IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, 26 pages.

Classical Free-Start Collision Attacks on Full-Round AES-256-DM and AES-256-MMO

Liyuan Tang[1], Yanzhao Shen[2,5] and Xiaoyang Dong[a,1,3,4]

[1] Tsinghua University, Beijing, P. R. China
[2] Shandong Institute of Blockchain, Shandong, P. R. China
[3] Zhongguancun Laboratory, Beijing, P. R. China
[4] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, Beijing, P. R. China
[5] National Financial Cryptography Research Center, Shandong, P. R. China

Abstract

Despite extensive cryptanalysis of hash modes based on the Advanced Encryption Standard (AES), no classical free-start collision attack has previously been reported for full-round AES-256-DM or AES-256-MMO. In this work, we address this problem by extending the guess-and-determine (GD) rebound methodology. Our main technical contribution is a precomputation-based method for handling conflicts involving guessed variables in the GD procedure. By precomputing solutions to the corresponding systems of byte equations, such conflicts can be resolved in $O(1)$ online time. We further exploit triangular dependencies among controllable guessed variables to reduce the memory required for precomputation.

Using this method, we construct classical free-start collision attacks applicable to both AES-256-DM and AES-256-MMO. For the full 14-round AES-256, the attacks require 2^{60} time and 2^{24} memory, providing the first classical free-start collision attacks on these full-round constructions. The same approach improves the time complexity of the full-round free-start differential q-multicollision attack from $q \cdot 2^{67}$ to $q \cdot 2^{60}$, with memory complexity 2^{24}. We also obtain free-start collision attacks on the 12-round variants with time complexity 2^{54} and memory complexity 2^{24}.

Keywords: Guess-and-Determine Rebound Attack, Free-Start Collision, AES-256

1 Introduction

Hash functions are fundamental cryptographic primitives underlying digital signatures, message authentication, integrity protection, and numerous other cryptographic applications. Their security is commonly evaluated with respect to three fundamental properties: collision resistance, preimage resistance, and second-preimage resistance.

This work focuses on block-cipher-based hash modes instantiated with the Advanced Encryption Standard (AES). Important examples include the Davies–Meyer (DM), Matyas–Meyer–Oseas (MMO), and Miyaguchi–Preneel (MP) modes from the PGV framework [PGV], which can be iterated using the Merkle–Damgård construction [Mer89, Dam89]. These constructions are of both theoretical and practical interest. In particular, AES-MMO has been adopted by Zigbee [All07] and recommended in documents issued by ISO [ISO10]

Emails: tangly22@mails.tsinghua.edu.cn (Liyuan Tang), shanyanzhao@sdibc.cn (Yanzhao Shen), xiaoyangdong@tsinghua.edu.cn (Xiaoyang Dong)
[a]Corresponding author.

This work is licensed under a "CC BY 4.0" license.

and the IETF [Cam12], while an AES-DM construction was submitted to NIST under the name AES-Hash [CL01]. Moreover, the availability of efficient hardware and software implementations of AES, including AES-NI, makes AES-based hashing attractive in modern cryptographic protocols such as multi-party computation [GKWY20, KOS16]. AES-like components are also employed in several dedicated hash functions, including Whirlpool [BR+00], Grøstl [GKM+09], Grindahl [KRT], ECHO [BBG+09], and Haraka v2 [KLMR16].

The security of AES-based hash modes has consequently received extensive attention. For collision attacks, Mendel *et al.* introduced the rebound attack and applied it to 5-round AES-MMO/MP [MRST09]. Subsequent developments based on the Super-Sbox technique extended the attacks to 6 rounds [GP10, LMR+09]. More recent studies have obtained increasingly strong collision attacks on reduced-round AES/Rijndael instantiated in the DM, MMO, and MP modes [TSI+24, CGL+25, QBD25]. Substantial progress has also been made on preimage attacks. Early results reached 7 rounds of AES-128 [Sas, WFW+], while automatic meet-in-the-middle techniques subsequently extended the attacks to 8 rounds [BDG+21]. Further advances reached 10 rounds of AES-192 and AES-256 in the DM mode [DHS+, CGL+24]. These results demonstrate the broad and continuing interest in the security of AES-based hashing.

Rebound Attacks.

Introduced by Mendel, Rechberger, Schläffer, and Thomsen at FSE 2009 [MRST09], the rebound attack has become a principal technique for analyzing AES-like hash functions. It divides a differential trail into an inbound phase, where available degrees of freedom are used to satisfy its low-probability part efficiently, and an outbound phase, where the remaining conditions are fulfilled probabilistically.

Subsequent extensions connected multiple inbound phases using key-schedule degrees of freedom and introduced the Super-Sbox technique for handling two consecutive rounds as a larger nonlinear component [GP10, LMR+09]. Further developments reduced memory costs, improved list merging and time–memory tradeoffs, and extended the inbound phase through triangulation and guess-and-determine techniques [SLW+10, Nay11, DDKS, BDF11].

Despite these advances, classical results on full-round AES-based hash modes remain limited. Biryukov *et al.* presented a free-start differential q-multicollision construction based on full-round AES-256 [BKN09], but it considers a different notion from the ordinary free-start collision studied here. A quantum collision attack on full-round AES-256-DM was later presented at CRYPTO 2026 [TQHD26]. Prior to this work, constructing a classical free-start collision attack on full-round AES-256-DM or AES-256-MMO remained an open problem.

Our Contributions

At CRYPTO 2025, Qin *et al.* [QBD25] introduced the guess-and-determine (GD) rebound attack by incorporating the GD technique of Bouillaguet, Derbez, and Fouque [BDF11] into the rebound framework. At CRYPTO 2026, Tang *et al.* [TQHD26] further extended this methodology and obtained quantum (multi-block) standard collision attacks on AES-DM. By considering only the last block of their attack on full-round AES-256-DM, a quantum free-start collision can be obtained with time complexity 2^{38}. Nevertheless, constructing a classical free-start collision attack on full-round AES-256-DM or AES-256-MMO remained an important open problem. In this work, we further develop the GD methodology and apply it to the DM and MMO hashing modes based on AES-256. Our attacks on AES-256-MMO target the one-block compression function with full freedom in the 256-bit AES key, rather than an iterated hash function. Our main contributions are summarized as follows.

- **Extended GD methodology.** We develop a systematic method for resolving conflicts involving guessed variables in the GD procedure. The resulting attacks cannot be obtained by directly applying the existing GD tool of Bouillaguet, Derbez, and Fouque [BDF11], but require the additional conflict-handling techniques introduced in this work. Specifically, we replace the probabilistic verification of such conflicts with precomputation, enabling $O(1)$ online resolution. Here, "online" refers to the GD computation performed after preprocessing. We further develop a memory-reduction strategy that exploits structural dependencies among controllable guessed variables to make the preprocessing requirements manageable.
- **First classical free-start collision attack on full-round AES-256-DM and AES-256-MMO.** The closest classical result is the free-start q-multicollision construction of Biryukov, Khovratovich, and Nikolić [BKN09], which is applicable to both hashing modes with time complexity $q \cdot 2^{67}$. Since this complexity exceeds the birthday bound 2^{64} for a 128-bit output, it does not improve upon generic collision search in the ordinary free-start collision setting. We present an attack applicable to both full-round AES-256-DM and AES-256-MMO, with time complexity 2^{60} and memory complexity 2^{24}. To the best of our knowledge, this is the first classical free-start collision attack on these full-round constructions.
- **Free-start collision attacks on 12-round AES-256-DM and AES-256-MMO.** We also apply our method to the 12-round variants of both hashing modes and obtain attacks with time complexity 2^{54} and memory complexity 2^{24}. To validate our method experimentally, we implement a partial attack following the 12-round differential trail, with an expected time complexity of 2^{42}. Our implementation successfully finds a message pair satisfying the expected differences up to the input of the S-boxes in round 10, without imposing the differential conditions on the two active S-boxes in that round. Further implementation details and experimental results are provided in Appendix B.

The results and comparisons with existing attacks are summarized in Table 1.

Table 1: Summary of attacks on AES-256-based hashing modes.

Target Mode	Attack	Rounds	Time	C-Mem	qRAM	Setting	Ref.
DM	Collision	6/14	2^{61}	-	-	Classic	[TSI+24]
		6/14	Practical	-	-	Classic	[QBD25]
		7/14	$2^{36.7}$	-	60	Quantum	[QBD25]
		9/14	2^{58}	2^{55}	-	Classic	[TSI+24]
		14/14	$2^{50.42}$	-	-	Quantum	[TQHD26]
	SFS Collision	9/14	Practical	-	-	Classic	[TSI+24]
MMO/DM	FS Collision	10/14	2^{49}	-	-	Classic	[BGLP24]
		10/14	$2^{25.6}$	-	-	Quantum	[BGLP24]
		12/14	$2^{61.3}$	-	-	Classic	[KPC+20]
		12/14	2^{54}	2^{24}	-	Classic	Sect. 5.2
		14/14	2^{60}	2^{24}	-	Classic	Sect. 4.3
	FS Multi-Collision	14/14	$q \cdot 2^{67}$	-	-	Classic	[BKN09]
		14/14	$q \cdot 2^{60}$	2^{24}	-	Classic	Sect. 4.4
	(Pseudo) Preimage	9/14	2^{120}	2^{8}	-	Classic	[BDG+21]
		9/14	2^{60}	-	2^{8}	Quantum	[DDS25]
		10/14	2^{120}	$2^{56}/2^{48}/2^{8}$	-	Classic	[DHS+, DDS25, ZHQD25]

2 Preliminaries

2.1 Notations and Symbols

The main notation used throughout this paper is summarized in Table 2, where the subscript d indicates that the corresponding quantity is associated with the selected

differential trail.

Table 2: Notations and symbols used throughout this paper.

Notation	Description
$\mathcal{D}$	Related-key differential trail.
$\mathcal{I}_d$, $\mathcal{O}_d$	Inbound and outbound phases associated with the differential trail, respectively.
p_d, p_o	Negative binary logarithms of the probabilities of the complete differential trail and the outbound phase, respectively.
$\mathbb{A}$, $\mathbb{P}$, $\mathbb{G}$, $\mathbb{D}$	Index sets of active S-box bytes, fixed plaintext bytes, guessed bytes, and deduced bytes, respectively.
$V[\mathbb{X}]$	Values assigned to the bytes indexed by $\mathbb{X}$, where $\mathbb{X} \in \{\mathbb{A}, \mathbb{P}, \mathbb{G}, \mathbb{D}\}$.
DoF_d	Number of available assignments for instantiating the differential trail.
DoF_I	Number of starting points provided by the inbound phase.
χ	Set of conflict byte equations produced by the GD procedure.

