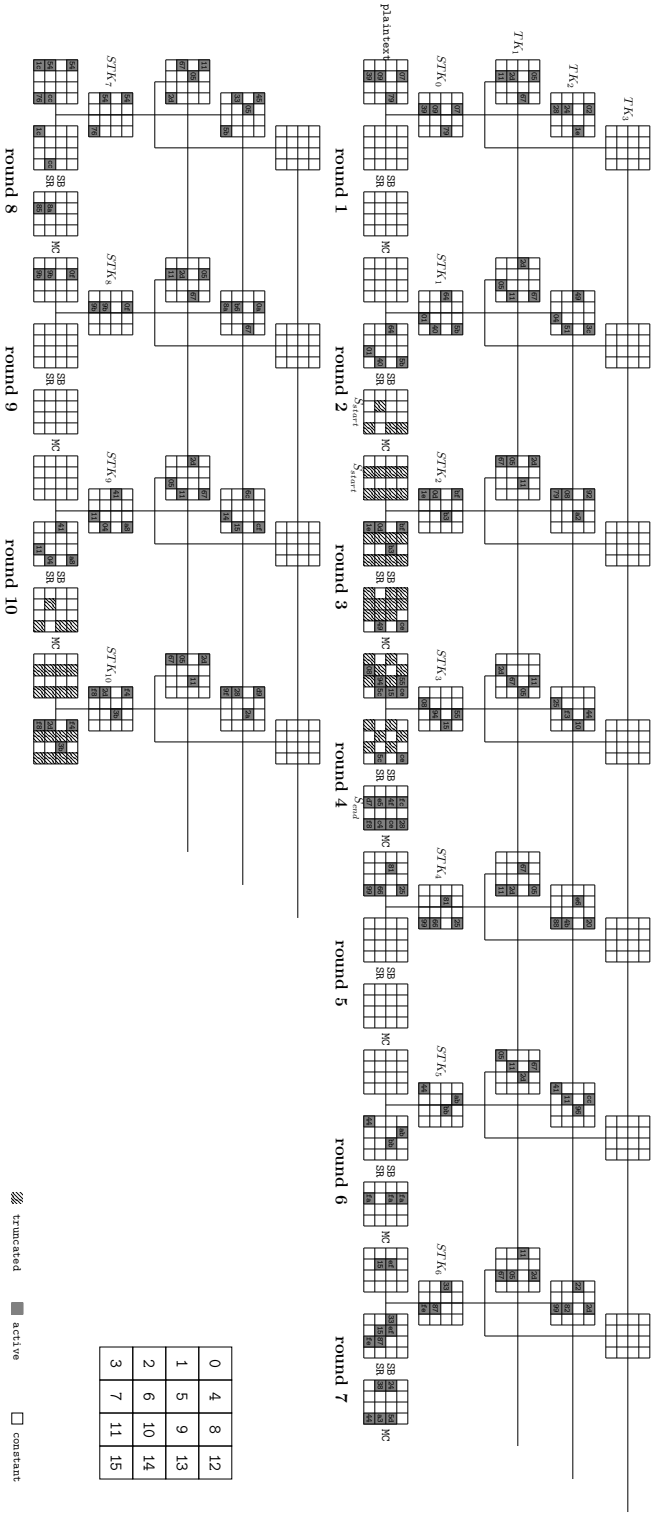


Figure 9: LBD attack on 10-round Deoxys-BC-256

complexity is also 2^{42}. The time complexity, as determined by Theorem 1, would be $2^{129-d_{in}-d_{out}} = 2^{129-28-39} = 2^{62}$ 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 intensive 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 for the 10-round Deoxys-BC-256 distinguisher, and the procedure for 13-round Deoxys-BC-384 is given in 6.3. The Inbound phase, 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 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.

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.

6.3 13-round Deoxys-BC-384 LBD Attack

The Inbound phase starts from Round 7 to Round 10 of Figure 10 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. Since 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.

