

Revisiting Single-Color Initial Structures in Meet-In-The-Middle Attacks

Shiyao Chen^{1,2}, Jian Guo³, Wenjie Nan³, Danping Shi^{4,5,6}, and
Tianyu Zhang^{3(✉)}

¹ School of Cyber Science and Technology, Shandong University, Qingdao, China
`shiyao.chen@sdu.edu.cn`

² State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China

³ Nanyang Technological University, Singapore
`guojian@ntu.edu.sg`, `{wenjie006,tianyu005}@e.ntu.edu.sg`

⁴ State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
`shidanping@iie.ac.cn`

⁵ State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, China

⁶ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. The meet-in-the-middle (MITM) attack framework is one of the most powerful cryptanalytic techniques with broad influence to preimage, key-recovery, and collision attacks. In this paper, we present two generic techniques. First, we observe that the constant space in MITM attacks is an exploitable source of degrees of freedom: its value-independence enables MITM-style partition and acceleration of attack subprocesses. With this intuition in mind, we revisit the single-color initial structure technique by Chen et al. at Asiacrypt 2025, and find an improved algorithm that relaxes the canonical optimization objective in automatic search for MITM attacks. Second, we extend the partial-target-preimage-to-collision conversion by Li, Isobe, and Shibutani at FSE 2012 used in MITM-based collision attacks to the settings of chosen-prefix collision and diamond structure construction. We demonstrate the practical relevance of our techniques by presenting a number of improved results in (pseudo-)preimage, (chosen-prefix) collision, and herding attacks on AES-like constructions over the state-of-the-art.

Keywords: Meet-in-the-Middle · Single-Color Initial Structure · Preimage · Chosen-Prefix Collision · Diamond Structure · Herding

1 Introduction

The meet-in-the-middle (MITM) attack is among the most influential techniques in cryptanalysis. It traces back to Diffie and Hellman’s seminal attack on Double-DES [22]. At Asiacrypt 2008, Sasaki and Aoki first formalized MITM attacks in preimage attacks on hash functions [51]. Their pioneering work established

MITM as a central framework for hash function cryptanalysis and was recognized by the IACR Test-of-Time Award⁷. The area developed rapidly through a sequence of key ideas, including splice-and-cut [2], initial structures [3,52], indirect and partial matching [1], probabilistic initial structures [33], bicliques [40], and precomputations in the middle [13].

Recursive MITM solver. At Crypto 2011, Bouillaguet, Derbez, and Fouque proposed a dedicated search tool (BDF11) for recursive MITM and guess-and-determine (GnD) key-recovery attacks on AES [10]. The solver models AES as an equation system, searches for an arbitrary-depth merging tree of MITM subroutines, and constructs it bottom-up from trivial single-variable enumerators. This makes the solver highly effective for constrained nonlinear equation systems. For example, Derbez and Fouque used the solver as a subroutine in their analysis of Demirci–Selçuk MITM attacks on AES, but only on the key-schedule equations [21]. Dinur and Jean later also adopted the solver in their attack on FIDES [23], and reported that the solver required many fine-tuning trials with artificially added constraints to return solutions of practical use. Thus, BDF11 is best used as a generic solver for small subsystems of non-linear equations. For large systems with many variables and long propagation paths, successful application of the BDF11 tool typically require manually or heuristically restricting the equation system beforehand.

Optimization-based MITM solvers. At Eurocrypt 2021, Bao *et al.* built the first MILP-based search tool for MITM attacks on AES-like constructions [5]. The tool has been enriched with a number of techniques, including the superposition structure and guess-and-determine [6]; precomputation [27]; linearization, distributed initial structures, and structural similarity [15]; triangulation [59]; and single-color initial structure (SCIS) [16]. This automatic tool was also generalized to sponge constructions [48,29,49] and feistel networks [34]. A parallel line of work, led by [53], aims at simplifying the modeling of MITM attacks on permutations and easing conversion to quantum attacks; this was extended to block ciphers and block-cipher-based hash functions with permutation-based key schedules [54], and later to ones with more complex key schedules [20]. These solvers follow a similar template: they first encode a set of *propagation rules* for two independent computation chunks (often termed as forward and backward) and *matching rules* for two-list merging into an optimization language (often MILP), then invoke an optimizer to find the optimal instantiation top-down. The optimization goal of these solvers is often modeled in the structure of $\min(d_b, d_r, d_m)$, where d_b and d_r denote the degrees of freedom (DoF) of the forward and backward chunks and d_m denotes the filtering power of the matching relation.

⁷ <https://www.iacr.org/testoftime/>

Single-color initial structure (SCIS). In MITM attacks, increasing the number of attacked rounds often raised the complexity of neutral word generation *i.e.*, computing the neutral words that comply with a chosen set of constants. The generic enumeration-and-sort method, termed precomputation [27], incur substantial time and memory costs outside the main MITM procedure, which becomes a bottleneck when converting MITM attacks to collision attacks. Introduced by Chen *et al.* at Asiacrypt 2025, the SCIS technique [16] identifies a class of MITM trails with efficient neutral word generation: constants are distributed asymmetrically so that part of the constant-induced constraints are solved on the fly. A detailed review is deferred to Section 2.2. Using SCIS, Chen *et al.* presented the first 7-round classical collision attacks on AES-MMO/MP mode, demonstrating the advantage of MITM-based methods for collision attacks on some targets when compared to differential-based or rebound-based attacks.

