Guess-and-Determine Rebound Revisited: Full Quantum Collision Attack on AES-256 in DM Hash Mode

Liyuan Tang[1], Lingyue Qin[1,2,3(✉)], Shiqi Hou[1(✉)], and Xiaoyang Dong[1,2,3(✉)]

[1] Tsinghua University, Beijing, P. R. China
tangly22@mails.tsinghua.edu.cn, {qinly,xiaoyangdong}@tsinghua.edu.cn, seventeenhsq@gmail.com
[2] Zhongguancun Laboratory, Beijing, P. R. China
[3] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, P. R. China

Abstract. At CRYPTO 2025, Qin *et al.* introduced the *guess-and-determine* (GD) *rebound attack*, which integrates the guess-and-determine approach by Bouillaguet, Derbez, and Fouque and the rebound attack by Mendel *et al.* Taking the GD rebound as a building block, this paper introduces several classical and quantum models to convert the semi-free-start (SFS) collision attack or free-start (FS) collision attack into collision attacks on DM hashing mode with AES.

As an application, the first full quantum collision attack on AES-256-DM is proposed. Despite numerous round-reduced quantum or classical attacks proposed against the three popular hash modes MMO/MP/DM with AES over the past two decades, this is the first full attack that targets one of the three fundamental security requirements: collision, (2nd) preimage resistance. Our full attack on AES-256-DM improves the best previous attack by Taiyama *et al.* at ASIACRYPT 2024 by 5 rounds. Besides, some improved results on AES-128-DM and AES-192-DM are also given, which have been verified partially or fully by experiments.

Keywords: Rebound Attack · Guess-and-Determine · AES-256 · Quantum Attack

1 Introduction

Hash functions are among the most important primitives in cryptography due to their wide and crucial applications in digital signatures, message integrity verification, and other cryptographic applications. To support these applications, collision resistance, preimage resistance, and second preimage resistance constitute the three fundamental security requirements for cryptographic hash functions. In this paper, we mainly focus on the security of hash functions built on the block cipher Advanced Encryption Standard (AES) [20] and the like (we call them AES hash modes). Typical examples are the three PGV-modes [61] – Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), and Miyaguchi-Preneel (MP), instantiated

with AES and Merkle-Damgård construction [57,21]. The MMO-mode instantiated with AES is standardized by Zigbee [1] and also suggested by ISO [43] and IETF [15]. The AES-DM hash function was once submitted to NIST in 2001 by Cohen and Laurie as a formal proposal under the name AES-Hash [19]. Recently, many feature-rich cryptographic protocols, *e.g.*, multi-party computation protocols [38,45], use AES hash mode like AES-MMO as building blocks due to its high efficiency when implemented with AES-NI. Besides, many hash functions like Whirlpool [5], Grøstl [31], Grindahl [47], ECHO [6], Haraka v2 [49], etc., adopt block ciphers or permutations with similar features to AES as the underlying building blocks.

There are rigorous and massive cryptanalysis results on AES hash modes, but none of them breaks any of the three fundamental security requirements (collision, preimage, or 2nd preimage) of the hashing modes (DM/MMO/MP) with full AES. At CRYPTO 2009, Biryukov *et al.* [9] introduced the free-start multi-collision attack on full AES-256-DM, *but no full collision attack has been found in the last 17 years*. At FSE 2009, Mendel *et al.* proposed the rebound attack [56], which leads to a collision attack on AES-MMO/MP up to 5 rounds. Later, [35,52] improved the attack to 6 rounds with the Super-Sbox techniques. At FSE 2011, Sasaki [63] introduced the first preimage attack on AES hashing modes with the underlying AES-128 reduced to 7 rounds. Then, the complexity of the attack was marginally improved by Wu *et al.* [68]. Ten years later, at EUROCRYPT 2020, Hosoyamada and Sasaki proposed the quantum collision attacks on 7-round AES-MMO [39], exceeding the classic attacks [35,52] by 1 round. At ASIACRYPT 2020, Dong *et al.* [27] reduced the memory complexity of the quantum attack. At EUROCRYPT 2021, the preimage attack on the hashing modes with AES-128 was finally improved up to 8 rounds via an automatic MitM attack [3]. At CRYPTO 2022, Schrottenloher and Stevens [65] proposed a 7-round quantum meet-in-the-middle attack on AES-128 hashing mode. At CRYPTO 2021 and EUROCRYPT 2024, Dong *et al.* [25] and Chen *et al.* [17] proposed the 10-round preimage attacks on AES-256 or AES-192 in DM mode, respectively. At CRYPTO 2022, the quantum collision attack on the hashing modes was also improved up to 8 rounds of AES-128 [24]. At ASIACRYPT 2024, Taiyama *et al.* systematically proposed (semi-free) collision attacks on DM modes with AES-128/-192/-256, respectively, and a 9-round collision attack on AES-256-DM was introduced [66]. Later in 2025, more round-reduced attacks on AES/Rijndael in DM/MMO/MP modes are given in [18,62]. Besides, the known/chosen-key attacks on AES that closely impact its hash modes are also studied [48,9,30,34].

The Rebound Attack, pioneered by Mendel, Rechberger, Schläffer, and Thomsen at the FSE 2009 [56], stands as a cornerstone methodology for analyzing the collision resistance of AES-like hash functions. Its core insight lies in splitting an attack into two distinct phases: an inbound phase, where degrees of freedom are harnessed to deterministically satisfy portions of a differential characteristic, and an outbound phase, where the remaining aspects of the characteristic are fulfilled probabilistically. To extend the rebound attack, Lamberger *et al.*

introduced a multi-inbound phase framework at ASIACRYPT 2009 [52], leveraging key-related degrees of freedom to connect these phases. Gilbert, Peyrin [35], and Lamberger *et al.* [52] further enhanced the inbound phase by treating two consecutive AES-like rounds as a unified "Super-Sbox". Subsequent advancements followed: at ASIACRYPT 2010, Sasaki *et al.* [64] reduced memory overhead by exploiting the differential properties of non-fully active Super-Sbox; Naya-Plasencia [58] refined the attack at CRYPTO 2011 through advanced algorithms for merging large lists; and Dinur *et al.* cut memory costs further at CRYPTO 2012 with the introduction of the dissection technique [23]. More recently, Dong *et al.* proposed the triangulating rebound attack at CRYPTO 2022, integrating multiple inbound rounds into a "super inbound" phase via triangulation algorithms [46]; Qin *et al.* then extended this lineage at CRYPTO 2025 with the guess-and-determine rebound attack, combining Bouillaguet, Derbez, and Fouque's guess-and-determine strategy [12] with the rebound techniques.

Quantum Attacks. The past decade has witnessed notable advances in the quantum cryptanalysis of symmetric-key primitives, such as the attacks on Feistel [50], EM construction [51,10], MACs and AEAD [44,11], FX construction [53], generic collision or multi-collision finding [16,42,54], k-xor [36,59] and generic attacks on Merkle-Damgård construction [7,26], etc.

At EUROCRYPT 2020, Hosoyamada and Sasaki [39] introduced the first quantum dedicated collision attacks by converting the classical rebound attacks into quantum rebound attacks. This showed that, under their respective bounds of generic algorithms, quantum attacks are able to penetrate more rounds than classical attacks. At ASIACRYPT 2020, Dong *et al.* [27] reduced the requirement of qRAM in the quantum rebound attack by exploiting non-full-active Super-Sbox technique [64], and Flórez-Gutiérrez et al. [29] proposed quantum collision attacks on Gimli. At CRYPTO 2021, Hosoyamada and Sasaki [41] introduced the quantum collision attacks on reduced SHA-2. At ASIACRYPT 2021, Dong *et al.* [28] studied the quantum free-start collision attacks.

Our Contributions. At CRYPTO 2025, Qin *et al.* [62] introduced the guess-and-determine (GD) rebound attack, by embedding Bouillaguet, Derbez, and Fouque's GD approach into the rebound attack. This combination allows an efficient improvement for the key collision attacks on AES. However, how to leverage the GD rebound attack to achieve powerful attacks on AES hashing modes is left open, especially the collision attacks. In this paper, we fill the gap by proposing two novel frameworks of converting GD-rebound-based semi-free-start (SFS) or free-start (FS) collision attacks on AES-DM into (multi-block) collision attacks in both classical and quantum scenarios.

– **Framework of converting SFS collision into collision.** This model converts the SFS collision attack into a 2-block (Block 0 with M_0 and Block 1 with M_1) classical or quantum collision attack on AES-DM. Suppose that in Block i ($i \geq 0$), the hash computes AES-DM(h_i, M_i)=h_{i+1}. The simple SFS

collision in Block 1 requires the related-key differential of $\Delta h_1 = 0, \Delta M_1 \neq 0$, and $\Delta h_2 = 0$, where there is no conditions on the value of h_1. One may compute many SFS collisions with different h_1s stored in a hash table, and then find an M_0 with the output chaining variable h_1 colliding to one of the elements in the hash table. However, this leads to huge memory to store SFS collisions when balancing the time complexities between Block 0 and 1. Inspired by the Hosoyamada and Sasaki's quantum collision attack on SHA-2 [41], we find the specific SFS collision in Block 1 with some byte conditions on h_1 fixed. The M_0 from Block 0 only needs to meet those conditions of h_1, and the degrees of freedom from M_1 are explored by the GD rebound approach to produce the final collision. This makes the full attack almost memory-less. The byte conditions on h_1 are discovered also by the GD rebound approach. In quantum setting, we convert the memory-less method into quantum collisions using quantum amplitude amplification (QAA) [13] with multiple nested Grover's algorithms [37]. The QAA algorithm is to find the good M_0 that leads to the final 2-block collision. In the setup phase of QAA, a nested Grover's algorithm is performed to build the superposition of M_0 meeting the conditions on h_1. In the checking phase of QAA, another nested Grover's algorithm is performed to exhaust the degrees of freedom of M_1 to find one M_1 leading the final collision.

Comparison with Hosoyamada and Sasaki's Quantum Collision Attack [41]. Hosoyamada *et al.* also proposed quantum ways to convert SFS collision on SHA-2 into 2-block collision. However, in their attack, a large portion of the M_1 and the internal states S_{start} within the inbound phase[4] are prefixed before the quantum attack, while our attack needs to explore the degrees of freedom of these states. Therefore, Hosoyamada *et al.*'s attack only needs one Grover's algorithm to meet the conditions of h_1 and find the final collision. However, our quantum attack needs the QAA algorithm with two nested Grover's algorithms to tradeoff the time complexities of setup phase and checking phase by exploring the full degrees of freedom of M_1 and S_{start}, to minimize the overall complexity.

– **Framework of converting FS collision into collision.** Based on our 2-block collision framework of converting SFS collision into collision, we build the 3-block collision framework of converting FS collision into collision. We put the FS collision attack in Block 2 and the 2-block collision framework in Blocks 0 and 1. Note that the 2-block collision framework outputs a pair (h_2, h_2') with $\Delta h_2 = 0$. However, when integrating the 2-block framework into the full 3-block attack, it outputs a pair of (h_2, h_2') with the given input difference $\Delta h_2 \neq 0$ of the FS collision in Block 2. Some byte conditions on h_2 are found first by GD-rebound, which should also be satisfied by the 2-block collision framework. In quantum setting, it also needs QAA algorithm with multiple nested Grover's algorithms to perform the full attack. In setup

[4] It is in fact the part with dense active bits of the differential trail, which is similar to the inbound phase of a rebound attack.

phase, superpositions of M_0, M_1 meeting the byte conditions of h_2 as well as the difference Δh_2 are built by the quantum 2-block collision framework. In checking phase, the degree of the freedoms from M_2 are explored by the GD rebound approach to check if any value of M_2 produces the final collision.

Comparison with Taiyama et al.*'s Model Converting FS to Collision [66]* . At ASIACRYPT 2024, Taiyama *et al.* proposed the framework to convert FS collision into collision. As shown in Fig. 8, their model requires N related-key differentials in Block 2 with different Δh_2 leading to FS collisions in h_3. To match one of these N differences via Block 0/1's output, the model generates and sorts the $\sqrt{2^{128}/N}$ values of h_2, resulting in huge memory complexity $\sqrt{2^{128}/N}$. For example, their 9-round AES-256-DM collision attack needs a impractical memory of 2^{55}. In contrast, our framework uses a single related-key differential in Block 2 (for FS conversion) and matches the difference Δh_2 via a GD-rebound attack in Block 0/1 instead of using Taiyama *et al.*'s birthday-based approach. Hence, our framework is (almost) memory-less and efficient.

As proof of work, we apply the above frameworks to AES-DM hashing, and achieve various new collision attacks with 1-5 rounds of improvements over the state-of-the-art. The results are summarized as follows and also in Table 1, along with comparisons to all existing attacks.

- **Collision attack on full-round** AES-256-DM[5]**.** *In spite of the fact that numerous results are proposed against* AES*-like hashing (especially* DM/MMO/MP *modes) in the past decades, no full-round attack has yet been reported against any of its three fundamental security requirements, i.e., collision, preimage, and second preimage resisitance.* The only full attack is a free-start multi-collision attack on AES-256-DM by Biryukov, Khovratovich, and Nikolić [9] at CRYPTO 2009, which gives a free-start q-collision (q pseudo-collisions) in time $q \cdot 2^{67}$, better than the trivial bound $O(q \cdot 2^{\frac{q-2}{q+2}128})$. Based on our quantum framework of converting FS collision into collision, we propose the first full collision attack on AES-256-DM in the quantum setting with time complexity $2^{50.42}$. Compared with the state-of-the-art results to date by Taiyama *et al.* [66] at ASIACRYPT 2024, our attack *improves the number of rounds by 5.*
- **Improved classical collision attack on** AES-192-DM**.** Based on our classic framework of converting FS collision into collision, we propose a 7-round classical collision attack on AES-192-DM with time complexity 2^{54}. It extends the best previous classic attack by Taiyama *et al.*'s [66] by 2 rounds.
- **Classical and quantum collision attacks on** AES-128-DM**.** Based on the framework of converting SFS collision into collision, we propose a 4-round classical collision attack on AES-128-DM with time complexity 2^{59} with 2^8

[5]In 2022, Baek *et al.* [2] claimed a full quantum collision attack AES-256-DM. However, this attack was found to be an error by Taiyama *et al.* [66, Section A]. We experimentally re-confirm this error and discuss it in the Supplementary Material H.

memory. Note that the best previous attack also achieves 4 rounds, however, with a huge memory complexity 2^{60}.

Table 1: Summary of Attacks on AES in DM Hashing Modes. S is the size of the quantum computer. To be better than generic attack (**Scenario 2** in Sect. 2.6), $3.51 < S < 2^{16.11}$ for AES-128 and $4.27 < S < 2^{25.07}$ for AES-256.

Target	Attack	Blocks	Rounds	Time	C-Mem	qRAM	Setting	Generic	Ref.
AES-256	Collision	1-block	6/14	2^{61}	-	-	Classic		[66]
		1-block	6/14	Practical	-	-	Classic		[62]
		1-block	7/14	$2^{36.7}$	-	60	Quantum	2^{64}	[62]
		2-block	9/14	2^{58}	2^{55}	-	Classic		[66]
		3-block	**14/14**	$\mathbf{2^{50.42}/\sqrt{S/4.27}}$	-	-	**Quantum**		**Sect. 4**
	SFS Collision	1-block	9/14	Practical	-	-	Classic	2^{64}	[66]
	FS Collision	1-block	10/14	2^{49}	-	-	Classic	2^{64}	[4]
		1-block	10/14	$2^{25.6}$	-	-	Quantum		[4]
	FS Multi-Collision	1-block	14/14	$q \cdot 2^{67}$	-	-	Classic	$q \cdot 2^{\frac{q-2}{q+2}n}$	[9]
	(Pseudo) Preimage	1-block	9/14	2^{120}	2^8	-	Classic	2^{128}	[3]
		1-block	9/14	2^{60}	-	2^8	Quantum	2^{64}	[22]
		1-block	10/14	2^{120}	$2^{56}/2^{48}/2^8$	-	Classic	2^{128}	[25,22,69]
AES-128	Collision	1-block	2/10	2^{49}	-	-	Classic		[66]
		1-block	2/10	Practical	-	-	Classic		[60,62]
		2-block	3/10	2^{60}	2^{52}	-	Classic		[66]
		1-block	3/10	Practical	-	-	Classic	2^{64}	[62]
		1-block	4/10	2^{60}	2^{60}	-	Classic		[18]
		2-block	4/10	2^{59}	2^8	-	Classic		Sect. E.1
		2-block	5/10	$2^{55.04}/\sqrt{S/3.51}$	-	-	Quantum		Sect. E.2
	SFS Collision	1-block	5/10	2^{57}	-	-	Classic	2^{64}	[66]
		1-block	5/10	2^{39}	-	-	Classic		[62]
AES-192	Collision	1-block	5/12	2^{61}	-	-	Classic		[66]
		1-block	6/12	$2^{38.7}$	-	44	Quantum	2^{64}	[62]
		3-block	7/12	2^{54}	2^8	-	Classic		Sect. F
	SFS Collision	1-block	7/12	2^{62}	-	-	Classic		[66]
		1-block	7/12	2^{56}	-	-	Classic	2^{64}	[60]
		1-block	7/12	Practical	-	-	Classic		[62]

In addition, we propose a quantum collision attack on 5-round AES-128-DM with time complexity $2^{55.04}$, which gains 1-round improvement over the state-of-the-art.

– **Practical experiments for verifications.** In order to experimentally verify our attacks, we perform two practical experiments.

  - *Near Collision on 4-round* AES-128-DM: To verify our framework of converting the SFS collision into collision within available computing resources, we conduct this near collision attack. By ignoring the last active bytes of the 4-round related-key differential, the complexity of the attack becomes $2^{59-7} = 2^{52}$. It has been practically implemented with 72 NVIDIA V100 GPUs for about 50 hours to obtain a near-collision, where only one byte of Δh_2 is nonzero. Note that the previous attack with a huge memory of 2^{60} will prevent any (partial) practical verification.

- *Full Collisions on 7-round* AES-192-DM: To verify our framework of converting FS collision into collision, we conduct this experiment, which requires about 2^{54} operations. The attack is practically implemented on 80 NVIDIA A800 GPUs together with 36 NVIDIA V100 GPUs, and a full-collision is obtained after about one week of computation.

All the source codes and experiments can be found via:
https://github.com/DSwutang/AES-DM-collision

2 Preliminaries

2.1 Notations and Symbols

- $\mathcal{D}$: Related-key differential characteristic.
- $|X|$: The cardinality of a given set X.
- $\mathcal{I}_d^{(i)}$: Inbound phase of the rebound attack in Block i, where d refers to the related-key differential $\mathcal{D}$.
- $\mathcal{O}_d^{(i)}$: Outbound phase of the rebound attack in Block i.
- p_d: The negative binary logarithm of the probability of $\mathcal{D}$.
- p_o: The negative binary logarithm of the probability of the outbound phase.
- $\mathbb{A}$, $\mathbb{G}$, $\mathbb{D}$: Sets of byte positions of the active bytes, guessed bytes, and deduced bytes, respectively, in the inbound phase.
- $V[\mathbb{X}]$: The set of possible values assigned to the bytes indexed by X, where $\mathbb{X}$ can be $\mathbb{A}$, $\mathbb{G}$, $\mathbb{D}$.
- DoF_d: Degree of freedom of the related-key differential $\mathcal{D}$.
- DoF_I: Degree of freedom of the inbound phase.
- χ: The system of conflict byte equations in the GD approach.

2.2 Quantum Computations and Grover's Algorithm

The state of an n-qubit quantum system is represented by a unit vector in $\mathbb{C}^{2^n}$ with respect to the computational basis $\{|0\cdots00\rangle, |0\cdots01\rangle, \ldots, |1\cdots11\rangle\}$, which is also denoted by $\{|i\rangle : 0 \le i < 2^n\}$. Quantum algorithms are implemented by applying a sequence of unitary transformations and measurements to such states. These unitary transformations can be decomposed into elementary quantum gates, such as single-qubit and two-qubit gates.

Superposition Oracles for Classical Circuits. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. The superposition oracle of f is the unitary transformation $\mathcal{U}_f$ acting on an $(n+1)$-qubit system, defined by $\mathcal{U}_f |x, y\rangle = |x, y \oplus f(x)\rangle$, where $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2$. By linearity, for a superposition state, we have

$$\mathcal{U}_f \left(\sum_{x \in \mathbb{F}_2^n} \alpha_x |x\rangle |0\rangle \right) = \sum_{x \in \mathbb{F}_2^n} \alpha_x |x\rangle |f(x)\rangle .$$

If f can be computed by an efficient classical circuit, then $\mathcal{U}_f$ can also be implemented efficiently in the quantum circuit model. This is done by first constructing a reversible circuit for f and then replacing the reversible gates with the corresponding quantum gates.

Grover's Algorithm [37]. Consider the problem of finding a marked element in a finite set X. Let $M \subseteq X$ denote the set of marked elements, and assume that the marked fraction $\epsilon = |M|/|X|$ is known. Classically, this problem can be solved by independent random sampling in expected time $O(1/\epsilon)$. While for the quantum setting, the search algorithms can be described by two basic procedures:

- a *Setup* procedure, which prepares a uniform superposition over all elements of X, with running time denoted by $|Setup|_{RT}$;
- a *Checking* procedure, which determines whether a given element is marked, with running time denoted by $|Checking|_{RT}$.

Grover's algorithm [37] performs a quantum search over X by iteratively applying these procedures. And the overall time complexity can be given as

$$\sqrt{1/\epsilon} \cdot (|Setup|_{RT} + |Checking|_{RT}),$$

where the *Checking* operation is implemented as a controlled-phase oracle marking elements of M. When both the *Setup* and *Checking* operations are efficient, Grover's algorithm achieves a quadratic speedup over classical exhaustive search.

Quantum Amplitude Amplification (QAA) Algorithm [13]. More generally, Grover's algorithm is a special instance of quantum amplitude amplification (QAA), introduced by Brassard, Hoyer, Mosca, and Tapp [13]. Let $\mathcal{A}$ be an algorithm that outputs a superposition over a good subspace $G \subseteq X$ and a bad subspace $B \subseteq X$, where a good output is produced with probability a. The overall running time of QAA to produce a good output is given by

$$\sqrt{1/a} \cdot (|\mathcal{A}|_{RT} + |Checking|_{RT}). \tag{1}$$

2.3 Specifications of AES

AES [20] operates on a 4×4 column-major order array of bytes, whose round function contains four major transformations as illustrated in Fig. 1: SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AK). The MixColumns is to multiply each column of the state by an MDS matrix. AES has three variants called AES-128, AES-192, and AES-256 with key lengths of 128 bits, 192 bits, and 256 bits, respectively. The key schedules for the three variants are given in Fig. 10, 11, and 12 in Supp. A.

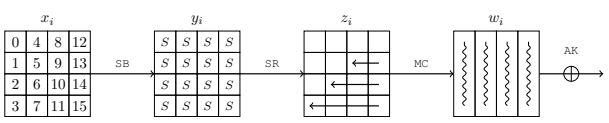


Fig. 1: The round function of AES

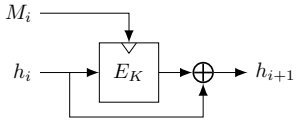


Fig. 2: Davies-Meyer Mode

2.4 Hash Modes and Collision Attacks

The Davies-Meyer (DM) mode (Fig. 2) is used to construct the compression function (CF) from a block cipher, which is then used to construct the hash function (denoted as H) by using the Merkle-Damgård construction [57,21]. All the hash functions in the MD-SHA hash family are built with DM modes. In DM mode, the message block serves as the key input to the block cipher, and the hash function's IV or chaining value serves as the plaintext input. The computation is defined as:

$$h_{i+1} = E_{M_i}(h_i) \oplus h_i, \tag{2}$$

with $i \geq 0$ and $h_0 = IV$.

- **Collision:** find one pair of (M, M') satisfying $H(IV, M) = H(IV, M')$.
- **Semi-free-start (SFS) Collision:** find two pairs (h, M) and (h, M') satisfying $H(h, M) = H(h, M')$, where $h \neq IV$.
  - **Key Collision:** A SFS collision on the DM hash mode is to find (h, M) and (h, M') satisfying $E_M(h) \oplus h = E_{M'}(h) \oplus h$, which is equal to $E_M(h) = E_{M'}(h)$. Note that M acts as the key in the block cipher E, and it is equivalent to find two keys (M, M') of E and encrypt the same plaintext h to get the same ciphertext. At ASIACRYPT 2024, Taiyama *et al.* [66] formally define this collision as *key collision*.
- **Free-start (FS) Collision:** find two pairs (h, M) and (h', M') satisfying $H(h, M) = H(h', M')$, where $h \neq h'$. A one-block FS collision on the DM mode is to find (h, M) and (h', M'), so that $E_M(h) \oplus h = E_{M'}(h') \oplus h'$.
- **Free-start Multi-Collision:** At CRYPTO 2009, Biryukov *et al.* [9] introduced the free-start q-multi-collision on AES-256-DM, which is to find a set of q ($q \geq 2$) pairs: $\{(P_1, K_1), (P_2, K_2), \ldots, (P_q, K_q)\}$ that satisfies

$$E_{K_1}(P_1) \oplus E_{K_1 \oplus \Delta_K}(P_1 \oplus \Delta_P) = E_{K_2}(P_2) \oplus E_{K_2 \oplus \Delta_K}(P_2 \oplus \Delta_P) = \\ = \cdots = E_{K_q}(P_q) \oplus E_{K_q \oplus \Delta_K}(P_q \oplus \Delta_P) = \Delta_C,$$

  where $P_i \oplus \Delta_P \neq P_j$ and $K_i \oplus \Delta_K \neq K_j$ with $i, j \in \{1, 2, \ldots, q\}$. They gave the free-start q-multi-collision in time $q \cdot 2^{67}$, better than the trivial bound $O(q \cdot 2^{\frac{q-2}{q+2}128})$.

2.5 Rebound Attack

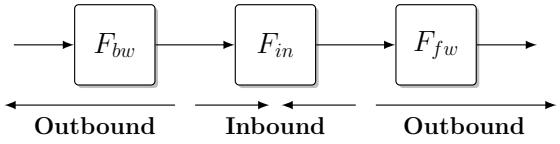


Fig. 3: Rebound Attack

The rebound attack was first introduced by Mendel *et al.* in [56], which consists of an inbound phase and an outbound phase as shown in Fig. 5, where F is an internal block cipher or permutation which is split into three subparts, then $F = F_{fw} \circ F_{in} \circ F_{bw}$.