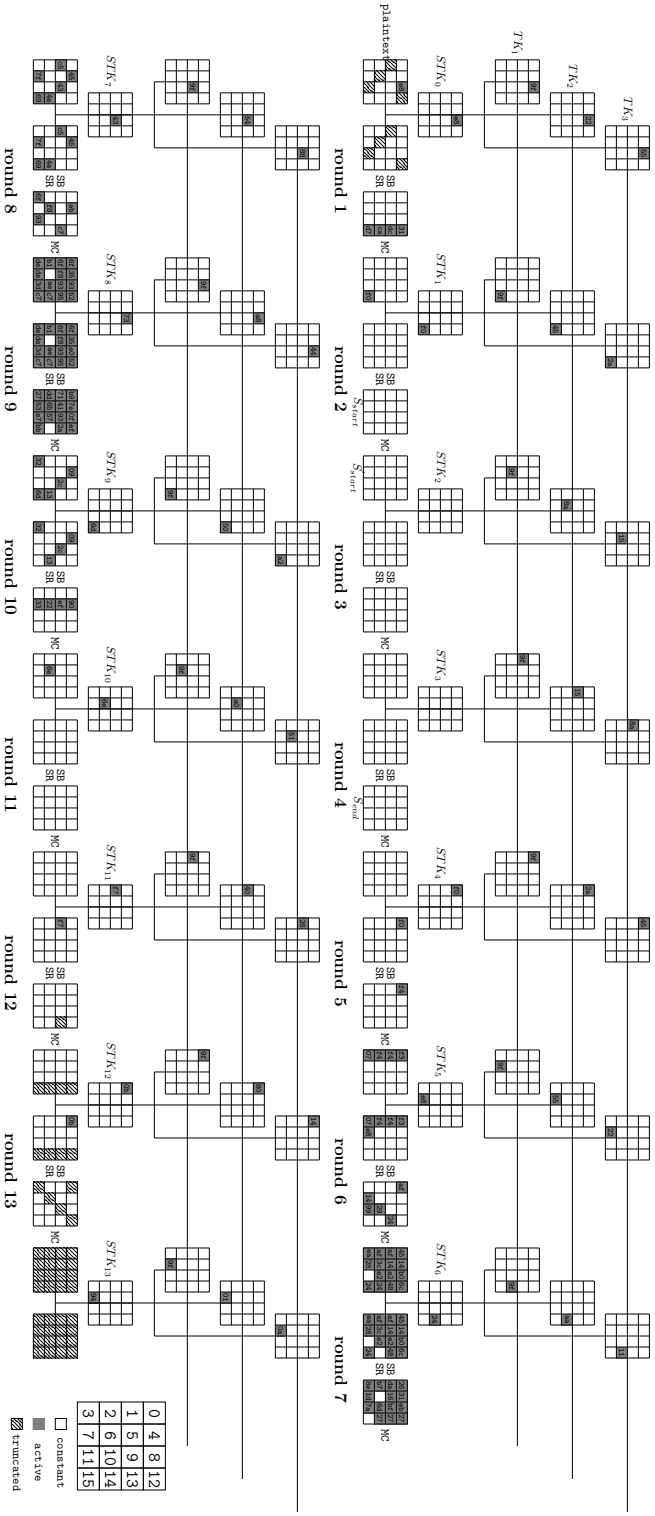


Figure 10: LBD attack on 13-round Deoxys-BC-384

Remark 3. In [76, 77], an 11-round related-key rectangle *distinguisher* on Deoxys-BC-384 was extended by adding one round at the beginning and two at the end to yield a *key-recovery* attack on 14 rounds. In [70], an 10-round related-key rectangle *distinguisher* on Deoxys-BC-384 was extended by adding one round at the beginning and four at the end to yield a *key-recovery* attack on 15 rounds. Converting that 11(10)-round distinguisher into a chosen-key distinguisher is straightforward; extending the 14(15)-round key-recovery to a chosen-key *distinguisher* is not. Our focus here is distinguishers, and we present a new 13-round chosen-key distinguisher. Similar separations between distinguishers and key-recovery also appear for AES-192 [9] and Deoxys-BC-256 [76].

Remark 4. Although our distinguisher does not currently yield key recovery, it exposes non-random behavior over nearly 81% of Deoxys-BC 's full-round configuration, thereby tightening the security-margin picture in the chosen-key/open-key setting that models hashing and tweakable deployments. This is complementary to longer-round key-recovery results (which can imply distinguishers): our results operate in a different, practically relevant threat setting and localize where distinguishability emerges—information that can guide future improvements in key-recovery techniques. We therefore view our findings as complementary to prior analyses and as a meaningful contribution to the overall understanding of Deoxys-BC's security profile.

7 Discussion and 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 automatic Constraint Programming search tools, 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 a 13-round related-key differential characteristic for Deoxys-BC-384, with acceptable probabilities for limited birthday attacks. For tweakable block ciphers, the inbound phase has been extended to 4 rounds by taking advantage of the available degrees of freedom from the tweaks. The free bytes of the tweaks also contribute to reducing 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. Besides, extending our analysis to Rijndael-256-256, especially in light of NIST's ongoing standardization efforts, is a promising direction for future work.

Acknowledgments

We would like to thank the anonymous reviewers of ToSC for their valuable comments and suggestions. This work was (in part) supported by: the National Key R&D Program of China (2024YFA1013000), the National Cryptologic Science Foundation of China (2025NCSF01012), the National Natural Science Foundation of China (62502494, 62532013, 62032014), the Fundamental Research Funds for the Central Universities, the Ministry of Education in Singapore under Grant RG102/24, the Fellowship Funds from China Postdoctoral Science Foundation (2024M763221), and the Zhongguancun Laboratory.

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] Andreeva, E., Bogdanov, A., Mennink, B.: Towards understanding the known-key security of block ciphers. In: FSE 2013. vol. 8424, pp. 348–366

[3] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 411–436. Springer (2015)

[4] Bao, Z., Ding, L., Guo, J., Wang, H., Zhang, W.: Improved meet-in-the-middle preimage attacks against AES hashing modes. IACR Trans. Symmetric Cryptol. **2019**(4), 318–347 (2019). https://doi.org/10.13154/tosc.v2019.i4.318-347, https://doi.org/10.13154/tosc.v2019.i4.318-347

[5] Bariant, A., Leurent, G.: Truncated boomerang attacks and application to aes-based ciphers. IACR Cryptol. ePrint Arch. p. 701 (2022), https://eprint.iacr.org/2022/701

[6] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016, Proceedings, Part II. pp. 123–153. Springer (2016)

[7] Biham, E.: New types of cryptanalytic attacks using related keys (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT '93, Proceedings. vol. 765, pp. 398–409

[8] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., Shamir, A.: Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In: EUROCRYPT 2010, Proceedings. vol. 6110, pp. 299–319

[9] Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: ASIACRYPT 2009, Proceedings. vol. 5912, pp. 1–18

[10] Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: CRYPTO 2009, Proceedings. vol. 5677, pp. 231–249

[11] Biryukov, A., Nikolic, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In: EUROCRYPT 2010, Proceedings. vol. 6110, pp. 322–344