Chosen-prefix collision attack (CPC). The CPC attack was introduced as a stronger variant of collision attacks [31,11]: given any two prefixes P_0, P_1 chosen by the adversary, a CPC attack finds suffixes S_0, S_1 such that $H(P_0 \| S_0) = H(P_1 \| S_1)$. The setting is strictly harder than identical-prefix collision yet carries far greater practical impact, as evidenced by the seminal attacks on MD5 by Stevens *et al.* [55,57,56] and on SHA-1 by Leurent and Peyrin [42,43]. Most recently, Chen *et al.* proposed the first CPC attacks on AES-like hashing at FSE 2025 [14]. As far as we know, all existing dedicated attacks can be reduced to finding a suitable differential trail, and no alternative approach has been proposed.

Diamond structure (DS) and herding attack. The herding attack (with a short suffix) was highlighted by Kelsey and Kohno at Eurocrypt 2006 [38]. The adversary first commits to a hash digest h_T , and upon being later challenged with a prefix P , produce a short suffix S such that $H(P \| S) = h_T$. A herding attack proceeds in two phases: an offline phase that builds a DS, a binary tree of 2^k chaining values (CVs) that converge to a single root used as the commitment, and an online phase that links the challenge prefix into one of the DS leaves. As such, the suffix is about $k + 1$ blocks in length, which can be practical in many situations, in contrast to long message second preimage attacks [38, Sect. 1.4]. For an n -bit hash function, the generic bound for herding attacks is approximately $O(2^{2n/3})$ [38,8]. The first dedicated herding attack was given by Zhang *et al.* at FSE 2024 on AES-like hashing [58]. The idea is to use a multi-target MITM attack to accelerate the online phase of finding a link message. At FSE 2024, Dong *et al.* improved this approach by using partial-target MITM attacks [26]. A notable gap is that existing dedicated herding attacks did not attempt to accelerate the offline phase.

1.1 Our Contributions

In this work, we present two contributions to the generic MITM attack framework, aimed at refining the algorithmic skeleton of the attack when applicable as well as expanding the influence of MITM to more cryptanalytic constructions.

Improved SCIS MITM attack framework. In [16], the constraint-light chunk is denoted as the red chunk, and the red-side computation is a plain enumerate-and-filter step, the accumulative cost of which across different choices of constants reaches 2^{n-d_b} . This cost is precisely the d_b term in the canonical balancing condition $\min(d_b, d_r, d_m)$ [9,51]. In this work, we observe that the constant space in MITM attacks has been an overlooked but exploitable source of DoF: it can be partitioned into subspaces that act as neutral words in MITM attacks. Following the intuition, we replace the red-side computation and filtering in [16] with a constant-as-neutral-word MITM subprocess, store the partial-candidates, then on the fly compute and merge the blue neutral words. This reduces the complexity of the red-side computation below the 2^{n-d_b} bound, thus relaxes the canonical balancing condition to roughly $\min(d_r, d_m)$ whenever the constant space supplies sufficient degrees of freedom. To our knowledge, this is the first relaxation of the canonical balancing rule since its formalization in [9,51], which expands the total search space of MITM attacks and leads to more efficient attacks.

MITM-based CPC and Herding Attacks. From the literature review, we see that existing CPC attacks rely on differential-based frameworks, which carry two intrinsic limitations. First, Kelsey and Kohno noted that differential-based CPC does not compose well with DS construction, since *one must fix the positions of the nodes within the DS in advance* [38, Sect. 2]. Second, modern design rationales such as the wide-trail strategy offer natural resistance to differential-based methods. These limitations partially explain why differential-based CPC attacks on AES-MMO reach only 5 rounds [14], why no CPC attack is known on Whirlpool, and why dedicated herding attacks still rely on generic methods for the offline diamond construction [58,26]. In this work, we show that MITM-based methods can be leveraged to mount CPC and herding attacks, and in contrast to differential-based methods, naturally compose with DS constructions to yield meaningful speedup. Specifically, we extend the generic conversion technique first proposed by Li, Isobe, and Shibutani [44], which reduces collision finding to repeated queries to a partial-target preimage (PTP) attack oracle, in two directions: from a single CV to two chosen CVs, yielding CPC attacks; and to perfect matching across the CVs of a diamond layer, yielding efficient diamond constructions and thus herding attacks. The extended conversion thereby introduces MITM attacks into finding CPC attacks, accelerating DS construction, and constructing double-phase accelerated herding attacks.