2.2 Specifications of AES

AES [DR02] operates on a 4×4 byte array in column-major order. As illustrated in Fig. 1, the round function consists of four major transformations: SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AK). The MixColumns transformation multiplies each column of the state by an MDS matrix. AES has three variants, namely AES-128, AES-192, and AES-256, with key lengths of 128 bits, 192 bits, and 256 bits, respectively.

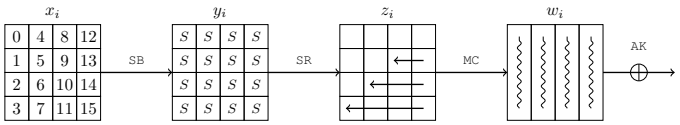


Figure 1: The round function of AES

The key schedules for the three variants are given in Fig. 2.

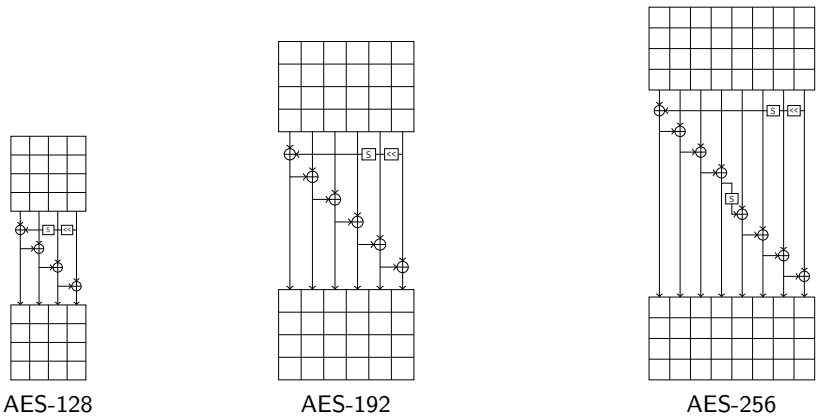


Figure 2: The key schedules of AES.

2.3 Hash Modes

Block-cipher-based hash functions construct a compression function from a block cipher by combining the block-cipher output with one of its inputs. Two commonly used constructions are the Davies–Meyer (DM) mode and the Matyas–Meyer–Oseas (MMO) mode.

In the DM mode, the message block m_i is used as the block-cipher key, while the previous chaining value h_{i-1} is used as the plaintext and is fed forward to the output:

$$h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1}.$$

In the MMO mode, the roles of the message block and chaining value are exchanged:

$$h_i = E_{h_{i-1}}(m_i) \oplus m_i.$$

Hence, by denoting the plaintext and key inputs of the block cipher by P and K, respectively, both modes can be represented by the common algebraic form

$$F(P, K) = E_K(P) \oplus P.$$

The structures of the two hashing modes are illustrated in Fig. 3.

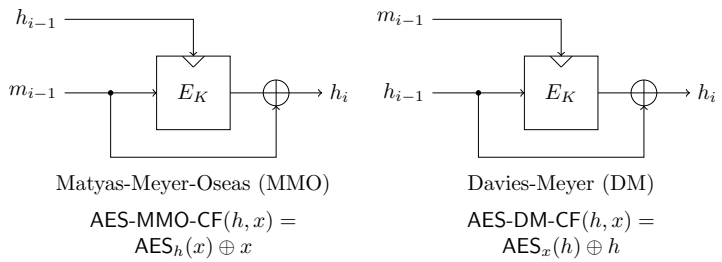


Figure 3: The Davies–Meyer and Matyas–Meyer–Oseas hash modes.

2.4 Collision Models

- **Collision.** A collision consists of two distinct messages M and M' such that

$$H(IV, M) = H(IV, M').$$

- **Semi-free-start (SFS) collision.** An SFS collision consists of two distinct inputs (h, M) and (h, M') satisfying

$$H(h, M) = H(h, M'),$$

where the chaining value h can be chosen by the attacker and is not required to equal the prescribed initial value IV.

- **Free-start (FS) collision.** An FS collision consists of two pairs (h, M) and (h', M') satisfying

$$H(h, M) = H(h', M'),$$

where the attacker can choose both chaining values and $h \neq h'$. In particular, a one-block FS collision for the DM mode satisfies

$$E_M(h) \oplus h = E_{M'}(h') \oplus h'.$$

- **Free-start differential q-multicollision.** Biryukov *et al.* [BKN09] introduced the free-start differential q-multicollision, whose objective is to find q pairs

$$\{(P_i, K_i)\}_{i=1}^{q}, \qquad q \geq 2,$$

such that, for fixed differences Δ_P, Δ_K, and Δ_C,

$$E_{K_i}(P_i) \oplus E_{K_i \oplus \Delta_K}(P_i \oplus \Delta_P) = \Delta_C, \qquad 1 \leq i \leq q.$$

The pairs additionally satisfy

$$P_i \oplus \Delta_P \neq P_j \quad \text{and} \quad K_i \oplus \Delta_K \neq K_j \qquad \text{for } i \neq j.$$

2.5 Rebound Attack

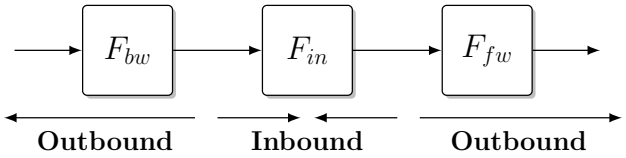


Figure 4: The framework of the rebound attack.

The rebound attack was introduced by Mendel *et al.* [MRST09]. As illustrated in Fig. 4, it divides an internal block cipher or permutation F into three consecutive components,

$$F = F_{\text{fw}} \circ F_{\text{in}} \circ F_{\text{bw}},$$

and consists of an inbound phase and an outbound phase.

- **Inbound phase.** The attacker uses the available degrees of freedom together with a meet-in-the-middle technique to efficiently satisfy the low-probability differential transitions within F_{in}. The solutions obtained in this phase are referred to as starting points for the outbound phase. The available degrees of freedom determine the number of starting points that can be generated.

- **Outbound phase.** Each starting point obtained from the inbound phase is propagated backward through F_{bw} and forward through F_{fw}. The attacker then checks whether the resulting pair satisfies the remaining differential conditions. Since these conditions are fulfilled probabilistically, sufficiently many starting points are tested in a brute-force manner until a valid pair is found.

2.6 Qin *et al.*'s Guess-and-Determine Rebound Attack

Qin *et al.* [QBD25] introduced the guess-and-determine (GD) rebound attack at CRYPTO 2025 by combining the rebound framework with the GD technique of Bouillaguet, Derbez, and Fouque [BDF11]. Its main purpose is to instantiate a multi-round inbound differential by solving the relations among its internal state and key bytes. The resulting solutions are then propagated through the remaining rounds and used as starting points for the outbound phase.

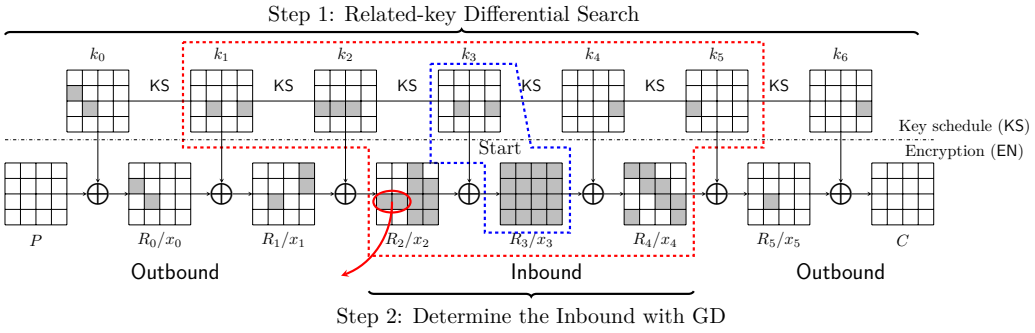


Figure 5: The guess-and-determine rebound framework of Qin *et al.* [QBD25].

Differential trail and phase separation. As illustrated in Fig. 5, the first step is to find a related-key differential trail. Qin *et al.* use the search tool of Gérault *et al.* [GLMS20] for this purpose. Let 2^{-p_d} denote the probability of the complete differential. The degrees of freedom of the differential are the state, plaintext, or key values that can be chosen independently while preserving the prescribed differences. We use DoF_d to denote the number of possible assignments to these free values. Thus, when both the plaintext P and the master key K are unrestricted, they provide

$$\mathrm{DoF}_d = 2^{|P|+|K|}$$

possible assignments before further conditions are imposed, where $|P|$ and $|K|$ denote their bit lengths. Since each assignment satisfies the differential with probability 2^{-p_d}, a basic counting requirement for obtaining at least one valid pair is $\mathrm{DoF}_d \geq 2^{p_d}$.

The differential trail is divided into an inbound phase $\mathcal{I}_d$ and an outbound phase $\mathcal{O}_d$. The inbound phase contains the low-probability conditions to be handled by the GD procedure, whereas the remaining conditions are verified during outward propagation. We write 2^{-p_o} for the probability of satisfying $\mathcal{O}_d$.

Guess-and-determine procedure for the inbound phase. The DDT entries associated with the active S-boxes in $\mathcal{I}_d$ determine their admissible values. Let $\mathbb{A}$ be the index set of these active S-boxes and let $V[\mathbb{A}]$ denote their assigned values. For AES, if $\mathcal{I}_d$ contains s_1 active S-box transitions of probability 2^{-7} and s_2 transitions of probability 2^{-6}, the number of possible assignments to $V[\mathbb{A}]$ is $2^{s_1+2s_2-1}$. For a key-collision attack, the fixed plaintext bytes are additionally represented by $V[\mathbb{P}]$, where $\mathbb{P}$ denotes their index set.