[12] Blondeau, C., Peyrin, T., Wang, L.: Known-key distinguisher on full PRESENT. In: CRYPTO 2015, Proceedings, Part I. vol. 9215, pp. 455–474

[13] Bouillaguet, C., Derbez, P., Fouque, P.: Automatic search of attacks on round-reduced AES and applications. In: Rogaway, P. (ed.) CRYPTO 2011, Proceedings. vol. 6841, pp. 169–187

[14] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: EUROCRYPT 2018, Proceedings, Part II. vol. 10821, pp. 683–714

[15] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: A security analysis of Deoxys and its internal tweakable block ciphers. IACR Transactions on Symmetric Cryptology pp. 73–107 (2017)

[16] Cogliati, B., Seurin, Y.: On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In: EUROCRYPT 2015, Proceedings, Part I. vol. 9056, pp. 584–613

[17] Cogliati, B., Seurin, Y.: Strengthening the known-key security notion for block ciphers. In: FSE 2016. vol. 9783, pp. 494–513

[18] Cui, T., Sun, L., Chen, H., Wang, M.: Statistical integral distinguisher with multi-structure and its application on AES. In: ACISP 2017, Proceedings, Part I. vol. 10342, pp. 402–420

[19] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)

[20] Daemen, J., Rijmen, V.: Plateau characteristics. IET information security **1**(1), 11–17 (2007)

[21] Derbez, P., Euler, M., Fouque, P., Nguyen, P.H.: Revisiting related-key boomerang attacks on AES using computer-aided tool. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13793, pp. 68–88. Springer (2022). https://doi.org/10.1007/978-3-031-22969-5_3, https://doi.org/10.1007/978-3-031-22969-5_3

[22] Derbez, P., Fouque, P., Jean, J.: Faster chosen-key distinguishers on reduced-round AES. In: INDOCRYPT 2012. vol. 7668, pp. 225–243

[23] Dobraunig, C., Eichlseder, M., Mendel, F.: Square attack on 7-round Kiasu-BC. In: ACNS 2016, Proceedings. vol. 9696, pp. 500–517

[24] Dobraunig, C., List, E.: Impossible-differential and boomerang cryptanalysis of round-reduced Kiasu-BC. In: CT-RSA 2017, Proceedings. vol. 10159, pp. 207–222

[25] Dong, X., Guo, J., Li, S., Pham, P.: Triangulating rebound attack on aes-like hashing. IACR Cryptol. ePrint Arch. p. 731 (2022), https://eprint.iacr.org/2022/731

[26] Dong, X., Qin, L., Sun, S., Wang, X.: Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In: EUROCRYPT 2022, Proceedings, Part III. vol. 13277, pp. 3–33

[27] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family. Submission to NIST (round 3) **7**(7.5), 3 (2010)

[28] Fouque, P.A., Jean, J., Peyrin, T.: Structural evaluation of aes and chosen-key distinguisher of 9-round aes-128. In: CRYPTO 2013. pp. 183–203. Springer (2013)

[29] Gérault, D., Lafourcade, P.: Related-key cryptanalysis of midori. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10095, pp. 287–304 (2016). https://doi.org/10.1007/978-3-319-49890-4_16, https://doi.org/10.1007/978-3-319-49890-4_16

[30] Gerault, D., Lafourcade, P., Minier, M., Solnon, C.: Computing aes related-key differential characteristics with constraint programming. Artificial intelligence **278**, 103183 (2020)

[31] Gérault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Rueher, M. (ed.) Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9892, pp. 584–601. Springer (2016). https://doi.org/10.1007/978-3-319-44953-1_37, https://doi.org/10.1007/978-3-319-44953-1_37

[32] Gilbert, H.: A simplified representation of AES. In: ASIACRYPT 2014, Proceedings, Part I. vol. 8873, pp. 200–222

[33] Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for aes-like permutations. In: FSE 2010, Seoul, Korea, February 7-10, 2010. pp. 365–383 (2010)

[34] Grassi, L., Rechberger, C.: Revisiting gilbert's known-key distinguisher. Des. Codes Cryptogr. **88**(7), 1401–1445 (2020). https://doi.org/10.1007/s10623-020-00756-5, https://doi.org/10.1007/s10623-020-00756-5

[35] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The led block cipher. In: Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13. pp. 326–341. Springer (2011)

[36] Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y.: Improved attacks on sLiSCP permutation and tight bound of limited birthday distinguishers. IACR Trans. Symmetric Cryptol. **2020**(4), 147–172 (2020). https://doi.org/10.46586/tosc.v2020.i4.147-172, https://doi.org/10.46586/tosc.v2020.i4.147-172

[37] Iwamoto, M., Peyrin, T., Sasaki, Y.: Limited-birthday distinguishers for hash functions. In: ASIACRYPT 2013, Proceedings, Part II. vol. 8270, pp. 504–523

[38] Jean, J.: TikZ for Cryptographers. https://www.iacr.org/authors/tikz/ (2016)

[39] Jean, J., Naya-Plasencia, M., Peyrin, T.: Multiple limited-birthday distinguishers and applications. In: SAC 2013. pp. 533–550. Springer (2013)

[40] Jean, J., Nikolic, I., Peyrin, T.: Kiasu v1. Submitted to the CAESAR competition (2014)

[41] Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the tweakey framework. In: ASIACRYPT 2014. pp. 274–288. Springer (2014)

[42] Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41. Submitted to CAESAR **124** (2016)

[43] Khovratovich, D., Biryukov, A., Nikolic, I.: Speeding up collision search for byte-oriented hash functions. In: CT-RSA 2009. pp. 164–181