- **Inbound phase**. In the inbound phase, the attackers efficiently fulfill the low probability part in the middle of the differential trail with a meet-in-the-middle technique. The degree of freedom is the number of matched pairs in the inbound phase, which will act as the starting point for the outbound phase.
- **Outbound phase**. In the outbound phase, the matched values of the inbound phase, *i.e.*, starting points, are computed backward and forward through F_{bw} and F_{fw} to obtain a pair of values which satisfy the outbound differential trail in a brute-force fashion.

2.6 Generic Quantum Collision Algorithms

The complexity of generic quantum collision-finding algorithms is highly dependent on the resources available to the adversary. Existing literature typically considers three scenarios, distinguished by the type and scale of memory and computing resources, each leading to different time–space tradeoffs.

- **Scenario 1.** Assuming a polynomial-size quantum computer with access to exponentially large quantum random access memory (qRAM), the BHT algorithm [14] achieves both time and qRAM complexity $2^{n/3}$. While this setting yields optimal time complexity for generic quantum collision finding, the requirement for large-scale qRAM poses significant practical challenges.
- **Scenario 2.** In the absence of exponentially large quantum or classical memory, the quantum analogue of the parallel rho method [67,8,39] employs S quantum processors to achieve a time-space tradeoff of $2^{n/2}/S$, making it suitable for memory-constrained settings with parallel quantum resources.
- **Scenario 3.** Allowing large classical memory while avoiding exponentially large qRAM, the CNS algorithm [16] finds collisions in time $2^{2n/5}$ using $2^{n/5}$ classical RAM, providing a balanced tradeoff between efficiency and resource availability.

In this paper, we focus on the **Scenario 2** setting, which is used in [39,27,41].

2.7 Converting SFS Collisions into 2-block Collision on SHA-2

For the DM-based hash function SHA-256, Mendel *et al.* [55] presented a framework to convert semi-free-start (SFS) collision attacks having some special properties into a 2-block collision. The framework is illustrated in Fig. 4. The attacker first analyzes the second block without fixing h_1. A semi-free-start collision attack that can work for 2^X choices, typically for any unfixed X bits, of h_1 is located in the second block. Then, the attacker tests 2^{n-X} messages for the first

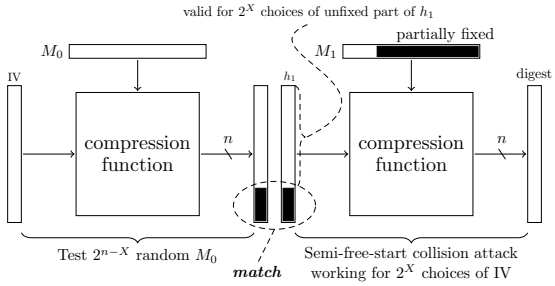


Fig. 4: Converting Semi-free-start Collisions into 2-block Collisions [41]

block to hit one of 2^X choices of h_1, typically to hit the fixed $n - X$ bits of h_1. Finally, the attacker determines the rest part of the second block to generate a 2-block collision.

The cost for the first block is 2^{256-X} for SHA-256. To be faster than the birthday paradox, X must satisfy $X > 128$. To achieve such a semi-free-start collision attack, the previous work [55] generated a differential characteristic such that the characteristic can be satisfied for any value of the first five message words. Hence, it achieves $X = 160$.

The quantum version of Mendel *et al.*'s attack was given by Hosoyamada and Sasaki [41] at CRYPTO 2021. They found that with Grover's algorithm [31], the 38-step semi-free-start collision [55] with $X = 20$ could be converted into a quantum 2-block collision attack, whose time complexity is $c \cdot \sqrt{2^{256-20}/S}$ with a quantum computer of size S and a small constant c. The idea is to apply Grover's algorithm to find a M_0 so that the output of the first block meets the $256 - X$ fixed bits in h_1 with time complexity about $\sqrt{2^{256-X}}$. Then, with the "good" h_1, they find a collision in the digest with a negligible time by similar approach of the 38-step semi-free-start collision [55].

2.8 Converting the FS collision into collision by Taiyama *et al.*'s

At ASIACRYPT 2024, Taiyama *et al.* [66] invented a model to convert the free-start collision attack into collision. The basic idea is to find N different related-key differentials with input differences $\Delta h_1^{(1)} \neq 0, \Delta h_1^{(2)} \neq 0, \ldots, \Delta h_1^{(N)} \neq 0$ with corresponding message differences $\Delta M_1^{(1)}, \Delta M_1^{(2)}, \ldots, \Delta M_1^{(N)}$ and output differences $\Delta h_2^{(1)} = 0, \Delta h_2^{(2)} = 0, \ldots, \Delta h_2^{(N)} = 0$. For each of the N related-key differentials with arbitrary $h_1^{(i)}$, a FS collision is found with a time $T_2 < 2^{64}$. Taiyama *et al.* first compute Block 0 by randomly choosing a pair of input messages M_0 to obtain a pair of h_1, which has a difference that is equal to one of N differences $\Delta h_1^{(1)}, \Delta h_1^{(2)}, \ldots, \Delta h_1^{(N)}$. Then, a FS collision attack on Block 1 is performed with the pair of input h_1 fixed to the value identified in Block 0.

The probability that Δh_1 coincides with one of N specified differences is about $N/2^{128}$. Therefore, upon collecting $2^{128}/N$ pairs of h_1, such a pair will be found with high probability. Due to the birthday paradox, $2^{128}/N$ pairs can be prepared from $\sqrt{2^{128}/N} = 2^{64}/N^{\frac{1}{2}}$ values of h_1. Thus, the total complexity is estimated as

$$\text{Time} : 2^{64}/N^{\frac{1}{2}} + T_2; \quad \text{Memory} : 2^{64}/N^{\frac{1}{2}}. \tag{3}$$

3 Models of Guess-and-Determine Rebound Collision Attack On AES-DM

3.1 Qin *et al.*'s Guess-and-Determine Rebound Attack [62]

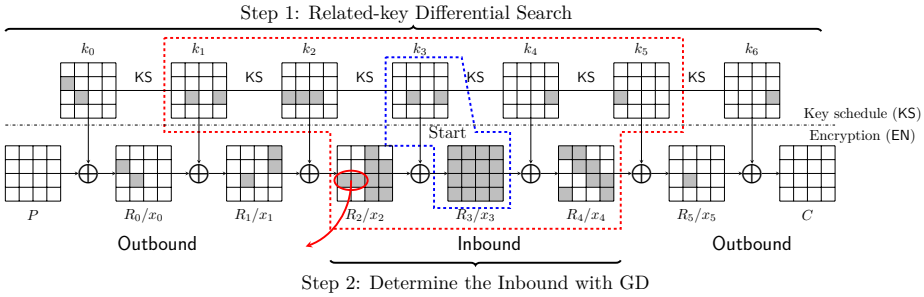


Fig. 5: Guess-and-Determine Rebound Attack [62]

At CRYPTO 2025, Qin *et al.* [62] proposed the guess-and-determine rebound attack based on the guess-and determine approach by Bouillaguet, Derbez, and Fouque [12]. The idea of the GD-rebound attack is to solve the multi-round inbound phase by the guess-and-determine technique. Given a fixed differential trail, the values of the internal states are determined by the DDT of active S-boxes. Then, the fixed and unfixed bytes (degrees of freedom from the state and key) form a nonlinear system. By applying Bouillaguet *et al.*'s GD method, the system is solved and the solutions serve as the starting points for the outbound phase. As shown in Fig. 5, the attack proceeds as follows:

1. **Related-key Differential:** Find a related-key differential by applying Gérault *et al.*'s search tool [32]. Given a differential, the attacker has the following information:
   - **Degree of Freedom of the differential**, denoted as DoF_d. It is defined as the number of independent assignments available for the variables involved in instantiating the differential. Usually, the master key K and the input P act as the DoF_d, *i.e.*, $\text{DoF}_d = 2^{|P|+|K|}$
   - **Probability of the Differential** (2^{-p_d}). To ensure that at least one pair satisfies the differential, $\text{DoF}_d \geq 2^{p_d}$.

2. **Determine the Inbound and Outbound Phase:** A portion of the related-key differential path is selected as the inbound phase (denoted as $\mathcal{I}_d$), which is usually determined through experience or traversal. The remainder of the differential path is the outbound phase (denoted as $\mathcal{O}_d$), which is satisfied with probability 2^{-p_o}.
3. **Guess-and-Determine for $\mathcal{I}_d$:** By access to the DDT, the values of the active S-boxes in $\mathcal{I}_d$ are marked as the known bytes, denoted as $V[\mathbb{A}]$, where $\mathbb{A}$ is the index set of the active S-boxes in $\mathcal{I}_d$. Due to the properties of the DDT of AES, given s_1 and s_2 active S-boxes with a probability of 2^{-7} and 2^{-6}, respectively, the number of possible values is $|V[\mathbb{A}]| = 2^{s_1+2s_2-1}$.
   If one considers a key collision attack [66], the plaintext P is also marked as known bytes, denoted as $V[\mathbb{P}]$, where $\mathbb{P}$ is the index set of P.
   Apply the guess-and-determine approach by Bouillaguet, Derbez, and Fouque [12] to figure out the unknown bytes (denoted as $V[\mathbb{U}]$, where $\mathbb{U}$ is the index set of the unknown bytes) to satisfy the known bytes ($V[\mathbb{A}]$,$V[\mathbb{P}]$). Denote the guess-and-determine approach is a function $\mathcal{F}$, with ($\mathcal{I}_d$,$V[\mathbb{A}]$,$V[\mathbb{P}]$) as input, and outputs the process to determine the $V[\mathbb{U}]$:
   - the bytes to guess $V[\mathbb{G}] \in \mathbb{F}_2^{8\cdot|\mathbb{G}|}$,
   - the bytes to deduce $V[\mathbb{D}]$,
   - the system of conflict byte equations to solve, denoted as

$$\chi(V[\mathbb{A}], V[\mathbb{P}], V[\mathbb{G}], V[\mathbb{D}]) = 0. \tag{4}$$

   The number of the conflict byte equations is denoted as $|\chi|$.
   Obviously, $V[\mathbb{U}] = (V[\mathbb{G}], V[\mathbb{D}])$, and

$$(V[\mathbb{G}], V[\mathbb{D}], \chi) \leftarrow \mathcal{F}(\mathcal{I}_d, V[\mathbb{A}], V[\mathbb{P}]). \tag{5}$$

   For a key collision attack with $V[\mathbb{P}] = 0$, the number of pairs satisfying $\mathcal{I}_d$ (i.e., starting points) is

$$\mathrm{DoF}_I = \frac{2^{s_1+2s_2-1} \times 2^{8\cdot|\mathbb{G}|}}{2^{8\cdot|\chi|}}, \tag{6}$$

   which should satisfy that $\mathrm{DoF}_I \geq 2^{p_o}$ to ensure that at least one starting point satisfies the outbound phase $\mathcal{O}_d$.
   To derive all the DoF_I starting points, one has to traverse $2^{s_1+2s_2-1}$ $V[\mathbb{A}]$ and $2^{8\cdot|\mathbb{G}|}$ $V[\mathbb{G}]$, with time complexity about $\mathcal{T}_{\mathrm{DoF}_I} = 2^{s_1+2s_2-1} \times 2^{8\cdot|\mathbb{G}|}$.

3.2 Converting SFS Collision into Collision Based on GD-Rebound

With Mendel *et al.* 's framework in Sect. 2.7, we introduce the guess-and-determine rebound attack in Block 1 as shown in Fig. 6, where $|h_i| = n$.

- **Related-key Differential:** In Block 1, the related-key differential meets $\Delta h_1 = \Delta h_2 = 0$, $\Delta M_1 \neq 0$, and $\mathrm{DoF}_d = 2^{|M_1|+|h_1|} \geq 2^{p_d}$.

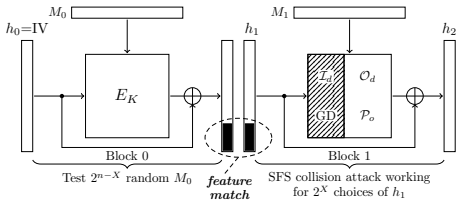


Fig. 6: Attack mode converting SFS collision into collision on AES-DM

– **Semi-Free-Start Collision:** Different from key collision, the plaintext $P = h_1$ is totally free in a typical SFS collision attack. Given an $\mathcal{I}_d$, the attacker run the GD approach with $V[\mathbb{P}]$ marked as unknown. Therefore, the Eq. (5) becomes

$$(V[\mathbb{G}], V[\mathbb{D}], \chi) \leftarrow \mathcal{F}(\mathcal{I}_d, V[\mathbb{A}]), \tag{7}$$

where $(V[\mathbb{G}], V[\mathbb{D}])$ includes $V[\mathbb{P}]$. Theoretically, there are at most $\mathrm{DoF}_I \cdot 2^{-p_o}$ SFS collisions. These SFS collisions are then stored in a hash table indexed by h_1. The index h_1 with non-empty entries is the valid choice of h_1, whose number is denoted as 2^X in Fig. 6.

– **Convert SFS Collision To Collision.** The trivial method is to test 2^{n-X} random M_0 to hit one of the 2^X h_1 stored. To be better than birthday attack, $2^{n-X} < 2^{n/2}$ holds, which leads to $2^X > 2^{n/2}$. This actually stops the attack, since the time to prepare and store 2^X h_1 will be higher than the birthday attack.
In practice, *e.g.*, Mendel *et al.* 's attack on SHA-2, the 2^X h_1 are not actually prepared and stored. **The valid h_1 has some conditions**, *i.e.*, it consists of the X-bit unfixed part and $n - X$ fixed part as shown in Fig. 6. Then, the attacker randomly tests 2^{n-X} M_0 to match the $n - X$ fixed part to find a valid h_1, which is then used to perform SFS collision attack in Block 1 to find a colliding pair (M_1, M_1'). Therefore, not all SFS collisions can be converted into collision.

Condition Detection of Valid h_1 in GD-Rebound. Note that in Fig. 6, X-bit of h_1 can be arbitrary value. However, according to Eq. (7), $h_1 = V[\mathbb{P}]$ is determined by $\mathcal{F}(\mathcal{I}_d, V[\mathbb{A}])$ and cannot be arbitrarily chosen. Inspired by the key collision attack, where $V[\mathbb{P}]$ can be arbitrarily chosen and act as known bytes in the GD approach as Eq. (5). Hence, we use Eq. (5) instead of Eq. (7) to find conditions of h_1. After solving Eq. (5) with GD approach, we analyze the conflict Eq. (4) and select only the equation system related to $(V[\tilde{\mathbb{A}}], V[\mathbb{P}])$ with $\tilde{\mathbb{A}} \subseteq \mathbb{A}$ to derive the conditions of h_1, *i.e.*,

$$\tilde{\chi}(V[\tilde{\mathbb{A}}], V[\mathbb{P}]) = 0, \tag{8}$$

which contains $|\tilde{\chi}|$ byte equations. Then, given $V[\tilde{\mathbb{A}}]$ (deduced by DDT), the Eq. (8) on $V[\mathbb{P}]$ is the conditions of h_1, which should be matched by the output

of Block 0. Given that there are $\tilde{s}_1$ and $\tilde{s}_2$ active S-boxes with a probability of 2^{-7} and 2^{-6}, respectively, the number of possible values $|V[\tilde{\mathbb{A}}]| = 2^{\tilde{s}_1+2\tilde{s}_2}$. Then, there will be $|V[\tilde{\mathbb{A}}]|$ valid systems on $V[\mathbb{P}]$ as Eq. (8). By randomly testing M_0, one of the $|V[\tilde{\mathbb{A}}]|$ systems should be satisfied at first. One example is given in Fig. 7, where the $\tilde{\chi}$ system has two byte equations, as

$$\begin{cases} P[15] \oplus k_0[15] \oplus x_0[15] = 0, \\ P[2] \oplus x_0[2] \oplus \mathsf{SB}(k_0[15]) \oplus \mathsf{SB}(x_0[0]) \oplus \mathsf{SB}(x_0[5]) \oplus 02 \cdot \mathsf{SB}(x_0[10]) \\ \qquad \oplus 03 \cdot \mathsf{SB}(x_0[15]) \oplus x_1[2] = 0. \end{cases} \tag{9}$$

The first conflict is directly derived from the known active bytes, and the second conflict is derived through the encryption and key schedule of the known active bytes. For $\tilde{\mathbb{A}} = \{k_0[15], x_0[0, 2, 5, 10, 15], x_1[2]\}$, traverse and store $|V[\tilde{\mathbb{A}}]|$ systems of $\{P[2], P[15]\}$ as Eq. (9).

Considering $\tilde{\chi}$ will be satisfied by M_0 first by fixing $V[\tilde{\mathbb{A}}]$, the DoF_I in (6) should be

$$\widehat{\mathrm{DoF}}_I = \frac{2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)} \times 2^{8\cdot|\mathbb{G}|}}{2^{8\cdot(|\chi|-|\tilde{\chi}|)}}. \tag{10}$$

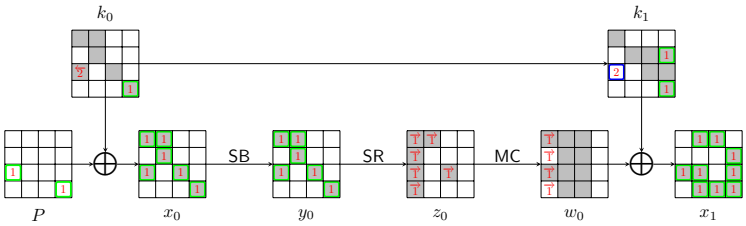


Fig. 7: An example of $\tilde{\chi}$ for AES-128-DM. The numbers indicates the corresponding steps of GD. The bytes in green boxes are the known bytes and in blue boxes are the deduced bytes at the beginning of each step.

Classical Attack: Converting SFS to Collision with GD-Rebound. The full collision attack is

1. In Block 1, find a related-key differential, determine the $\mathcal{I}_d$, and apply Eq. (5). Derive and store $|V[\tilde{\mathbb{A}}]|$ systems Eq. (8).
2. In Block 0, randomly choosing $2^{|\tilde{\chi}|\cdot 8-(\tilde{s}_1+2\tilde{s}_2)+t}$ different values of M_0, compute h_1 with fixed $h_0 = IV$. It is expected that 2^t M_0 will satisfy one of the $|V[\tilde{\mathbb{A}}]|$ systems Eq. (8) as valid h_1.
3. Recompute Eq. (5) with valid $h_1 = V[\mathbb{P}]$ and the remaining possible values of $V[\mathbb{A}]$, whose number is $|V[\mathbb{A}]| = 2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)}$. And there are $2^{8|\mathbb{G}|}$ values for $V[\mathbb{G}]$.

(a) If $2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)} \times 2^{8\cdot|\mathbb{G}|} \geq 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}$, *i.e.*, $\widehat{\mathrm{DoF}}_I \geq 2^{p_o}$, then one valid h_1 derived in Step 2 is enough to find the collision. The total time complexity will be

$$2^{8|\tilde{\chi}|-(\tilde{s}_1+2\tilde{s}_2)+0} + 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}. \tag{11}$$

(b) If $2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)} \times 2^{8\cdot|\mathbb{G}|} < 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}$, *i.e.*, $\widehat{\mathrm{DoF}}_I < 2^{p_o}$, then the total time complexity will be

$$2^{8|\tilde{\chi}|-(\tilde{s}_1+2\tilde{s}_2)+t} + 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}, \tag{12}$$

where $2^t = \frac{2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}}{2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)+8\cdot|\mathbb{G}|}} = 2^{p_o}/\widehat{\mathrm{DoF}}_I$.

Quantum Version. Two cases are analyzed based on the above classical steps:

- In Case 3(a), two runs of the Grover's algorithm will be performed.
  - First, run Grover's algorithm on M_0 to find a valid h_1. *In order to achieve memory-less quantum steps, a single valid* $|V[\tilde{\mathbb{A}}]|$ *and Eq.* (8) *are fixed and satisfied by* M_0. Then, the time of this step is about $\frac{\pi}{4} \cdot 2^{4|\tilde{\chi}|}$.
  - Second, run Grover's algorithm on the remaining possible values of $V[\mathbb{A}]$ and $V[\mathbb{G}]$, to find the collision, with time complexity about $\frac{\pi}{4} \cdot 2^{\frac{p_o+8(|\chi|-|\tilde{\chi}|)}{2}}$.
  - Overall time is
$$\frac{\pi}{4} \cdot 2^{4|\tilde{\chi}|} + \frac{\pi}{4} \cdot 2^{\frac{p_o+8(|\chi|-|\tilde{\chi}|)}{2}}. \tag{13}$$
- In Case 3(b), still fix $|V[\tilde{\mathbb{A}}]|$ and Eq. (8), we need to run the quantum amplitude amplification (QAA) with nested Grover's algorithm. Let $S := \{(M_0, h_1) : \tilde{\chi}(V[\tilde{\mathbb{A}}], h_1) = 0\}$.
  - In *Setup* phase, the quantum algorithm $\mathcal{A}$ is to run Grover's algorithm on M_0 to produce a superposition $|\phi\rangle := \frac{1}{\sqrt{|S|}} \sum_{M_0 \in S} |M_0, h_1\rangle$. The time complexity is $\mathcal{T}_{\mathcal{A}} = \frac{\pi}{4} \cdot 2^{4|\tilde{\chi}|}$.
  - In *Checking* phase, the checking algorithm is $\mathcal{U}_f$ that
$$\mathcal{U}_f(|M_0\rangle\,|b\rangle) = |M_0\rangle\,|b \oplus f(M_0)\rangle \tag{14}$$
where $f(M_0) = 1$ iff there exist one value from the $2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)+8\cdot|\mathbb{G}|}$ $(V[\mathbb{A}\backslash\tilde{\mathbb{A}}], V[\mathbb{G}])$ values, that leads to a collision in the second block. Obviously, $f(M_0) = 1$ holds with probability 2^{-t} with t given in Eq. (12). The implementation of f is to run a nested Grover's algorithm on $(V[\mathbb{A} \setminus \tilde{\mathbb{A}}], V[\mathbb{G}])$ and check if it leads to collision. If M_0 leads to a collision, the time complexity to run the nested Grover's algorithm (also the time complexity of f) is
$$\mathcal{T}_f = \frac{\pi}{4} \cdot \sqrt{2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)+8\cdot|\mathbb{G}|}}. \tag{15}$$
  - The full complexity is

$$\mathcal{T} = \frac{\pi}{4} \cdot 2^{t/2}(\mathcal{T}_{\mathcal{A}} + \mathcal{T}_f) = \frac{\pi}{4} \cdot 2^{t/2} \cdot (\frac{\pi}{4} \cdot 2^{4|\tilde{\chi}|} + \frac{\pi}{4} \cdot \sqrt{2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)+8\cdot|\mathbb{G}|}}). \tag{16}$$

Comparison with Hosoyamada and Sasaki Quantum Attack Model. At CRYPTO 2021, Hosoyamada and Sasaki convert Mendel *et al.* 's framework into the quantum attack [41]. In their quantum attack on SHA-256, given a valid h_1, a full message M_1 (a portion of M_1 is already fixed when preparing the attack) is deduced, and checked if it leads to a collision in Block 1. Therefore, a single run of Grover's algorithm to find the "good" h_1 that leads to the final collision is enough to perform the quantum attack.

3.3 Converting the FS Collision to Collision Based on GD-Rebound

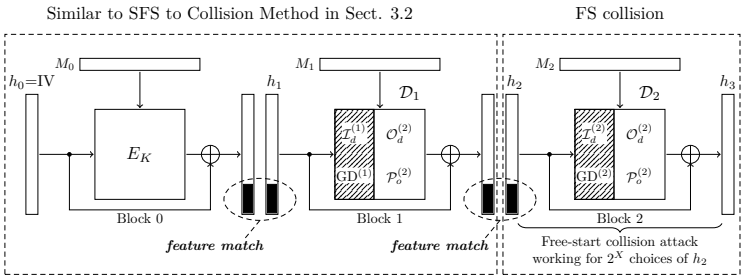


Fig. 8: Attack mode converting FS collision into collision on AES-DM

Based on the framework in Sect. 3.2, we further introduce the guess-and-determine rebound attack in Block 1 and 2 as shown in Fig. 8, to extend the free-start collisions into collision attacks.

- **Related-key Differential:**
  - In Block 2, the related-key differential characteristic $\mathcal{D}_2$ meets $\Delta h_3 = 0$, $\Delta M_2 \neq 0$, $\Delta h_2 \neq 0$, and $\mathrm{DoF}_d^{(2)} = 2^{|M_2|+|h_2|} \geq 2^{p_d^{(2)}}$.
  - In Block 1, the related-key differential characteristic $\mathcal{D}_1$ meets $\Delta h_1 = 0$, $\Delta M_1 \neq 0$, and $\mathrm{DoF}_d^{(1)} = 2^{|M_1|+|h_1|} \geq 2^{p_d^{(1)}}$. The output difference is equal to $\Delta h_2 \neq 0$.
- **Free-Start Collision:** In Block 2, there is a free-start collision trivially, if the value of h_2 is totally free and its difference is Δh_2. Given an $\mathcal{I}_d^{(2)}$ in Block 2, the attacker runs the GD approach with $V[\mathbb{P}^{(2)}] = h_2$ marked as unknown. Therefore, the Eq. (5) becomes

$$(V[\mathbb{G}^{(2)}], V[\mathbb{D}^{(2)}], \chi^{(2)}) \leftarrow \mathcal{F}(\mathcal{I}_d^{(2)}, V[\mathbb{A}^{(2)}]), \tag{17}$$

  where $(V[\mathbb{G}^{(2)}], V[\mathbb{D}^{(2)}])$ includes $V[\mathbb{P}^{(2)}]$. Theoretically, there are at most $\mathrm{DoF}_I^{(2)} \cdot 2^{-p_o^{(2)}}$ FS collisions. These FS collisions are then stored in a hash table L indexed by h_2. The index h_2 with non-empty entries is the valid choice of h_2, whose number is denoted as 2^X in Fig. 8.

– **Trivially Connecting Block 1 and 2.** A trivial method is to find 2^{n-X} output values h_2 of Block 1, which all satisfy the differential characteristic in Block 1, to expect at least one hit against L. However, the time is trivially not better than birthday attack as analyzed in the SFS collision in Sect. 3.2.
– **Conditions of valid h_2.** Similar to the analysis in Sect. 3.2, we use Eq. (5) instead of Eq. (17) to find conditions of h_2. After solving (5) with GD approach, we select

$$\tilde{\chi}^{(2)}(V[\tilde{\mathbb{A}}^{(2)}], V[\mathbb{P}^{(2)}]) = 0, \tag{18}$$

which contains $|\tilde{\chi}^{(2)}|$ byte equations. Given that $|V[\tilde{\mathbb{A}}^{(2)}]|$ has $2^{\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)}}$ possible values, satisfying Eq. (18) requires $2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})}$ values of $V[\mathbb{P}^{(2)}]$.