After fixing $V[\mathbb{A}]$ and $V[\mathbb{P}]$, the remaining unknown bytes are divided into guessed bytes $V[\mathbb{G}]$ and bytes $V[\mathbb{D}]$ deduced from the cipher relations. The GD procedure is represented by a function $\mathcal{F}$:

$$\big(V[\mathbb{G}], V[\mathbb{D}], \chi\big) \leftarrow \mathcal{F}\big(\mathcal{I}_d, V[\mathbb{A}], V[\mathbb{P}]\big).$$

During the deductions, some bytes may be computed through more than one path. The resulting consistency conditions form the set of conflict byte equations

$$\chi = \big\{\chi_i\big(V[\mathbb{A}], V[\mathbb{P}], V[\mathbb{G}], V[\mathbb{D}]\big) = 0\big\}_{i=0}^{|\chi|-1},$$

where $|\chi|$ denotes the number of conflict byte equations.

Starting points. Assuming that each conflict byte equation imposes an independent 8-bit condition, the number of inbound solutions in the key-collision setting is

$$\mathrm{DoF}_I = \frac{2^{s_1+2s_2-1} \cdot 2^{8|\mathbb{G}|}}{2^{8|\chi|}}.$$

These solutions serve as starting points for the outbound phase. Consequently, the condition

$$\mathrm{DoF}_I \geq 2^{p_o}$$

ensures that at least one starting point is expected to satisfy $\mathcal{O}_d$.

Generating all candidates requires traversing the possible assignments to both $V[\mathbb{A}]$ and $V[\mathbb{G}]$. Before filtering the conflict equations, the corresponding time complexity is approximately

$$\mathcal{T}_{\mathrm{DoF}_I} = 2^{s_1+2s_2-1} \cdot 2^{8|\mathbb{G}|}.$$

3 A Precomputation-Based Method for Solving Conflicts in GD Rebound Attacks

3.1 Conflicts Related to Guessed Variables

Recall that χ denotes the set of conflict byte equations produced by the GD procedure. We distinguish these equations according to their dependence on the guessed variables $V[\mathbb{G}]$. Let $\widetilde{\chi} \subseteq \chi$ denote the subset of conflicts that are independent of $V[\mathbb{G}]$. Such conflicts are handled separately by selecting appropriate admissible values for the active S-boxes and are therefore outside the scope of this work. We focus on the remaining conflicts $\chi_{\mathbb{G}} = \chi \setminus \widetilde{\chi}$, which depend on one or more guessed variables.

Consider a byte $g_k \in V[\mathbb{G}]$ guessed at Step k. As illustrated in Fig. 6, suppose that, in a subsequent step, an internal byte Y is derived through two computation paths:

$$Y = F_0(g_k, \mathbf{C}_0) \qquad \text{and} \qquad Y = F_1(g_k, \mathbf{C}_1),$$

where $\mathbf{C}_0$ and $\mathbf{C}_1$ contain only values determined before g_k is assigned. The corresponding conflict is

$$F_0(g_k, \mathbf{C}_0) = F_1(g_k, \mathbf{C}_1).$$

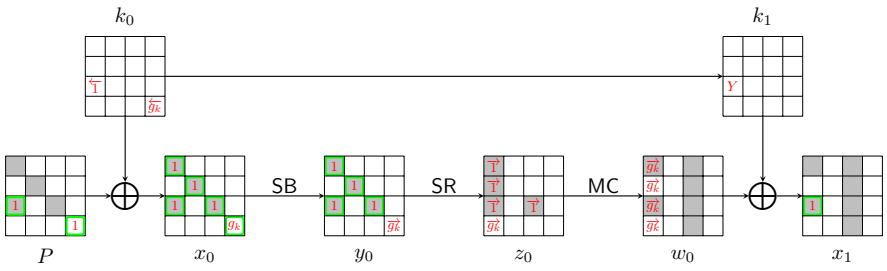


Figure 6: Illustration of a conflict related to a guessed byte g_k.

In Fig. 6, the symbol 1 marks the state bytes whose values have already been determined from the active-S-box conditions and other prescribed conditions. The symbols $\overrightarrow{1}$ and $\overleftarrow{1}$ mark the bytes obtained by computing forward and backward, respectively, from these initially determined bytes.

In this example, $g_k = x_0[15]$ and hence $k_0[15] = g_k \oplus P[15]$. The key relation gives

$$Y = k_0[2] \oplus \mathrm{SB}(g_k \oplus P[15]) \oplus const,$$

whereas the round-function relation gives

$$Y = x_1[2] \oplus z_0[0] \oplus z_0[1] \oplus 02 \cdot z_0[2] \oplus 03 \cdot \mathrm{SB}(g_k).$$

Consequently, the guessed value of g_k must satisfy

$$\begin{aligned} k_0[2] \oplus \mathrm{SB}(g_k \oplus P[15]) \oplus const = x_1[2] \oplus z_0[0] \oplus z_0[1] \\ \oplus\, 02 \cdot z_0[2] \oplus 03 \cdot \mathrm{SB}(g_k). \end{aligned} \tag{1}$$

In a GD procedure, g_k is assigned independently and the conflict equation is checked afterward. As with the other byte conflicts in the GD procedure, this equation is satisfied with probability 2^{-8}.

Instead of guessing g_k independently and subsequently rejecting inconsistent candidates, we aim to determine a valid assignment from the values available before Step k. However, the propagation from g_k to the conflict may pass through several nonlinear layers and involve multiple undetermined intermediate variables, making it difficult to solve the conflict directly for g_k. We therefore retain these intermediate variables and resolve the resulting system of byte equations through precomputation.

3.2 Precomputation for Solving Conflict Equations

When the propagation from a guessed byte to a conflict passes through multiple nonlinear layers, the resulting consistency condition may involve several undetermined intermediate variables. In this case, directly eliminating these variables to obtain a single equation in g_k may be impractical. Instead, we retain the intermediate variables and represent the conflict by a system of byte equations.

Before assigning the guessed byte g_k at Step k, let

$$\mathbf{U} = (X_0, \ldots, X_{m-1}) \in \mathbb{F}_{2^8}^m$$

denote the vector of undetermined byte variables involved in the conflict, one of which is g_k. By collecting the relations among these variables and moving all previously determined terms to the right-hand side, the system can be written as

$$\begin{cases} F_0(\mathbf{U}) = C_0, \\ F_1(\mathbf{U}) = C_1, \\ \quad \vdots \\ F_{m-1}(\mathbf{U}) = C_{m-1}, \end{cases} \tag{2}$$

where each C_i is computed only from values determined before Step k. The functions $F_0, \ldots, F_{m-1}$ are determined by the key relations or round-function relations and may contain several nonlinear operations.

For example, the conflict in Eq. (1) can be represented in the form of Eq. (2). By introducing

$$X_0 = g_k \qquad \text{and} \qquad X_1 = g_k \oplus P[15],$$

we obtain

$$\begin{cases} \mathrm{SB}(X_1) \oplus 03 \cdot \mathrm{SB}(X_0) = C_0, \\ X_1 \oplus X_0 = C_1, \end{cases}$$

where

$$\begin{aligned} C_0 &= k_0[2] \oplus const \oplus x_1[2] \oplus z_0[0] \oplus z_0[1] \oplus 02 \cdot z_0[2], \\ C_1 &= P[15]. \end{aligned}$$

Thus, this example corresponds to $m = 2$ with $\mathbf{U} = (X_0, X_1)$ and $\mathbf{C} = (C_0, C_1)$.

Define the mapping

$$\Phi : \mathbb{F}_{2^8}^m \to \mathbb{F}_{2^8}^m, \qquad \Phi(\mathbf{U}) = \big(F_0(\mathbf{U}), \ldots, F_{m-1}(\mathbf{U})\big).$$

In the preprocessing phase, we enumerate the possible values of $\mathbf{U}$ and construct a lookup table indexed by $\mathbf{C} = (C_0, \ldots, C_{m-1})$. For each value of $\mathbf{C}$ for which a solution exists, the table stores one corresponding solution $\mathbf{U}$ satisfying

$$\Phi(\mathbf{U}) = \mathbf{C}.$$

If all the other variables can be recovered directly after a particular byte X_j of $\mathbf{U}$ is determined, it is sufficient to store only the corresponding value of X_j.

During the online GD procedure, all bytes in $\mathbf{C}$ are known before Step k and are therefore used directly as the table index. A retrieved table value gives either a complete solution to Eq. (2) or one byte from which the complete solution $\mathbf{U}$ can be recovered. The lookup and the subsequent recovery require only a constant number of byte operations, and hence the online cost remains $O(1)$.

Since $\mathbf{C}$ consists of m bytes, the table contains 2^{8m} entries. Its preprocessing time and memory complexities are therefore both $O(2^{8m})$, up to the size of each table entry.

3.3 Memory Reduction Based on Triangular Dependencies

The precomputation method described above requires a table indexed by the m-byte vector $\mathbf{C} = (C_0, \ldots, C_{m-1})$ and therefore requires up to 2^{8m} table entries. When the GD procedure contains additional guessed bytes whose values remain controllable, these degrees of freedom can be used to fix some components of $\mathbf{C}$, thereby reducing the number of the table entries.

Let $\mathbf{u} = (u_0, \ldots, u_{r-1})$ denote r controllable guessed bytes. We analyze the dependencies of the components of $\mathbf{C}$ on these bytes. After appropriately reordering the selected components and guessed bytes, suppose that they form the following triangular dependency structure:

$$\begin{cases} C_{i_0} = G_0(u_0; \mathbf{D}_0), \\ C_{i_1} = G_1(u_0, u_1; \mathbf{D}_1), \\ \quad \vdots \\ C_{i_{r-1}} = G_{r-1}(u_0, \ldots, u_{r-1}; \mathbf{D}_{r-1}), \end{cases} \tag{3}$$

where each $\mathbf{D}_j$ contains only values that have already been determined before u_j is assigned.

The controllable bytes are assigned sequentially. For each j, after $u_0, \ldots, u_{j-1}$ have been determined, we select u_j such that C_{i_j} takes a prescribed value $c^*_{i_j}$. This requires the corresponding equation to be efficiently solvable with respect to u_j. Owing to the triangular dependency structure, the subsequent assignments of $u_{j+1}, \ldots, u_{r-1}$ do not change the previously fixed components $C_{i_0}, \ldots, C_{i_j}$.