[44] Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: ASIACRYPT 2007, Proceedings. vol. 4833, pp. 315–324

[45] Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: FSE 2011. pp. 306–327. Springer (2011)

[46] Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. In: ASIACRYPT 2009, Tokyo, Japan, December 6-10, 2009. Proceedings. pp. 126–143 (2009)

[47] Lauridsen, M.M., Rechberger, C.: Linear distinguishers in the key-less setting: Application to PRESENT. In: FSE 2015. vol. 9054, pp. 217–240

[48] Leurent, G., Pernot, C.: New representations of the AES key schedule. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021, Proceedings, Part I. vol. 12696, pp. 54–84. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_3, https://doi.org/10.1007/978-3-030-77870-5_3

[49] Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: CRYPTO 2002. pp. 31–46. Springer (2002)

[50] Liu, F., Isobe, T., Meier, W.: Automatic verification of differential characteristics: Application to reduced gimli. In: CRYPTO 2020, Proceedings, Part III. vol. 12172, pp. 219–248

[51] Liu, Y., Zhang, W., Sun, B., Rijmen, V., Liu, G., Li, C., Fu, S., Cao, M.: The phantom of differential characteristics. Designs, Codes and Cryptography **88**(11), 2289–2311 (2020)

[52] Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. In: SAC 2009, Revised Selected Papers. vol. 5867, pp. 16–35

[53] Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced whirlpool and grøstl. In: FSE 2009, Leuven, Belgium, February 22-25, 2009. pp. 260–276 (2009)

[54] Mennink, B., Preneel, B.: On the impact of known-key attacks on hash functions. In: ASIACRYPT 2015, Proceedings, Part II. vol. 9453, pp. 59–84

[55] Minematsu, K.: Building blockcipher from small-block tweakable blockcipher. Designs, Codes and Cryptography **74**(3), 645–663 (2015)

[56] Minier, M., Phan, R.C., Pousse, B.: Distinguishers for ciphers and known key attack against rijndael with large blocks. In: AFRICACRYPT 2009, Proceedings. vol. 5580, pp. 60–76

[57] Minier, M., Solnon, C., Reboul, J.: Solving a symmetric key cryptographic problem with constraint programming. In: ModRef 2014, Workshop of the CP 2014 Conference. p. 13 (2014)

[58] Naito, Y.: Full prf-secure message authentication code based on tweakable block cipher. In: International Conference on Provable Security. pp. 167–182. Springer (2015)

[59] Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: International Conference on Principles and Practice of Constraint Programming. pp. 529–543. Springer (2007)

[60] Nikolic, I., Pieprzyk, J., Sokolowski, P., Steinfeld, R.: Known and chosen key differential distinguishers for block ciphers. In: ICISC 2010. vol. 6829, pp. 29–48

[61] Peyrin, T., Seurin, Y.: Counter-in-tweak: authenticated encryption modes for tweakable block ciphers. In: CRYPTO 2016. pp. 33–63. Springer (2016)

[62] Prud'homme, C., Fages, J.G., Lorca, X.: Choco solver documentation. TASC, INRIA Rennes, LINA CNRS UMR **6241**, 13–42 (2016)

[63] Sadeghi, S., Rijmen, V., Bagheri, N.: 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). https://doi.org/10.1007/s10623-021-00904-5, https://doi.org/10.1007/s10623-021-00904-5

[64] Sasaki, Y.: Known-key attacks on rijndael with large blocks and strengthening *ShiftRow* parameter. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. vol. 6434, pp. 301–315

[65] Sasaki, Y.: Improved related-tweakey boomerang attacks on Deoxys-BC. In: Africacrypt 2018. pp. 87–106. Springer (2018)

[66] Sasaki, Y., Emami, S., Hong, D., Kumar, A.: Improved known-key distinguishers on feistel-sp ciphers and application to camellia. In: ACISP 2012. vol. 7372, pp. 87–100

[67] Sasaki, Y., Li, Y., Wang, L., Sakiyama, K., Ohta, K.: Non-full-active Super-Sbox analysis: Applications to ECHO and grøstl. In: ASIACRYPT 2010, Singapore, December 5-9, 2010. Proceedings. pp. 38–55 (2010)

[68] Sasaki, Y., Yasuda, K.: Known-key distinguishers on 11-round feistel and collision attacks on its hashing modes. In: FSE 2011. vol. 6733, pp. 397–415

[69] Schroeppel, R., Orman, H.: The hasty pudding cipher. AES candidate submitted to NIST p. M1 (1998)

[70] Song, L., Yang, Q., Chen, Y., Hu, L., Weng, J.: Probabilistic extensions: A one-step framework for finding rectangle attacks and beyond. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 339–367. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_12, https://doi.org/10.1007/978-3-031-58716-0_12

[71] Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018). https://doi.org/10.13154/tosc.v2018.i3.93-123, https://doi.org/10.13154/tosc.v2018.i3.93-123

[72] Sun, S., Gérault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., Hu, L.: Analysis of aes, skinny, and others with constraint programming. IACR Trans. Symmetric Cryptol. **2017**(1), 281–306 (2017). https://doi.org/10.13154/tosc.v2017.i1.281-306, https://doi.org/10.13154/tosc.v2017.i1.281-306

[73] Tolba, M., Abdelkhalek, A., Youssef, A.M.: A meet in the middle attack on reduced round Kiasu-BC. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **99-A**(10), 1888–1890 (2016). https://doi.org/10.1587/transfun.E99.A.1888, https://doi.org/10.1587/transfun.E99.A.1888