Main Results. We apply the two techniques mentioned above to AES-like hash functions (see [Section A](#) for a brief introduction), and obtain a series of improvements over the state-of-the-art as summarized in [Table 1](#).

First, the improved SCIS MITM attack framework yields improved classical preimage and collision attacks compared with SOTA: (1) the first 8-round collision attack on Kiasu-BC hashing at time complexity 2^{56} , overcoming the neutral-word generation bottleneck identified at Crypto 2022 [6]; (2) the first 8-round pseudo-preimage result on Saturnin-hash at time complexity 2^{224} (preimage attack with time complexity 2^{240} after conversion [4]), improving by one round over the best classical result at FSE 2023 [54]; (3) the first 7-round classical collision result on Saturnin-hash at 2^{120} , improving by two rounds over the classical rebound attack at Asiacrypt 2021, and matching the best attack round in quantum [28].

Second, the extended LIS12 conversion generically converts MITM-based collision attacks to CPC attacks, yielding: (1) the first 7-round classical CPC attack on AES-MMO, improving by two rounds over the best classical attack, and by one round over the best quantum attack [14]; (2) the first classical CPC attack on 6-round Whirlpool, filling a gap left open in the classical setting, and matching the best quantum attacked rounds [14]; (3) the first 7-round classical CPC result on Saturnin-hash at 2^{120} .

Third, the conversion applies to the acceleration of the offline phase (DS construction) in herding attacks, together with the online acceleration technique proposed in [58], we obtain double-phase accelerated herding attacks on Saturnin-hash, AES-MMO, and Whirlpool.

Organization. Our paper is organized as follows. Preliminaries on MITM attacks are given in [Section 2](#). Next, a detailed narrative of our algorithmic refinements and the extended LIS12 conversion can be found in [Section 3](#) and [Section 4](#). The main results obtained are presented in [Section 5](#) and [Section 6](#). Finally, [Section 7](#) concludes the paper.

2 Preliminaries on MITM Attacks

This section is intended as a review of the MITM attack framework. We will introduce the overall MITM structure in [Section 2.1](#), relevant techniques in the literature on MITM-based collision attacks and the single-color initial structure.

2.1 High-level Description of MITM Attacks

As depicted in [Figure 1](#), an MITM attack divides the computation of a primitive into two computation-independent chunks, which structurally meet at an intermediate state. The two chunks are denoted as forward and backward chunks, or equivalently *blue* and *red* chunks for abbreviation and visualization. The term *neutral word* for a chunk represents a value whose computation in the chunk has a constant impact to the other chunk. An equivalent expression would be: when

Table 1. Summary of the results on preimage, collision, and herding attacks.

Kiasu-BC							
Target	Attack	Rounds	Time	Memory	Setting	Technique	Reference
Hashing Mode	Collision	6/10	2^{56}	2^{32}	Classic	Super S-box	[41,32]
		6/10	2^{48}	2^{32}	Classic	Rebound	[25]
		7/10	2^{60}	2^{60}	Classic	SCIS	[16]
		8/10	2^{56}	2^{56}	Classic	Improved SCIS	Section 5.1
Saturnin-hash							
Target	Attack	Rounds	Time	Memory	Setting	Technique	Reference
Hash function	Preimage	7/16	$2^{208} (2^{232})$	2^{48}	Classic	MITM	[27]
		7/16	$2^{192} (2^{224})$	2^{160}	Classic	MITM	[54]
		7/16	$2^{115.55}$	$2^{32 \dagger}$	Quantum	MITM	[54]
		8/16	$2^{224} (2^{240})^{\S}$	2^{176}	Classic	Improved SCIS	Section 5.2
	Collision	5/16	2^{64}	2^{66}	Classic	RK Rebound*	[28]
		7/16	$2^{113.5}$	—	Quantum	RK Rebound	[28]
		7/16	2^{120}	2^{120}	Classic	Improved SCIS	Section B.1
	CPC	6/16	2^{112}	2^{64}	Classic	RK Rebound	[14]
		7/16	$2^{118.7}$	—	Quantum	RK Rebound	[14]
		7/16	2^{120}	2^{120}	Classic	SCIS, Conversion	Section B.2
	Herding	7/16	2^{160}	2^{160}	Classic	SCIS, Conversion	Section B.2
		Any	$2^{172.4}$	$2^{172.4}$	Classic	—	[38,8]
AES-MMO/MP							
Target	Attack	Rounds	Time	Memory	Setting	Technique	Reference
Hash function	CPC	5/10	2^{57}	2^{32}	Classic	RK Rebound	[14]
		6/10	$2^{61.5}$	—	Quantum	RK Rebound	[14]
		7/10	2^{60}	2^{60}	Classic	SCIS, Conversion	Section 6.1
	Herding	6/10	$2^{82.7}$	$2^{82.2}$	Classic	MITM	[58]
		6/10	2^{77}	2^{76}	Classic	MITM	[26]
		7/10	2^{83}	2^{82}	Classic	MITM	[26]
		7/10	2^{80}	2^{80}	Classic	SCIS, Conversion	Section 6.1
	Any	$2^{88.1}$	$2^{87.8}$	Classic	—	[38,8]	
Whirlpool							
Target	Attack	Rounds	Time	Memory	Setting	Technique	Reference
Hash function	CPC	6/10	$2^{205.4}$	—	Quantum	RK Rebound	[14]
		6/10	2^{240}	2^{240}	Classic	SCIS, Conversion	Section 6.2
	Herding	6/10	2^{334}	2^{333}	Classic	MITM	[26]
		6/10	2^{320}	2^{320}	Classic	SCIS, Conversion	Section 6.2
		Any	$2^{344.7}$	$2^{344.2}$	Classic	—	[38,8]