After applying this procedure successively, the r components $C_{i_0}, \ldots, C_{i_{r-1}}$ are fixed and need not be included in the table index. The table is therefore indexed only by the remaining $m - r$ components of $\mathbf{C}$, reducing the number of entries from 2^{8m} to

$$2^{8(m-r)}.$$

The resulting conflict can still be resolved by a single table lookup. Since the controllable bytes can be determined sequentially with a fixed number of operations, the online complexity remains $O(1)$. Each fixed guessed byte reduces the available degrees of freedom by at most 8 bits, and the reduction is exactly 8 bits when its value is uniquely determined.

4 Free-start Collision Attack on Full-Round AES-256-DM and AES-256-MMO

For both the Davies–Meyer and Matyas–Meyer–Oseas constructions, let P and K denote the plaintext and key inputs of AES, respectively. Although P and K play different roles

in the two constructions, their compression functions can be expressed in the common form $F(P,K) = \mathsf{AES}_K(P) \oplus P$. Therefore, finding a free-start collision amounts to finding two input pairs (P,K) and $(P \oplus \Delta P, K \oplus \Delta K)$ such that

$$\mathsf{AES}_K(P) \oplus P = \mathsf{AES}_{K\oplus\Delta K}(P \oplus \Delta P) \oplus (P \oplus \Delta P).$$

Equivalently, the collision condition is

$$\mathsf{AES}_K(P) \oplus \mathsf{AES}_{K\oplus\Delta K}(P \oplus \Delta P) = \Delta P. \tag{4}$$

To satisfy Eq. (4), we employ the full-round differential trail $\mathcal{D}$ from [TQHD26], as shown in Fig. 7. We define the first eight rounds of the key schedule together with the first five rounds of the encryption path as the inbound phase $\mathcal{I}_d$, while the remaining parts constitute the outbound phase $\mathcal{O}_d$. The differential probabilities of the inbound and outbound phases are 2^{-214} and 2^{-60}, respectively.

Since the plaintext input P is unrestricted in the free-start collision setting, its bytes are not initially marked as known in the guess-and-determine procedure. The GD procedure for $\mathcal{D}$ is described in Sect. 4.1, and the corresponding equations are summarized in Table 3.

4.1 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d$

As marked by 1 in Fig. 8, the $V[\mathbb{A}] = (x_0[0-15], x_1[0,3,4,8,9,14], x_2[0,4], x_3[0,4], x_4[0], k_6[12], k_7[12-15])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-15]$, $\Delta y_0[0-15]$, $\Delta x_1[0,3,4,8,9,14]$, $\Delta y_1[0,3,4,8,9,14]$, $\Delta x_2[0,4]$, $\Delta y_2[0,4]$, $\Delta x_3[0,4]$, $\Delta y_3[0,4]$, $\Delta x_4[0]$, $\Delta y_4[0]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 3.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute forward to get $z_0[0-15]$, $w_0[0-15]$ (marked by $\overrightarrow{1}$).
   (b) In round 1, compute $k_1[0,3,4,8,9,14] = (x_1 \oplus w_0)[0,3,4,8,9,14]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_1[0,4-8]$, $w_1[4-7]$ (marked by $\overrightarrow{1}$).
   (c) In round 2, compute $k_2[4] = (x_2 \oplus w_1)[4]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_2[0,4]$ (marked by $\overrightarrow{1}$).
2. Guess $x_1[1,2,5-7,10,11,13,15]$ (marked by 2).
   (a) Compute backward to get $k_1[1,2,5-7,10,11,13,15]$ (marked by $\overleftarrow{2}$).
   (b) Compute forward to get $z_1[1-3,9-11,13-15]$ and $w_1[0-3,8-11]$ (marked by $\overrightarrow{2}$). Then compute $k_2[0] = x_2[0] \oplus w_1[0]$ (marked by $\overrightarrow{2}$).
3. Guess $k_2[5,12-15]$ (marked by 3).
   (a) According to the key relations, compute forward to get $k_3[0-11,13-15]$ (marked by $\overrightarrow{3}$) from $k_2[12-15]$ and k_1.
   (b) In round 2, compute forward to get $x_2[5]$ and $z_2[1]$ (marked by $\overrightarrow{3}$).
   (c) From $k_3[0,4]$, compute backward to get $w_2[0,4] = (k_3 \oplus x_3)[0,4]$ (marked by $\overleftarrow{3}$).
   (d) From $k_3[13]$, compute forward to get $k_4[0,4]$ (marked by $\overrightarrow{3}$). Then compute backward to get $w_3[0]$ (marked by $\overleftarrow{3}$).
4. According to $\mathrm{SB}(k_6[12]) \oplus k_7[12] = k_3[4] \oplus k_3[12]$, deduce $k_3[12]$ (marked by 4). Similarly, from $k_3[5-7,13-15]$ and $k_7[13-15]$ deduce $k_6[13-15]$ (marked by 4).

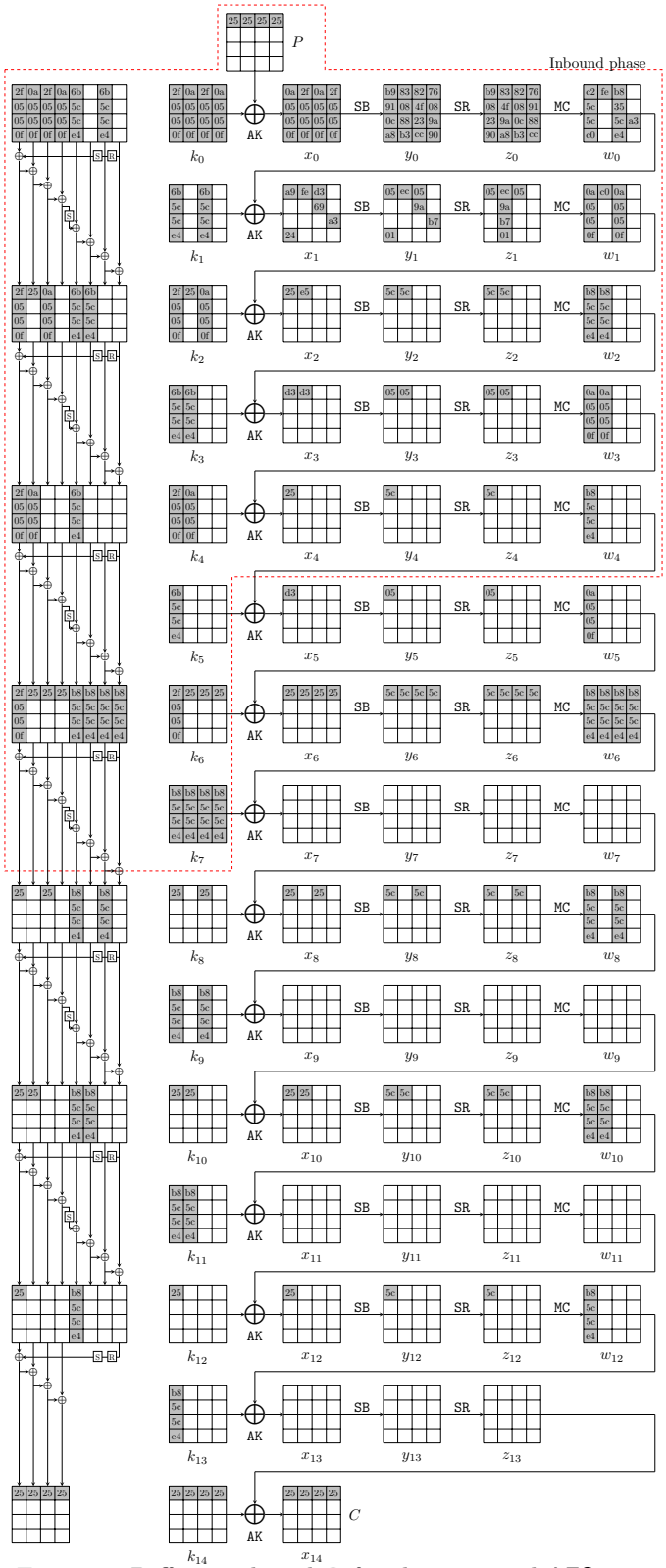


Figure 7: Differential trail $\mathcal{D}$ for the 14-round AES-256.

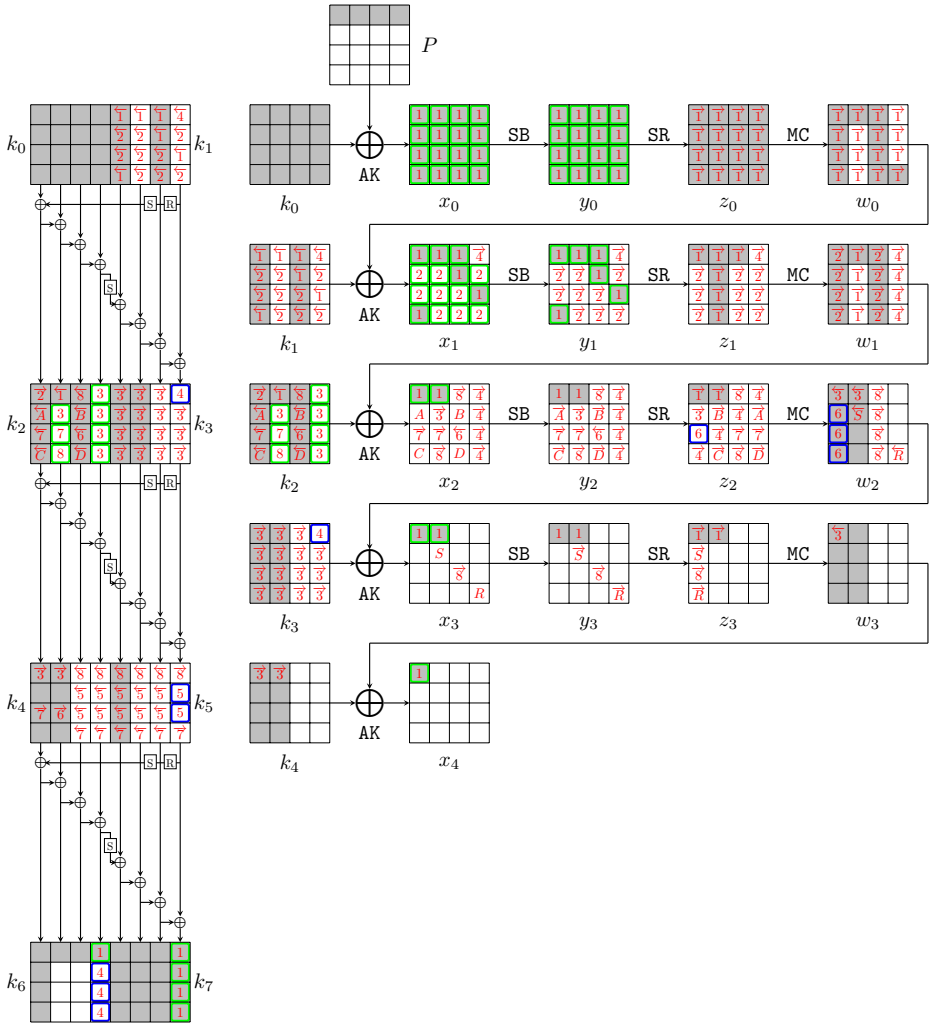


Figure 8: The GD process of $\mathcal{I}_d$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

(a) According to the key relations, compute backward to get $k_1[12]$ (marked by $\overleftarrow{4}$).