[74] Wang, H., Peyrin, T.: Boomerang switch in multiple rounds. application to AES variants and Deoxys. IACR Transactions on Symmetric Cryptology pp. 142–169 (2019)

[75] Yang, Q., Song, L., Zhang, N., Shi, D., Wang, L., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. J. Cryptol. **37**(2), 19 (2024). https://doi.org/10.1007/S00145-024-09499-1, https://doi.org/10.1007/s00145-024-09499-1

[76] Zhao, B., Dong, X., Jia, K.: New related-tweakey boomerang and rectangle attacks on Deoxys-BC including bdt effect. IACR Transactions on Symmetric Cryptology pp. 121–151 (2019)

[77] Zhao, B., Dong, X., Jia, K., Meier, W.: Improved related-tweakey rectangle attacks on reduced-round Deoxys-BC-384 and deoxys-i-256-128. In: INDOCRYPT 2019. vol. 11898, pp. 139–159

[78] Zhou, N.F., Kjellerstrand, H., Fruhman, J.: Constraint solving and planning with Picat. Springer (2015)

[79] Zhu, C., Wang, G., Zhu, B.: Fast chosen-key distinguish attacks on round-reduced AES-192. In: ACISP 2019. vol. 11547, pp. 573–587

Supplementary Material

A Key pair of AES-192 distinguisher

Table 4: The key pair conforming to differential characteristic used in the 12-round AES-192 distinguisher.

Round	k	k'	$k \oplus k'$
0	D22174C1 47260B2D C670A1C9 363DFE66	FF1C727A 4B260B2D FBED3C75 363DFE66	2D3D06BB 0C000000 3D9D9DBC 00000000
1	43EC0780 00000000 B04217A2 F7641C8F	4FEC0780 2D9D9DBC A04217A2 EB641C8F	0C000000 2D9D9DBC 10000000 1C000000
2	3114BD46 07294320 44C544A0 44C544A0	108920FA 26B4DE9C 6958D91C 44C544A0	219D9DBC 219D9DBC 2D9D9DBC 00000000
3	1459F7B9 E33DEB36 D2295670 D5001550	0459F7B9 EF3DEB36 FFB4CBCC D9001550	10000000 0C000000 2D9D9DBC 0C000000
4	91C551F0 D5001550 7300A4BA 903D4F8C	B058CC4C F49D88EC 5E9D3906 B1A0D230	219D9DBC 219D9DBC 2D9D9DBC 219D9DBC
5	421419FC 97140CAC 06D15D5C D3D1480C	4E1419FC 97140CAC 274CC0E0 D3D1480C	0C000000 00000000 219D9DBC 00000000
6	45525ADC D56F1550 977B0CAC 006F0000	68CFC760 D96F1550 977B0CAC 006F0000	2D9D9DBC 0C000000 00000000 00000000
7	06BE5D5C D56F1550 FD0B09DF 28641C8F	2723C0E0 F4F288EC F10B09DF 28641C8F	219D9DBC 219D9DBC 0C000000 00000000
8	BF1F1023 BF701023 B9CE4D7F 6CA1582F	BF1F1023 BF701023 9853D0C3 6CA1582F	00000000 00000000 219D9DBC 00000000
9	EF611C8F C7050000 781A1023 C76A0000	E3611C8F CB050000 741A1023 CB6A0000	0C000000 0C000000 0C000000 0C000000
10	7EA44D7F 12051550 C4384F46 033D4F46	5339D0C3 3F9888EC E5A5D2FA 2EA0D2FA	2D9D9DBC 2D9D9DBC 219D9DBC 2D9D9DBC
11	7B275F65 BC4D5F65 C2E9121A D0EC074A	5ABAC2D9 91D0C2D9 C2E9121A FD719AF6	219D9DBC 2D9D9DBC 00000000 2D9D9DBC
12	8AFD9936 89C0D670 F2E78915 4EAAD670	C61D90AE E8BD4254 B207808D 23D74254	4CE00998 617D9424 40E00998 6D7D9424