[†] means quantum-accessible quantum memory (QRAQM).

[§] The preimage attack on the compression function has the time complexity 2^{224} , which can be converted into the preimage attack on Saturnin-hash with 2^{240} .

* short for related-key rebound.

[‡] short for triangulating rebound.

some constant values are chosen during the attack, a *neutral word* must comply with these constants or satisfy the constraints brought by the constants during propagation.

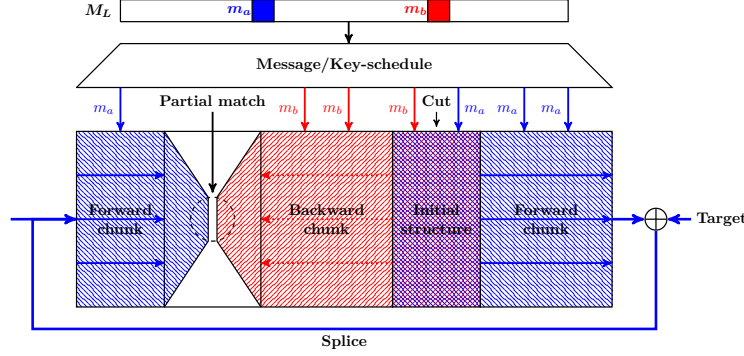


Fig. 1. A high-level overview of MITM attacks [50].

A conventional MITM attack is orchestrated as follows: To start with, a set of constants will be chosen. The neutral words complying with the constants values will be computed. The step is referred to as *neutral word generation* (for trivial cases this step is skipped). Then, the neutral words are plugged into the chunks and computed in parallel to two intermediate states E^+ and E^- , which are connected with some matching relation. A matching step is conducted to use the relation to filter the neutral words. Without loss of generality, we assume $d_b \leq d_r$, an algorithmic description can be found in [Algorithm 1](#). The computational complexity of the above attack is evaluated as follows:

$$2^{n-(d_b+d_r)} \cdot (2^{d_b} + 2^{d_r} + 2^{d_b+d_r-d_m}) \simeq 2^{n-\min(d_b, d_r, d_m)}. \quad (1)$$

2.2 SCIS MITM Attacks

SCIS [16] denotes a structural property of MITM trails that facilitates efficient neutral word generation. It assigns constants, thus constant-induced constraints, asymmetrically to the two chunks. For simplicity and without loss of generality, the constraint-heavy chunk is referred as the *blue* chunk and spans only a few rounds, the neutral word generation of which is easier to be handled by established tools like the BDF11 solver [10]. The constraint-light chunk is referred as the *red* chunk, which is kept relatively free of constants, propagates across more rounds from both sides to the matching point. As a result, only the red values are used for matching.

For formalization, [16] classifies the constants into enumerating constants *ec* and indexing constants *ic*. By definition, *ec* are those constants for which blue

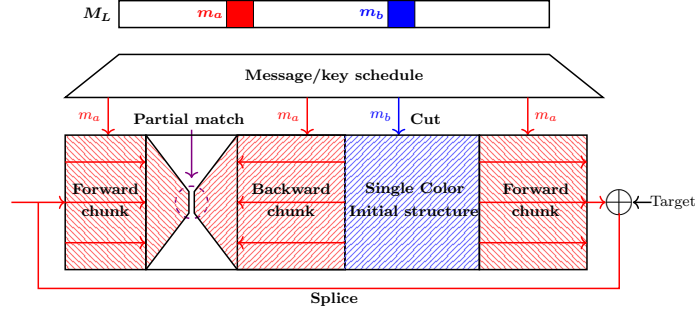


Fig. 2. The SCIS structure [16]

neutral words can be generated with amortized cost $O(1)$ for any fixed value of them. All remaining constants in the blue chunk, *i.e.*, the ones that cannot be handled efficiently, are grouped into ic . For each fixed ec , the attacker enumerates all blue neutral words complying with ec , stores them in a table W_{blue} indexed by the value at ic , and then enumerates red neutral words r and applies a d_m -bit partial-match filter at the matching point. The algorithm is given in [Algorithm 2](#).

SCIS MITM attacks still follow the typical MITM framework, and does not change the rule-of-thumb balancing condition $\min(d_b, d_r, d_m)$. Instead, it provides concrete guidance on how to control the complexity of neutral word generation rather than plain enumerate-and-sort, thereby enabling a range of MITM-based collision attacks on meaningful rounds that outperform differential-based techniques.