Classical Attack: Converting FS to Collision with GD-Rebound. The full collision attack is

1. In Block 2, find a related-key differential $\mathcal{D}_2$ with input difference Δh_2, determine the $\mathcal{I}_d^{(2)}$, apply Eq. (5). Derive and store $|V[\tilde{\mathbb{A}}^{(2)}]|$ systems Eq. (18) as $\tilde{\chi}^{(2)}$. It is expected that $2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})}$ h_2 will satisfy one of the $|V[\tilde{\mathbb{A}}^{(2)}]|$ systems Eq. (18) as valid h_2, and assume that $2^{t^{(2)}}$ valid h_2 are needed to obtain a collision in h_3.
2. In Block 1, set the input difference to 0 and the output difference to Δh_2, then find a related-key differential $\mathcal{D}_1$, determine $\mathcal{I}_d^{(1)}$. In Block 0, $h_0 = IV$ and there are no differences in M_0 and h_1. The strategy in Block 0 and Block 1 is to find $2^{t^{(2)}}$ valid h_2 which meets:
   – Condition I: the difference Δh_2,
   – Condition II: the conditions of Eq. (18).

   The method in Block 0 and 1 is almost the same to the classical attack of converting SFS to collision ($\Delta h_2 = 0$) in Sect. 3.2, where the input and output differences are 0, while in this step, the output has fixed difference Δh_2.

   Therefore, we perform one time of the classical attack strategy of converting SFS to collision in Sect. 3.2 to find a pair (M_0, M_1) and (M_0, M_1') to "Condition I" first, which needs the GD-rebound attack in Block 1. Then, "Condition II" is satisfied randomly with probability $2^{(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})-8|\tilde{\chi}^{(2)}|}$.

   The **Full Classical Attacks** are summarized in the following two cases:

   (a) If $2^{s_1^{(2)}+2s_2^{(2)}-1-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} \times 2^{8\cdot|\mathbb{G}^{(2)}|} \geq 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}$, *i.e.*, $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}} \geq 2^{p_o^{(2)}}$, then one valid h_2 satisfying "Condition I and II" is enough to find a collision in h_3. Then, perform $2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})}$ times of the attack of converting SFS to collision in Sect. 3.2, it is expected to find the valid h_2. The total time complexity will be

$$\mathcal{T}_{SFS} \cdot 2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} + 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}, \tag{19}$$

   where $\mathcal{T}_{SFS}$ is given by Eq. (11) or Eq. (12).

(b) If $2^{s_1^{(2)}+2s_2^{(2)}-1-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} \times 2^{8\cdot|\mathbb{G}^{(2)}|} < 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}$, *i.e.*, $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}} < 2^{p_o^{(2)}}$, then the total time complexity will be

$$\mathcal{T}_{SFS} \cdot 2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} \cdot 2^{t^{(2)}} + 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}. \tag{20}$$

where $2^{t^{(2)}} = \frac{2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}}{2^{s_1^{(2)}+2s_2^{(2)}-1-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})+8\cdot|\mathbb{G}^{(2)}|}} = 2^{p_o^{(2)}}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}}$, and $\mathcal{T}_{SFS}$ is given by Eq. (11) or Eq. (12).

Quantum Strategy of Converting FS Collision to Collision. In quantum attacks, it is expected that more nest Grover's algorithms are needed to find the final collision in h_3, which makes the quantum attacks extremely complex and ugly. *To simplify the quantum algorithm, we simplify the quantum steps in Block 0 and 1 by fixing the values $V[\mathbb{A}^{(1)}]$ and $V[\mathbb{G}^{(1)}]$ globally in Block 1, and fixing the values $V[\tilde{\mathbb{A}}^{(2)}]$ globally in Block 2.* Then, two cases are analyzed based on the above classical attack steps,

– In Case 2(a), two runs of the Grover's algorithm will be performed.
  • First, run Grover's algorithm on M_0 to find a valid h_2 with
    ∗ a probability of $2^{-8|\chi^{(1)}|-p_o^{(1)}}$ to satisfy Δh_2 (Condition I),
    ∗ a probability of $2^{-8|\tilde{\chi}^{(2)}|}$ to satisfy Condition II in h_2.
    The time of this step to find a valid h_2 is

$$\frac{\pi}{4} \cdot 2^{\frac{8|\chi^{(1)}|+p_o^{(1)}+8|\tilde{\chi}^{(2)}|}{2}}. \tag{21}$$

  • Second, run Grover's algorithm on the remaining possible values of $V[\mathbb{A}^{(2)}]$ and $V[\mathbb{G}^{(2)}]$, to find the collision, with time complexity about

$$\frac{\pi}{4} \cdot 2^{\frac{p_o^{(2)}+8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}{2}}.$$

  • Overall time:

$$\frac{\pi}{4} \cdot 2^{\frac{8|\chi^{(1)}|+p_o^{(1)}+8|\tilde{\chi}^{(2)}|}{2}} + \frac{\pi}{4} \cdot 2^{\frac{p_o^{(2)}+8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}{2}} \tag{22}$$

– In Case 2(b), we need to run the quantum amplitude amplification (QAA) with nested Grover's algorithm. Let $S := \{(M_0, h_2) : \textit{Condition I and II hold}\}$. Note that given the fixed values $V[\mathbb{A}^{(1)}]$, $V[\mathbb{G}^{(1)}]$, and $V[\tilde{\mathbb{A}}^{(1)}]$, it is easy to determine h_2 with M_0 as following

$$M_0 \xrightarrow{\mathsf{DM}} h_1 \xrightarrow[\text{Condition I}]{V[\mathbb{A}^{(1)}],V[\mathbb{G}^{(1)}]} \Delta h_2 \xrightarrow[\text{Condition II}]{V[\tilde{\mathbb{A}}^{(1)}]} h_2. \tag{23}$$

For a randomly given M_0, the probability that $(M_0, h_2) \in S$ holds is on average $2^{-8|\chi^{(1)}|-p_o^{(1)}-8|\tilde{\chi}^{(2)}|}$.

- In *Setup* phase, the quantum algorithm $\mathcal{A}$ is to run Grover's algorithm on M_0 to produce a superposition $|\phi\rangle := \frac{1}{\sqrt{|S|}} \sum_{M_0 \in S} |M_0, h_2\rangle$. The time complexity is

$$\mathcal{T}_{\mathcal{A}} = \frac{\pi}{4} \cdot 2^{\frac{8|\chi^{(1)}| + p_o^{(1)} + 8|\tilde{\chi}^{(2)}|}{2}}. \tag{24}$$

- In *Checking* phase, the checking algorithm is $\mathcal{U}_f$ that

$$\mathcal{U}_f(|M_0\rangle\,|b\rangle) = |M_0\rangle\,|b \oplus f(M_0)\rangle \tag{25}$$

where $f(M_0) = 1$ if and only if there exists one value from all possible ones of $2^{s_1^{(2)} + 2s_2^{(2)} - 1 - (\tilde{s}_1^{(2)} + 2\tilde{s}_2^{(2)}) + 8\cdot|\mathbb{G}^{(2)}|}$ $(V[\mathbb{A}^{(2)} \setminus \tilde{\mathbb{A}}^{(2)}], V[\mathbb{G}^{(2)}])$, that leads to a collision in Block 2. Obviously, $f(M_0) = 1$ holds with probability $2^{-t^{(2)}}$ with $t^{(2)}$ given in Eq. (20). The implementation of f is to run a nested Grover's algorithm on $(V[\mathbb{A}^{(2)} \setminus \tilde{\mathbb{A}}^{(2)}], V[\mathbb{G}^{(2)}])$ and check if it leads to a collision. If M_0 leads to a collision, the time complexity to run the nested Grover's algorithm (also the time complexity of f) is

$$\mathcal{T}_f = \frac{\pi}{4} \cdot \sqrt{2^{s_1^{(2)} + 2s_2^{(2)} - 1 - (\tilde{s}_1^{(2)} + 2\tilde{s}_2^{(2)}) + 8\cdot|\mathbb{G}^{(2)}|}}. \tag{26}$$

- The full complexity is

$$\begin{aligned} \mathcal{T} &= \frac{\pi}{4} \cdot 2^{t^{(2)}/2} (\mathcal{T}_{\mathcal{A}} + \mathcal{T}_f) \\ &= \frac{\pi}{4} \cdot 2^{t^{(2)}/2} \cdot \left(\frac{\pi}{4} \cdot 2^{\frac{8|\chi^{(1)}| + p_o^{(1)} + 8|\tilde{\chi}^{(2)}|}{2}} + \frac{\pi}{4} \cdot \sqrt{2^{s_1^{(2)} + 2s_2^{(2)} - 1 - (\tilde{s}_1^{(2)} + 2\tilde{s}_2^{(2)}) + 8\cdot|\mathbb{G}^{(2)}|}} \right). \end{aligned} \tag{27}$$

Comparison with Taiyama *et al.*'s model [66]. At ASIACRYPT 2024, Taiyama *et al.* invented a model to convert the free-start collision attack into collision. As shown in Sect. 2.8, the two-block collision attack includes FS collisions with N different input differences of the second block. Then, in the first block, the birthday approach is applied to generate an output difference that is equal to one of the N input differences, which leads to a huge memory as shown in Eq. (3). Besides, the improvement may be marginal if the number of related-key differential characteristic is small, *i.e.*, N is not large. Given N, the improvement factor is about $N^{1/2}$. Therefore, the collision attack based on this strategy is usually close to the birthday bound, *e.g.*, the time complexity of the 9-round collision attack given by Taiyama *et al.* is 2^{58} with a memory of 2^{55}, while the birthday attack is 2^{64} time.

In our attack of converting FS collision to collision, a single related-key differential that constructing the FS collision is used in Block 2 with GD-rebound technique. In Block 0 and 1, the GD-rebound is also applied to find the pairs of (M_0, M_1), (M_0, M_1') that directly satisfy input differences and values of Block 2. No birthday method is applied.

4 Quantum Collision Attack on Full AES-256-DM

In Supp. B, we give a heuristic search method to search differential characteristics and the corresponding collision attacks on AES-DM.

Based on the quantum strategy of converting free-start (FS) Collision into Collision introduced in Sect. 3.3, we find a 3-block quantum collision attack on the full 14-round AES-256-DM. The attack follows the framework in Fig. 8 in Sect. 3.3, where the related-key differential characteristics $\mathcal{D}_1$ ($\Delta h_1 = 0$, $\Delta M_1 \neq 0$, $\Delta h_2 \neq 0$) and $\mathcal{D}_2$ ($\Delta h_3 = 0$, $\Delta M_2 \neq 0$, $\Delta h_2 \neq 0$) are introduced to Block 1 and 2. For this purpose, we find two 14-round differential trails for AES-256 using the automatic tool in [32,33]. The first trail $\mathcal{D}_1$ satisfies

$$\mathsf{AES}_{M_1}(h_1) \oplus \mathsf{AES}_{M_1 \oplus \Delta M_1}(h_1) = \Delta h_2 \neq 0 \quad \text{(Block 1)}, \tag{28}$$

and the second trail $\mathcal{D}_2$ satisfies

$$\mathsf{AES}_{M_2}(h_2) \oplus \mathsf{AES}_{M_2 \oplus \Delta M_2}(h_2 \oplus \Delta h_2) = \Delta h_2 \quad \text{(Block 2)}. \tag{29}$$

4.1 Differential Characteristics Search

Following the framework given in Sect. 3.3,

- The differentials $\mathcal{D}_1$ and $\mathcal{D}_2$ should satisfy $2^{256+128} \geq 2^{p_d^{(1)}}$ and $2^{256+128} \geq 2^{p_d^{(2)}}$, as well as the Eq. (28) and (29).
  When searching for the 14-round related-key differential trail of AES-256, we did not find any trail with probability higher than 2^{-256}. Therefore, the degrees of freedom of the key are insufficient to satisfy the probability of the entire differential trail. Taking $\mathcal{D}_2$ as an example, at least $2^{p_d^{(2)}-256}$ values of $P = h_2$ are required to satisfy the probability.
- Given $\mathcal{D}_1$ and $\mathcal{D}_2$, run the guess-and-determine rebound [62,12] to determine all the parameters discussed in Sect. 3.3. In this attack, we found the probability of the outbound phases are usually lower than the birthday bound, *i.e.* $p_o^{(2)} > 64$, which makes the classical attack better than 2^{64} impossible according to Eq. (19) and (20). Therefore, we concentrate on the quantum attack instead, and check if any differential with GD-rebound attack parameters makes Eq. (27) lower than 2^{64}.

Finally, we find the differential trails $\mathcal{D}_1$ and $\mathcal{D}_2$ given in Fig. 13 and Fig. 14 in Supp. C. For both trails we choose the first 8 rounds of the subkeys and the first 3 rounds of the encryption path as the inbound phases $\mathcal{I}_d^{(1)}$ and $\mathcal{I}_d^{(2)}$, and the remaining parts as the outbound phases $\mathcal{O}_d^{(1)}$ and $\mathcal{O}_d^{(2)}$. For $\mathcal{D}_1$ of Block 1 in Fig. 13, the probabilities of the inbound and outbound phases are 2^{-191} and $2^{-p_o^{(1)}} = 2^{-78}$, respectively. For $\mathcal{D}_2$ of Block 2 in Fig. 14, they are 2^{-196} and 2^{-78}. The guess-and-determine steps are given in the Sect. 4.3 for $\mathcal{D}_1$ and Sect. 4.2 for $\mathcal{D}_2$, and the detailed equations are listed in Table 3 and Table 2.

4.2 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d^{(2)}$

Run the Eq. (5), with given $V[\mathbb{P}] = h_2$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 9. The $V[\mathbb{A}] = (x_0[0-15], x_1[0,3,4,8,9,14], x_2[0,4], k_6[12], k_7[12-15])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-15]$, $\Delta y_0[0-15]$, $\Delta x_1[0,3,4,8,9,14]$, $\Delta y_1[0,3,4,8,9,14]$, $\Delta x_2[0,4]$, $\Delta y_2[0,4]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 2.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute $k_0[0-15] = (x_0 \oplus P)[0-15]$ (marked by $\overleftarrow{1}$). 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 $\overleftarrow{1}$).

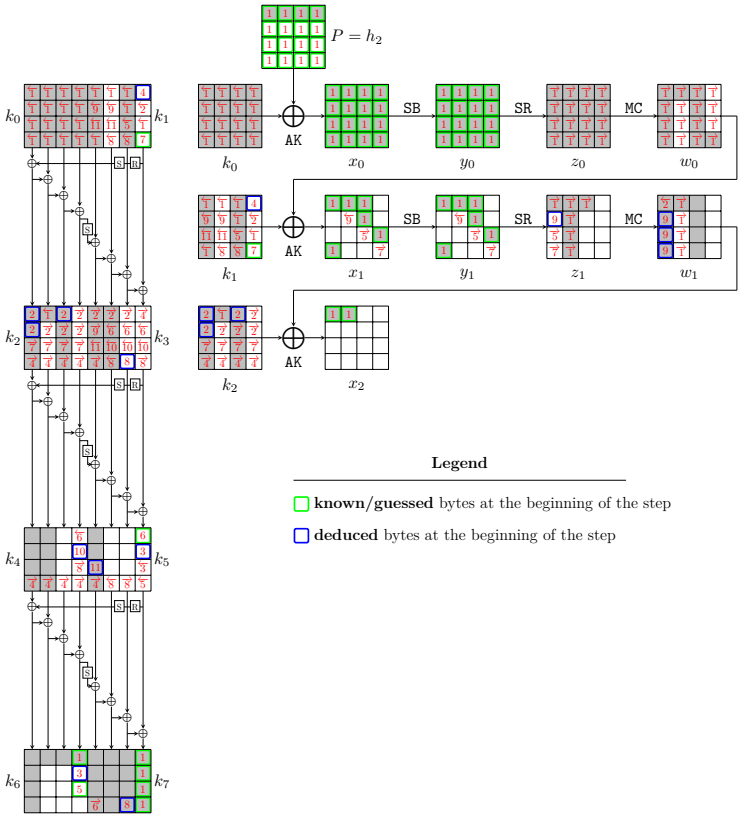


Fig. 9: The GD process of $\mathcal{I}_d^{(2)}$.

2. According to the key relations, deduce $k_2[1]$, $k_2[0]$ and $k_2[8]$ (marked by 2).
   (a) Compute backward to get $k_1[13]$ (marked by $\overleftarrow{2}$).
   (b) Compute forward to get $k_2[5, 9, 13]$ (marked by $\overrightarrow{2}$). Similarly, compute $k_2[12]$ and $k_3[0, 4, 8]$ (marked by $\overrightarrow{2}$).
3. Deduce $k_6[13]$ and $k_5[13]$(marked by 3). Then compute backward to get $k_5[14]$ (marked by $\overleftarrow{3}$).
4. Similar to the previous step, deduce $k_1[12]$ (marked by 4). Using $k_1[12]$ and $k_3[8]$, we can compute $k_3[12]$ (marked by $\overrightarrow{4}$). Then using $k_1[3, 12]$ and $k_0[3, 7, 11, 15]$, we can compute $k_2[3, 7, 11, 15]$, $k_3[3]$, and subsequently compute $k_4[3, 7, 11, 15]$, $k_5[3]$ (marked by $\overrightarrow{4}$).
5. Guess $k_6[14]$ (marked by 5). Then deduce $k_5[15]$ (marked by $\overleftarrow{5}$).
6. Guess $k_5[12]$ (marked by 6).
   (a) Compute forward to get $k_7[3]$ (marked by $\overrightarrow{6}$).
   (b) Compute backward to get $k_4[12]$ and $k_3[5, 9, 13]$ (marked by $\overleftarrow{6}$).
7. Guess $k_1[15]$ (marked by 7).
   (a) In round 1, Compute forward to get $x_1[15]$, $y_1[15]$ and $z_1[3]$ (marked by $\overrightarrow{7}$).
   (b) According to the key relations, compute forward to get $k_2[2, 6, 10, 14]$ (marked by $\overrightarrow{7}$).
8. Deduce $k_7[11]$ (marked by 8). Then from $k_7[11] = k_7[3] \oplus k_3[11]$, deduce $k_3[11]$ (marked by 8).
   (a) Using $k_3[11]$ to compute forward to get $k_3[15]$ (marked by $\overrightarrow{8}$). Then compute forward to get $k_5[11]$ (marked by $\overrightarrow{8}$).
   (b) Using $k_5[11]$ to compute backward to get $k_5[7]$, $k_3[7]$ and $k_1[7, 11]$ in sequence (marked by $\overleftarrow{8}$).
9. For column 1 over the MC operation in round 1, deduce $z_1[1]$ and $w_1[1-3]$ (marked by 9) from $z_1[0, 2, 3]$ and $w_1[0]$. In round 1, compute backward to get $y_1[5]$, $x_1[5]$ and $k_1[5]$ (marked by $\overleftarrow{9}$).
   (a) Using $k_1[5]$ to compute forward to get $k_3[1]$ (marked by $\overrightarrow{9}$).
   (b) Using $k_3[1]$ to compute backward to get $k_1[1]$ (marked by $\overleftarrow{9}$).
10. Deduce $k_4[13]$ (marked by 10). Then compute backward to get $k_3[6, 10, 14]$ (marked by $\overleftarrow{10}$).
11. Deduce $k_5[12]$ (marked by 11). Compute backward to get $k_3[2]$ and $k_1[2, 6]$ (marked by $\overleftarrow{11}$). Then we get all the key bytes.

Degree of freedom and complexity.

– In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}^{(2)}]$ for the active bytes from the input/output differences in the $\mathcal{I}_d^{(2)}$. There are 29 active S-boxes, where $s_1^{(2)} = 22$ S-boxes are of probability 2^{-7}, and $s_2^{(2)} = 7$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}^{(2)}]| = 2^{s_1^{(2)}+2s_2^{(2)}-1} = 2^{35}$ choices for the bytes marked by 1 (except bytes in P) in Fig. 9.

1.	$k_0[0-15] = (x_0 \oplus P)[0-15]$	$k_1[0,3,4,8,9,14] = (x_1 \oplus w_0)[0,3,4,8,9,14]$
	$k_2[4] = (x_2 \oplus w_1)[4]$	
2.	$k_2[1] = \mathrm{SB}(k_1[14]) \oplus k_0[1]$	$k_2[0] = (k_0 \oplus k_2)[4]$
	$k_2[5,8,9,12,13] = k_0[5,8,9,12,13] \oplus k_2[1,4,5,8,9]$	$\mathrm{SB}(k_1[13]) = (k_0 \oplus k_2)[0] \oplus const$
	$k_3[0] = k_1[0] \oplus \mathrm{SB}(k_2[12])$	$k_3[4,8] = k_1[4,8] \oplus k_3[0,4]$
3.	$\mathrm{SB}(k_6[13]) = k_7[13] \oplus k_1[9] \oplus k_1[13]$	$\mathrm{SB}(k_5[13]) = k_6[12] \oplus k_0[8] \oplus k_0[12] \oplus const$
	$\mathrm{SB}(k_5[14]) = k_6[13] \oplus k_0[9] \oplus k_0[13]$	
4.	$k_1[12] = k_7[12] \oplus \mathrm{SB}(k_6[12]) \oplus k_1[8]$	$k_3[12] = k_3[8] \oplus k_1[12]$
	$k_2[3] = k_0[3] \oplus \mathrm{SB}(k_1[12])$	$k_2[7,11,15] = k_0[7,11,15] \oplus k_2[3,7,11]$
	$k_4[3] = k_2[3] \oplus \mathrm{SB}(k_3[12])$	$k_4[7,11,15] = k_2[7,11,15] \oplus k_4[3,7,11]$
	$k_3[3] = \mathrm{SB}(k_2[15]) \oplus k_1[3]$	$k_5[3] = \mathrm{SB}(k_4[15]) \oplus k_3[3]$
5.	$\mathrm{SB}(k_5[15]) = \underset{\sim}{k_6[14]} \oplus k_0[10] \oplus k_0[14]$	
6.	$k_7[3] = k_5[3] \oplus \mathrm{SB}(k_4[3] \oplus k_4[7] \oplus k_4[11] \oplus k_4[14] \oplus \mathrm{SB}(\underset{\sim}{k_5[12]}))$	
	$\mathrm{SB}(k_4[12]) = k_3[0] \oplus k_3[4] \oplus k_3[8] \oplus k_3[12] \oplus \underset{\sim}{k_5[12]}$	
	$\mathrm{SB}(k_3[13]) = k_2[0] \oplus k_2[4] \oplus k_2[8] \oplus k_2[12] \oplus k_4[12] \oplus const$	
	$k_3[9] = k_1[13] \oplus k_3[13]$	$k_3[5] = k_1[9] \oplus k_3[9]$
7.	$k_2[2] = k_0[2] \oplus \mathrm{SB}(\underset{\sim}{k_1[15]})$	$k_2[6,10,14] = k_0[6,10,14] \oplus k_2[2,6,10]$
8.	$k_7[11] = k_7[15] \oplus k_5[15]$	$k_3[11] = k_7[15] \oplus k_7[11]$
	$k_3[15] = k_3[11] \oplus k_1[15]$	$k_5[11] = k_5[15] \oplus k_3[15]$
	$k_5[7] = k_3[11] \oplus k_5[11]$	$k_3[7] = k_5[7] \oplus k_5[3]$
	$k_1[11] = k_3[11] \oplus k_3[7]$	$k_1[7] = k_3[7] \oplus k_3[3]$
9.	$w_1[0,1,2,3] = \mathrm{MC}(z_1[0,1,2,3])$	$k_1[5] = x_1[5] \oplus w_0[5]$
	$k_3[1] = k_3[5] \oplus k_1[5]$	$k_1[1] = k_2[1] \oplus \mathrm{SB}(k_2[13])$
10.	$\mathrm{SB}(k_4[13]) = k_3[1] \oplus k_3[5] \oplus k_3[9] \oplus k_3[13] \oplus k_5[13]$	
	$\mathrm{SB}(k_3[14]) = k_2[1] \oplus k_2[5] \oplus k_2[9] \oplus k_2[13] \oplus k_4[13]$	
	$k_3[10] = k_3[14] \oplus k_1[14]$	$k_3[6] = k_3[10] \oplus k_1[10]$
11.	$k_5[2] = k_3[6] \oplus k_3[10] \oplus k_3[14] \oplus k_5[14]$	$k_3[2] = k_5[2] \oplus \mathrm{SB}(k_4[14])$
	$k_1[5] = k_3[1] \oplus k_3[5]$	$k_1[1] = k_3[1] \oplus \mathrm{SB}(k_2[13])$

Table 2: Equations in the GD steps of $\mathcal{I}_d^{(2)}$. Waved bytes are guessed.

- Given $V[\mathbb{A}]$, three bytes $V[\mathbb{G}^{(2)}] = (k_6[14], k_5[12], k_1[15])$ (marked by a wavy line) are guessed in Steps 5, 6, and 7. There is no conflict in the $\mathcal{I}_d^{(2)}$, *i.e.*, $|\chi^{(2)}| = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d^{(2)}$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}} = \frac{2^{35+3\times 8}}{2^{8\times 0}} = 2^{59}$.
- 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 $2^{-p_o^{(2)}} = 2^{-78}$, which means that at least $2^{18} = 2^{p_o^{(2)}}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}}$ distinct values of $P = h_2$ are required to generate 2^{78} starting points to expect one collision in h_3.

4.3 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d^{(1)}$

Run the Eq. (5), with given $V[\mathbb{P}] = h_1$ and $V[\mathbb{A}^{(1)}]$, which are marked by 1 in Fig. 15. The $V[\mathbb{A}^{(1)}] = (x_0[0-15], x_1[0,5,8,10,15], x_2[0,4], k_6[12], k_7[12-15])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-15]$, $\Delta y_0[0-15]$,

$\Delta x_1[0,3,4,8,9,14]$, $\Delta y_1[0,3,4,8,9,14]$, $\Delta x_2[0,4]$, $\Delta y_2[0,4]$, and also the fixed differences in the key schedule.

As shown in Fig. 13 and 14, the differential trail $\mathcal{D}_1$ and $\mathcal{D}_2$ are quite similar. The guess-and-determine steps are also similar, which are given in Supp. C. The guess-and-determine steps are also summarized in Table 3.

Degree of freedom and complexity. The analysis is almost the same to that in the last Sect. 4.2 due to the very similar differential trails in Block 1 and Block 2.