B AES-256 Differential Characteristics

Table 5: 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

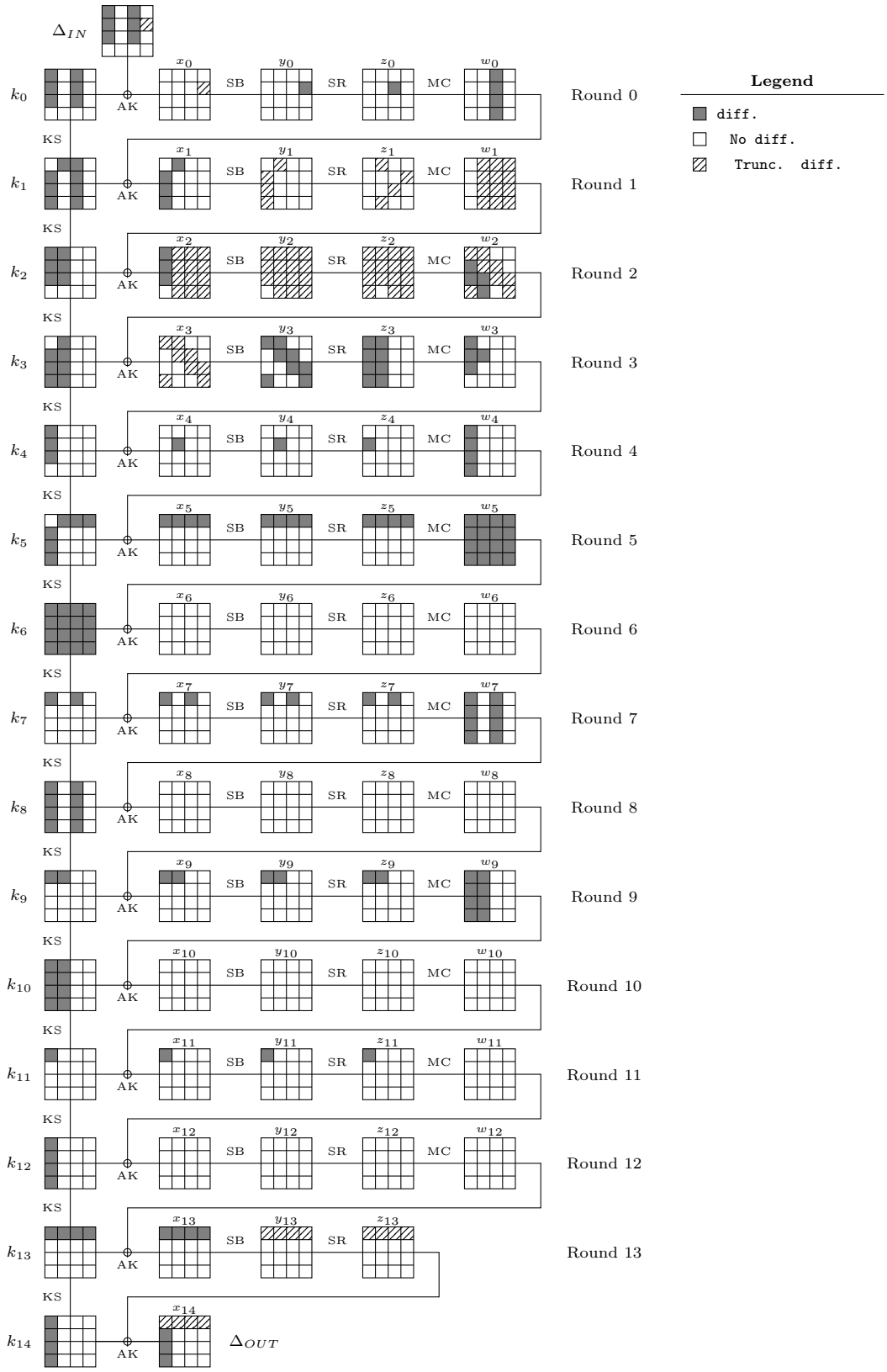


Figure 11: Differential trail of 14-round AES-256 used in the distinguisher.

C Kiasu-BC Differential Characteristics

Table 6: Differential characteristics used in the distinguisher [28] of 9 rounds Kiasu-BC.

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 7: The key pair conforming to differential characteristic used in the 9-round Kiasu-BC distinguisher [28].