(b) In round 1, compute $x_1[12] = k_1[12] \oplus w_0[12]$, then compute forward to get $z_1[12]$ and $w_1[12-15]$ (marked by $\overrightarrow{4}$).

(c) In round 2, compute $x_2[12-15] = (k_2 \oplus w_1)[12-15]$, then compute forward to get $z_2[3,6,9,12]$ (marked by $\overrightarrow{4}$).

5. According to key relations, deduce $\mathrm{SB}(k_5[13]) = k_2[4] \oplus k_2[12] \oplus k_6[12] \oplus const$ (marked by 5). Similarly deduce $k_5[14]$ from $k_2[5,13]$ and $k_6[13]$ (marked by 5).

   (a) From $k_5[13,14]$ compute backward to get $k_5[1,2,5,6,9,10]$ and $k_4[9,10,13,14]$ (marked by $\overleftarrow{5}$).

6. For column 0 over the MC operation in round 2, deduce $z_2[2]$ (marked by 6) from $z_2[0,1,3]$ and $w_2[0]$. Then compute backward to get $x_2[10]$ and $k_2[10]$ (marked by $\overleftarrow{6}$). Then compute $k_4[6] = k_2[10] \oplus k_4[10]$ (marked by $\overrightarrow{6}$).

7. Guess $k_2[6]$ (marked by 7).
   (a) Compute $k_4[2] = k_2[6] \oplus k_4[6]$ (marked by $\overrightarrow{7}$). Then compute backward to get $k_2[2]$ (marked by $\overleftarrow{7}$).
   (b) In round 2, compute forward to get $x_2[2, 6]$ and $z_2[10, 14]$ (marked by $\overrightarrow{7}$).
   (c) According to key relations, compute $k_5[15]$ from $k_2[6, 14]$ and $k_6[14]$ (marked by $\overrightarrow{7}$). Then compute backward to get $k_5[3, 7, 11]$ and $k_4[11, 15]$ (marked by $\overleftarrow{7}$).
8. Guess $k_2[7]$ (marked by 8).
   (a) According to key relations, compute $k_5[12]$ from $k_2[7, 15]$ and $k_6[15]$ (marked by $\overrightarrow{8}$). Then compute backward to get $k_5[0, 4, 8]$ and $k_4[8, 12]$ (marked by $\overleftarrow{8}$). Then from $k_4[8]$ and $k_2[12]$, compute backward to get $k_2[8]$ (marked by $\overleftarrow{8}$).
   (b) In round 2, compute forward to get $x_2[7, 8]$, $z_2[8, 11]$ and $w_2[8 - 11]$(marked by $\overrightarrow{8}$).
   (c) In round 3, compute forward to get $x_3[10]$ and $z_3[2]$ (marked by $\overrightarrow{8}$).

Table 3: Equations in the GD steps of $\mathcal{I}_d$. The waved bytes are guessed in corresponding steps.

1.	$k_1[0, 3, 4, 8, 9, 14] = (x_1 \oplus w_0)[0, 3, 4, 8, 9, 14]$	$k_2[4] = (x_2 \oplus w_1)[4]$
2.	$k_1[1, 2, 5-7, 10, 11, 13, 15] = x_1[1, 2, 5-7, 10, 11, 13, 15] \oplus w_0[1, 2, 5-7, 10, 11, 13, 15]$	
	$k_2[0] = x_2[0] \oplus w_1[0]$	
3.	$k_3[0-3] = \mathrm{SB}(k_2[12-15]) \oplus k_1[0-3]$	
	$k_3[4-11, 13-15] = k_1[4-11, 13-15] \oplus k_3[0-7, 9-11]$	
	$x_2[5] = w_1[5] \oplus k_2[5]$	$w_2[0, 4] = (k_3 \oplus x_3)[0, 4]$
4.	$k_3[12] = \mathrm{SB}(k_6[12]) \oplus k_7[12] \oplus k_3[4]$	$\mathrm{SB}(k_6[13]) = k_3[5] \oplus k_3[13] \oplus k_7[13]$
	$\mathrm{SB}(k_6[14]) = k_3[6] \oplus k_3[14] \oplus k_7[14]$	$\mathrm{SB}(k_6[15]) = k_3[7] \oplus k_3[15] \oplus k_7[15]$
	$k_1[12] = k_3[8] \oplus k_3[12]$	$x_1[12] = k_1[12] \oplus w_0[12]$
	$w_1[12-15] = \mathrm{MC}(z_1[12-15])$	$x_2[12-15] = (k_2 \oplus w_1)[12-15]$
5.	$\mathrm{SB}(k_5[14]) = k_6[13] \oplus k_2[5] \oplus k_2[13] \oplus const$	$\mathrm{SB}(k_5[13]) = k_6[12] \oplus k_2[4] \oplus k_2[12] \oplus const$
	$k_5[1, 2, 5, 6, 9, 10] = (k_3 \oplus k_5)[5, 6, 9, 10, 13, 14]$	
	$\mathrm{SB}(k_4[13, 14]) = (k_3 \oplus k_5)[1, 2]$	$k_4[9, 10] = (k_2 \oplus k_4)[13, 14]$
6.	$w_2[0-3] = \mathrm{MC}(z_2[0-3])$	$k_2[10] = x_2[10] \oplus w_1[10]$
	$k_4[6] = k_2[10] \oplus k_4[10]$	
7.	$k_4[2] = k_2[6] \oplus k_4[6]$	$k_2[2] = \mathrm{SB}(k_3[15]) \oplus k_4[2] \oplus const$
	$x_2[2, 6] = (k_2 \oplus w_1)[2, 6]$	$\mathrm{SB}(k_5[15]) = k_6[14] \oplus k_2[6] \oplus k_2[14] \oplus const$
	$\mathrm{SB}(k_4[15]) = k_3[3] \oplus k_5[3]$	$k_4[11] = k_2[15] \oplus k_4[15]$
8.	$\mathrm{SB}(k_5[12]) = k_6[15] \oplus k_2[7] \oplus k_2[15] \oplus const$	$k_5[0] = k_3[4] \oplus k_3[8] \oplus k_3[12] \oplus k_5[12]$
	$\mathrm{SB}(k_4[12]) = k_3[0] \oplus k_5[0]$	$k_2[8] = k_2[12] \oplus k_4[4] \oplus k_4[12]$
	$x_2[7, 8] = (k_2 \oplus w_1)[7, 8]$	$w_2[8-11] = \mathrm{MC}(z_2[8-11])$
	$x_3[10] = k_3[10] \oplus w_2[10]$	

After Step 8 of the GD process, there remain four undetermined variables in x_2. Denote these variables by $A = x_2[1]$, $B = x_2[9]$, $C = x_2[3]$, and $D = x_2[11]$. By further introducing $R = x_3[15]$ and $S = x_3[5]$, we can derive the following equations:

1. From key relations:

$$(A \oplus w_1[1]) \oplus \mathrm{SB}(k_3[14]) \oplus k_2[5] \oplus (B \oplus w_1[9]) = k_4[9]$$

$$(C \oplus w_1[3]) \oplus \mathrm{SB}(k_3[12]) \oplus k_2[7] \oplus (D \oplus w_1[11]) = k_4[11]$$

2. From MC relations in round 2:

$$w_2[4] = 02 \cdot z_2[4] \oplus 03 \cdot \mathrm{SB}(B) \oplus z_2[6] \oplus \mathrm{SB}(C)$$
$$S \oplus k_3[5] = z_2[4] \oplus 02 \cdot \mathrm{SB}(B) \oplus 03 \cdot z_2[6] \oplus \mathrm{SB}(C)$$
$$R \oplus k_3[15] = 03 \cdot z_2[12] \oplus \mathrm{SB}(A) \oplus z_2[14] \oplus 02 \cdot \mathrm{SB}(D)$$

3. From MC relations in round 3:

$$w_3[0] = 02 \cdot z_3[0] \oplus 03 \cdot \mathrm{SB}(S) \oplus z_3[2] \oplus \mathrm{SB}(R)$$

Thus, we obtain:

$$\begin{cases} A \oplus B = c_0 \\ C \oplus D = c_1 \\ 03 \cdot \mathrm{SB}(B) \oplus \mathrm{SB}(C) = c_2 \\ 02 \cdot \mathrm{SB}(B) \oplus \mathrm{SB}(C) \oplus S = c_3 \\ \mathrm{SB}(A) \oplus 02 \cdot \mathrm{SB}(D) \oplus R = c_4 \\ 03 \cdot \mathrm{SB}(S) \oplus \mathrm{SB}(R) = c_5 \end{cases}$$

where

$$\begin{aligned} c_0 &= w_1[1] \oplus \mathrm{SB}(k_3[14]) \oplus k_2[5] \oplus w_1[9] \oplus k_4[9] \\ c_1 &= w_1[3] \oplus \mathrm{SB}(k_3[12]) \oplus k_2[7] \oplus w_1[11] \oplus k_4[11] \\ c_2 &= w_2[4] \oplus 02 \cdot z_2[4] \oplus z_2[6] \\ c_3 &= k_3[5] \oplus z_2[4] \oplus 03 \cdot z_2[6] \\ c_4 &= k_3[15] \oplus 03 \cdot z_2[12] \oplus z_2[14] \\ c_5 &= w_3[0] \oplus 02 \cdot z_3[0] \oplus z_3[2] \end{aligned}$$

To achieve $O(1)$ online solving time, we construct a lookup table in the preprocessing phase. The right-hand side of the above system is determined by six constants over $\mathbb{F}_{2^8}$, namely $(c_0, c_1, c_2, c_3, c_4, c_5)$. Hence, the table is indexed by $2^{6\cdot 8} = 2^{48}$ possible right-hand-side values.