- In $\mathcal{I}_d^{(1)}$, $|V[\mathbb{A}^{(1)}]| = 2^{s_1^{(1)}+2s_2^{(1)}-1} = 2^{28}$.
- Given $V[\mathbb{A}^{(1)}]$, the only difference is that in $\mathcal{I}_d^{(1)}$ four bytes $V[\mathbb{G}^{(1)}] = (k_6[14], k_3[12], k_5[12], k_1[3])$ (marked by a wavy line) are guessed in Steps 4, 5, 6, and 7. There is no conflict in the $\mathcal{I}_d^{(1)}$, *i.e.*, $|\chi^{(1)}| = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d^{(1)}$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(1)}} = \frac{2^{28+4\times 8}}{2^{8\times 0}} = 2^{60}$.
- Since the probability of the outbound phase is $2^{-p_o^{(1)}} = 2^{-78}$, This means that at least $2^{18} = 2^{p_o^{(1)}}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(1)}}$ distinct values of $P = h_1$ are required to generate 2^{78} starting points to expect one pair satisfying the difference Δh_2.

4.4 Quantum Collision Attack on the 14-Round AES-256-DM

Based on the quantum strategy of converting free-start (FS) Collision into Collision introduced in Sect. 3.3, we give the 3-block quantum collision attack on the full 14-round AES-256-DM. There are two quantum models in Sect. 3.3, *i.e.*, Case 2(a) and Case 2(b). In our attack, according to the analysis in Sect. 4.2, $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}} < 2^{p_o^{(2)}}$ leads to the Case 2(b) works, which combines the quantum amplitude amplification (QAA) algorithm with nested Grover's algorithm.

Setup Phase. As the analysis in Sect. 4.2 and 4.3, there is no conflict in the inbound phases $\mathcal{I}_d^{(2)}$ and $\mathcal{I}_d^{(1)}$, *i.e.*, $|\chi^{(1)}| = |\chi^{(2)}| = |\tilde{\chi}^{(2)}| = 0$. By fixing the values $V[\mathbb{A}^{(1)}]$ and $V[\mathbb{G}^{(1)}]$ in $\mathcal{I}_d^{(1)}$, one determines h_2 with M_0 by Eq. (23) and the GD steps in Sect. 4.3. Therefore, randomly given a M_0, $(M_0, h_2) \in S := \{(M_0, h_2) : \textit{Condition I and II hold}\}$ holds with probability of $2^{-p_o^{(1)}} = 2^{-78}$. The algorithm $\mathcal{A}$ to produce the superposition $|\phi\rangle := \frac{1}{\sqrt{|S|}} \sum_{M_0 \in S} |M_0, h_2\rangle$ is to run Grover's algorithm on $g : \mathbb{F}_2^{256} \mapsto \mathbb{F}_2$, where $g(M_0) = 1$ if and only if $(M_0, h_2) \in S$. The implementation of g is given in Alg. 1.

Complexity of $\mathcal{U}_g$. For the full AES-256, there are $16 \times 14 + 52 = 276$ S-boxes. In Alg. 1, three full AES-256 with $276 \times 3 = 828$ S-boxes is used to compute h_1, h_2, and h_2'. And 22 S-boxes are applied in Table 3. Taking the uncomputing into account, the time complexity of g is about

$$\mathcal{T}_g = (828 + 22) \times 2 \approx 2^{10.73} \quad \text{S-box applications.} \tag{30}$$

It needs about $(28 + 4) \times 8 = 256$ qubits to store $V[\mathbb{A}^{(1)}]$ and $V[\mathbb{G}^{(1)}]$.

Algorithm 1: $\mathcal{U}_g$ of the quantum attack on full AES-256-DM

Input: $|M_0\rangle|y\rangle$
Output: $|M_0\rangle|y \oplus g(M_0)\rangle$

Compute $h_1 = \mathsf{AES\text{-}256\text{-}DM}(IV, M_0)$.
With $V[\mathbb{A}^{(1)}]$ and $V[\mathbb{G}^{(1)}]$ fixed, take h_1 as P and run GD steps in Table 3 to retrieve the key material M_1.
Compute $h_2 = \mathsf{AES\text{-}256\text{-}DM}(h_1, M_1)$.
Compute $h_2' = \mathsf{AES\text{-}256\text{-}DM}(h_1, M_1 \oplus \Delta M_1)$.
if $h_2 \oplus h_2' = \Delta h_2$ *(Condition I)* `/* a probability of` 2^{-78} `*/`
then
  return $|M_0\rangle|y \oplus 1\rangle$
else
  return $|M_0\rangle|y\rangle$

Complexity of $\mathcal{A}$. According to Eq. (24), the time complexity of $\mathcal{A}$ is

$$\mathcal{T}_{\mathcal{A}} = \frac{\pi}{4} \cdot \sqrt{2^{78}} \cdot 2^{10.73} \approx 2^{49.38} \quad \text{S-box applications.} \tag{31}$$

Checking Phase. Given a valid h_2, the checking phase is to find if there exists a M_2 that leads to the final collision in h_3. In order to apply Grover's search over the space of $V[\mathbb{A}^{(2)}]$, we use a built-in lookup circuit to access all the 2^{35} possible values of $V[\mathbb{A}^{(2)}]$ in the following method.

- Deduce the byte pairs (v_i^0, v_i^1) $(i = 0, 1, \cdots, 21)$ for the $s_1^{(2)}$=22 active S-boxes with differential probability of 2^{-7}. For i-th of the 22 S-boxes, introduce a bit $\ell_i \in \{0, 1\}$, and $\ell_i = 0$ if and only if v_i^0 is selected by $V[\mathbb{A}^{(2)}]$. Deduce the byte pairs (v_j^{00}, v_j^{10}) and (v_j^{01}, v_j^{11}) $(j = 22, 23, \cdots, 28)$ for $s_2^{(2)}$=7 S-boxes with differential probability of 2^{-6}. For j-th of the 7 S-boxes, introduce two bit $\tilde{\ell}_j \in \{0, 1\}^2$, and $v_j^{\tilde{\ell}_j}$ is selected by $V[\mathbb{A}^{(2)}]$.
- Store all the pairs (v_i^0, v_i^1) $(i = 1, \cdots, 21)$[6], (v_j^{00}, v_j^{10}), and (v_j^{01}, v_j^{11}) $(j = 22, 23, \cdots, 28)$ in a qRAM L indexed by $\mathsf{addr} = (\ell_1, \cdots, \ell_{21}, \tilde{\ell}_{22}, \cdots, \tilde{\ell}_{28}) \in \mathbb{F}_2^{35}$, which needs about $72 \times 8 = 576$ qubits. The quantum oracle of qRAM L is defined as $\mathcal{U}_L$:

$$\mathcal{U}_L(|\mathsf{addr}\rangle\,|0\rangle) = |\mathsf{addr}\rangle\,|v_1^{\ell_1}, \cdots, v_{21}^{\ell_{21}}, v_{22}^{\tilde{\ell}_{22}}, \cdots, v_{28}^{\tilde{\ell}_{28}}\rangle\,. \tag{32}$$

Define $\tilde{g}(h_2; \cdot) : \mathbb{F}_2^{35+24} \mapsto \mathbb{F}_2$, where $\tilde{g}(h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]) = 1$ if and only if there exists $\mathsf{addr} \in \mathbb{F}_2^{35}, V[\mathbb{G}^{(2)}] \in \mathbb{F}_2^{24}$ that leads to a collision in h_3. The implementation of $\tilde{g}(h_2; \cdot)$ is given in Alg. 2.

[6] To achieve $2^{(22+2\cdot 7)-1}$ possible $V[\mathbb{A}^{(2)}]$.

Algorithm 2: $\mathcal{U}_{\tilde{g}(h_2;\cdot)}$ of the quantum attack on full AES-256-DM

Input: $|h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]\rangle|y\rangle$
Output: $|h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]\rangle|y \oplus \tilde{g}(h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}])\rangle$

Get $V[\mathbb{A}^{(2)}] = (v_0^0, v_1^{\ell_1}, \cdots, v_{21}^{\ell_{21}}, v_{22}^{\tilde{\ell}_{22}}, \cdots, v_{28}^{\tilde{\ell}_{28}})$ with the addr by Eq. (32)
With values of $V[\mathbb{A}^{(2)}], V[\mathbb{G}^{(2)}]$, take h_2 as P and run GD steps in Table 2 to retrieve the key material M_2. `/* 22 S-box applications */`
Compute $h_3 = \mathsf{AES\text{-}256\text{-}DM}(h_2, M_2)$.
Compute $h_3' = \mathsf{AES\text{-}256\text{-}DM}(h_2 \oplus \Delta h_2, M_2 \oplus \Delta M_2)$.
if $h_3 \oplus h_3' = 0$ `/* a probability of` 2^{-78} `*/`
then
  return $|h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]\rangle|y \oplus 1\rangle$
else
  return $|h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]\rangle|y\rangle$

Algorithm 3: $\mathcal{U}_f$ of the quantum attack on full AES-256-DM

Input: $|h_2\rangle|y\rangle$
Output: $|h_2\rangle|y \oplus f(h_2)\rangle$

Run Grover's algorithm on $\tilde{g}(h_2;\cdot)$ to find good $(\mathsf{addr}, V[\mathbb{G}^{(2)}]) \in \mathbb{F}_2^{59}$.
Get $V[\mathbb{A}^{(2)}] = (v_0^0, v_1^{\ell_1}, \cdots, v_{21}^{\ell_{21}}, v_{22}^{\tilde{\ell}_{22}}, \cdots, v_{28}^{\tilde{\ell}_{28}})$ with the addr by Eq. (32).
Retrieve M_2 by $V[\mathbb{A}^{(2)}]$ and $V[\mathbb{G}^{(2)}]$ from GD steps in Table 2.
Compute $h_3 = \mathsf{AES\text{-}256\text{-}DM}(h_2, M_2)$.
Compute $h_3' = \mathsf{AES\text{-}256\text{-}DM}(h_2 \oplus \Delta h_2, M_2 \oplus \Delta M_2)$.
if $h_3 \oplus h_3' = 0$ `/* a probability of` 2^{-78} `*/`
then
  return $|h_2\rangle|y \oplus 1\rangle$
else
  return $|h_2\rangle|y\rangle$

Complexity of $\mathcal{U}_{\tilde{g}}$. Taking the uncomputing into account, the time of $\tilde{g}$ is

$$(22 + 276 + 276) \times 2 = 2^{10.16} \quad \text{S-box applications.} \tag{33}$$

Define $f(h_2) = 1$ if and only if there exists $\mathsf{addr}, V[\mathbb{G}^{(2)}]$ that $\tilde{g}(h_2; \mathsf{addr}, V[\mathbb{G}^{(2)}]) = 1$. The implementation of f is given in Alg. 3.

Complexty of $\mathcal{U}_f$. The time including the uncomputing is

$$\mathcal{T}_f = \frac{\pi}{4} \cdot \sqrt{2^{59}} \cdot 2^{10.16} + (22 + 276 \times 2) \times 2 \approx 2^{39.32} \quad \text{S-box applications.} \tag{34}$$

The Full Attack. The success probability of $\mathcal{A}$ that output the final collision is $2^{-t^{(2)}}$ according to Eq. (27), where $t^{(2)} = p_o^{(2)} - 59 = 19$. So the final time

complexity is

$$\mathcal{T}_{full} = \frac{\pi}{4} \cdot 2^{t^{(2)}/2} \cdot (\mathcal{T}_{\mathcal{A}} + \mathcal{T}_f) = \frac{\pi}{4} \cdot 2^{9.5} \cdot (2^{49.38} + 2^{39.32}) \approx 2^{58.53} \quad \text{S-boxes}, \tag{35}$$

$$= 2^{58.53}/276 \approx 2^{50.42} \quad \text{AES-256-DM}. \tag{36}$$

In Supp. D, we give a detailed evaluation on the size S_0 of quantum circuit of the full attack of AES-256-DM, which is about $S_0 = 4.27$ 1-block AES-256-DM. According to [40], assume we have a quantum computer of size S AES-256-DM, the time will be $T = 2^{50.42}/\sqrt{S/S_0}$ 1-block AES-256-DM. To be better than generic attack (**Scenario 2** in Sect. 2.6), *i.e.*, $T = 2^{50.42}/\sqrt{S/S_0} < 2^{64}/S$, we get $4.27 < S < 2^{25.07}$.

5 Applications to Reduced AES-128-DM and AES-192-DM

Following the framework converting the SFS collision into collision in Sect. 3.2, we give a classical collision attack on 4-round AES-128-DM in time 2^{59}, and a quantum collision attack on 5-round AES-128-DM with time 2^{55}. Please find the detailed attacks in Supp. E.1 and E.2. Based on the classic strategy of converting FS Collision into Collision introduced in Sect. 3.3, we propose a 3-block collision attack on 7-round AES-192-DM in time 2^{54} in Supp. F.

6 Verification Experiments

To validate the proposed methodology, we have conducted two experiments on 4-round AES-128-DM and 7-round AES-192-DM respectively within attainable computing resources.

Experiments on 4-round AES-128-DM. Following the procedure provided in Supp. E.1, we have implemented experiments to search for near-collisions on 4-round AES-128-DM. To be noted, we deliberately relaxed the differential condition on one active S-box in the third round of the differential trail $\mathcal{D}$. This simplification is only applied to the practical experiments and does not reflect the full attack setting. As a result, the time complexity is reduced from 2^{59} to approximately 2^{52}, making the experiments computationally feasible. The search was conducted on 72 NVIDIA V100 GPUs and required about 50 hours to obtain a valid near-collision. And the details of this practical near-collision are given in Supp. G, including the corresponding message pair in Table 8, and the differences of Block 1 in Fig. 24.

Experiments on 7-round AES-192-DM. Further, we have implemented experiments to search for collisions on 7-round AES-192-DM following the procedure detailed in Supp. F. The theoretical time complexity is 2^{54}. In practice, we employed 80 NVIDIA A800 GPUs together with 36 NVIDIA V100 GPUs, and the

search required about one week to obtain a message M_0 together with the corresponding $(V[\mathbb{A}^{(1)}], V[\mathbb{G}^{(1)}])$ that leads to a valid h_2. Using this valid h_2, we were able to find a collision on h_3. In Supp. G, we provide the details of the full-collision: the corresponding message pair is presented in Table 9, while the detailed differences in Block 2 and Block 3 are illustrated in Fig. 25 and Fig. 26, respectively.

7 Conclusion

This work proposed new methods to construct collision attacks from semi-free-start (SFS) and free-start (FS) collisions on DM hashing with AES through a revisited guess-and-determine (GD) rebound attack. Our models have yielded the first full quantum collision attack on AES-256-DM, extending the previous state-of-the-art by 5 rounds. This is the first quantum full collision attack against a well-studied hash function since Hosoyamada and Sasaki invented the seminal dedicated quantum collision at EUROCRYPT 2020. Furthermore, the practical implementations on the improved attacks on AES-128-DM and AES-192-DM demonstrate the correctness of our collision attack frameworks.

Acknowledgement. We thank the anonymous reviewers from CRYPTO 2026 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.

References

1. ZigBee Alliance. ZigBee 2007 specification. *Online: http://www.zigbee.org/*, 2007.
2. Seungjun Baek, Sehee Cho, and Jongsung Kim. Quantum cryptanalysis of the full AES-256-based davies–meyer, hirose and MJH hash functions. *Quantum Information Processing*, 21:163, 2022.
3. 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.
4. 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.
5. PSLM Barreto, Vincent Rijmen, et al. The whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14, 2000.
6. 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.
7. Barbara Jiabao Benedikt, Marc Fischlin, and Moritz Huppert. Nostradamus goes quantum. In *ASIACRYPT 2022*, pages 583–613. Springer, 2022.

8. Daniel J Bernstein. Cost analysis of hash collisions: Will quantum computers make sharcs obsolete. *SHARCS*, 9:105, 2009.
9. 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.
10. Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In *ASIACRYPT 2019*, pages 552–583.
11. Xavier Bonnetain, Gaëtan Leurent, María Naya-Plasencia, and André Schrottenloher. Quantum linearization attacks. In *ASIACRYPT 2021*, volume 13090, pages 422–452. Springer, 2021.
12. 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.
13. Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.
14. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN 1998*, pages 163–169. Springer, 1998.
15. Matthew Campagna. A cryptographic suite for embedded systems (suitee). *Internet-Draft*, 2012.
16. André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT 2017*, volume 10625, pages 211–240. Springer, 2017.
17. 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.
18. 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.
19. Bram Cohen and Ben Laurie. AES-hash. *NIST: Modes of Operation for Symmetric Key Block Ciphers*, 2001.
20. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
21. Ivan Damgård. A design principle for hash functions. In *CRYPTO '89*, pages 416–427, 1989.
22. 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.
23. 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.
24. Xiaoyang Dong, Jian Guo, Shun Li, and Phuong Pham. Triangulating rebound attack on aes-like hashing. In *CRYPTO 2022*, volume 13507, pages 94–124. Springer, 2022.
25. 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.
26. Xiaoyang Dong, Shun Li, Phuong Pham, and Guoyan Zhang. Quantum attacks on hash constructions with low quantum random access memory. *IACR Cryptol. ePrint Arch.*, page 1286, 2023.

27. Xiaoyang Dong, Siwei Sun, Danping Shi, Fei Gao, Xiaoyun Wang, and Lei Hu. Quantum collision attacks on AES-like hashing with low quantum random access memories. In *ASIACRYPT 2020*, volume 12492, pages 727–757.
28. Xiaoyang Dong, Zhiyu Zhang, Siwei Sun, Congming Wei, Xiaoyun Wang, and Lei Hu. Automatic classical and quantum rebound attacks on AES-like hashing by exploiting related-key differentials. In *ASIACRYPT 2021*, volume 13090, pages 241–271. Springer, 2021.
29. Antonio Flórez-Gutiérrez, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, André Schrottenloher, and Ferdinand Sibleyras. New results on gimli: Full-permutation distinguishers and improved collisions. In *ASIACRYPT 2020*, volume 12491, pages 33–63. Springer, 2020.
30. Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In *CRYPTO 2013*, volume 8042, pages 183–203.
31. 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.
32. David Gérault, Pascal Lafourcade, Marine Minier, and Christine Solnon. Computing AES related-key differential characteristics with constraint programming. *Artificial Intelligence*, 278:103183, 2020.
33. David Gérault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In *CP 2016*, volume 9892, pages 584–601. Springer, 2016.
34. Henri Gilbert. A simplified representation of AES. In *ASIACRYPT 2014*, volume 8873, pages 200–222. Springer, 2014.
35. Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010*, pages 365–383, 2010.
36. Lorenzo Grassi, María Naya-Plasencia, and André Schrottenloher. Quantum algorithms for the k-xor problem. In *ASIACRYPT 2018*, volume 11272, pages 527–559. Springer, 2018.
37. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC 1996*, pages 212–219, 1996.
38. 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.
39. Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In *EUROCRYPT 2020*, volume 12106, pages 249–279.
40. Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In *EUROCRYPT 2020*, pages 249–279. Springer, 2020.
41. Akinori Hosoyamada and Yu Sasaki. Quantum collision attacks on reduced SHA-256 and SHA-512. In *CRYPTO 2021*, volume 12825, pages 616–646. Springer, 2021.
42. Akinori Hosoyamada, Yu Sasaki, and Keita Xagawa. Quantum multicollision-finding algorithm. In *ASIACRYPT 2017*, volume 10625, pages 179–210.
43. 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.

44. Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *CRYPTO 2016*, pages 207–237, 2016.
45. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS 2016*, pages 830–842, 2016.
46. Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic. Speeding up collision search for byte-oriented hash functions. In *CT-RSA 2009*, volume 5473, pages 164–181.
47. Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl hash functions. In *FSE 2007*, volume 4593, pages 39–57.
48. Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *ASIACRYPT 2007*, volume 4833, pages 315–324. Springer, 2007.
49. 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.
50. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *ISIT 2010*, pages 2682–2685, 2010.
51. Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type Even-Mansour cipher. In *ISITA 2012*, pages 312–316, 2012.
52. 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.
53. Gregor Leander and Alexander May. Grover Meets Simon - quantumly attacking the FX-construction. In *ASIACRYPT 2017*, pages 161–178, 2017.
54. Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In *EUROCRYPT 2019*, volume 11478, pages 189–218. Springer.
55. Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced SHA-256. In *EUROCRYPT 2013*, volume 7881, pages 262–278. Springer, 2013.
56. 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.
57. Ralph C. Merkle. A certified digital signature. In *CRYPTO '89, 1989*, pages 218–238, 1989.
58. María Naya-Plasencia. How to improve rebound attacks. In *CRYPTO 2011*, volume 6841, pages 188–205. Springer, 2011.
59. María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum k-xor and k-xor-sum algorithms. In *EUROCRYPT 2020*, volume 12106, pages 311–340. Springer, 2020.
60. Jianqiang Ni, Yingxin Li, Fukang Liu, and Gaoli Wang. Practical key collision on AES and kiasu-bc. *IACR Cryptol. ePrint Arch.*, page 462, 2025.
61. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93*, volume 773, pages 368–378.
62. 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.
63. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In *FSE 2011*, pages 378–396.

64. 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.
65. André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. In *CRYPTO 2022*, volume 13509, pages 717–747. Springer, 2022.
66. Kodai Taiyama, Kosei Sakamoto, Ryoma Ito, Kazuma Taka, and Takanori Isobe. Key collisions on AES and its applications. *Cryptology ePrint Archive*, 2024.
67. Paul C Van Oorschot and Michael J Wiener. Parallel collision search with cryptanalytic applications. *Journal of cryptology*, 12(1):1–28, 1999.
68. 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.
69. 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.

Supplementary Material

A AES's Key Schedules

The key schedules for the three variants are given in Fig. 10, 11, and 12.

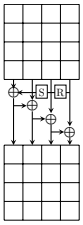


Fig. 10: AES-128

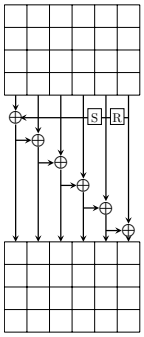


Fig. 11: AES-192

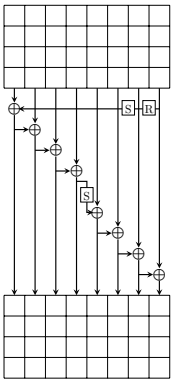


Fig. 12: AES-256

B Heuristic Method to Search our Collision Attacks

Our general procedures for searching differential characteristics and constructing the corresponding AES-DM collision attacks are summarized as follows.

1. **Searching for a differential characteristic.** We first search for a differential characteristic satisfying the degree-of-freedom requirement, namely, $\mathrm{DoF}_d \geq 2^{p_d}$. In addition, the characteristic should satisfy the differential constraints imposed by the corresponding attack model. For the differential characteristic used in the SFS-to-collision attack and the Block 1 differential characteristic used in the FS-to-collision attack, we require $\Delta P = 0$. For the Block 2 differential characteristic used in the FS-to-collision attack, we require $\Delta P = \Delta C$. Among the characteristics satisfying these requirements, we search for the one with overall probability as large as possible based on Gérault *et al.*'s search tool [32] and take it as the candidate for the subsequent GD-rebound attack.
2. **Determining the inbound phase.** After a candidate differential characteristic is obtained, we will determine the inbound phase heuristically. According to the analysis in Sect. 3.1, the input message P is fixed during the GD process and degrees of freedom only come from the key schedule. We want as many active S-box values as possible to be satisfied by the free key bytes. Therefore, we let the inbound phase cover as many active S-boxes

involved in the GD process as possible. Starting from the case where the number of such active S-boxes equals the number of free key bytes, we enumerate the choices obtained by increasing or decreasing this number. Based on this principle, we enumerate different choices of the inbound phase and run Bouillaguet, Derbez, and Fouque's GD tool [12]. In the SFS-to-collision framework, fixing the inbound phase determines all parameters in the overall complexity formula, namely Eq. (16). In the FS-to-collision framework, we search the inbound phases for Block 1 and Block 2 separately, treating their parameters as independent. Although Eq. (27) involves parameters from both blocks, during the separate search for one block, the parameters of the other block can be treated as fixed. The Eq. (27) allow us to compare different inbound choices for the two blocks, and select the ones leading to the lowest overall complexity. Experimentally, with fixed P, the inbound phase should start from the first round to efficiently satisfy the fixed P. The resulting attacks are not claimed to be optimal.

Following the above procedures, we successfully found feasible GD-rebound attacks on AES-DM. All the source codes and experiments can be found in:

https://github.com/DSwutang/AES-DM-collision

C Differential Trails $\mathcal{D}_1$ and $\mathcal{D}_2$, and the GD Steps of the Inbound Phase $\mathcal{I}_d^{(1)}$ of the Attack on AES-256-DM

Differential trail $\mathcal{D}_1$ for the 14-round AES-256-DM is given in Fig. 13. Differential trail $\mathcal{D}_2$ for the 14-round AES-256-DM is given in Fig. 14.

For $\mathcal{D}_1$, run the Eq. (5), with given $V[\mathbb{P}] = h_1$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 15. The $V[\mathbb{A}] = (x_0[0-15], x_1[0,5,8,10,15], x_2[0,4], k_6[12], k_7[12-15])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-15]$, $\Delta y_0[0-15]$, $\Delta x_1[0,3,4,8,9,14]$, $\Delta y_1[0,3,4,8,9,14]$, $\Delta x_2[0,4]$, $\Delta y_2[0,4]$, 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 $k_0[0-15] = (x_0 \oplus P)[0-15]$ (marked by $\overleftarrow{1}$). 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 $k_2[0] = (x_2 \oplus w_1)[0]$ (marked by $\overleftarrow{1}$).
2. According to the key relations, deduce $k_1[13]$, $k_2[2]$ and $k_2[4]$ (marked by 2). Compute forward to get $k_2[6,10,14]$ (marked by $\overrightarrow{2}$).Similarly, Compute $k_2[8,12]$ and $k_3[0]$ (marked by $\overrightarrow{2}$).
3. Deduce $k_1[12]$ and $k_5[12]$(marked by 3). Using $k_1[12]$, compute forward to get $k_2[3,7,11,15]$ (marked by $\overrightarrow{4}$).