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

Table 8: 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

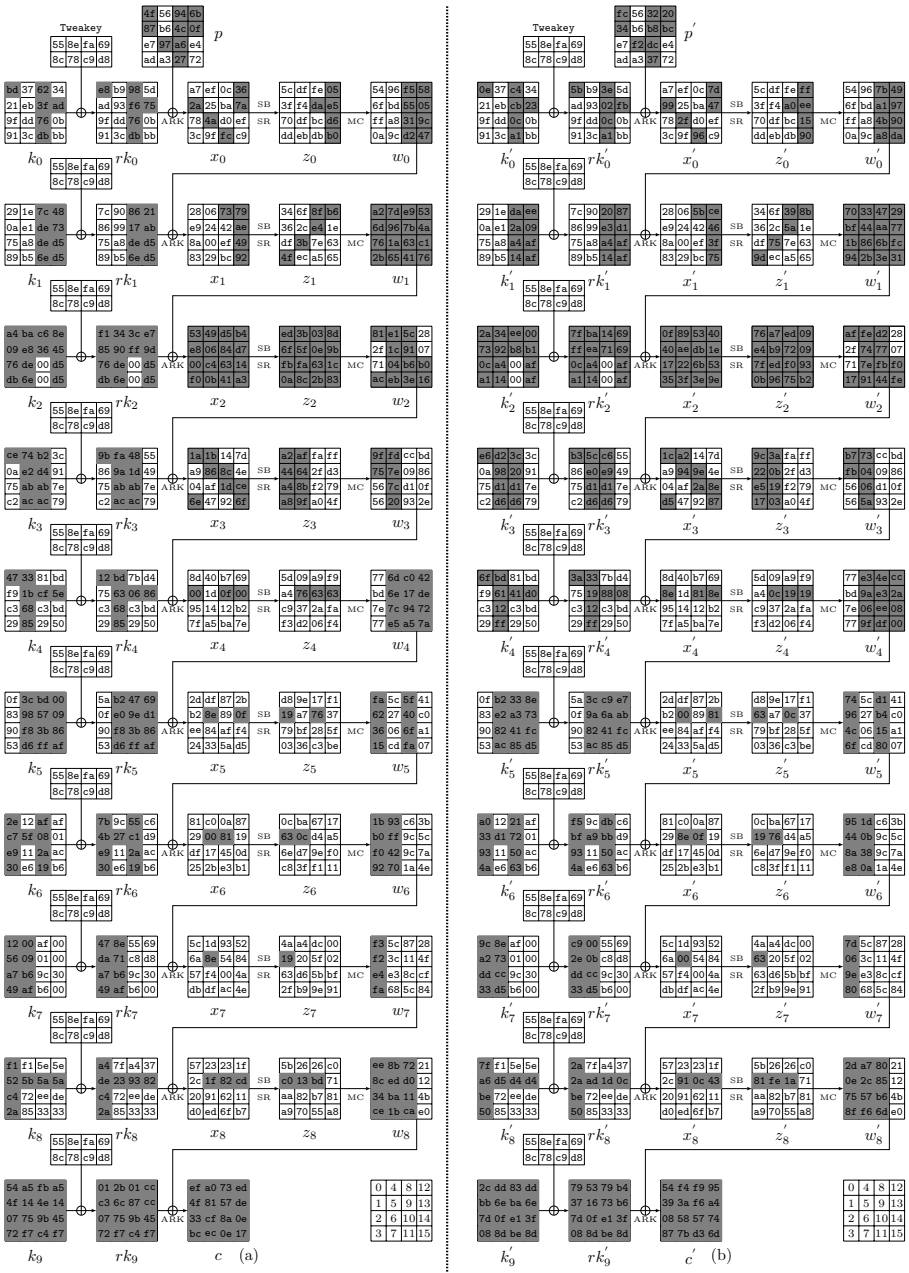


Figure 12: A pair conforming to the 9-round `Kiasu-BC` distinguisher.

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

D Step 2 Constraint Models

This appendix provides detailed constraint formulations used in Step 2 of our two-step search methodology. Step 2 instantiates truncated differential trails from Step 1 into concrete differential characteristics by upgrading binary activity variables to exact byte differences and enforcing cryptographic compatibility constraints.

D.1 Basic Round Function Constraints

The fundamental round function constraints model the exact byte-level propagation through each AES operation, upgrading the binary constraints from Step 1.

D.1.1 AddRoundKey (ARK) Operation

The AddRoundKey operation applies XOR between state and subkey bytes. For each round r, position (i, j):

$$\Delta Y_r[i,j] \oplus \Delta K_r[i,j] = \Delta X_{r+1}[i,j] \tag{7}$$

where $\Delta Y_r[i,j] \in \{0,1,\ldots,255\}$ represents the exact byte difference after MixColumns, $\Delta K_r[i,j]$ the subkey difference, and $\Delta X_{r+1}[i,j]$ the resulting state difference.

This constraint is implemented using precomputed XOR tuples:

$$\text{XOR_Table} = \{(x,y,z) \mid x,y,z \in \{0,\ldots,255\}, x \oplus y = z\}$$

D.1.2 ShiftRows (SR) Operation

ShiftRows applies a fixed permutation to state rows. For AES-128 with shift offsets $\text{SHIFT} = [0,1,2,3]$:

$$\Delta X_r[i,(j+\text{SHIFT}[i]) \bmod 4] = \Delta Z_r[i,j] \tag{8}$$

where $\Delta Z_r[i,j]$ represents the state after ShiftRows. This constraint ensures exact propagation of byte differences through the deterministic permutation.

D.2 DDT Constraints: S-box Input/Output Compatibility

The most critical constraints in Step 2 enforce S-box differential distribution table (DDT) compatibility, ensuring that input/output difference pairs correspond to valid S-box transitions.

D.2.1 S-box Differential Constraint

For each S-box in round r at position (i, j):

$$(\Delta X_r[i,j], \Delta SB_r[i,j], P_r[i,j]) \in \text{DDT_Table} \tag{9}$$

where:

- $\Delta X_r[i,j]$ is the input difference
- $\Delta SB_r[i,j]$ is the output difference
- $P_r[i,j]$ is the differential probability (expressed as $-\log_2(\Pr)$)

The DDT table is precomputed as:

$$\text{DDT_Table} = \{(\delta_{in}, \delta_{out}, p) \mid \delta_{in}, \delta_{out} \in \{0,\ldots,255\}, p = -\log_2(\Pr[\delta_{in} \to \delta_{out}])\}$$

D.2.2 DDT Table Construction

The differential distribution table is constructed by computing all valid S-box transitions:

$$\text{For each } \delta_{in} \in \{1, \ldots, 255\}:$$

$$\text{Count}[\delta_{in}][\delta_{out}] = |\{x \mid S(x) \oplus S(x \oplus \delta_{in}) = \delta_{out}\}| \tag{10}$$

$$\Pr[\delta_{in} \to \delta_{out}] = \frac{\text{Count}[\delta_{in}][\delta_{out}]}{256} \tag{11}$$

For $\delta_{in} = 0$, we enforce $\delta_{out} = 0$ with probability 1 (encoded as $p = 0$).

D.2.3 Special Case: Free Input Differences

At round 0, input differences can be chosen freely (plaintext control), so active S-boxes receive maximum probability:

$$\text{If } \Delta X_0[i,j] \neq 0 \text{ then } P_0[i,j] = 6 \text{ (probability } 2^{-6}) \tag{12}$$

D.3 Exact Arithmetic: Finite Field Operations for MixColumns

MixColumns performs matrix multiplication in $\text{GF}(2^8)$ with polynomial $x^8+x^4+x^3+x+1$. Step 2 models this exactly using precomputed multiplication tables.