The table is generated by enumerating all 2^{48} possible values of $(c_0, c_1, c_2, c_3, c_4, B)$. For each choice, the remaining variables are computed as

$$A = B \oplus c_0, \quad C = \mathrm{SB}^{-1}(c_2 \oplus 03 \cdot \mathrm{SB}(B)), \quad D = C \oplus c_1,$$
$$S = c_3 \oplus 02 \cdot \mathrm{SB}(B) \oplus \mathrm{SB}(C), \quad R = c_4 \oplus \mathrm{SB}(A) \oplus 02 \cdot \mathrm{SB}(D),$$

and the corresponding value of c_5 is then obtained from the last equation. The value of B is stored at the resulting index $(c_0, \ldots, c_5)$. If several solutions produce the same index, it is sufficient to retain one corresponding value of B. Therefore, the table contains 2^{48} entries, each storing at most one byte, and the space complexity is 2^{48}.

During the online guess-and-determine procedure, a stored value of B can be retrieved by a single table lookup. Once B is obtained, the remaining variables are recovered using the equations above. Thus, the complete tuple (A, B, C, D, R, S) can be recovered in $O(1)$ online time.

Degree of freedom and complexity.

- In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}]$ for the active bytes from the input/output differences in the $\mathcal{I}_d$. There are 32 active S-boxes, where $s_1 = 22$ S-boxes are of probability 2^{-7}, and $s_2 = 10$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}]| = 2^{s_1+2s_2-1} = 2^{41}$ choices for the bytes marked by 1 in Fig. 8.

- Given $V[\mathbb{A}]$, 16 bytes $V[\mathbb{G}] = (x_1[1, 2, 5-7, 10, 11, 13, 15], k_2[5-7, 12-15])$ (marked by a wavy line) are guessed in Steps 2, 3, 7 and 8. After applying the precomputation-based method, no unresolved probabilistic conflict remains in $\mathcal{I}_d$. Hence, no additional probability loss is introduced by the conflict equations. Hence, the degrees of freedom of the $\mathcal{I}_d$ are $\mathrm{DoF}_{\mathcal{I}_d} = \frac{2^{41+16\times 8}}{2^{8\times 0}} = 2^{169}$.
- Since there is no conflict in the inbound phase, the time of the GD to find one starting point is $\mathcal{T}_{\mathsf{GD}} = 1$. Since the probability of the outbound phase is $2^{-p_o} = 2^{-60}$, the degrees of freedom of $\mathcal{I}_d$ are sufficient to ensure the existence of a collision.

4.2 Memory Reduction

The original equation system contains six byte constants $(c_0, \ldots, c_5)$, and directly storing a solution for every possible right-hand side requires 2^{48} table entries, which is impractical. We therefore apply the memory-reduction method described in Sect. 3.3. More precisely, we replace the guesses of $k_2[14]$, $k_2[6]$, and $k_2[7]$ with deterministic assignments performed in this order. Under this ordering, c_0 is fixed first, followed by c_4 and c_1, while the previously fixed constants remain unchanged. The dependencies of the selected constants on the three controllable bytes can be written as

$$\begin{cases} c_0 = G_0(k_2[14]; \mathbf{D}_0), \\ c_4 = G_1(k_2[14], k_2[6]; \mathbf{D}_1), \\ c_1 = G_2(k_2[14], k_2[6], k_2[7]; \mathbf{D}_2), \end{cases}$$

where each $\mathbf{D}_i$ contains values determined before the corresponding byte is assigned. Hence, these dependencies form the triangular structure defined in Eq. (3).

We first consider

$$c_0 = w_1[1] \oplus \mathrm{SB}(k_3[14]) \oplus k_2[5] \oplus w_1[9] \oplus k_4[9].$$

The bytes $w_1[1]$ and $w_1[9]$ are determined in Step 2, while $k_2[5]$ is available before $k_2[14]$ is determined. After guessing $k_2[12]$ and $k_2[13]$ in Step 3, the subsequent deterministic deductions provide $k_4[9]$, whose value does not depend on $k_2[14]$. Therefore, instead of guessing $k_2[14]$, we first set the target value

$$k_3^*[14] = \mathrm{SB}^{-1}\left(w_1[1] \oplus k_2[5] \oplus w_1[9] \oplus k_4[9]\right).$$

According to the AES-256 key schedule,

$$k_3[14] = \mathrm{SB}(k_2[14]) \oplus k_1[14] \oplus k_1[10] \oplus k_1[6] \oplus k_1[2].$$

We thus determine $k_2[14]$ as

$$k_2[14] = \mathrm{SB}^{-1}\left(k_3^*[14] \oplus k_1[14] \oplus k_1[10] \oplus k_1[6] \oplus k_1[2]\right),$$

which guarantees $c_0 = 0$.

We next consider

$$c_4 = k_3[15] \oplus 03 \cdot z_2[12] \oplus z_2[14].$$

Before Step 7, $k_3[15]$ and $z_2[12]$ have already been determined, while

$$z_2[14] = \mathrm{SB}(k_2[6] \oplus w_1[6]).$$

Instead of guessing $k_2[6]$ in Step 7, we determine it as

$$k_2[6] = w_1[6] \oplus \mathrm{SB}^{-1}\left(k_3[15] \oplus 03 \cdot z_2[12]\right),$$

which guarantees $c_4 = 0$. Since $k_2[6]$ does not affect c_0, this assignment preserves the condition $c_0 = 0$.

Finally, we consider

$$c_1 = w_1[3] \oplus \mathrm{SB}(k_3[12]) \oplus k_2[7] \oplus w_1[11] \oplus k_4[11].$$

The bytes $w_1[3]$ and $w_1[11]$ are determined in Step 2, $k_3[12]$ is obtained from the previous deductions, and $k_4[11]$ is determined after $k_2[6]$ has been fixed in Step 7. We postpone the determination of $k_2[7]$ to Step 8 and set

$$k_2[7] = w_1[3] \oplus \mathrm{SB}(k_3[12]) \oplus w_1[11] \oplus k_4[11],$$

which guarantees $c_1 = 0$. Since $k_2[7]$ affects neither c_0 nor c_4, this final assignment preserves both previously imposed conditions. The three deterministic assignments therefore guarantee

$$c_0 = c_1 = c_4 = 0.$$

Under these conditions, the original equation system reduces to

$$\begin{cases} A \oplus B = 0, \\ C \oplus D = 0, \\ 03 \cdot \mathrm{SB}(B) \oplus \mathrm{SB}(C) = c_2, \\ 02 \cdot \mathrm{SB}(B) \oplus \mathrm{SB}(C) \oplus S = c_3, \\ \mathrm{SB}(A) \oplus 02 \cdot \mathrm{SB}(D) \oplus R = 0, \\ 03 \cdot \mathrm{SB}(S) \oplus \mathrm{SB}(R) = c_5. \end{cases}$$

In particular, $A = B$ and $C = D$. For each choice of (c_2, c_3, B), the remaining variables can be computed as

$$\begin{aligned} C = D &= \mathrm{SB}^{-1}\left(c_2 \oplus 03 \cdot \mathrm{SB}(B)\right), \\ S &= c_2 \oplus c_3 \oplus \mathrm{SB}(B), \\ R &= 02 \cdot c_2 \oplus 07 \cdot \mathrm{SB}(B). \end{aligned}$$

We therefore construct a lookup table indexed by (c_2, c_3, c_5), with each entry storing at most one corresponding value of B The table is generated by enumerating all 2^{24} possible values of (c_2, c_3, B), computing the corresponding value of c_5, and storing B at the resulting index. During the online GD procedure, (c_2, c_3, c_5) is used directly as the table index. A retrieved value of B allows all the remaining variables to be recovered using a constant number of byte operations.

This construction is an instance of the triangular memory-reduction method with $m = 6$ and $r = 3$. Fixing c_0, c_4, and c_1 leaves only (c_2, c_3, c_5) as the table index and reduces the number of table entries from $2^{8m} = 2^{48}$ to $2^{8(m-r)} = 2^{24}$. Accordingly,

$$\mathcal{T}_{\mathrm{pre}} = 2^{24}, \qquad \mathcal{M}_{\mathrm{pre}} = 2^{24}, \qquad \mathcal{T}_{\mathsf{GD}} = 1.$$

Thus, the preprocessing time and memory complexities are reduced from 2^{48} to 2^{24}, while the online solving time remains $O(1)$. Fixing the three guessed bytes reduces the available degrees of freedom by $3 \cdot 8 = 24$ bits.

4.3 Free Start Collision Attack on the 14-Round AES-256

Based on the differential trail $\mathcal{D}$ shown in Fig. 7, we construct a free-start collision attack on the 14-round AES-256. In the inbound phase, the preprocessing step requires 2^{24} time and 2^{24} memory to build the lookup table for the remaining system of equations. After

this preprocessing, the guess-and-determine procedure can find one starting point with time complexity $\mathcal{T}_{\mathsf{GD}} = 1$.

For the outbound phase, the probability is 2^{-60}. Hence, about 2^{60} starting points are required to obtain one valid collision, and the corresponding time complexity is 2^{60}. Since three additional guessed bytes are fixed in the memory-reduction procedure, the degrees of freedom in the inbound phase are reduced from 2^{169} to $2^{169-3\times 8} = 2^{145}$. Nevertheless, 2^{145} is still much larger than 2^{60}, and hence the available starting points remain sufficient to find a collision in expectation.

Therefore, the overall time complexity of the free-start collision attack is 2^{60}, and the memory complexity is 2^{24}.

4.4 Improving Biryukov et al.'s q-Multicollision Attack

Biryukov *et al.* [BKN09] constructed a differential q-multicollision for the full AES-256 by producing q right pairs following the same differential trail. Their trail contains 41 active S-boxes. By fixing admissible inputs to 30 of them and applying the triangulation algorithm, they obtained 18 free byte variables, while the remaining 11 active S-boxes were not covered by the triangulation procedure. A randomly generated candidate satisfies the transitions of these remaining S-boxes with probability 2^{-67}. Consequently, finding one right pair requires 2^{67} operations in expectation, and independently repeating this procedure q times gives a differential q-multicollision with time complexity $q \cdot 2^{67}$.