Algorithm 1: Canonical MITM framework

```

1 while True do
2   Assign arbitrary values to the constants in the predefined constraints;
3   Initialize an empty table  $W$ ;
4   Generate blue neutral words based on the constants;
5   Generate red neutral words based on the constants;
6   foreach blue neutral value  $b$  do
7     Compute forward to  $E^+$  and obtain value  $m^+$  as partial-match index;
8      $W[m^+] \leftarrow b$ ;
9   foreach red neutral value  $r$  do
10    Compute backward to  $E^-$  and obtain  $m^-$  as the partial-match index;
11    foreach  $b \in W[m^-]$  do
12      Get matched candidate  $(b, r)$ ;
13      if a preimage / correct key / collision pair is found then
14        return  $(b, r)$ ;

```

Algorithm 2: Core SCIS MITM function $f(g_{ec})$

```

// Assume we perform the single-color match on red neutral words
1 Generate blue neutral words that comply with  $g_{ec}$  with amortized cost  $O(1)$ ,
   store in  $T_{blue}$ , indexed by its value at  $g_{ic}$ ;
2 for  $g_{ic}$  in  $2^{d_{g_{ic}}}$  values do
3   for  $r$  in  $2^{d_r}$  red neutral words do
4     Compute to the matching point;
5     if pass the partial-match filter with probability  $2^{-d_m}$  then
6       for  $b$  in  $2^{d_b}$  entries of  $T_{blue}[g_{ic}]$  do
7         Get candidate  $(b, r)$ ;
8         Check if a preimage/correct key/collision pair has been found;

```

2.3 Chosen-Prefix Collisions

A chosen-prefix collision (CPC) attack finds suffixes S_0 and S_1 for any two arbitrarily chosen prefixes P_0 and P_1 such that $H(P_0||S_0) = H(P_1||S_1)$. The CPC problem is at least as hard as collision search. Moreover, a CPC attack is *downward compatible* with a collision attack, since a collision instance reduces to CPC by setting $P_0 = P_1 = IV$. The generic complexity of a CPC attack is $2^{n/2+1}$, matching the birthday bound up to a constant factor.

The canonical CPC attack framework, introduced by Stevens, Lenstra and de Weger [55] and used in subsequent related works, follows a two-phase differential-based approach: the first phase finds a pair of message blocks with output difference δ in a structured set $\mathcal{S}$ via a birthday attack; the second phase utilizes $\mathcal{S}$ and applies a dedicated differential-based attack to produce a collision, instantiated as the message modification technique on MD5 and SHA-1 [55,42] or as related-key rebound attacks on AES-like hashing [14].

2.4 Herding Attacks

At Eurocrypt 2006, Kelsey and Kohno proposed the *herding attack* [38], which works in a commitment setting: given a hash function H with a fixed initial value IV , the adversary first commits to a target hash value h_T . After later receiving an arbitrary message prefix P of predetermined length from the challenger, the adversary appends a short suffix S such that $H(P||S) = h_T$.

As illustrated in Figure 3, the generic herding attack proceeds in two phases. The offline phase builds a DS with 2^k leaves $\{x_i\}_{i \in [1, 2^k]}$, such that $h_T = H(x_i||M_i)$ for any i . The attacker then announces h_T and receives P . During the online phase, it finds a linking message M_{link} that connects $H(P)$ to a leaf x_j , thus obtaining $S = M_{\text{link}}||M_j$.

In the classical setting, the DS construction costs about $O(\sqrt{k} \cdot 2^{(n+k)/2})$ [38,8], and the online search costs 2^{n-k} . Balancing the two phases yields a total complexity of $O(2^{2n/3})$ approximately, achieved at $k \approx n/3$.

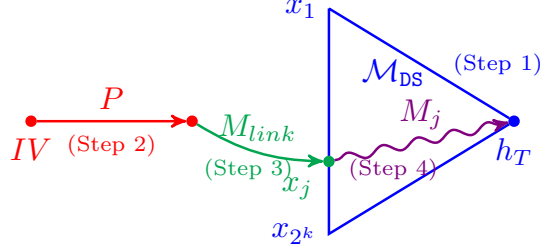


Fig. 3. Herding Attack on MD Constructions

Dedicated Herding Attacks. So far, the speedup of dedicated herding attacks [58,26] over generic bound stems from using an MITM attack to accelerate the online phase (step 3 in Figure 3). The DS construction during offline phase is kept generic. Thus, if an MITM attack finds a linking message with advantage τ over generic, the overall complexity is $\sqrt{k} \cdot 2^{(n+k)/2} + 2^{n-k-\tau}$, which is approximately minimized to $2^{(2n-\tau)/3}$ at $k \approx 2^{(n-2\tau)/3}$.