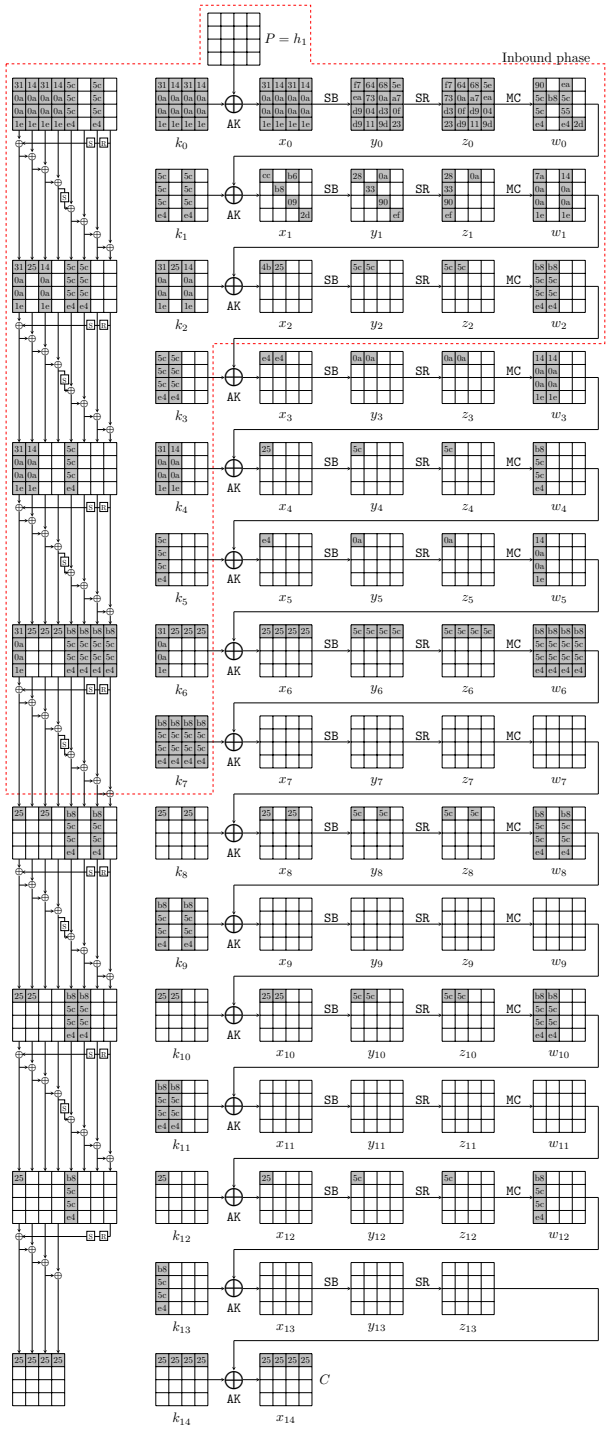


Fig. 13: Differential trail $\mathcal{D}_1$ for the 14-round AES-256-DM ($\Delta P = 0$).

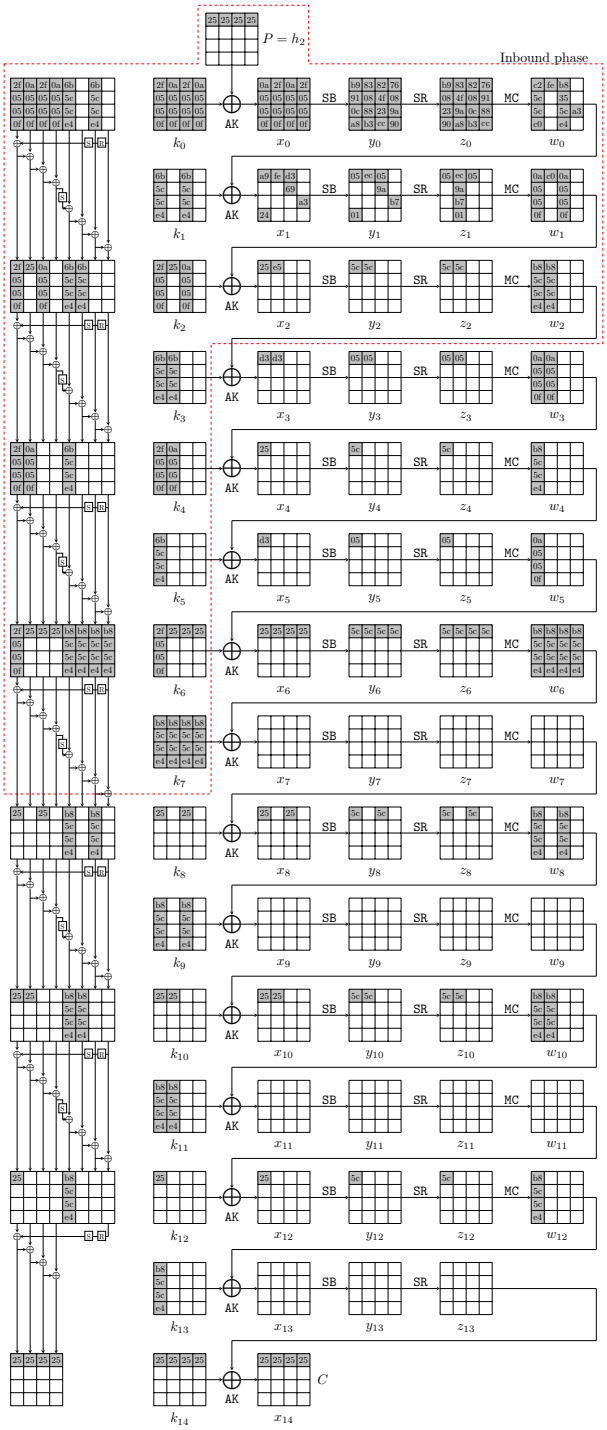


Fig. 14: Differential trail $\mathcal{D}_2$ for the 14-round AES-256-DM.

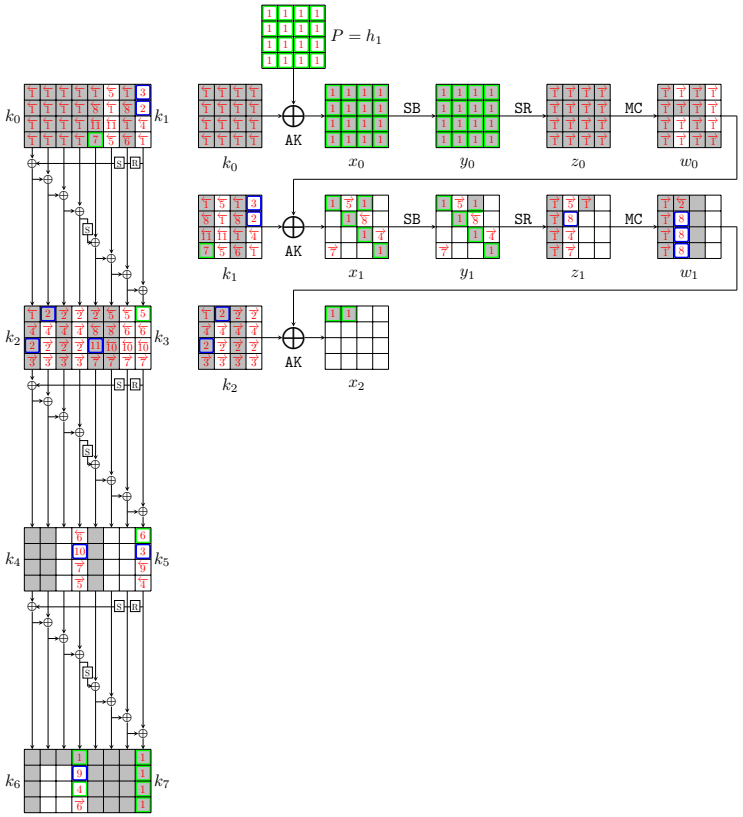


Fig. 15: The GD process of $\mathcal{I}_d^{(1)}$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

4. Guess $k_6[14]$ (marked by 4). Then deduce $k_5[15]$ and $k_1[14]$ (marked by $\overleftarrow{4}$).
   (a) In round 1, compute forward to get $x_1[14]$, $y_1[14]$ and $z_1[6]$ (marked by $\overrightarrow{4}$).
   (b) According to the key relations, compute forward to get $k_1[1, 5, 9, 13]$ (marked by $\overrightarrow{4}$).
5. Guess $k_3[12]$ (marked by 5).
   (a) Compute forward to get $k_4[15]$ (marked by $\overrightarrow{5}$).
   (b) Compute backward to get $k_3[4, 8]$ and $k_1[4]$ (marked by $\overleftarrow{5}$).
   (c) In round 1, compute forward to get $x_1[4]$, $y_1[4]$ and $z_1[4]$ (marked by $\overrightarrow{5}$).
6. Guess $k_5[12]$ (marked by 6).
   (a) Compute forward to get $k_7[3]$ (marked by $\overrightarrow{6}$).
   (b) Compute backward to get $k_4[12]$ and $k_3[9, 13]$ (marked by $\overleftarrow{6}$).
7. Guess $k_1[3]$ (marked by 7).

(a) In round 1, Compute forward to get $x_1[3]$, $y_1[3]$ and $z_1[7]$ (marked by $\overrightarrow{7}$).
(b) According to the key relations, compute forward to get $k_3[3,7,11,15]$ and $k_4[14]$ (marked by $\overrightarrow{7}$).

8. For column 2 over the MC operation in round 1, deduce $z_1[5]$ and $w_1[5-7]$ (marked by 8) from $z_1[4,6,7]$ and $w_1[4]$. In round 1, compute backward to get $y_1[9]$, $x_1[9]$ and $k_1[9]$ (marked by $\overleftarrow{8}$).
   (a) Using $k_1[9]$ to compute forward to get $k_3[5]$ (marked by $\overrightarrow{8}$).
   (b) Using $k_3[5]$ to compute backward to get $k_3[1]$ and $k_1[1]$ (marked by $\overleftarrow{8}$).
9. Deduce $k_6[13]$ (marked by 9). Then compute backward to get $k_5[14]$ (marked by $\overleftarrow{9}$).
10. Deduce $k_4[13]$ (marked by 10). Then compute backward to get $k_3[6,10,14]$ (marked by $\overleftarrow{10}$).
11. Deduce $k_3[2]$ (marked by 11). Compute backward to get $k_1[2,6]$ (marked by $\overleftarrow{11}$). Then we get all the key bytes.

1.	$k_0[0-15] = (x_0 \oplus P)[0-15]$	$k_1[0,5,8,10,15] = (x_1 \oplus w_0)[0,5,8,10,15]$
	$k_2[0] = (x_2 \oplus w_1)[0]$	
2.	$k_2[2] = \mathrm{SB}(k_1[15]) \oplus k_0[2]$	$k_2[4] = k_0[4] \oplus k_2[0]$
	$\mathrm{SB}(k_1[13]) = k_0[0] \oplus k_2[0] \oplus const$	$k_2[6,10,14] = k_0[6,10,14] \oplus k_2[2,6,10]$
	$k_2[8,12] = k_0[8,12] \oplus k_2[4,8]$	$k_3[0] = k_1[0] \oplus \mathrm{SB}(k_2[12])$
3.	$k_7[12] \oplus \mathrm{SB}(k_6[12]) = k_1[8] \oplus k_1[12]$	$\mathrm{SB}(k_5[13]) \oplus k_6[12] = k_0[8] \oplus k_0[12] \oplus const$
	$k_2[3] = k_0[3] \oplus \mathrm{SB}(k_1[12])$	$k_2[7,11,15] = k_0[7,11,15] \oplus k_2[3,7,11]$
4.	$k_1[14] = k_1[10] \oplus \mathrm{SB}(k_6[14]) \oplus k_7[14]$	$\mathrm{SB}(k_5[15]) = k_6[14] \oplus k_0[10] \oplus k_0[14]$
	$k_2[1,5,9,13] = k_0[1,5,9,13] \oplus \{\mathrm{SB}(k_1[14]), k_2[1,5,9]\}$	$x_1[14] = k_1[14] \oplus w_0[14]$
5.	$k_4[15] = \mathrm{SB}(k_3[12]) \oplus k_0[7] \oplus k_0[15]$	$k_3[8] = k_1[12] \oplus k_3[12]$
	$k_3[4] = k_1[8] \oplus k_3[8]$	$k_1[4] = k_3[0] \oplus k_3[4]$
	$x_1[4] = k_1[4] \oplus w_0[4]$	
6.	$k_6[15] = \mathrm{SB}(k_5[12]) \oplus k_0[11] \oplus k_0[15]$	
	$k_1[11] = \mathrm{SB}(k_6[15]) \oplus k_7[15] \oplus k_1[15]$	$\mathrm{SB}(k_4[12]) = k_5[12] \oplus k_1[4] \oplus k_1[12]$
	$\mathrm{SB}(k_3[13]) = k_4[12] \oplus k_0[8] \oplus k_0[12] \oplus const$	$k_3[9] = k_3[13] \oplus k_1[13]$
7.	$k_3[3] = \mathrm{SB}(k_2[15]) \oplus k_1[3]$	$k_3[7,11,15] = k_1[7,11,15] \oplus k_3[3,7,11]$
	$x_1[3] = k_1[3] \oplus w_0[3]$	
8.	$w_1[4,5,6,7] = \mathrm{MC}(z_1[4,5,6,7])$	$k_1[9] = x_1[9] \oplus w_0[9]$
	$k_3[5] = k_3[9] \oplus k_1[9]$	$k_3[1] = k_3[5] \oplus k_1[5]$
	$k_1[1] = k_3[1] \oplus \mathrm{SB}(k_2[13])$	
9.	$\mathrm{SB}(k_6[13]) = k_1[9] \oplus k_1[13] \oplus k_7[13]$	$\mathrm{SB}(k_5[14]) = k_0[9] \oplus k_0[13] \oplus k_6[13]$
10.	$\mathrm{SB}(k_4[13]) = k_3[1] \oplus k_3[5] \oplus k_3[9] \oplus k_3[13] \oplus k_5[13]$	$\mathrm{SB}(k_3[14]) = k_2[1] \oplus k_2[5] \oplus k_2[9] \oplus k_2[13] \oplus k_4[13]$
	$k_3[10] = k_3[14] \oplus k_1[14]$	$k_3[6] = k_3[10] \oplus k_1[10]$
11.	$k_3[2] = \mathrm{SB}(k_4[14]) \oplus k_3[6] \oplus k_3[10] \oplus k_3[14] \oplus k_5[14]$	$k_1[6] = k_3[2] \oplus k_3[6]$
	$k_1[2] = k_3[2] \oplus \mathrm{SB}(k_2[14])$	

Table 3: Equations in the GD steps of $\mathcal{I}_d^{(1)}$. Waved bytes are guessed.

Degree of freedom and complexity. The analysis is almost the same to that in the last Sect. 4.2 due to the very similar differential trails in Block 1 and Block 2.

- In $\mathcal{I}_d^{(1)}$, $|V[\mathbb{A}]| = 2^{s_1^{(1)}+2s_2^{(1)}-1} = 2^{35}$.
- Given $V[\mathbb{A}]$, the only difference is that in $\mathcal{I}_d^{(1)}$ four bytes $V[\mathbb{G}] = (k_6[14], k_3[12], k_5[12], k_1[3])$ (marked by a wavy line) are guessed in Steps 4, 5, 6, and 7. There is no conflict in the $\mathcal{I}_d^{(1)}$, *i.e.*, $|\chi| = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d^{(1)}$ are $\widehat{\mathrm{DoF}}_I = \frac{2^{35+4\times 8}}{2^{8\cdot 0}} = 2^{67}$.
- Since the probability of the outbound phase is $2^{-p_o^{(2)}} = 2^{-78}$, This means that at least $2^{11} = 2^{p_o^{(2)}}/\widehat{\mathrm{DoF}}_I$ distinct values of $P = h_1$ are required to generate 2^{78} starting points to expect one pair satisfying the difference Δh_2.

D Quantum Circuit Evaluation on the AES-256-DM Attack

On size of the quantum circuit of the full attack. We measure the depth and width of a quantum circuit in units of the depth and width of a quantum circuit for one-block AES-256-DM, respectively, so that the cost estimation is independent of specific implementation choices. According to [40], one S-box application is equivalent to 1/276 of one AES-256-DM computation, since AES-256 requires 276 S-box applications. The compact circuit for 1-block AES-256-DM needs 768 qubits (256 for the key, 128 for the internal state, 128 for the feed-forward state, about 256 for the S-Box ancilla qubits).

- Depth: The depth or the time complexity of the full attack is $2^{50.42}$ 1-block AES-256-DM in Eq. (36).
- Width: The width includes the circuit of $\mathcal{U}_g$ in Alg. 1 and $\mathcal{U}_f$ in Alg. 3.
  1. $\mathcal{U}_g$ in Alg. 1: M_0 needs 256 qubits to store; y needs 1 qubit; In Line 1, 768 qubits are needed to implement AES-256-DM; In Line 2, 256 qubits are used to store $V[\mathbb{A}^{(1)}]$ and $V[\mathbb{G}^{(1)}]$; Line 3 and 4 reuse the circuit of AES-256-DM from Line 1. In total, the width of $\mathcal{U}_g$ is about
  $$(256 + 1 + 768 + 256)/768 = 1281/768 \approx 1.67 \text{ AES-256-DM}.$$
  2. $\mathcal{U}_f$ in Alg. 3: Line 1 of Alg. 3 runs a nested Grover's algorithm on $\mathcal{U}_{\tilde{g}}$, which is implemented with Alg. 2. For $\mathcal{U}_{\tilde{g}}$, the input is 128-qubit h_2, 35-qubit addr, 24-qubit $V[\mathbb{G}^{(2)}]$, and 1-qubit y; In Line 2 of Alg. 2, 72 bytes are stored in $\mathcal{U}_L$, which needs about 576 qubits, and $V[\mathbb{A}^{(2)}]$ needs 29 bytes (or 232 qubits); In Line 3 and 4 reuse the implementation of AES-256-DM in Alg. 1, and 256 qubits are needed to store h_3 and h_3'. Totally, 128+35+24+1+576+232+256=1252 qubits are needed for $\mathcal{U}_{\tilde{g}}$. In Line 2 of Alg. 3, $V[\mathbb{A}^{(2)}]$ needs 29 bytes (or 232 qubits); Line 3 needs 256 qubits to store M_2; In Line 4-5, 256 qubits are needed to store h_3 and h_3'. Therefore, to implement $\mathcal{U}_f$, 1252+232+256+256=1996 qubits are needed.

  In total, the width of the full circuit is about
  $$(1281 + 1996)/768 = 3277/768 \approx 4.27 \text{ AES-256-DM}.$$

E Collision Attack on Round-Reduced AES-128-DM

Following the framework converting the SFS collision into collision in Sect. 3.2, we give a classical collision attack on 4-round AES-128-DM and a quantum collision attack on 5-round AES-128-DM.

E.1 Collision attack on 4-round AES-128-DM

Based on the classic strategy of converting semi-free-start (SFS) Collision into Collision introduced in Sect. 3.2, we propose a 2-block collision attack on 4-round AES-128-DM. The attack follows the framework in Fig. 6 in Sect. 3.2, where the related-key differential characteristic $\mathcal{D}$ ($\Delta h_1 = 0$, $\Delta M_1 \neq 0$, $\Delta h_2 = 0$) is introduced to Block 1. For this purpose, we find a 4-round differential trail for AES-128 using the automatic tool in [32,33]. It satisfies

$$\mathsf{AES}_{M_1}(h_1) \oplus \mathsf{AES}_{M_1 \oplus \Delta M_1}(h_1) = 0 \quad \text{(Block 1)}. \tag{37}$$

The differential trail $\mathcal{D}$ is given in Fig. 16. We choose the whole state of the subkeys and the first 2 rounds of the encryption path and one active S-box in round 2 as the inbound phase $\mathcal{I}_d$, and the remaining parts as the outbound phase $\mathcal{O}_d$. For $\mathcal{D}$ of Block 1 in Fig. 16, the inbound and outbound probabilities are 2^{-135} and $2^{-p_o} = 2^{-27}$, respectively. The guess-and-determine steps are given in Sect. E.1 and the detailed equations are in Sect. 4.

Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d$. Run the Eq. (5), with given $V[\mathbb{P}] = h_1$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 17. The $V[\mathbb{A}] = (x_0[0,4,5,8], x_1[0-11,13], x_2[9], k_1[12], k_2[13]$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0,4,5,8]$, $\Delta y_0[0,4,5,8]$, $\Delta x_1[0-11,13]$, $\Delta y_1[0-11,13]$, $\Delta x_2[9]$, $\Delta y_2[9]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 4.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute $k_0[0,4,5,8]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_0[0,1,4,8]$ (marked by $\overrightarrow{1}$).
   (b) In round 1, compute forward to get $z_1[0-2,4,5,7-11,13-15]$ and $w_1[8-11]$ (marked by $\overrightarrow{1}$). Compute backward to get $w_0[13]$ (marked by $\overleftarrow{1}$).
   (c) In round 2, compute backward to get $k_2[9]$ (marked by $\overleftarrow{1}$). Since $k_2[13] = k_1[13] \oplus k_2[9]$, there is a conflict with a probability of 2^{-8}.
2. Guess $k_1[14]$ (marked by 2). Then according to the key relations, deduce $k_0[13]$, $k_1[0,4,8,9,14]$ and $k_2[0,4,5,8]$ (marked by 2) as Table 4.
3. For column 2 over the MC operation in round 0, deduce $z_0[10,11]$ and $w_0[10,11]$ (marked by 3) from $z_0[8,9]$ and $w_0[8,9]$. Then compute backward to get $k_0[2,7]$ (marked by $\overleftarrow{3}$) and compute forward to get $k_1[10,11]$ and $k_0[14]$ (marked by $\overrightarrow{3}$).

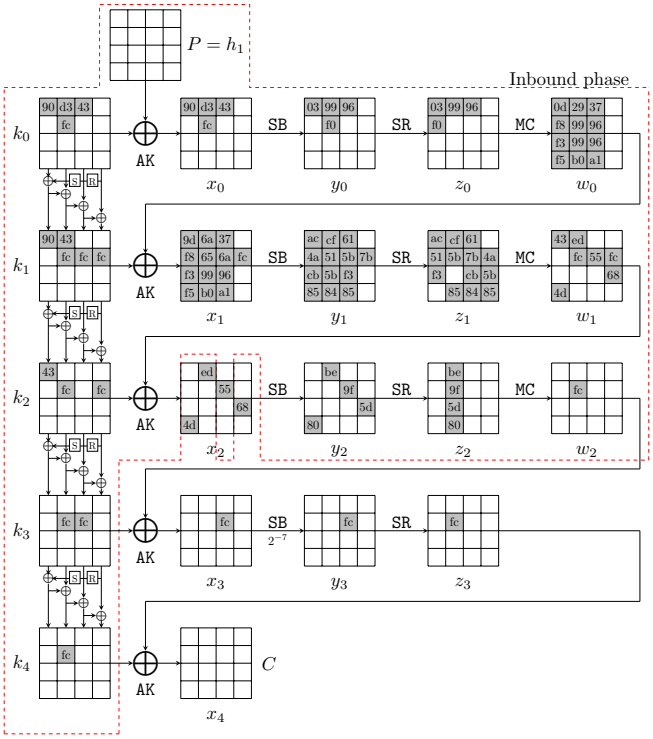


Fig. 16: Differential trail $\mathcal{D}$ for the 4-round AES-128-DM ($\Delta P = \Delta C = 0$).

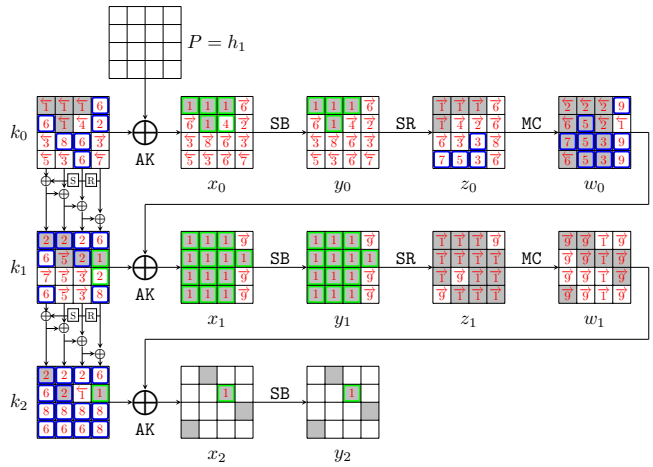


Fig. 17: The GD process of $\mathcal{I}_d$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

4. Guess $x_0[9]$ (marked by 4). Compute backward to get $k_0[9]$ (marked by $\overleftarrow{4}$) and compute forward to get $z_0[5]$ (marked by $\overrightarrow{4}$).
5. For column 1 over the MC operation in round 0, deduce $z_0[7]$ and $w_0[5,6,7]$ (marked by 5) from $z_0[4,5,6]$ and $w_0[4]$. Then compute backward to get $k_0[3]$ (marked by $\overleftarrow{5}$) and compute forward to get $k_1[5,6,7]$ (marked by $\overrightarrow{5}$). Since $k_1[5] = k_1[9] \oplus k_0[9]$ and $k_1[9]$, $k_0[9]$ are deduced in step 2 and 4, there is a conflict with a probability of 2^{-8}.
6. According to the key relations, deduce $k_0[1,10,11,12]$, $k_1[1,3,12]$ and $k_3[1,3,7,11,12]$ (marked by 6) as Table 4.
7. For column 0 over the MC operation in round 0, since there are six values known, there are two conflicts with a probability of 2^{-16}.. Then deduce $z_0[3]$ and $w_0[2]$ (marked by 7) from $z_0[0,1,2]$ and $w_0[0,1,3]$. Then compute backward to get $k_0[15]$ (marked by $\overleftarrow{7}$) and compute forward to get $k_1[2]$ (marked by $\overrightarrow{7}$). Since $\mathsf{SB}(k_0[15]) \oplus k_1[2] = k_0[2]$ and $k_0[2]$ is deduced in step 3, there is a conflict with a probability of 2^{-8}.
8. According to the key relations, deduce $k_0[6]$, $k_1[15]$ and $k_2[2,6,10,14,15]$ (marked by 8) as Table 4.
9. For column 3 over the MC operation in round 0, since there are five values known, there is a conflict with a probability of 2^{-8}.. Then deduce $w_0[12,14,15]$ (marked by 9) from $z_0[12-15]$ and $w_0[13]$. Then compute forward to get $z_1[3,6,12]$ and $w_1[0-7,12-15]$(marked by $\overrightarrow{9}$). Then we get all states of the starting point.