Our free-start collision attack can be extended in the same manner. The differential trail $\mathcal{D}$ fixes the differences ΔP, ΔK, and $\Delta C = \Delta P$. Hence, each valid pair generated by the attack satisfies

$$E_{K_i}(P_i) \oplus E_{K_i \oplus \Delta K}(P_i \oplus \Delta P) = \Delta P.$$

By independently assigning fresh values to the free variables, we can generate q distinct pairs $\{(P_1, K_1), (P_2, K_2), \ldots, (P_q, K_q)\}$ such that

$$\begin{aligned} E_{K_1}(P_1) \oplus E_{K_1 \oplus \Delta K}(P_1 \oplus \Delta P) &= \cdots \\ &= E_{K_q}(P_q) \oplus E_{K_q \oplus \Delta K}(P_q \oplus \Delta P) = \Delta P. \end{aligned}$$

Therefore, these pairs form a differential q-multicollision.

The lookup table used in the inbound phase is constructed only once, requiring 2^{24} time and 2^{24} memory. After preprocessing, each right pair is obtained with time complexity 2^{60}. The total time complexity is thus

$$2^{24} + q \cdot 2^{60} = O(q \cdot 2^{60}),$$

while the memory complexity remains 2^{24}. Moreover, the inbound phase provides 2^{145} degrees of freedom after the memory-reduction procedure, which is sufficient to generate the required distinct pairs as long as $q \cdot 2^{60} \leq 2^{145}$. Therefore, our attack reduces the time complexity of constructing the differential q-multicollision from $q \cdot 2^{67}$ to $q \cdot 2^{60}$.

5 Free-start Collision Attack on 12-Round AES-256-DM and AES-256-MMO

We consider the same type of free-start collision as in the attack on the 14-round AES-256. More precisely, the objective is to find two input pairs (P, K) and $(P \oplus \Delta P, K \oplus \Delta K)$ satisfying

$$E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \Delta P) = \Delta P,$$

which yields a collision after the Davies–Meyer feed-forward.

We construct the 12-round differential trail by removing the last two rounds of the full-round trail and fixing the active positions in rounds 2–11 to match those of the full-round trail. Under these constraints, we use automated tools to search for a 12-round truncated differential characteristic. We then select the inbound and outbound phases and further search for concrete differences to optimize the outbound differential probability. The resulting trail, denoted by $\mathcal{D}_{12}$, is provided in Fig. 10 in Appendix A. The trail contains 41 active S-boxes, of which 36 belong to the encryption path and 5 belong to the key schedule. We divide $\mathcal{D}_{12}$ into an inbound phase $\mathcal{I}_d$ and an outbound phase $\mathcal{O}_d$. The first five rounds of the encryption path together with the corresponding part of the key schedule form $\mathcal{I}_d$, whereas the remaining seven rounds form $\mathcal{O}_d$. Their differential probabilities are 2^{-205} and 2^{-54}, respectively, resulting in an overall trail probability of 2^{-259}.

5.1 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d$

As marked by 1 in Fig. 9, the $V[\mathbb{A}] = (x_0[0-15], x_1[0,5,8,10,15], x_2[0,4], x_3[0,4], x_4[0,4], k_6[12], k_7[12-15])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-15]$, $\Delta y_0[0-15]$, $\Delta x_1[0,5,8,10,15]$, $\Delta y_1[0,5,8,10,15]$, $\Delta x_2[0,4]$, $\Delta y_2[0,4]$, $\Delta x_3[0,4]$, $\Delta y_3[0,4]$, and also the fixed differences in the key schedule.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute forward to get $z_0[0-15]$, $w_0[0-15]$ (marked by $\overrightarrow{1}$).
   (b) In round 1, compute $k_1[0,5,8,10,15] = (x_1 \oplus w_0)[0,5,8,10,15]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_1[0-4,8]$, $w_1[0-4]$ (marked by $\overrightarrow{1}$).
   (c) In round 2, compute $k_2[0] = (x_2 \oplus w_1)[0]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_2[0,4]$ (marked by $\overrightarrow{1}$).
2. Guess $x_1[1-4,6,7,9,11,13,14]$ (marked by 2).
   (a) Compute backward to get $k_1[1-4,6,7,9,11,13,14]$ (marked by $\overleftarrow{2}$).
   (b) Compute forward to get $z_1[4-7,9-11,13-15]$ and $w_1[4-11]$ (marked by $\overrightarrow{2}$). Then compute $k_2[4] = x_2[4] \oplus w_1[4]$ (marked by $\overrightarrow{2}$).
3. After Step 2, the state of the guess-and-determine procedure is identical to that obtained after Step 2 in Sect. 4.1. Therefore, the remaining GD steps are the same as those described previously.

Degree of freedom and complexity.

- In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}]$ for the active bytes from the input/output differences in the $\mathcal{I}_d$. There are 31 active S-boxes, where $s_1 = 19$ S-boxes are of probability 2^{-7}, and $s_2 = 12$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}]| = 2^{s_1+2s_2-1} = 2^{42}$ choices for the bytes marked by 1 in Fig. 9.
- Given $V[\mathbb{A}]$, 17 bytes $V[\mathbb{G}] = (x_1[1-4,6,7,9,11,13,14], k_2[5-7,12-15])$ (marked by a wavy line) are guessed in Steps 2, 3, 7 and 8. After applying the precomputation-based method, no unresolved probabilistic conflict remains in $\mathcal{I}_d$. Hence, no additional probability loss is introduced by the conflict equations. Hence, the degrees of freedom of the $\mathcal{I}_d$ are $\mathrm{DoF}_{\mathcal{I}_d} = \frac{2^{42+17\times 8}}{2^{8\times 0}} = 2^{178}$.
- Since there is no conflict in the inbound phase, the time of the GD to find one starting point is $\mathcal{T}_{\mathsf{GD}} = 1$. Since the probability of the outbound phase is $2^{-p_o} = 2^{-54}$, the degrees of freedom of $\mathcal{I}_d$ are sufficient to ensure the existence of a collision.

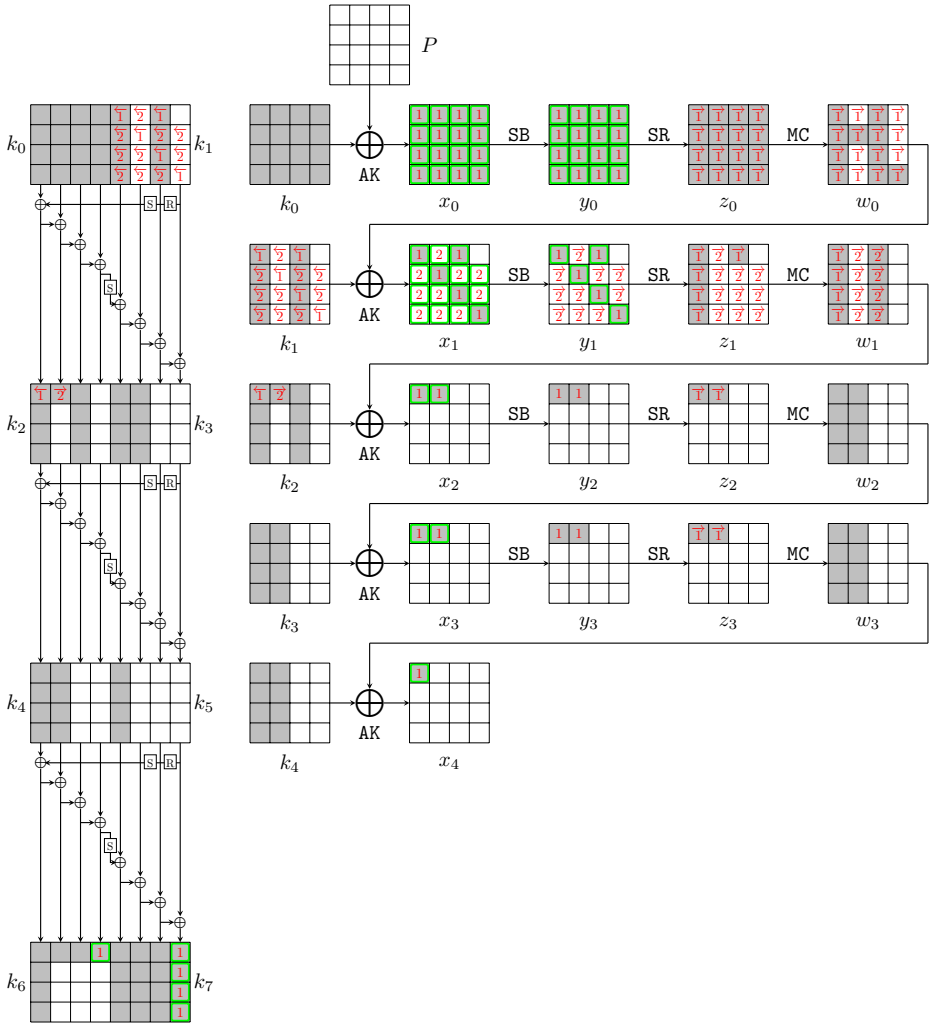


Figure 9: The GD process of $\mathcal{I}_d$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

5.2 Free Start Collision Attack on the 12-Round AES-256

Based on the differential trail $\mathcal{D}_{12}$ shown in Fig. 10, we construct a free-start collision attack on the 12-round AES-256. As explained above, after Step 2, the GD state and all subsequent deductions are identical to those in the 14-round attack. Consequently, the remaining system of equations and the triangular dependencies used for memory reduction remain unchanged. By applying the same memory-reduction technique as in Sect. 4.2, the preprocessing phase requires 2^{24} time and 2^{24} memory, after which one starting point can be generated with $\mathcal{T}_{\mathsf{GD}} = 1$.

The outbound phase has probability 2^{-54}, and hence approximately 2^{54} starting points are required to obtain a collision. The inbound phase initially provides 2^{178} degrees of freedom. After fixing three guessed bytes for memory reduction, this number becomes $2^{178-3\times 8} = 2^{154}$, which remains sufficient since $2^{154} \gg 2^{54}$.