3 Improved SCIS MITM Attack Framework

In this section, we propose an improved SCIS MITM attack algorithm. We explain the core idea in Section 3.1 and formalize it in Section 3.2, and derive complexity formulas for preimage, key-recovery, and collision attacks in Section 3.3. Throughout this paper, we will follow the notation in [16] (reviewed in Section 2.2), use blue and red to denote the constraint-heavy and constraint-light chunks.

3.1 Accelerating the Red-Side Computation

Recall from Section 2.2 that under a fixed enumerating constant value ec , the SCIS core function enumerates over the $2^{d_{ic}}$ values of indexing constants. For each value of ic , it enumerates 2^{d_r} red neutral words, computes from both sides to the matching point, and keeps only the ones that pass the d_m -bit partial-match filter. This is a plain enumerate-and-filter step that costs 2^{d_r} to obtain $2^{d_r-d_m}$ matched red candidates. In (pseudo-)preimage attacks, a total of $2^{n-d_b-d_r}$ combinations of (ec, ic) must be tested, so the red-side computation accumulates to

$$2^{n-d_b-d_r} \cdot 2^{d_r} = 2^{n-d_b},$$

which corresponds to the d_b term in the canonical MITM balancing condition $\min(d_b, d_r, d_m)$ of Equation (1).

In this work, we propose a non-trivial optimization of the red-side computation by utilizing the DoF of the constant space and accelerating it with an MITM procedure. Constants in MITM attacks are conventionally assigned arbitrary values before the parallel computation and matching (as in Algorithm 3). Their role

is to restrict the neutral words during generation and to enforce computational independence between the forward and backward chunks. The enabling observation of this work is that the choice of constant values is itself a source of DoF, as in standard MITM setups these values can be assigned independently⁸. On the SCIS red-side closed loop, the constants on the two paths can therefore be partitioned into two independently iterable subspaces that play the role of forward and backward neutral words, and matched on the d_m -bit partial-matching filter. As an effect, the red-side computation is replaced by a constant-as-neutral-word MITM process over the two constant subspaces, and the accumulated red-side computational cost can drop below the bound of 2^{n-d_b} , thus provides a relaxation of the canonical balancing condition. We formalize this idea and detail in the following subsection.

3.2 Description of New Attack Framework

We partition the constant space into three subsets: enumerating constants ec , forward constants fc , and backward constants bc . The enumerating constants are chosen heuristically similar to [16]: with any predetermined value of ec , there exists a neutral word generation algorithm for the blue (constant-dense) chunk that solves blue neutral words with amortized cost $O(1)$. The role of ec is to push blue neutral word generation on the fly as far as possible, so as to minimize memory complexity and computational overhead. The forward and backward constants fc and bc are the remaining constants, positioned on the red-chunk computation paths from both sides to the matching point. They constitute the two constant subspaces for the constant-as-neutral-word MITM. We refer the reader to example trails used in our attacks (*e.g.*, Figure 4) for an illustration of how the constants are categorized.

Under a fixed ec , the attack framework consists of two stages. The first stage is a constant-as-neutral-word MITM that computes and filters the red side. Red values take the typical constant role of outer enumeration, and the two constant subspaces fc and bc take the neutral-word roles of forward and backward enumeration. The output is a table of *partial candidates*, denoted W , indexed by the constant values (fc, bc) and storing the matched red values under each index. The second stage solves the blue neutral words and merges them with W to form full candidates for testing. Unlike the SCIS procedure in Algorithm 2 of [16], we do not materialize a blue neutral word table; instead, blue solutions are merged with W on the fly to optimize the memory complexity. We adopt the core MITM function format of [16] for easier adaptation to preimage, key-recovery, and collision attacks. An algorithmic description is given in Algorithm 3.

Stage 1: Constant-as-neutral-word MITM for red-side computation.

The first stage computes and filters the red values. For each fixed r , the forward

⁸ There exist cases where constants exhibit tractable dependencies due to primitive structure, *e.g.*, the targeted cases of the structural similarity technique in [15]. In this paper, we focus on cases where constant values can be assigned independently.

Algorithm 3: MITM core function $f(ec)$

```
1 Initialize an empty table  $W$ ;  
   // Stage 1: Constant-as-neutral-word MITM for partial-match  
2 foreach  $r$  in  $2^{d_r}$  red values do  
   // Forward computation over  $fc$   
3   Initialize an empty temporary table  $U$ ;  
4   foreach  $fc$  in  $2^{d_{fc}}$  values do  
5     Compute value for partial-match  $m^+$ ;  
6      $U[m^+] \leftarrow fc$ ;  
   // Backward computation over  $bc$   
7   foreach  $bc$  in  $2^{d_{bc}}$  values do  
8     Compute value for partial-match  $m^-$ ;  
9     foreach  $fc$  in  $U[m^-]$  do  
10     $W[(fc, bc)] \leftarrow r$ ;  
  
   // Stage 2: Solve blue neutral words and join with partial  
   candidates  
11 Generate blue neutral words that comply with the predetermined  $ec$ ;  
12 foreach  $b$  in  $2^{d_b+d_{fc}+d_{bc}}$  solutions computed on the fly do  
13   Get forward and backward constant values it corresponds to  $(fc, bc)$ ;  
14   foreach  $r$  in  $W[(fc, bc)]$  do  
15     // Form candidate and verify  
16     Get candidate  $(ec, fc, bc, b, r)$ ;  
     Check if a preimage/correct key/collision pair has been found;
```