Analysis of the conflict byte equation system χ. As stated above, the conflict system has 6 equations, and we can resolve two conflicts in advance as follows:

- For the conflict $k_2[13] = k_1[13] \oplus k_2[9]$ in Step 1, it can be re-expressed as a combination of bytes marked by 1, as

$$k_2[13] \oplus k_1[13] \oplus x_2[9] \oplus \mathsf{SB}(x_1[8]) \oplus 02 \cdot \mathsf{SB}(x_1[13]) \oplus 03 \cdot \mathsf{SB}(x_1[2]) \oplus \mathsf{SB}(x_1[7]) = 0. \tag{38}$$

  In Eq. (38), there are 2^9 possible values of the 7 involved bytes. We fix them to

$$k_1[13] = \texttt{0xfc},\ k_2[13] = \texttt{0xf7},\ x_2[9] = \texttt{0},$$
$$x_1[2] = \texttt{0x64}, x_1[7] = \texttt{0xec},\ x_1[8] = \texttt{0x83},\ x_1[13] = \texttt{0x0f},$$

  which make the conflict satisfied with probability 1.
- In Step 4 of the GD procedure, $x_0[9]$ is guessed, and then leads to a conflict in Step 5. We represent the conflict by the deduced bytes in Step 1 to 3 and $x_0[9]$, which is

$$x_0[9] \oplus \mathsf{SB}(x_0[9]) = 03 \cdot z_0[4] \oplus 02 \cdot z_0[6] \oplus w_0[4] \oplus x_0[5] \oplus k_1[9] \oplus P[9]. \tag{39}$$

  So we build a hash table $H_{x_0[9]}$ before the GD process, which store the value of $x_0[9]$ indexed by the value $x_0[9] \oplus \mathsf{SB}(x_0[9])$. Then in Step 4, we first

calculate the value of the right part of Eq. (39), then access H to get right $x_0[9]$. So the conflict in Step 5 is satisfied with probability 1.

Degree of freedom and complexity.

- In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}]$ for active bytes from the input/output differences in the $\mathcal{I}_d$. Totally, there 20 active S-boxes, where 7 active bytes of Eq. (38) in $V[\mathbb{A}]$ are prefixed to solve the conflict in Step 1. There are 13 active S-boxes remaining, where $s_1 = 10$ S-boxes are of probability 2^{-7}, and $s_2 = 3$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}]| = 2^{s_1+2s_2} = 2^{16}$ choices[7] for the bytes marked by 1 (except bytes in P) in Fig. 17.
- Given $V[\mathbb{A}]$, one byte $V[\mathbb{G}] = (k_1[14])$ (marked by a wavy line) is guessed in Step 4, another "guessed" byte $x_0[9]$ is actually deduced (without guessing) by the hash table $H_{x_0[9]}$. There are four conflicts in the $\mathcal{I}_d$, *i.e.*, $|\chi| = 4$ and $|\tilde{\chi}| = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(1)}} = \frac{2^{16+1\times 8}}{2^{8\times 4}} = 2^{-8}$.
- Since there are four conflicts in the inbound phase, the time of the GD to find one starting point is $\mathcal{T}_{\mathsf{GD}} = 2^{32}$. Since the probability of the outbound phase is $2^{-p_o} = 2^{-27}$, which means that at least $2^t = 2^{27+8} = 2^{35} = 2^{p_o}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d}$ distinct values of $P = h_1$ are required to generate 2^{27} starting points to expect one collision. The memory complexity is 2^8 of storing $H_{x_0[9]}$.

Classic Collision Attack on the 4-Round AES-128-DM Based on $\mathcal{D}$ shown in Fig. 16, we can construct a 2-block collision attack. According to the attack procedure in Sect. 3.2 and the complexity analysis above, we have

$$s_1 = 10,\ s_2 = 3,\ \tilde{s}_1 = \tilde{s}_2 = 0,\ |\mathbb{G}| = 1,\ |\chi| = 4,\ |\tilde{\chi}| = 0,\ p_o = 27,$$

Since $2^{s_1+2s_2-1-(\tilde{s}_1+2\tilde{s}_2)} \times 2^{8\cdot|\mathbb{G}|} < 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)}$, according to Eq. (12), the expected time complexity of the collision attack on 4-Round AES-128-DM becomes

$$2^{|\tilde{\chi}|\cdot 8-(\tilde{s}_1+2\tilde{s}_2)+t} + 2^{p_o} \cdot 2^{8(|\chi|-|\tilde{\chi}|)} = 2^{59}. \tag{40}$$

E.2 Quantum Collision attack on 5-round AES-128-DM

Based on the quantum strategy of converting semi-free-start (SFS) Collision into Collision introduced in Sect. 3.2, we propose a 2-block collision attack on 5-round AES-128-DM. The attack follows the framework in Fig. 6 in Sect. 3.2, where the related-key differential characteristic $\mathcal{D}$ ($\Delta h_1 = 0, \Delta M_1 = 0, \Delta h_2 = 0$) is introduced to Block 1. For this purpose, we find a 5-round differential trails for AES-128 using the automatic tool in [32,33]. It satisfies

$$\mathsf{AES}_{M_1}(h_1) \oplus \mathsf{AES}_{M_1\oplus\Delta M_1}(h_1) = 0 \quad \text{(Block 1)}. \tag{41}$$

[7] Since the 7 active bytes of $V[\mathbb{A}]$ are already fixed, the number of non-repetitive combinations $V[\mathbb{A}]$ will be $2^{s_1+2s_2}$ without dividing by 2.

1.	$k_0[0,4,5,8] = (x_0 \oplus P)[0,4,5,8]$	$k_2[13] \stackrel{?}{=} k_1[13] \oplus k_2[9]$
2.	$k_0[13] = k_2[13] \oplus k_0[5] \oplus \mathsf{SB}(k_1[14])$	$k_1[0] = k_0[0] \oplus \mathsf{SB}(k_0[13]) \oplus const$
	$k_1[4,8] = k_0[4,8] \oplus k_1[0,4]$	$k_1[9] = k_1[13] \oplus k_0[13]$
	$k_2[0] = k_1[0] \oplus \mathsf{SB}(k_1[13]) \oplus const$	$k_2[4,8] = k_1[4,8] \oplus k_2[0,4]$
	$k_2[5] = k_2[9] \oplus k_1[9]$	
3.	$w_0[10,11], z_0[10,11] = \mathtt{MC}(z_0[8,9], w_0[8,9])$	$k_0[2,7] = P[2,7] \oplus \mathsf{SB}^{-1}(z_0[10,11])$
	$k_1[10,11] = w_0[10,11] \oplus x_1[10,11]$	$k_0[14] = k_1[14] \oplus k_1[10]$
4.	$k_0[9] = P[9] \oplus x_0[9]$	$z_0[5] = \mathsf{SB}(x_0[9])$
5.	$w_0[5,6,7], z_0[7] = \mathtt{MC}(z_0[4,5,6], w_0[4])$	$k_0[3] = P[3] \oplus \mathsf{SB}^{-1}(z_0[7])$
	$k_1[5,6,7] = w_0[5,6,7] \oplus x_1[5,6,7]$	$k_1[5] \stackrel{?}{=} k_1[9] \oplus k_0[9]$
6.	$k_1[1,3] = k_1[5,7] \oplus k_0[5,7]$	$k_0[10,11] = k_1[10,11] \oplus k_1[6,7]$
	$k_0[1] = k_1[1] \oplus \mathsf{SB}(k_0[14])$	$k_0[12] = \mathsf{SB}^{-1}(k_1[3] \oplus k_0[3])$
	$k_1[12] = k_0[12] \oplus k_1[8]$	$k_2[1,3] = k_1[1,3] \oplus \mathsf{SB}(k_1[14,12])$
	$k_2[7,11,12] = k_1[7,11,12] \oplus k_2[3,7,9]$	
7.	$w_0[2], z_0[3] = \mathtt{MC}(z_0[0,1,2], w_0[0,1,3])$?	$k_0[15] = P[15] \oplus \mathsf{SB}^{-1}(z_0[3])$
	$k_1[2] = w_0[2] \oplus x_1[2]$	$k_1[2] \oplus \mathsf{SB}(k_0[15]) \stackrel{?}{=} k_0[2]$
8.	$k_0[6] = k_1[6] \oplus k_1[2]$	$k_1[15] = k_0[15] \oplus k_1[11]$
	$k_2[2] = k_1[2] \oplus \mathsf{SB}(k_1[15])$	$k_2[6,10,14,15] = k_1[6,10,14,15] \oplus k_2[2,6,10,11]$
9.	$w_0[12,14,15] = \mathtt{MC}(z_0[12-15], w_0[13])$?	

Table 4: Equations in the GD steps for $\mathcal{I}_d$. Waved bytes are guessed. The red equations are conflicts.

The differential trails $\mathcal{D}$ is given in Fig. 18. We choose the whole state of the subkeys and the first 3 rounds of the encryption path as the inbound phase $\mathcal{I}_d$, and the remaining parts as the outbound phase $\mathcal{O}_d$. For $\mathcal{D}$ of Block 1 in Fig. 18, the inbound and outbound probabilities are 2^{-175} and $2^{-p_o} = 2^{-21}$, respectively. The guess-and-determine steps are given in Sect. E.2 and the detailed equations are in Sect. 5.

Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d$. Run the Eq. (5), with given $V[\mathbb{P}] = h_1$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 17. The $V[\mathbb{A}] = (x_0[0-2,6,10,13], x_1[0,2,5,8,11-13], x_2[0,1,5,6,10-12,15], k_0[13], k_1[14], k_2[14], k_3[14])$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-2,6,10,13]$, $\Delta y_0[0-2,6,10,13]$, $\Delta x_1[0,2,5,8,11-13]$, $\Delta y_1[0,2,5,8,11-13]$, $\Delta x_2[0,1,5,6,10-12,15]$, $\Delta y_2[0,1,5,6,10-12,15]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 4.

1. Deduce bytes from 1 bytes:
   (a) In round 0, since $k_0[13] = x_0[13] \oplus P[13]$ and the values of $k_0[13], x_0[13]$ and $P[13]$ are fixed, there is a conflict with probability of 2^{-8}. Then compute $k_0[0-2,6,10] = (x_0 \oplus P)[0-2,6,10]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_0[0,2,9,10,13,14]$ (marked by $\overrightarrow{1}$).
   (b) In round 1, compute forward to get $z_1[0,1,8-10,12,15]$ (marked by $\overrightarrow{1}$).

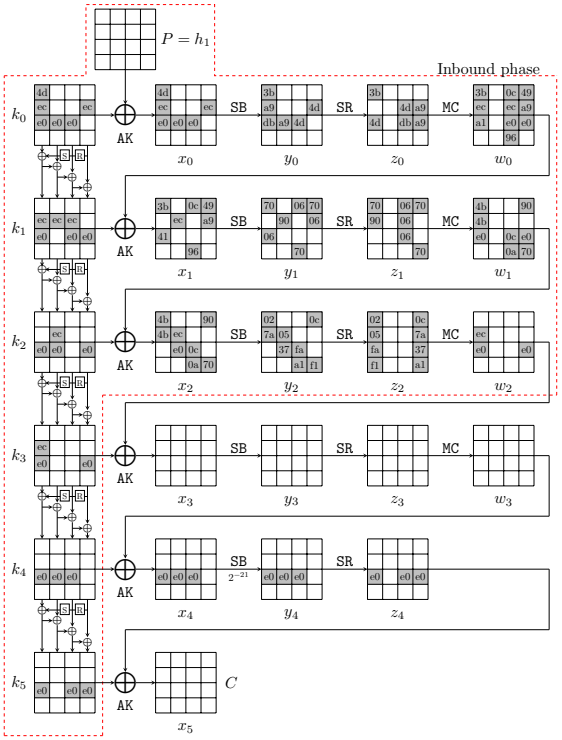


Fig. 18: Differential trail $\mathcal{D}$ for the 5-round AES-128-DM ($\Delta P = \Delta C = 0$).

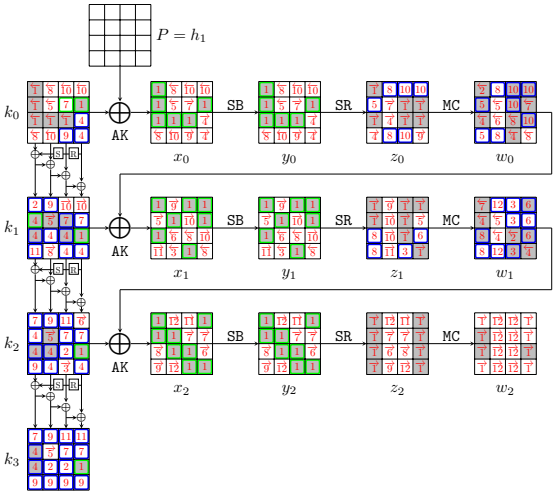


Fig. 19: The GD process of $\mathcal{I}_d$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

(c) In round 2, compute forward to get $z_2[0-1, 12-15]$ (marked by $\overleftarrow{1}$).

2. According to the key relations, deduce $k_1[0]$, $k_2[10]$ and $k_3[6, 10]$ (marked by 2) as Table 5. Compute backward to get $w_0[0]$ and $w_1[10]$ (marked by $\overleftarrow{2}$).
3. For column 2 over the MC operation in round 1, deduce $z_1[11]$ and $w_1[8, 9, 11]$ (marked by 3) from $z_1[8-10]$ and $w_1[10]$. Then compute backward to get $x_1[7]$ (marked by $\overleftarrow{3}$) and $k_2[11]$ (marked by $\overrightarrow{3}$).
4. Guess $k_1[2]$ (marked by 4). Then deduce $k_0[14, 15]$, $k_1[1, 6, 10, 11, 15]$, $k_2[1, 2, 6, 7, 15]$ and $k_3[1, 2]$ (marked by 4) as Table 5. Since $k_2[15]$ is be deduced twice from different known bytes, there is a conflict of with probability of 2^{-8}.
5. For column 0 over the MC operation in round 0, since five values are known, there is a conflict with a probability of 2^{-8}. Then deduce $z_0[1]$ and $w_0[1, 3]$ (marked by 5) from $z_0[0, 2, 3]$ and $w_0[0, 2]$.
   (a) Compute backward to get $k_0[5]$ (marked by $\overleftarrow{5}$).
   (b) Compute forward to get $x_1[1]$ and $z_1[13]$ (marked by $\overrightarrow{5}$).
   (c) Due to the key relation, deduce $k_1[5]$, $k_2[5]$ and $k_3[5]$ (marked by $\overrightarrow{5}$). And then compute backward to $w_0[5]$ and $w_1[5]$ (marked by $\overleftarrow{5}$).
6. For column 3 over the MC operation in round 1, deduce $z_1[14]$ and $w_1[12-14]$ (marked by 6) from $z_1[12, 13, 15]$ and $w_1[15]$. Then compute backward to get $w_0[6]$ (marked by $\overleftarrow{6}$) and $k_2[12]$ (marked by $\overrightarrow{6}$).
7. Guess $k_0[9]$ (marked by 7). Then deduce $k_1[9, 13]$, $k_2[0, 9, 13]$ and $k_3[0, 9, 13]$ (marked by 7) as Table 5. Since $k_2[15]$ is be deduced twice from different known bytes, there is a conflict of with probability of 2^{-8}.
8. For column 1 over the MC operation in round 0, deduce $z_0[4, 7]$ and $w_0[4, 7]$ (marked by 8) from $z_0[5, 6]$ and $w_0[5, 6]$. For column 0 over the MC operation in round 1, deduce $z_1[2, 3]$ and $w_1[2, 3]$ (marked by 8) from $z_1[0, 1]$ and $w_1[0, 1]$.
   (a) In round 0, compute backward to get $k_0[3, 4]$ (marked by $\overleftarrow{8}$).
   (b) In round 1, compute forward to $k_1[7]$ (marked by $\overrightarrow{8}$) and compute backward to get $x_1[10, 15]$ and $w_0[10, 15]$ (marked by $\overleftarrow{8}$).
   (c) In round 2, compute forward to $x_2[2]$ and $z_2[10]$ (marked by $\overrightarrow{8}$).
9. According to the key relations, deduce $k_0[11]$, $k_1[4]$, $k_2[3, 4]$ and $k_3[3, 4, 7, 11, 15]$ (marked by 9) as Table 5.
10. For column 2 over the MC operation in round 0, deduce $z_0[8, 11]$ and $w_0[8, 9]$ (marked by 10) from $z_0[9, 10]$ and $w_0[10, 11]$. For column 3 over the MC operation in round 0, deduce $z_0[12]$ and $w_0[12, 14]$ (marked by 10) from $z_0[13-15]$ and $w_0[13, 15]$, where has a conflict with a probability of 2^{-8}.
    (a) Compute backward to get $k_0[7, 8, 12]$ (marked by $\overleftarrow{10}$).
    (b) Compute forward to get $k_1[8, 12]$ and $z_1[5, 6]$ (marked by $\overrightarrow{10}$).
    (c) Since $k_1[8] \oplus k_1[12] = k_0[12]$, $k_1[8] \oplus k_0[8] = k_1[4]$ and $k_1[4]$ is deduced in step 9, there are two conflicts with a probability of 2^{-8}.
11. According to the key relations, deduce $k_1[3]$, $k_2[8]$ and $k_3[8, 12]$ (marked by 11). Since $k_1[3]$ is deduced three times from known values, there are two conflicts with a probability of 2^{-8}. $k_2[8]$ is deduced twice from known values, where has one conflict with a probability of 2^{-8}.

12. For column 1 over the MC operation in round 1, since six values are known, there are two conflict with a probability of 2^{-8}. Then deduce $w_1[4,7]$ (marked by 12) from $z_1[4-7]$ and $w_1[5,6]$. Then all states of the starting points are determined.

1.	$k_0[0-2,6,10] = (x_0 \oplus P)[0-15]$	
2.	$k_2[10] = k_2[14] \oplus k_1[14]$	$k_3[10] = k_2[14] \oplus k_2[14]$
	$k_1[0] = k_0[0] \oplus \mathsf{SB}(k_0[13]) \oplus const$	$k_3[6] = k_3[10] \oplus k_2[10]$
3.	$w_1[8,9,11], z_1[11] = \mathtt{MC}(z_1[8,9,10], w_1[10])$	$k_2[11] = w_1[11] \oplus x_2[11]$
4.	$k_0[15] = \mathsf{SB}^{-1}(k_0[2] \oplus k_1[2])$	$k_1[6] = k_0[6] \oplus k_1[2]$
	$k_1[10] = k_0[10] \oplus k_1[6]$	$k_0[14] = k_1[14] \oplus k_1[10]$
	$k_2[6,2] = k_2[10,6] \oplus k_1[10,6]$	$k_1[15] = \mathsf{SB}^{-1}(k_2[2] \oplus k_1[2])$
	$k_1[11] = k_1[15] \oplus k_0[15]$	$k_2[7] = k_2[11] \oplus k_1[11]$
	$k_3[2] = k_3[6] \oplus k_2[6]$	$k_2[15] = \mathsf{SB}^{-1}(k_3[2] \oplus k_2[2]) \stackrel{?}{=} k_1[15] \oplus k_2[11]$
	$k_1[1] = k_0[1] \oplus \mathsf{SB}(k_0[14])$	$k_2[1] = k_1[1] \oplus \mathsf{SB}(k_1[14])$
	$k_3[1] = k_2[1] \oplus \mathsf{SB}(k_2[14])$	
5.	$w_0[1,3], z_0[1] = \mathtt{MC}(z_0[0,2,3], w_0[0,2])?$	$k_0[5] = P[5] \oplus \mathsf{SB}^{-1}(z_0[1])$
	$k_1[5] = k_0[5] \oplus k_1[1]$	$k_2[5] = k_1[5] \oplus k_2[1]$
	$k_3[5] = k_2[5] \oplus k_3[1]$	
6.	$w_1[12-14], z_1[14] = \mathtt{MC}(z_1[12,13,15], w_1[15])$	$k_2[12] = w_1[12] \oplus x_2[12]$
7.	$k_1[9] = k_0[9] \oplus k_1[5]$	$k_1[13] = k_1[9] \oplus k_0[13]$
	$k_2[0] = k_1[0] \oplus \mathsf{SB}(k_1[13]) \oplus const$	$k_2[9,13] = k_1[9,13] \oplus k_2[5,9]$
	$k_3[0] = k_2[0] \oplus \mathsf{SB}(k_2[13]) \oplus const$	$k_3[9,13] = k_2[9,13] \oplus k_3[5,9]$
8.	$w_0[4,7], z_0[4,7] = \mathtt{MC}(z_0[5,6], w_0[5,6])$	$w_1[2,3], z_1[2,3] = \mathtt{MC}(z_1[0,1], w_1[0,1])$
	$k_0[3,4] = P[3,4] \oplus \mathsf{SB}^{-1}(z_0[7,4])$	$k_1[7] = w_0[7] \oplus x_1[7]$
9.	$k_1[4] = k_0[4] \oplus k_1[0]$	$k_2[4] = k_1[4] \oplus k_2[0]$
	$k_3[4] = k_2[4] \oplus k_3[0]$	$k_2[3] = k_2[7] \oplus k_1[7]$
	$k_0[11] = k_1[11] \oplus k_1[7]$	$k_3[3] = k_2[3] \oplus \mathsf{SB}(k_2[12])$
	$k_3[7,11,15] = k_2[7,11,15] \oplus k_3[3,7,11]$	
10.	$w_0[8,9], z_0[8,11] = \mathtt{MC}(z_0[9,10], w_0[10,11])$	$w_0[12,14], z_0[12] = \mathtt{MC}(z_0[13-15], w_0[13,15])?$
	$k_0[7,8,12] = P[7,8,12] \oplus \mathsf{SB}^{-1}(z_0[11,8,12])$	$k_1[8,12] = w_0[8,12] \oplus x_1[8,12]$
	$k_1[12] \oplus k_0[12] \stackrel{?}{=} k_1[8]$	$k_1[8] \oplus k_0[8] \stackrel{?}{=} k_1[4]$
11.	$k_1[3] = k_0[3] \oplus \mathsf{SB}(k_0[12]) \stackrel{?}{=} k_1[7] \oplus k_0[7] \stackrel{?}{=} k_2[3] \oplus \mathsf{SB}(k_1[12])$	
	$k_2[8] = k_1[8] \oplus k_2[4] \stackrel{?}{=} k_1[12] \oplus k_2[12]$	$k_3[8,12] = k_2[8,12] \oplus k_3[4,8]$
12.	$w_1[5,6] = \mathtt{MC}(z_1[4-7], w_1[4,7])?$	

Table 5: Equations in the GD steps for $\mathcal{I}_d$. Waved bytes are guessed. The red equations are conflicts.

Degree of freedom and complexity.

– In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}]$ for active bytes from the input/output differences in the $\mathcal{I}_d$. There are 25 active S-boxes, where $s_1 = 25$ S-boxes are of probability 2^{-7}, and $s_2 = 0$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}]| = 2^{s_1+2s_2-1} = 2^{24}$ choices for the bytes marked by 1 (except bytes in P) in Fig. 19.

– Given $V[\mathbb{A}]$, two bytes $V[\mathbb{G}] = (k_1[2], k_0[9])$ (marked by a wavy line) is guessed in Step 4 and 7. There are 12 conflicts in the $\mathcal{I}_d$, *i.e.*, $|\chi| = 12$ and one conflict in χ is only related to $(V[\tilde{\mathbb{A}}], V[\mathbb{P}])$ in Step 1(a), *i.e.*, $|\tilde{\chi}| = 1$, where $V[\tilde{\mathbb{A}}] = (k_0[13], x_0[13])$ with $\tilde{s_1} = 2$ and $\tilde{s_2} = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d} = \frac{2^{24-2+2\times 8}}{2^{8\times 11}} = 2^{-50}$.
– Since the inbound phase $\mathcal{I}_d$ contains a total of 12 conflicts, and one of them (given in (8)) is resolved differently, the time of the GD to find a starting point is based on the remaining 11 conflicts: $\mathcal{T}_{\mathsf{GD}} = 2^{11\times 8} = 2^{88}$. Since the probability of the outbound phase is $2^{-p_o} = 2^{-21}$, which means that at least $2^{71} = 2^{p_o}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d}$ distinct values of $P = h_1$ are required to generate 2^{21} starting points to expect one collision.

Quantum Collision Attack on the 5-Round AES-128-DM. Based on the quantum strategy of converting semi-free-start (SFS) Collision into Collision introduced in Sect. 3.2, we give the 2-block quantum collision attack on the 5-round AES-128-DM. There are two quantum models in Sect. 3.2, *i.e.*, Case 3(a) and Case 3(b). In our attack, according to the analysis, $\widehat{\mathrm{DoF}}_{\mathcal{I}_d} < 2^{p_o}$ leads to the Case 3(b) works, which combines the quantum amplitude amplification (QAA) algorithm with nested Grover's algorithm.

Setup Phase. As the analysis in Sect. E.2, there are 12 conflicts in the inbound phase $\mathcal{I}_d$, one of which is in (8), *i.e.*, $|\chi| = 12, |\tilde{\chi}| = 1$. In the quantum attack, only one value for $V[\tilde{\mathbb{A}}] = (k_0[13], x_0[13])$ is chosen in order to achieve memoryless. Randomly given a M_0, $(M_0, h_1) \in S := \{(M_0, h_1) : \tilde{\chi}(V[\tilde{\mathbb{A}}], h_1) = 0\}$ holds with of $2^{-|\tilde{\chi}|\cdot 8} = 2^{-8}$. The algorithm $\mathcal{A}$ to produce the superposition $|\phi\rangle := \frac{1}{\sqrt{|S|}} \sum_{M_0 \in S} |M_0, h_1\rangle$ is to run Grover's algorithm on $g : \mathbb{F}_2^{128} \mapsto \mathbb{F}_2$, where $g(M_0) = 1$ if and only if $(M_0, h_1) \in S$. The implementation of g is given in Alg. 4.

Algorithm 4: $\mathcal{U}_g$ of the quantum attack on 5-round AES-128-DM

Input: $|M_0\rangle|y\rangle$
Output: $|M_0\rangle|y \oplus g(M_0)\rangle$

Compute $h_1 =$ 5-round AES-128-DM(IV, M_0).
With $V[\tilde{\mathbb{A}}]$ fixed, take h_1 as P and check $\tilde{\chi}(V[\tilde{\mathbb{A}}], V[\mathbb{P}])$
if $\tilde{\chi}(V[\tilde{\mathbb{A}}], V[\mathbb{P}]) = 0$ `/* a probability of` 2^{-8} `*/`
then
  return $|M_0\rangle|y \oplus 1\rangle$
else
  return $|M_0\rangle|y\rangle$

Complexity of $\mathcal{U}_g$. For the 5-round AES-128, there are $16\times 5+20 = 100$ S-boxes. In Alg. 4, a 5-round AES-128 with 100 S-boxes is used to compute h_1. Taking the uncomputing into account, the time complexity of g is about

$$\mathcal{T}_g = 100 \times 2 \approx 2^{7.64} \quad \text{S-box applications.} \tag{42}$$

It needs about $2 \times 8 = 16$ qubits to store $V[\tilde{\mathbb{A}}] = (k_0[13], x_0[13])$.