Therefore, the free-start collision attack on the 12-round AES-256 has time complexity 2^{54} and memory complexity 2^{24}.

6 Conclusion

In this work, we further developed the guess-and-determine (GD) rebound methodology by introducing a precomputation-based method for resolving conflicts involving guessed variables, together with a memory-reduction technique based on triangular dependencies. Applying this methodology, we obtained the first classical free-start collision attack applicable to both full-round AES-256-DM and AES-256-MMO, with time complexity 2^{60} and memory complexity 2^{24}. Our attack also improves the free-start differential q-multicollision construction of Biryukov *et al.* [BKN09] from $q \cdot 2^{67}$ to $q \cdot 2^{60}$ time while requiring 2^{24} memory. Furthermore, for the 12-round variants of both hashing modes, we obtained free-start collision attacks with time complexity 2^{54} and memory complexity 2^{24}. We also experimentally validated our method through a partial attack following the 12-round differential trail, verifying the expected differences up to the input of the round-10 S-boxes. These results demonstrate that resolving conflicts through precomputation can substantially extend the applicability of the GD rebound framework to classical collision attacks on full-round AES-based hashing modes.

Acknowledgement.

We thank the anonymous reviewers for their insightful comments. This work is supported by the National Key R&D Program of China (2024YFA1013000), the National Natural Science Foundation of China (U25B2014, 62272257, 62302250), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM114), and the Zhongguancun Laboratory.

7 Bibliography

References

[All07] ZigBee Alliance. ZigBee 2007 specification. *Online: http://www.zigbee.org/*, 2007.

[BBG+09] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. SHA-3 proposal: ECHO. *Submission to NIST (updated)*, page 113, 2009.

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In *CRYPTO 2011*, volume 6841, pages 169–187. Springer, 2011. doi:10.1007/978-3-642-22792-9_10.

[BDG+21] Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In *EUROCRYPT 2021*, volume 12696, pages 771–804. Springer, 2021.

[BGLP24] Zhenzhen Bao, Jian Guo, Shun Li, and Phuong Pham. Automatic quantum multi-collision distinguishers and rebound attacks with triangulation algorithm. In *ACISP 2024*, volume 14896, pages 24–43. Springer, 2024.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *CRYPTO 2009*, volume 5677, pages 231–249. Springer, 2009. doi:10.1007/978-3-642-03356-8_14.

[BR+00] PSLM Barreto, Vincent Rijmen, et al. The whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14, 2000.

[Cam12] Matthew Campagna. A cryptographic suite for embedded systems (suitee). *Internet-Draft*, 2012.

[CGL+24] Shiyao Chen, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang. Diving deep into the preimage security of AES-like hashing. In *EUROCRYPT 2024*, volume 14651, pages 398–426. Springer, 2024. doi:10.1007/978-3-031-58716-0_14.

[CGL+25] Shiyao Chen, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang. Scrutinizing the security of aes-based hashing and one-way functions. In *ASIACRYPT 2025*, volume 16245, pages 157–188. Springer, 2025. doi:10.1007/978-981-95-5018-0_6.

[CL01] Bram Cohen and Ben Laurie. AES-hash. *NIST: Modes of Operation for Symmetric Key Block Ciphers*, 2001. URL: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf.

[Dam89] Ivan Damgård. A design principle for hash functions. In *CRYPTO '89*, pages 416–427, 1989.

[DDKS] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *CRYPTO 2012*, volume 7417, pages 719–740. Springer.

[DDS25] Mathieu Degré, Patrick Derbez, and André Schrottenloher. Simplified meet-in-the-middle preimage attacks on aes-based hashing. *IACR Cryptol. ePrint Arch.*, page 2213, 2025. URL: https://eprint.iacr.org/2025/2213.

[DHS+] Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO 2021*, volume 12827, pages 278–308.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[GKM+09] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. In *Symmetric Cryptography 2009*, 2009.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers. In *IEEE S&P 2020*, pages 825–841, 2020.

[GLMS20] David Gérault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Computing AES related-key differential characteristics with constraint programming. *Artificial Intelligence*, 278:103183, 2020. doi:10.1016/j.artint.2019.103183.

[GP10] Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010*, pages 365–383, 2010.

[ISO10] ISO/IEC. 10118-2:2010 Information technology — Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher. 3rd ed., International Organization for Standardization, Geneve, Switzerland, October, 2010.

[KLMR16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - efficient short-input hashing for post-quantum applications. *IACR Trans. Symmetric Cryptol.*, 2016(2):1–29, 2016. doi:10.13154/tosc.v2016.i2.1-29.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS 2016*, pages 830–842, 2016.

[KPC+20] Hangi Kim, Myungseo Park, Jaehyung Cho, Jihun Kim, and Jongsung Kim. Weaknesses of some lightweight blockciphers suitable for iot systems and their applications in hash modes. *Peer-to-Peer Networking and Applications*, 13(2):489–513, 2020.

[KRT] Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl hash functions. In *FSE 2007*, volume 4593, pages 39–57.

[LMR+09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT 2009*, pages 126–143, 2009.

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO '89, 1989*, pages 218–238, 1989.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009*, pages 260–276, 2009.

[Nay11] María Naya-Plasencia. How to improve rebound attacks. In *CRYPTO 2011*, volume 6841, pages 188–205. Springer, 2011. doi:10.1007/978-3-642-22792-9_11.

[PGV] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93*, volume 773, pages 368–378.

[QBD25] Lingyue Qin, Wenquan Bi, and Xiaoyang Dong. Guess-and-determine rebound: Applications to key collisions on AES. In *CRYPTO 2025*, volume 16004, pages 139–171. Springer, 2025. doi:10.1007/978-3-032-01901-1_5.

[Sas] Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In *FSE 2011*, pages 378–396.

[SLW+10] Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox analysis: Applications to ECHO and grøstl. In *ASIACRYPT 2010*, pages 38–55, 2010.

[TQHD26] Liyuan Tang, Lingyue Qin, Shiqi Hou, and Xiaoyang Dong. Guess-and-determine rebound revisited: full quantum collision attack on aes-256 in dm hash mode. In *Annual International Cryptology Conference*, pages 435–467. Springer, 2026.

[TSI+24] Kodai Taiyama, Kosei Sakamoto, Ryoma Ito, Kazuma Taka, and Takanori Isobe. Key collisions on AES and its applications. *Cryptology ePrint Archive*, 2024.

[WFW+] Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (pseudo) preimage attack on round-reduced Grøstl hash function and others. In *FSE 2012*, pages 127–145.

[ZHQD25] Boxin Zhao, Qingliang Hou, Lingyue Qin, and Xiaoyang Dong. Triangulating meet-in-the-middle attack. In *CRYPTO 2025*, volume 16004, pages 260–292. Springer, 2025. doi:10.1007/978-3-032-01901-1_9.

A Differential Trail for the 12-Round AES-256

Figure 10 presents the complete related-key differential trail $\mathcal{D}_{12}$ used in our attack on the 12-round AES-256.

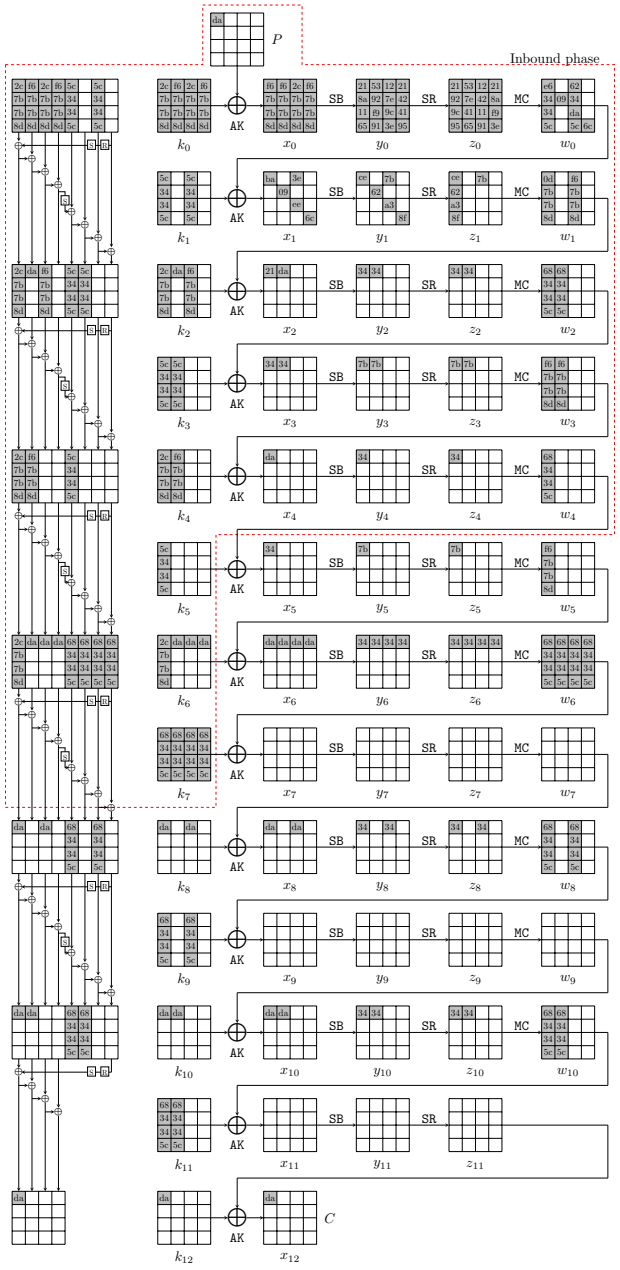


Figure 10: Differential trail $\mathcal{D}_{12}$ for the 12-round AES-256.

B Experimental Verification of the Partial Attack

To validate our GD-based rebound method, we implement a partial attack following the 12-round differential trail $\mathcal{D}_{12}$. The objective is to find two plaintext-key pairs whose

differences follow the trail up to the input of the S-boxes in round 10, without imposing the differential conditions on the two active S-boxes in that round. The expected time complexity of this partial attack is $O(2^{42})$.

Using an NVIDIA V100 GPU, our implementation successfully found such a pair in 67 minutes. The resulting pair and the search and verification code are publicly available at https://github.com/DSwutang/Free-Start-AES-Collision.