D.3.1 MixColumns Forward Constraint

For each column j in round r:

$$\Delta Y_{r+1}[0,j] = 2 \cdot \Delta Z_r[0,j] \oplus 3 \cdot \Delta Z_r[1,j] \oplus \Delta Z_r[2,j] \oplus \Delta Z_r[3,j] \tag{13}$$

$$\Delta Y_{r+1}[1,j] = \Delta Z_r[0,j] \oplus 2 \cdot \Delta Z_r[1,j] \oplus 3 \cdot \Delta Z_r[2,j] \oplus \Delta Z_r[3,j] \tag{14}$$

$$\Delta Y_{r+1}[2,j] = \Delta Z_r[0,j] \oplus \Delta Z_r[1,j] \oplus 2 \cdot \Delta Z_r[2,j] \oplus 3 \cdot \Delta Z_r[3,j] \tag{15}$$

$$\Delta Y_{r+1}[3,j] = 3 \cdot \Delta Z_r[0,j] \oplus \Delta Z_r[1,j] \oplus \Delta Z_r[2,j] \oplus 2 \cdot \Delta Z_r[3,j] \tag{16}$$

D.3.2 Finite Field Multiplication Tables

Multiplication by constants in $\text{GF}(2^8)$ is implemented using precomputed tables:

$$\text{MUL2_Table} = \{(x,y) \mid x \in \{0,\ldots,255\}, y = 2 \cdot x \bmod (x^8+x^4+x^3+x+1)\} \tag{17}$$

$$\text{MUL3_Table} = \{(x,y) \mid x \in \{0,\ldots,255\}, y = 3 \cdot x \bmod (x^8+x^4+x^3+x+1)\} \tag{18}$$

For efficient constraint propagation, combined operations are precomputed:

$$\text{MUL2XOR3_Table} = \{(z,x,y) \mid z = (2 \cdot x) \oplus (3 \cdot y)\}$$

D.3.3 MixColumns Inverse Constraint

The inverse MixColumns constraint ensures bidirectional consistency:

$$\Delta Z_r[0,j] = 14 \cdot \Delta Y_{r+1}[0,j] \oplus 11 \cdot \Delta Y_{r+1}[1,j] \oplus 13 \cdot \Delta Y_{r+1}[2,j] \oplus 9 \cdot \Delta Y_{r+1}[3,j] \tag{19}$$

$$\Delta Z_r[1,j] = 9 \cdot \Delta Y_{r+1}[0,j] \oplus 14 \cdot \Delta Y_{r+1}[1,j] \oplus 11 \cdot \Delta Y_{r+1}[2,j] \oplus 13 \cdot \Delta Y_{r+1}[3,j] \tag{20}$$

$$\Delta Z_r[2,j] = 13 \cdot \Delta Y_{r+1}[0,j] \oplus 9 \cdot \Delta Y_{r+1}[1,j] \oplus 14 \cdot \Delta Y_{r+1}[2,j] \oplus 11 \cdot \Delta Y_{r+1}[3,j] \tag{21}$$

$$\Delta Z_r[3,j] = 11 \cdot \Delta Y_{r+1}[0,j] \oplus 13 \cdot \Delta Y_{r+1}[1,j] \oplus 9 \cdot \Delta Y_{r+1}[2,j] \oplus 14 \cdot \Delta Y_{r+1}[3,j] \tag{22}$$

Additional multiplication tables are precomputed for constants $\{9, 11, 13, 14\}$ to support inverse operations.

D.4 Key Schedule Constraints

The AES key schedule generates round keys through S-box applications and linear operations. Step 2 models exact key schedule propagation with DDT constraints on key S-boxes.

D.4.1 Key Schedule S-box Constraints

For AES-128, S-boxes in the key schedule occur in specific positions. For each key S-box at position (i, j):

$$(\Delta K_{\text{in}}[i,j], \Delta K_{\text{out}}[i,j], P_K[i,j]) \in \text{DDT_Table} \tag{23}$$

D.4.2 Key Schedule Linear Operations

Linear key schedule operations are modeled as exact XOR constraints, similar to AddRoundKey but operating on key bytes across rounds.

D.5 Probability Optimization: Minimize Total Attack Complexity

The objective function in Step 2 minimizes the total differential characteristic probability, which directly relates to attack complexity.

D.5.1 Objective Function

The optimization objective aggregates all S-box probability contributions:

$$\text{minimize} \sum_{r=r_{start}}^{r_{end}} \sum_{i=0}^{3} \sum_{j=0}^{3} P_r[i,j] \tag{24}$$

where the summation covers the outbound rounds (typically $r \in \{4, \ldots, n-2\}$ for the rebound attack structure).

D.5.2 Probability Bounds

The objective variable is bounded based on the number of active S-boxes from Step 1:

$$6 \cdot \text{nbSBoxes} \leq \text{obj} \leq 7 \cdot \text{nbSBoxes} \tag{25}$$

where:

- Lower bound: 2^{-6} per S-box (best case differential probability)
- Upper bound: 2^{-7} per S-box (conservative estimate)

D.5.3 Search Strategy

The Choco solver employs domain-over-weighted-degree heuristic for variable ordering, prioritizing key schedule variables to detect inconsistencies early. Value selection uses DDT-guided assignment to prefer high-probability S-box transitions.

E New AES Key Schedule Representation

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:

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

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:

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

Following the new key-schedule representation, the key bytes $k[12], k[13], k[14]$, and $k[15]$, where the SubByte 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:

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

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

Proposition 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 column of the subkey at round 1 k_1.*

In the previous example, admissible active S-box values for k_6, k_7, k_8, and k_9 are selected sequentially so that four active S-boxes at k_2 and k_1 are fulfilled in roughly 2^8 time complexity under the following constraints:

$$k_5[13] = s_5[8] = \texttt{SB}(s_8[4]) \oplus s_9[8] = \texttt{SB}(k_8[14]) \oplus k_9[13],$$

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

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 be further reduced by filtering out trails that do not meet these above constraints.