Complexity of $\mathcal{A}$. The time complexity of $\mathcal{A}$ is

$$\mathcal{T}_{\mathcal{A}} = \frac{\pi}{4} \cdot 2^4 \cdot 2^{7.64} \approx 2^{11.30} \quad \text{S-box applications.}$$

Checking Phase. Given a valid h_1, the checking phase is to find if there exists a M_1 that leads to the final collision in h_2. In order to apply Grover's search over the space of $V[\mathbb{A} \setminus \tilde{\mathbb{A}}]$, we use a built-in lookup circuit to access all the 2^{22} possible values of $V[\mathbb{A} \setminus \tilde{\mathbb{A}}]$ in the following method.

- Deduce the byte pairs (v_i^0, v_i^1) $(i = 0, 1, \cdots, 22)$ for the $s_1 - \tilde{s_1}$=23 active S-boxes with differential probability of 2^{-7}. For i-th of the 23 S-boxes, introduce a bit $\ell_i \in \{0, 1\}$, and $\ell_i = 0$ if and only if v_i^0 is selected by $V[\mathbb{A} \setminus \tilde{\mathbb{A}}]$.
- Store all the pairs (v_i^0, v_i^1) $(i = 1, \cdots, 22)$[8] in a qRAM L indexed by $\mathsf{addr} = (\ell_1, \cdots, \ell_{22}) \in \mathbb{F}_2^{22}$, which needs about $46 \times 8 = 288$ qubits. The quantum oracle of qRAM L is defined as $\mathcal{U}_L$:

$$\mathcal{U}_L(|\mathsf{addr}\rangle\,|0\rangle) = |\mathsf{addr}\rangle\,|v_1^{\ell_1}, \cdots, v_{22}^{\ell_{22}}\rangle\,. \tag{43}$$

Define $\tilde{g}(h_1; \cdot) : \mathbb{F}_2^{22+16} \mapsto \mathbb{F}_2$, where $\tilde{g}(h_1; \mathsf{addr}, V[\mathbb{G}]) = 1$ if and only if there exists $\mathsf{addr} \in \mathbb{F}_2^{22}, V[\mathbb{G}] \in \mathbb{F}_2^{16}$ that leads to a collision in h_2. The implementation of $\tilde{g}(h_1; \cdot)$ is given in Alg. 5.

Complexity of $\mathcal{U}_{\tilde{g}}$. Taking the uncomputing into account, the time of is

$$(35 + 100 + 100) \times 2 = 2^{7.88} \quad \text{S-box applications.} \tag{44}$$

Define $f(h_1) = 1$ if and only if there exists $(\mathsf{addr}, V[\mathbb{G}])$ that $\tilde{g}(h_1; \mathsf{addr}, V[\mathbb{G}]) = 1$. The implementation of f is given in Alg. 6.

Complexty of $\mathcal{U}_f$. The time including the uncomputing is

$$\mathcal{T}_f = \frac{\pi}{4} \cdot \sqrt{2^{38}} \cdot 2^{7.88} + (35 + 100 \times 2) \times 2 \approx 2^{26.53} \quad \text{S-box applications.} \tag{45}$$

The Full Attack. The success probability of $\mathcal{A}$ that output the final collision is 2^{-t} according to Eq. (16) and (12), where $t = p_o - 38 = 71$. So the final time complexity is

$$\mathcal{T}_{full} = \frac{\pi}{4} \cdot 2^{t/2} \cdot (\mathcal{T}_{\mathcal{A}} + \mathcal{T}_f) = \frac{\pi}{4} \cdot 2^{35.5} \cdot (2^{11.30} + 2^{26.53}) \approx 2^{61.68} \quad \text{S-boxes,} \tag{46}$$

$$= 2^{61.68}/100 \approx 2^{55.04} \quad \text{5-round AES-128-DM.} \tag{47}$$

[8] To achieve 2^{23-1} possible $V[\mathbb{A} \setminus \tilde{\mathbb{A}}]$.

Algorithm 5: $\mathcal{U}_{\tilde{g}(h_1;\cdot)}$ of the quantum attack on 5-round AES-128-DM

Input: $|h_1; \mathsf{addr}, V[\mathbb{G}]\rangle|y\rangle$
Output: $|h_1; \mathsf{addr}, V[\mathbb{G}]\rangle|y \oplus \tilde{g}(h_1; \mathsf{addr}, V[\mathbb{G}])\rangle$

Get $V[\mathbb{A}] = (v_0^0, v_1^{\ell_1}, \cdots, v_{22}^{\ell_{22}})$ with the addr according to Eq. (43)
With values of $V[\mathbb{A} \setminus \tilde{\mathbb{A}}], V[\mathbb{G}]$, take h_1 as P and run GD steps in Table 5 to retrieve the key material M_1. `/* 35 S-box applications */`
Compute $h_2 = 5\text{-round AES-128-DM}(h_1, M_1)$.
Compute $h_2' = 5\text{-round AES-128-DM}(h_1, M_1 \oplus \Delta M_1)$.
if $h_2 \oplus h_2' = 0$ `/* a probability of` $2^{-11\times 8-21} = 2^{-109}$ `*/`
then
  return $|h_1; \mathsf{addr}, V[\mathbb{G}]\rangle|y \oplus 1\rangle$
else
  return $|h_1; \mathsf{addr}, V[\mathbb{G}]\rangle|y\rangle$

Algorithm 6: $\mathcal{U}_f$ of the quantum attack on 5-round AES-128-DM

Input: $|h_1\rangle|y\rangle$
Output: $|h_1\rangle|y \oplus f(h_1)\rangle$

Run Grover's algorithm on $\tilde{g}(h_1;\cdot)$ to find good $(\mathsf{addr}, V[\mathbb{G}]) \in \mathbb{F}_2^{38}$.
Get $V[\mathbb{A}] = (v_0^0, v_1^{\ell_1}, \cdots, v_{22}^{\ell_{22}})$ with the addr according to Eq. (43).
Retrieve M_1 by $V[\mathbb{A}]$ and $V[\mathbb{G}]$ from GD steps in Table 5.
Compute $h_2 = 5\text{-round AES-128-DM}(h_1, M_1)$.
Compute $h_2' = 5\text{-round AES-128-DM}(h_1, M_1 \oplus \Delta M_1)$.
if $h_2 \oplus h_2' = 0$ `/* a probability of` 2^{-109} `*/`
then
  return $|h_1\rangle|y \oplus 1\rangle$
else
  return $|h_1\rangle|y\rangle$

On size of the quantum circuit of the full attack. We use the depth and width of a quantum circuit for one-block AES-128-DM as the basic units for measuring circuit depth and width, respectively. This normalization makes our cost estimation less dependent on specific implementation choices for the underlying primitives. According to [40], one S-box application is equivalent to 1/100 of one 5-round AES-128-DM computation, since 5-round AES-128 requires 100 S-box applications. The compact circuit for 1-block AES-128-DM needs 640 qubits (128 for key, 128 for internal state, 128 for feedforward state, about 256 for S-Box ancilla qubits).

- Depth of the full attack: Following the above assumptions and taking the uncomputing into account, the full depth or time complexity of the attack is $2^{55.04}$ 5-round AES-128-DM in (46).
- Width: The width of the full circuit includes the circuit of $\mathcal{U}_g$ in Alg. 4 and $\mathcal{U}_f$ in Alg. 6.
  1. Analysis of $\mathcal{U}_g$ in Alg. 4: M_0 needs 128 qubits to store; y needs 1 qubit; In Line 1, 640 qubits are needed to implement 5-round AES-128-DM; In Line 2, 16 qubits are used to store $V[\tilde{\mathbb{A}}]$. In total, the width of $\mathcal{U}_g$ is about
  $$(128 + 1 + 640 + 16)/640 = 785/640 \approx 1.23 \quad \text{5-round AES-128-DM}.$$
  2. Analysis of $\mathcal{U}_f$ in Alg. 6: Line 1 of Alg. 6 runs a nested Grover's algorithm on $\mathcal{U}_{\tilde{g}}$, which is implemented with Alg. 5. For $\mathcal{U}_{\tilde{g}}$, the input is 128-qubit h_1, 22-qubit addr, 16-qubit $V[\mathbb{G}]$, and 1-qubit y; In Line 2 of Alg. 5, 46 bytes are stored in $\mathcal{U}_L$, which needs about 288 qubits, and $V[\mathbb{A}]$ needs 23 bytes (or 184 qubits); In Line 3 and 4 reuse the implementation of 5-round AES-128-DM in Alg. 4, and 256 qubits are needed to store h_3 and h'_3. Totally, 128+22+16+1+288+184+256=895 qubits are needed for $\mathcal{U}_{\tilde{g}}$. In Line 2 of Alg. 6, $V[\mathbb{A}]$ needs 23 bytes (or 184 qubits); Line3 needs 128 qubits to store M_1; In Line 4-5, 256 qubits are needed to store h_3 and h'_3. In total, to implement $\mathcal{U}_f$, 895+184+128+256=1463 qubits are needed. In total, the width of the full circuit is about
  $$(785 + 1463)/640 = 2248/640 \approx 3.51 \quad \text{5-round AES-128-DM}.$$

With a quantum computer of size $S_0 = 3.51$ 5-round AES-128-DM, the time of the attack is $T = 2^{55.04}$ 5-round AES-128-DM. According to [40], assume we have a quantum computer of size S 5-round AES-128-DM, the time will be $T = 2^{55.04}/\sqrt{S/S_0}$ 5-round AES-128-DM. To be better than generic attack (**Scenario 2** in Sect. 2.6), *i.e.*, $T = 2^{55.04}/\sqrt{S/S_0} < 2^{64}/S$, we get $3.51 < S < 2^{16.11}$.

F Collision Attack on Round-Reduced AES-192-DM

Based on the classic strategy of converting free-start (FS) Collision into Collision introduced in Sect. 3.3, we propose a 3-block collision attack on 7-round AES-192-DM. The attack follows the framework in Fig. 8 in Sect. 3.3, where the

related-key differential characteristics $\mathcal{D}_1$ ($\Delta h_1 = 0$, $\Delta M_1 \neq 0$, $\Delta h_2 \neq 0$) and $\mathcal{D}_2$ ($\Delta h_3 = 0$, $\Delta M_2 \neq 0$, $\Delta h_2 \neq 0$) are introduced to Block 1 and 2. For this purpose, we find two 7-round differential trails for AES-192 using the automatic tool in [32,33]. The first trail $\mathcal{D}_1$ satisfies

$$\mathsf{AES}_{M_1}(h_1) \oplus \mathsf{AES}_{M_1 \oplus \Delta M_1}(h_1) = \Delta h_2 \neq 0 \quad \text{(Block 1)}, \tag{48}$$

and the second trail $\mathcal{D}_2$ satisfies

$$\mathsf{AES}_{M_2}(h_2) \oplus \mathsf{AES}_{M_2 \oplus \Delta M_2}(h_2 \oplus \Delta h_2) = \Delta h_2 \quad \text{(Block 2)}. \tag{49}$$

The differential trails $\mathcal{D}_1$ and $\mathcal{D}_2$ are given in Fig. 20 and Fig. 21. For $\mathcal{D}_1$, we choose the first 3 rounds of the subkeys and the first 3 rounds except $x_2[7]$ of the encryption path as the inbound phase $\mathcal{I}_d^{(1)}$, and the remaining parts as the outbound phase $\mathcal{O}_d^{(1)}$. For $\mathcal{D}_2$, we choose the first 4 rounds of the subkeys and the first 4 rounds except $x_3[12]$ of the encryption path as the inbound phase $\mathcal{I}_d^{(2)}$, and the remaining parts as the outbound phase $\mathcal{O}_d^{(2)}$. For $\mathcal{D}_1$ of Block 1 in Fig. 20, the inbound and outbound probabilities are 2^{-157} and 2^{-54}, respectively. For $\mathcal{D}_2$ of Block 2 in Fig. 21, they are 2^{-157} and 2^{-34}. The guess-and-determine steps are given in Sect. F.1 and Sect. F.2, and the detailed equations are listed in Sect. 6 and Sect. 7.

F.1 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d^{(1)}$

Run the Eq. (5), with given $V[\mathbb{P}] = h_1$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 22. The $V[\mathbb{A}] = (x_0[0-3, 8, 12], x_1[0-3, 5-14], x_2[0, 2, 13], k_1[4]$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-3, 8, 12]$, $\Delta y_0[0-3, 8, 12]$, $\Delta x_1[0-3, 5-14]$, $\Delta y_1[0-3, 5-14]$, $\Delta x_2[0, 2, 13]$, $\Delta y_2[0, 2, 13]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 6.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute backward to get $k_0[0-3, 8, 12]$ (marked by $\overleftarrow{1}$). Compute forward to get $z_0[0, 7, 8, 10, 12, 13]$(marked by $\overrightarrow{1}$).
   (b) In round 1, compute forward to get $z_1[0-2, 5-15]$ and $w_1[8-15]$(marked by $\overrightarrow{1}$).
   (c) In round 2, compute backward to get $k_2[13]$ (marked by $\overleftarrow{1}$).
2. According to the key relations, deduce $k_1[11]$ (marked by 2). Then compute backward to get $w_0[11]$ (marked by $\overleftarrow{2}$).
3. Guess $k_1[7]$ (marked by 3).
   (a) Compute forward to get $k_1[10]$ (marked by $\overrightarrow{3}$).
   (b) Compute backward to get $w_0[7, 10]$ (marked by $\overleftarrow{3}$).
4. For column 2 over the MC operation in round 0, deduce $z_0[9, 11]$ and $w_0[8, 9]$ (marked by 4).

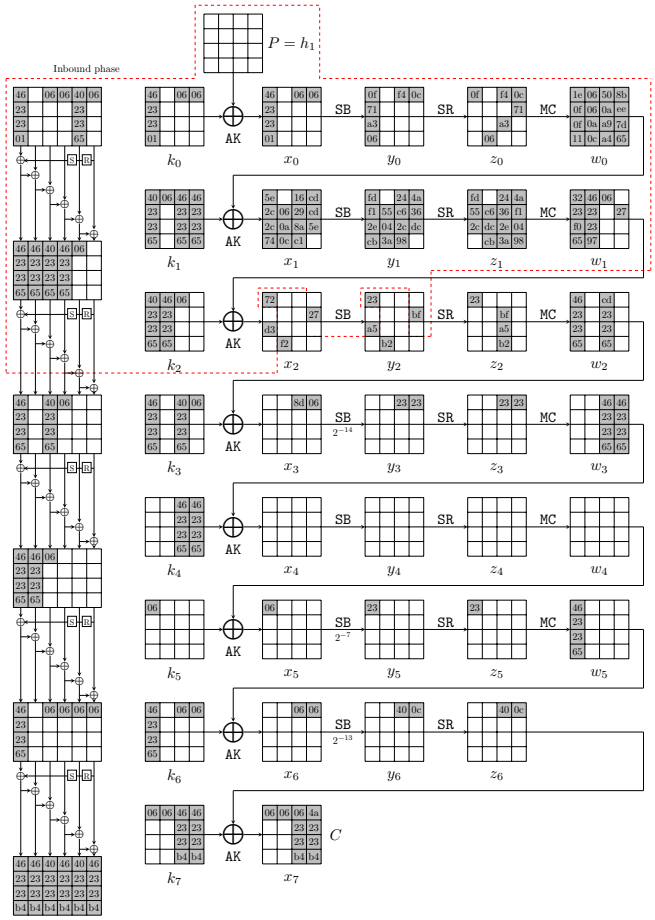


Fig. 20: Differential trail $\mathcal{D}_1$ for the 7-round AES-192-DM ($\Delta P = 0$).

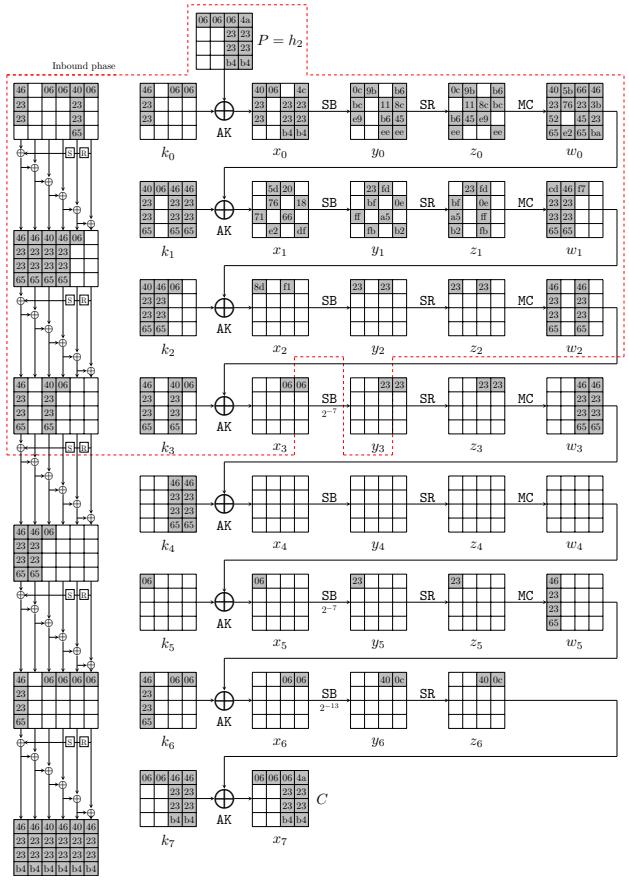


Fig. 21: Differential trail $\mathcal{D}_2$ for the 7-round AES-192-DM ($\Delta P = \Delta C$).

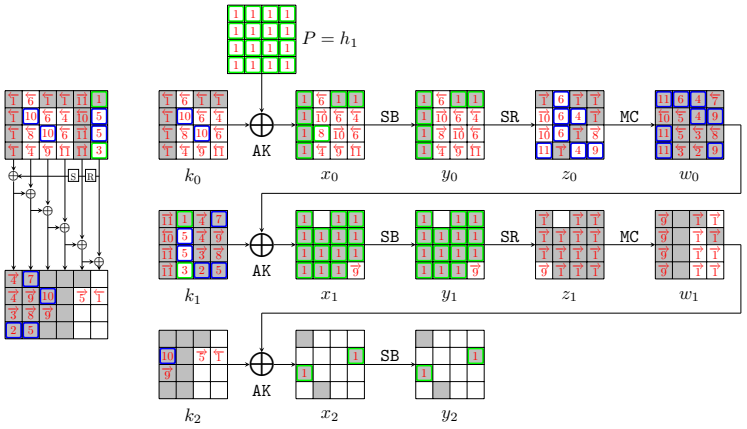


Fig. 22: The GD process of $\mathcal{I}_d^{(1)}$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

(a) Compute backward to get $k_0[7, 13]$ (marked by $\overleftarrow{4}$).
(b) Compute forward to get $k_1[8, 9]$ (marked by $\overrightarrow{4}$).

5. According to the key relations, deduce $k_1[5, 6, 15]$ (marked by 4). Then from $k_1[5, 6]$ compute backward to get $w_0[6, 15]$ (marked by $\overleftarrow{4}$).
6. For column 1 over the MC operation in round 0, deduce $z_0[4-6]$ and $w_0[4]$ (marked by 6).
   (a) Compute backward to get $k_0[4, 9, 14]$ (marked by $\overleftarrow{6}$).
   (b) Compute forward to get $x_1[4]$ (marked by $\overrightarrow{6}$). Then compute forward to get $w_1[4-7]$ (marked by $\overrightarrow{6}$).
7. According to the key relations, deduce $k_1[12]$ (marked by 7). Then compute backward to get $w_0[12]$ (marked by $\overleftarrow{7}$).
8. Guess $x_0[6]$ (marked by 8).
   (a) Compute backward to get $k_0[6]$ (marked by $\overleftarrow{8}$).
   (b) From $k_0[6]$, compute forward to get $k_1[14]$ (marked by $\overrightarrow{8}$). Then compute backward to get $w_0[14]$ (marked by $\overleftarrow{8}$).
9. For column 3 over the MC operation in round 0, since five values are known, there is a conflict with a probability of 2^{-8}. Then deduce $z_0[15]$ and $w_0[13, 15]$ (marked by 9).
   (a) Compute backward to get $k_0[11]$ (marked by $\overleftarrow{9}$).
   (b) Compute forward to get $k_1[13]$ and $x_1[15]$. Then for column 0 over the MC operation in round 1, deduce $w_1[0-3]$ and compute forward to get $k_2[2]$ (marked by $\overrightarrow{9}$).
10. According to key relations, deduce $k_0[5, 10]$ and $k_2[1]$ (marked by 10).
    (a) From $k_0[5, 10]$, compute forward to get $x_0[5, 10]$ (marked by $\overrightarrow{10}$).
    (b) From $k_2[1]$, compute backward to get $k_1[1]$ (marked by $\overleftarrow{10}$). Then from $k_1[1]$ compute backward to get $w_0[1]$ (marked by $\overleftarrow{10}$).

11. For column 0 over the MC operation in round 0, deduce $z_0[3]$ and $w_0[0,2,3]$ (marked by 11).
    (a) Compute backward to get $k_0[15]$ (marked by $\overleftarrow{11}$).
    (b) Compute forward to get $k_1[0,2,3]$ (marked by $\overrightarrow{11}$). Then all states of the starting points are determined.

1.	$k_0[0-3,8,12] = (x_0 \oplus P)[0-3,8,12]$	$w_1[8-11] = \mathrm{MC}(z_1[8-11])$
	$w_1[12-15] = \mathrm{MC}(z_1[12-15])$	$k_2[13] = (x_2 \oplus w_1)[13]$
2.	$k_1[11] = k_0[3] \oplus \mathrm{SB}(k_1[4])$	$w_0[11] = k_1[11] \oplus x_1[11]$
3.	$k_1[10] = \mathrm{SB}(k_1[7]) \oplus k_0[2]$	$w_0[7,10] = (k_1 \oplus x_1)[7,10]$
4.	$w_0[8-11] = \mathrm{MC}(z_0[8-11])$	$k_0[7,13] = (x_0 \oplus P)[7,13]$
	$k_1[8,9] = (w_0 \oplus x_1)[8,9]$	
5.	$\mathrm{SB}(k_1[6]) = k_0[1] \oplus k_1[9]$	$\mathrm{SB}(k_1[5]) = k_0[0] \oplus k_1[8] \oplus const$
	$k_1[15] = k_1[11] \oplus k_0[7]$	$w_0[5,6] = (k_1 \oplus x_1[5,6])$
6.	$w_0[4-7] = \mathrm{MC}(z_0[4-7])$	$k_0[4,9,14] = (x_0 \oplus P)[4,9,14]$
7.	$k_1[12] = k_1[8] \oplus k_0[4]$	$w_1[0] = k_2[0] \oplus x_2[0]$
8.	$k_0[6] = (x_0 \oplus P)[6]$	$k_1[14] = k_0[6] \oplus k_1[10]$
	$w_0[14] = k_1[14] \oplus x_1[14]$	
9.	$w_0[12-15] \stackrel{?}{=} \mathrm{MC}(z_0[12-15])$	$k_0[11] = (x_0 \oplus P)[11]$
	$k_1[13] = w_0[13] \oplus x_1[13]$	$x_1[14] = w_0[14] \oplus k_1[14]$
	$w_1[0-3] = \mathrm{MC}(z_1[0-3])$	$k_2[2] = w_1[2] \oplus x_2[2]$
10.	$k_0[5] = k_1[9] \oplus k_1[13]$	$k_0[10] = k_1[14] \oplus k_2[2]$
	$k_2[1] = k_0[9] \oplus k_1[13]$	$x_0[5,10] = (k_0 \oplus P)[5,10]$
	$k_1[1] = k_0[13] \oplus k_2[1] \oplus k_2[9]$	$w_0[1] = k_1[1] \oplus x_1[1]$
11.	$w_0[0-3] = \mathrm{MC}(z_0[0-3])$	$k_0[15] = (x_0 \oplus P)[15]$
	$k_1[0,2,3] = (w_0 \oplus x_1)[0,2,3]$	

Table 6: Equations in the GD steps for $\mathcal{I}_d^{(1)}$. Waved bytes are guessed. The red equations are conflicts.

Analysis of the conflict byte equation system χ. As stated above, the conflict system has 1 equation, and we can resolve the conflict in advance as follows: In Step 8 of the GD procedure, $x_0[6]$ is guessed, and then leads to a conflict in Step 9. We represent the conflict by the deduced bytes in Step 1 to 7 and $x_0[6]$, which is

$$x_0[6] \oplus \mathsf{SB}(x_0[6]) = 07 \cdot z_0[12] \oplus 04 \cdot z_0[13] \oplus 03 \cdot w_0[12] \oplus x_1[14] \oplus k_1[10] \oplus P[6]. \quad (50)$$

So we build a hash table $H_{x_0[6]}$ before the GD process, which store the value of $x_0[6]$ indexed by the value $x_0[6] \oplus \mathsf{SB}(x_0[6])$. Then in Step 8, we first calculate the value of the right part of Eq. (50), then access H to get right $x_0[6]$. So the conflict in Step 8 is satisfied with probability 1.

Degree of freedom and complexity.

– In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}^{(1)}]$ for active bytes from the input/output differences in the $\mathcal{I}_d^{(1)}$. There are 24 active S-boxes, where $s_1^{(1)} = 19$ S-boxes are of probability 2^{-7}, and $s_2^{(1)} = 4$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}^{(1)}]| = 2^{s_1^{(1)}+2s_2^{(1)}-1} = 2^{26}$ choices for the bytes marked by 1 (except bytes in P) in Fig. 22.
– Given $V[\mathbb{A}]$, one byte $V[\mathbb{G}^{(1)}] = (k_1[7])$ (marked by a wavy line) is guessed in Step 3, another "guessed" byte $x_0[6]$ is actually deduced (without guessing) by the hash table $H_{x_0[6]}$. There is no conflict in the $\mathcal{I}_d^{(1)}$, *i.e.*, $|\chi^{(1)}| = 0$. Hence, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d^{(1)}$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(1)}} = \frac{2^{26+1\times 8}}{2^{8\times 0}} = 2^{34}$.
– 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 $2^{-p_o^{(1)}} = 2^{-54}$, which means that at least $2^t = 2^{20} = 2^{p_o^{(1)}}/\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(1)}}$ distinct values of $P = h_1$ are required to generate 2^{54} starting points to expect one pair satisfying the difference Δh_2.
In summary, we have $s_1^{(1)} = 19$, $s_2^{(1)} = 4$, $\tilde{s}_1^{(1)} = \tilde{s}_2^{(1)} = 0$, $|V[\mathbb{G}^{(1)}]| = 1$, $|\chi^{(1)}| = |\tilde{\chi}^{(1)}| = 0$, $p_o^{(1)} = 54$. According to Eq. (12), the time to find one one pair satisfying the difference Δh_2 is

$$\mathcal{T}_{SFS} = 2^{20} + 2^{54} \approx 2^{54}. \tag{51}$$

F.2 Guess-and-Determine Steps of the Inbound Phase $\mathcal{I}_d^{(2)}$

Run the Eq. (5), with given $V[\mathbb{P}] = h_2$ and $V[\mathbb{A}]$, which are marked by 1 in Fig. 22. The $V[\mathbb{A}] = (x_0[0{-}2, 4, 8{-}15], x_1[2, 4, 5, 7, 8, 10, 12, 14], x_2[0, 8], x_3[8], k_1[4]$ are deduced by accessing the DDT with the fixed differences $\Delta x_0[0-2, 4, 8-15]$, $\Delta y_0[0-2, 4, 8-15]$, $\Delta x_1[2, 4, 5, 7, 8, 10, 12, 14]$, $\Delta y_1[2, 4, 5, 7, 8, 10, 12, 14]]$, $\Delta x_2[0, 8]$, $\Delta y_2[0, 8]$, $\Delta x_3[8]$, $\Delta y_3[8]$, and also the fixed differences in the key schedule. The guess-and-determine steps are given in the following and also summarized in Table 7.