and backward enumerations cost $2^{d_{fc}}$ and $2^{d_{bc}}$, respectively, and the expected number of joined pairs that pass the d_m -bit filter is $2^{d_{fc}+d_{bc}-d_m}$. The expected time complexity of Stage 1 is therefore

$$T_1 = 2^{d_r} \cdot (2^{d_{fc}} + 2^{d_{bc}} + 2^{d_{fc}+d_{bc}-d_m}). \quad (2)$$

The table W stores all partial candidates indexed by (fc, bc) and dominates the memory cost of this stage; its size equals the expected number of matched triples (fc, bc, r) , namely $2^{d_r+d_{fc}+d_{bc}-d_m}$. Under each of the $2^{d_{fc}+d_{bc}}$ indices of W , the expected number of red values is $2^{d_r-d_m}$. The temporary table U has size $2^{\min(d_{fc}, d_{bc})}$ and is reused across different r values.

Stage 2: Solve blue neutral words and join with partial candidates on the fly. The second stage turns partial candidates into candidates by providing the missing blue part. We solve for the blue neutral word values that comply with ec . This can be done either by a triangulation process as in [24] or by the procedure of [16], leveraging the BDF11 tool of [10]. For each blue solution b , we derive on the fly its corresponding (fc, bc) values, look up $W[(fc, bc)]$ to retrieve

the matched red values r , and form each *candidate* (ec, fc, bc, b, r) for full match checking.

The total blue solution space complying with a predetermined ec has size $2^{d_b+d_{fc}+d_{bc}}$, uniformly distributed over (fc, bc) indices, so each value of (fc, bc) expects 2^{d_b} blue solutions. Hence Stage 2 has time complexity

$$T_2 = 2^{d_{fc}+d_{bc}} \cdot (2^{d_b} + 2^{d_b+d_r-d_m}) \quad (3)$$

and produces on the fly

$$N_f = 2^{d_{fc}+d_{bc}+d_b+d_r-d_m} \quad (4)$$

candidates to be checked for a full match.

Complexity of the core function. Combining Equations (2) and (3), the time complexity of one call of the core function is

$$T_f = 2^{d_r} \cdot (2^{d_{fc}} + 2^{d_{bc}} + 2^{d_{fc}+d_{bc}-d_m}) + 2^{d_{fc}+d_{bc}} \cdot (2^{d_b} + 2^{d_b+d_r-d_m}), \quad (5)$$

and the memory complexity is dominated by the size of W .

Relaxation of the canonical balancing condition. The core balancing condition has changed from the canonical $\min(d_b, d_r, d_m)$ to $\min(d_b + d_{bc}, d_b + d_{fc}, d_r, d_m)$. In typical cases, since the constant space contributes enough degrees of freedom, the condition collapses to $\min(d_r, d_m)$. This is, to our knowledge, the first relaxation of the canonical $\min(d_b, d_r, d_m)$ rule since its formalization in [9,51].

3.3 Applications to Preimage, Key-Recovery, and Collision Attacks

We now derive the time and memory complexity of the framework for preimage, key-recovery, and collision attacks.

Preimage and key-recovery attacks. For preimage and key-recovery attacks, the candidate test in Algorithm 3 is typically one call of the primitive, and one success is expected after collecting 2^{n-d_m} candidates satisfying the d_m -bit partial-match condition. Since one call of the core function produces N_f candidates as in Equation (4), we need about $2^{n-d_m}/N_f$ calls to obtain one success. The total time complexity is therefore

$$\frac{2^{n-d_m}}{N_f} \cdot T_f = 2^{n-d_{fc}-d_{bc}-d_b-d_r} \cdot T_f \quad (6)$$

$$= 2^{n-d_b-d_{bc}} + 2^{n-d_b-d_{fc}} + 2^{n-d_b-d_m} + 2^{n-d_r} + 2^{n-d_m} \quad (7)$$

$$\approx 2^{n-\min(d_b+d_{bc}, d_b+d_{fc}, d_r, d_m)}. \quad (8)$$

As the checking is done on the fly, the memory complexity is dominated by the size of W , which is $2^{d_r+d_{fc}+d_{bc}-d_m}$.

Collision-type attacks. For collision attacks, the full check additionally maintains a hash table W_{dp} storing distinguished points for collision detection. For each candidate (ec, fc, bc, b, r) produced, the attacker reconstructs the corresponding input/output pair (x, h) and queries $W_{dp}[h]$. If the bucket is non-empty, a collision is found; otherwise, x is stored under index h .