1. Deduce bytes from 1 bytes:
   (a) In round 0, compute backward to get $k_0[0-2, 4, 8-15]$ (marked by $\overleftarrow{1}$).
   (b) In round 1, compute forward to get $w_1[8-11]$(marked by $\overrightarrow{1}$).
   (c) In round 2, compute backward to get $k_2[8]$ (marked by $\overleftarrow{1}$).
2. For column 1 over the MC operation in round 0, deduce $z_0[7]$ and $w_0[5-7]$ (marked by 2).
   (a) Compute backward to get $k_0[3]$ (marked by $\overleftarrow{2}$).
   (b) Compute forward to get $k_1[5, 7]$ (marked by $\overrightarrow{2}$).
3. According to the key relations, deduce $k_1[8, 10, 11]$ (marked by 3).
   (a) From $k_1[8]$, compute forward to get $k_1[12]$ (marked by $\overrightarrow{3}$).
   (b) From $k_1[8, 10]$, compute backward to get $w_0[8, 10]$ (marked by $\overleftarrow{3}$).

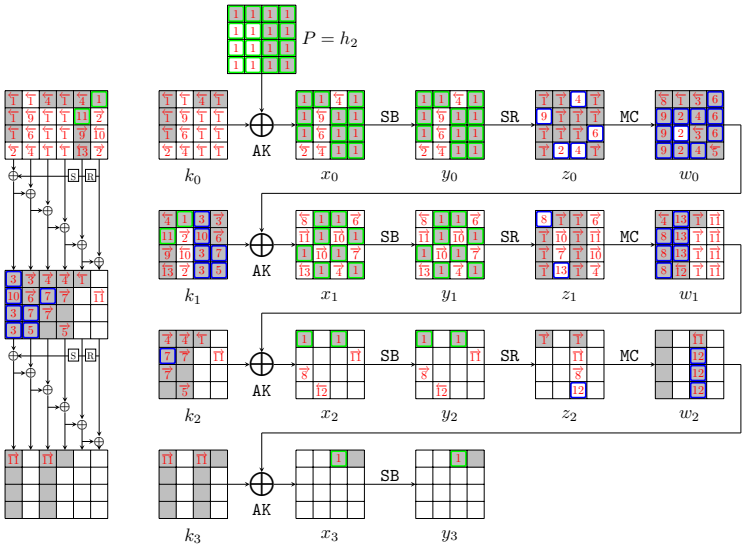


Fig. 23: The GD process of $\mathcal{I}_d^{(2)}$. The bytes in green boxes are the known/guessed bytes at the beginning of each step. The bytes in blue boxes are the deduced bytes at the beginning of each step.

4. For column 2 over the MC operation in round 0, deduce $z_0[8, 11]$ and $w_0[9, 11]$ (marked by 4).
   (a) Compute backward to get $k_0[7, 8]$ (marked by $\overleftarrow{4}$).
   (b) From $k_0[8]$, subsequently $k_2[0, 4]$ (marked by $\overrightarrow{4}$) and $k_1[0]$ (marked by $\overleftarrow{4}$). Then from $k_2[0]$, compute backward to get $w_1[0]$ (marked by $\overleftarrow{4}$).
   (c) Compute forward to get $x_1[11]$ (marked by $\overrightarrow{4}$).
5. According to the key relations, deduce $k_1[15]$ (marked by 5). Then compute forward to get $k_2[3, 7]$ (marked by $\overrightarrow{4}$). Compute backward to get $x_1[11]$ (marked by $\overleftarrow{5}$).
6. For column 3 over the MC operation in round 0, deduce $z_0[14]$ and $w_0[12-14]$ (marked by 6).
   (a) Compute backward to get $k_0[6]$ (marked by $\overleftarrow{6}$).
   (b) Compute forward to get $k_1[13]$ and $x_1[12]$ (marked by $\overrightarrow{6}$).
7. According to the key relations, deduce $k_1[14]$ and $k_2[1]$ (marked by 7).
   (a) From $k_[14]$, compute forward to get $k_2[2]$ (marked by $\overrightarrow{7}$).
   (b) From $k_2[1]$, compute forward to get $k_2[5]$(marked by $\overrightarrow{7}$).
8. For column 0 over the MC operation in round 1, deduce $z_1[1]$ and $w_1[1-3]$ (marked by 8).
   (a) Compute backward to get $w_0[0]$ (marked by $\overleftarrow{8}$).
   (b) Compute froward to get $x_2[2]$ (marked by $\overrightarrow{8}$).
9. For column 0 over the MC operation in round 0, deduce $z_0[1]$ and $w_0[1-3]$ (marked by 9).

(a) Compute backward to get $k_0[5]$ (marked by $\overleftarrow{9}$).
(b) Compute froward to get $k_1[2]$ (marked by $\overrightarrow{9}$).

10. According to key relations, deduce $k_1[9]$ (marked by 10).
(a) Compute backward to get $k_1[6]$ (marked by $\overleftarrow{10}$).
(b) From $k_1[9]$ and $k_1[6]$, compute forward to get $x_1[6,9]$ (marked by $\overrightarrow{10}$).
11. Guess $k_1[1]$ (marked by 11). Using key relations, subsequently compute forward to get $k_2[13]$ and $k_3[0,8]$ (marked by $\overrightarrow{11}$).
(a) In round 1, compute forward to get $x_1[1]$, then compute forward to get $w_1[12-15]$ (marked by $\overrightarrow{11}$).
(b) In round 2, compute forward to get $x_2[13]$ (marked by $\overrightarrow{11}$).
(c) From $k_3[8]$, compute backward to get $w_2[8]$ (marked by $\overleftarrow{11}$).
12. For column 2 over the MC operation in round 2, deduce $z_2[11]$ and $w_2[9-11]$ (marked by 12). Then compute backward to get $w_1[7]$ (marked by $\overleftarrow{12}$).
13. For column 1 over the MC operation in round 1, deduce $z_1[7]$ and $w_1[4-6]$ (marked by 13). Then compute backward to get $k_1[3]$ (marked by $\overleftarrow{13}$). Then all states of the starting points are determined.

1.	$k_0[0-2,4,8-15] = (x_0 \oplus P)[0-2,4,8-15]$	
	$k_2[8] = (x_2 \oplus w_1)[8]$	$w_0[4] = k_1[4] \oplus x_1[4]$
2.	$w_0[4-7] = \mathrm{MC}(z_0[4-7])$	$k_0[3] = (x_0 \oplus P)[3]$
	$k_1[5,7] = (x_1 \oplus w_0)[5,7]$	
3.	$k_1[8,10,11] = \mathrm{SB}(k_1[5,7,4]) \oplus k_0[0,2,3] \oplus \{const,0,0\}$	
	$k_1[12] = k_1[8] \oplus k_0[4]$	$w_0[8,10] = k_1[8,10] \oplus x_1[8,10]$
4.	$w_0[8-11] = \mathrm{MC}(z_0[8-11])$	$k_0[7,8] = (x_0 \oplus P)[7,8]$
	$k_2[0,4] = \{k_1[12], k_2[0]\} \oplus k_0[8,12]$	$k_1[0] = k_2[4] \oplus k_2[8]$
	$w_1[0] = k_2[0] \oplus x_2[0]$	$x_1[11] = w_0[11] \oplus k_1[11]$
5.	$k_1[15] = k_1[11] \oplus k_0[7]$	$k_2[7] = k_0[11] \oplus k_0[15] \oplus k_1[15]$
	$w_0[15] = k_1[15] \oplus x_1[15]$	
6.	$w_0[12-15] = \mathrm{MC}(z_0[12-15])$	$k_0[6] = (x_0 \oplus P)[6]$
	$k_1[13] = w_0[13] \oplus x_1[13]$	$x_1[12] = k_1[12] \oplus w_0[12]$
7.	$k_1[14] = k_1[10] \oplus k_0[6]$	$k_2[1] = k_1[13] \oplus k_0[9]$
	$k_2[2] = k_1[14] \oplus k_0[10]$	$k_2[5] = k_2[1] \oplus k_0[13]$
8.	$w_1[0-3] = \mathrm{MC}(z_1[0-3])$	$w_0[0] = k_1[0] \oplus x_1[0]$
	$x_2[2] = w_1[2] \oplus k_2[2]$	
9.	$w_0[0-3] = \mathrm{MC}(z_0[0-3])$	$k_0[5] = (x_0 \oplus P)[5]$
	$k_1[2] = w_0[2] \oplus x_1[2]$	
10.	$k_1[9] = k_1[13] \oplus k_0[5]$	$\mathrm{SB}(k_1[6]) = k_1[9] \oplus k_0[1]$
	$x_1[6,9] = k_1[6,9] \oplus w_0[6,9]$	
11.	$x_1[1] = k_1[1] \oplus w_0[1]$	$w_1[12-15] = \mathrm{MC}(z_1[12-15])$
	$k_2[13] = k_1[1] \oplus k_1[5] \oplus k_2[5]$	$x_2[13] = k_2[12] \oplus w_1[13]$
	$k_3[0] = \mathrm{SB}(k_2[13]) \oplus k_1[8] \oplus const$	$k_3[8] = k_1[12] \oplus k_2[0] \oplus k_3[0]$
	$w_2[8] = k_3[8] \oplus x_3[8]$	
12.	$w_2[8-11] = \mathrm{MC}(z_2[8-11])$	$w_1[7] = x_2[7] \oplus k_2[7]$
13	$w_1[4-7] = \mathrm{MC}(z_1[4-7])$	$k_1[3] = x_1[3] \oplus w_0[3]$

Table 7: Equations in the GD steps for $\mathcal{I}_d^{(2)}$. Waved bytes are guessed.

Degree of freedom and complexity.

- In Step 1 of the above procedures, we deduce the values $V[\mathbb{A}^{(2)}]$ for active bytes from the input/output differences in the $\mathcal{I}_d^{(1)}$. There are 23 active S-boxes, where $s_1^{(2)} = 19$ S-boxes are of probability 2^{-7}, and $s_2^{(2)} = 4$ S-boxes are of probability 2^{-6}. Therefore, there are $|V[\mathbb{A}^{(2)}]| = 2^{s_1^{(2)}+2s_2^{(2)}-1} = 2^{26}$ choices for the bytes marked by 1 (except bytes in P) in Fig. 23.
- Given $V[\mathbb{A}]$, one byte $V[\mathbb{G}^{(2)}] = (k_1[1])$ (marked by a wavy line) is guessed in Step 11. Since there is no conflict in the $\mathcal{I}_d^{(2)}$, *i.e.*, $|\chi^{(2)}| = 0$, according to Eq. (10), the degrees of freedom of the $\mathcal{I}_d^{(2)}$ are $\widehat{\mathrm{DoF}}_{\mathcal{I}_d^{(2)}} = \frac{2^{26+1\times 8}}{2^{8\times 0}} = 2^{34}$.
- 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 $2^{-p_o^{(2)}} = 2^{-34}$, a single value of $P = h_2$ can generate 2^{34} starting points, which is sufficient to expect one collision in h_3.

F.3 Classic Collision Attack on the 7-Round AES-192-DM

Based on $\mathcal{D}$ shown in Fig. 20 and Fig. 21, we can construct a 3-block collision attack. According to the attack procedure in Sect. 3.3 and the complexity analysis above, we have

$$s_1^{(2)} = 19,\ s_2^{(2)} = 4,\ \tilde{s}_1^{(2)} = \tilde{s}_2^{(2)} = 0,\ |V[\mathbb{G}^{(2)}]| = 1,\ |\chi^{(2)}| = |\tilde{\chi}^{(2)}| = 0,\ p_o^{(2)} = 34,$$
$$s_1^{(1)} = 19,\ s_2^{(1)} = 4,\ \tilde{s}_1^{(1)} = \tilde{s}_2^{(1)} = 0,\ |V[\mathbb{G}^{(1)}]| = 1,\ |\chi^{(1)}| = |\tilde{\chi}^{(1)}| = 0,\ p_o^{(1)} = 54,$$

Since $2^{s_1^{(2)}+2s_2^{(2)}-1-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} \times 2^{8\cdot|\mathbb{G}^{(2)}|} \geq 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)}$, according to Eq. (20) and (51), the expected time complexity of the collision attack on 7-Round AES-192-DM becomes

$$\mathcal{T}_{SFS} \cdot 2^{8|\tilde{\chi}^{(2)}|-(\tilde{s}_1^{(2)}+2\tilde{s}_2^{(2)})} + 2^{p_o^{(2)}} \cdot 2^{8(|\chi^{(2)}|-|\tilde{\chi}^{(2)}|)} \approx 2^{54}. \tag{52}$$

G Experiment results

This section provides the details of the experimental results reported in Section 6. For 4-round AES-128-DM, we present a near-collision, with the message pair shown in Tab. 8 and the detailed procedure illustrated in Fig. 24. For 7-round AES-192-DM, we present a collision, with the message pair shown in Tab. 9 and the details given in Fig. 25 and 26.

H Guess-and-Determine results on Baek et al.'s trail[2]

We searched for a differential trail, shown in Fig. 27, which follows the chosen-key differential trail pattern described in [2]. Following the methodology of [2], we fix a total of 25 active S-boxes in the first four rounds, as well as 5 active S-boxes in the key schedule. We then employ a guess-and-determine approach to analyze this trail. The results are summarized as follows.

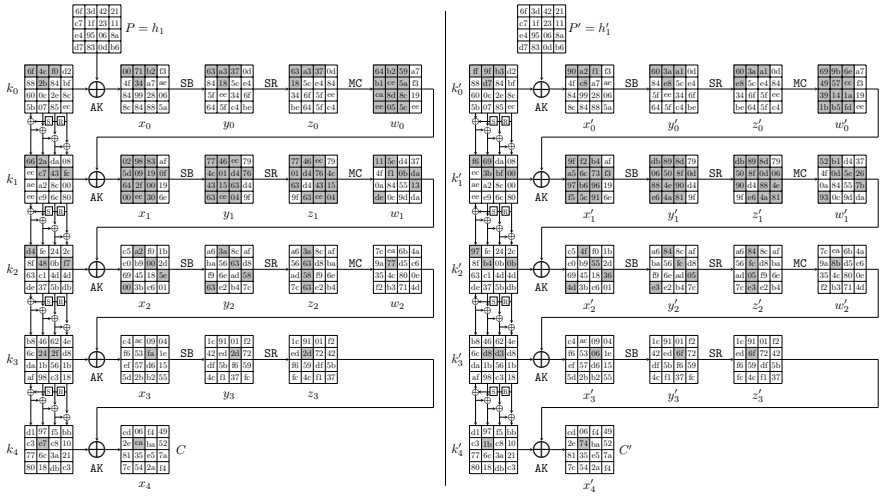


Fig. 24: Details of 4-round AES-128-DM near-Collision on Block 1, with h_2 and h'_2 differing only at $x_4[5]$.

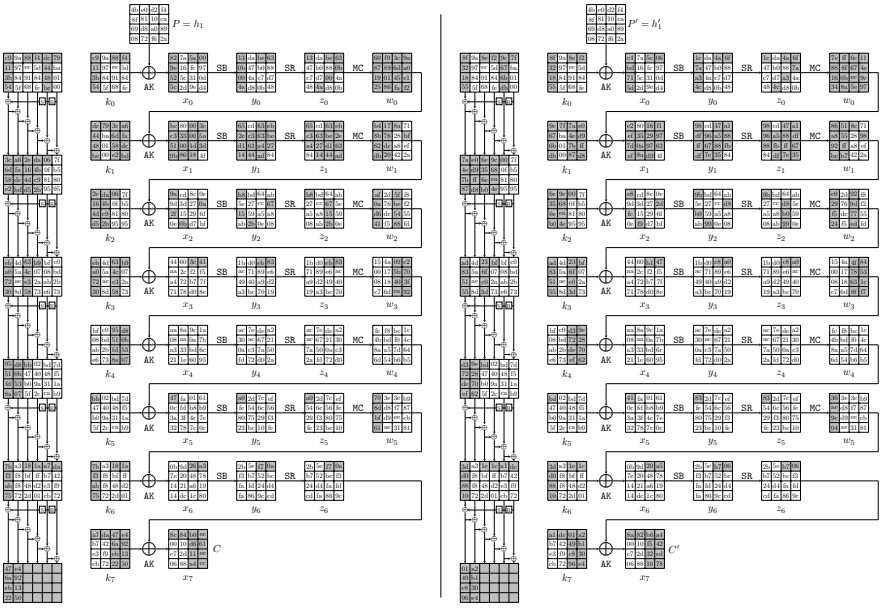


Fig. 25: Details of 7-round AES-192-DM full-collision on Block 1.

M_0	h_1	M_1/M_1'	h_2/h_2'
72 8a fe 86	6f 3d 42 21	6f 4c f0 d2	a2 3b b6 68
42 7e f4 ce	c7 1f 23 11	88 2b 84 bf	e9 d5 99 43
10 25 ec d3	e4 95 06 8a	60 0c 2e 8c	65 a0 e3 f0
b1 61 0b eb	d7 83 0d b6	5b 07 85 ec	ab d7 27 42
72 8a fe 86	6f 3d 42 21	ff 9f b3 d2	a2 3b b6 68
42 7e f4 ce	c7 1f 23 11	88 d7 84 bf	e9 6b 99 43
10 25 ec d3	e4 95 06 8a	60 0c 2e 8c	65 a0 e3 f0
b1 61 0b eb	d7 83 0d b6	5b 07 85 ec	ab d7 27 42

Table 8: Message pairs leading to a 4-round AES-128-DM near-collision($IV = 0$)

M_0	h_1	M_1/M_1'	h_2/h_2'	M_2/M_2'	h_3/h_3'
66 9b 36 b7 3d ba	4b e0 d2 f4	c9 9a 88 f4 dc 79	c7 64 62 1a	71 6a 3c ec 5b d3	4c 73 42 ab
4f ed 56 4f f3 97	8f 81 10 ca	11 97 ec 5d 44 ba	8f 91 c6 ab	ff 40 9f 25 91 dc	3f e0 1d a8
41 39 bb fe 92 b2	69 d8 a0 89	3b 84 91 84 48 01	ae f5 b1 67	91 b4 03 67 93 76	99 ef c8 67
7f b8 63 16 32 6f	08 72 f6 2a	54 5f 68 fe be 00	0e fa 52 e6	24 83 cc e6 31 13	e0 39 7b 08
66 9b 36 b7 3d ba	4b e0 d2 f4	8f 9a 8e f2 9c 7f	c1 62 64 50	37 6a 3a ea 1b d5	4c 73 42 ab
4f ed 56 4f f3 97	8f 81 10 ca	32 97 ec 5d 67 ba	8f 91 e5 88	dc 40 9f 25 b2 dc	3f e0 1d a8
41 39 bb fe 92 b2	69 d8 a0 89	18 84 91 84 6b 01	ae f5 92 44	b2 b4 03 67 b0 76	99 ef c8 67
7f b8 63 16 32 6f	08 72 f6 2a	55 5f 68 fe db 00	0e fa e6 52	24 83 cc e6 54 13	e0 39 7b 08

Table 9: Message pairs leading to a 7-round AES-192-DM full-collision($IV = 0$)

1. **Fixed P.** According to the two-block collision model in [2], a valid pair (h_1, h_1') as (P, P') can be obtained with complexity $2^{44.52}$, after which the value of P is fixed. Therefore, the analysis with fixed P is consistent with the collision model in [2]. Using the guess-and-determine approach, we find that this trail contains 6 free variables and 4 conflicts. Moreover, among the already fixed variables, there exists one additional conflict, as shown in Eq. (53).

$$\begin{cases} k_2[0] = x_0[0] \oplus P[0] \\ \qquad \oplus \mathrm{SB}(03 \cdot \mathrm{SB}(x_0[12]) \oplus \mathrm{SB}(x_0[1]) \oplus \mathrm{SB}(x_0[6]) \\ \qquad\quad \oplus 02 \cdot \mathrm{SB}(x_0[11]) \oplus x_1[15]) \oplus const \\ k_2[0] = x_2[0] \oplus 02 \cdot \mathrm{SB}(x_1[0]) \\ \qquad \oplus 03 \cdot \mathrm{SB}(x_1[5]) \oplus \mathrm{SB}(x_1[10]) \oplus \mathrm{SB}(x_1[15]). \end{cases} \tag{53}$$

2. **Unfixed P.** When P is not fixed, the analysis follows the attack model in [9]. The guess-and-determine procedure shows that this trail has 18 free variables and no conflicts. This number of free variables is consistent with the results reported in [2] and [9].

According to the two-block collision model in [2], the analysis should be conducted with fixed P. Based on the guess-and-determine results, the number of degrees of freedom for this trail is $2^{6 \cdot 8} = 2^{48}$, while the probability of the trail

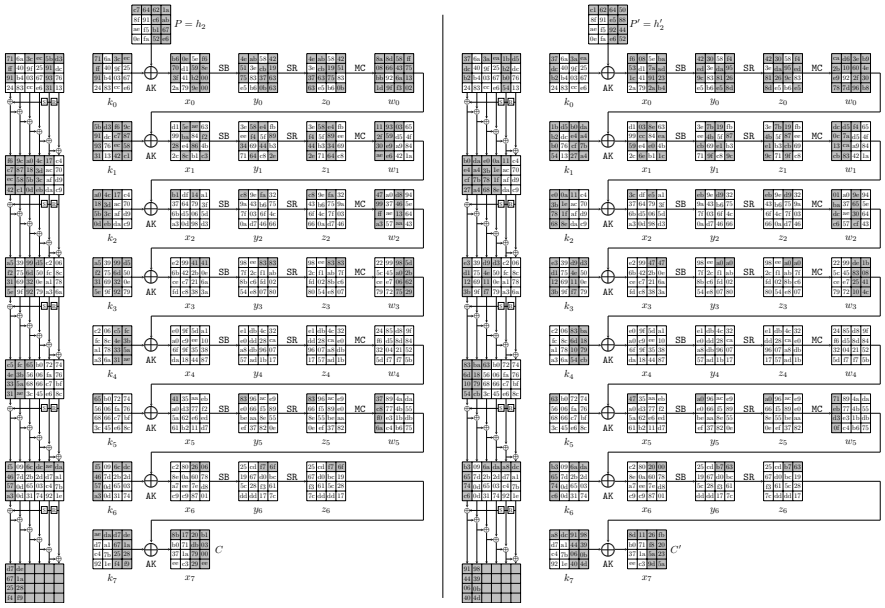


Fig. 26: Details of 7-round AES-192-DM full-collision on Block 2.

is $2^{-5\cdot 8-93} = 2^{-133}$. Therefore, the available degrees of freedom are insufficient to cover the trail probability, implying that a collision cannot be obtained using only a single P.

The verification code can be found in:

https://github.com/DSwutang/AES-DM-collision/BCK21

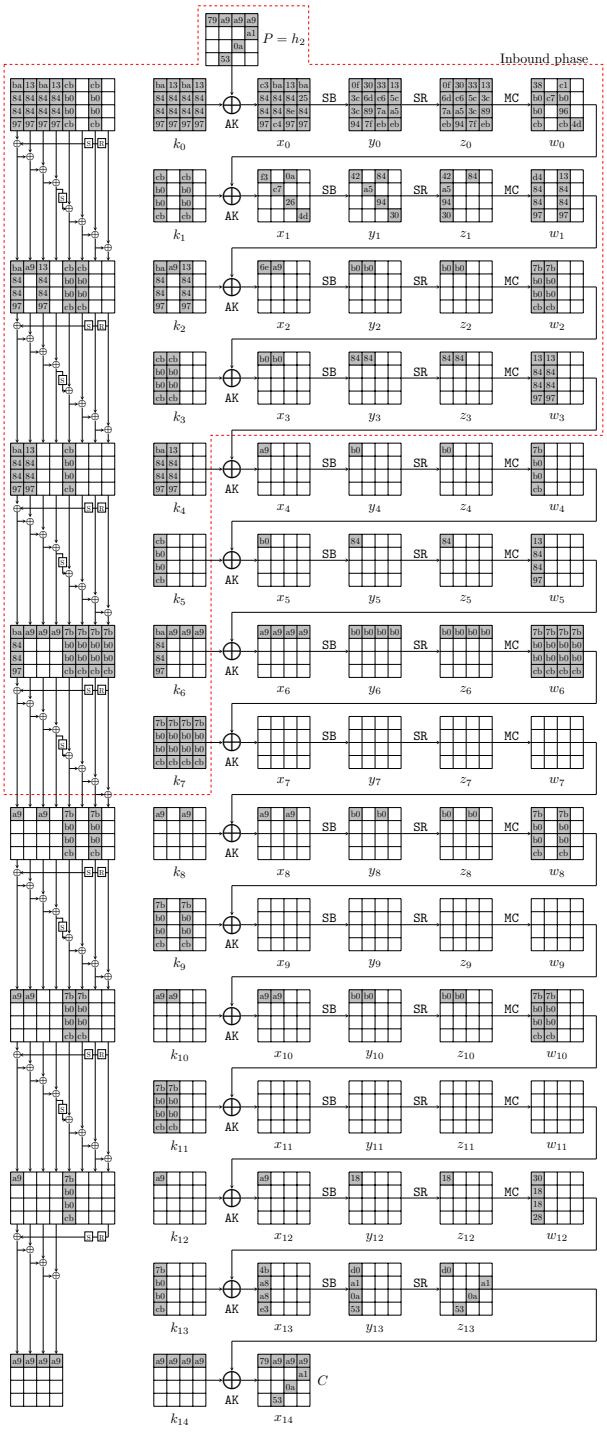


Fig. 27: A chosen-key differential trail satisfying the pattern described in [2].