One call of the core function produces N_f candidates sharing the same d_m -bit partial-match constraint on the red side. Since one collision is expected after collecting $2^{(n-d_m)/2}$ such candidates, we need about $2^{(n-d_m)/2}/N_f$ iterations. The total time complexity is

$$\frac{2^{(n-d_m)/2}}{N_f} \cdot T_f = 2^{(n+d_m)/2-d_{fc}-d_{bc}-d_b-d_r} \cdot T_f \quad (9)$$

$$\approx 2^{(n+d_m)/2-\min(d_b+d_{bc}, d_b+d_{fc}, d_r, d_m)}. \quad (10)$$

The memory complexity is dominated by the size of W_{dp} , which is $2^{(n-d_m)/2}$.

4 Extending MITM to CPC and Herding Attacks

At FSE 2012, Li, Isobe, and Shibutani [44] showed that a specific type of MITM preimage attacks can be converted into collision attacks. In this section, we extend their generic conversion into CPC attacks (Section 4.2), DS constructions and herding attacks (Section 4.3).

4.1 The LIS12 Conversion

Let $H : \{0, 1\}^n \times \{0, 1\}^\ell \rightarrow \{0, 1\}^n$ be a compression function iterated under the Merkle-Damgård paradigm with initial value IV . The LIS12 conversion [44] reduces collision finding to repeated queries to an efficient *partial-target preimage (PTP) generator*, defined as follows.

Definition 1 (PTP generator). Let $\mu \in \{0, 1\}^n$ be a bit mask with $\text{wt}(\mu) = t$, which denotes the Hamming weight of μ , and let

$$\Pi_\mu : \{0, 1\}^n \rightarrow \{0, 1\}^t$$

denote the projection that extracts the t bits of its input at the positions selected by μ . Fix a target $d \in \{0, 1\}^t$. A t -bit partial-target preimage generator $\mathcal{A}_{\mu,d}$ is a randomized algorithm that, on input a chaining value $CV \in \{0, 1\}^n$, outputs a message block $M \in \{0, 1\}^\ell$ satisfying

$$\Pi_\mu(H(CV, M)) = d.$$

Repeated invocations on the same CV return distinct messages M . Conditioned on the fixed constraint $\Pi_\mu(H(CV, M)) = d$, the remaining unconstrained part of the output, namely the $n-t$ bits outside the mask μ , is modeled as an independent uniform element of $\{0, 1\}^{n-t}$ across invocations and across the chaining values used below. The amortized per-call time complexity of $\mathcal{A}_{\mu,d}$ is 2^s .

Lemma 1 ([44]). *Fix a mask μ and target $d \in \{0, 1\}^t$, and invoke $\mathcal{A}_{\mu,d}(IV)$ a total of $2^{(n-t)/2}$ times to obtain messages $M_1, \dots, M_{2^{(n-t)/2}}$. Under the heuristic randomness assumption of $n - t$ unconstrained output bits, the birthday bound gives a colliding pair $H(IV, M_i) = H(IV, M_j)$ with constant probability. The total time complexity is $2^{s+(n-t)/2}$, which beats the generic bound $2^{n/2}$ if and only if $s < t/2$. We thus note a PTP generator $\mathcal{A}_{\mu,d}$ is efficient if and only if $s < t/2$.*

4.2 Extension to CPC Attacks

In the LIS12 conversion [Section 4.1](#), the PTP generator $\mathcal{A}_{\mu,d}(\cdot)$ is invoked under the single initial value IV . We extend this conversion to the CPC setting by invoking the generator on two distinct chaining values, one induced by each chosen prefix, and then searching for a cross-list collision between the two sets of outputs.

Let $P_0, P_1 \in \{0, 1\}^*$ be two distinct chosen prefixes and $CV_b = H(IV, P_b)$ the chaining values they induce, for $b \in \{0, 1\}$. Fix a mask μ with $\text{wt}(\mu) = t$ and a target $d \in \{0, 1\}^t$.

Attack procedure. A cross-list collision search under two distinct chosen prefixes can be performed as follows.

1. Query $\mathcal{A}_{\mu,d}(CV_0)$ a total of L times to obtain L pairs $(M_0, H(CV_0, M_0))$, store them in $\mathcal{L}_0$; build $\mathcal{L}_1$ analogously from $\mathcal{A}_{\mu,d}(CV_1)$.
2. Find a cross-list collision $H(CV_0, M_0) = H(CV_1, M_1)$ between $\mathcal{L}_0$ and $\mathcal{L}_1$. Each such collision yields a CPC with prefixes P_0, P_1 and suffixes M_0, M_1 .

Following the same birthday-bound argument in the unconstrained domain of size 2^{n-t} , taking $L \approx 2^{(n-t)/2}/2$ samples in each list gives a cross-list collision with constant probability. The total time complexity is then

$$T = 2L \cdot 2^s \approx 2^{s+(n-t)/2}, \quad (11)$$

which beats the generic bound $2^{n/2}$ if and only if $s < t/2$.

Lemma 2 (CPC attack complexity based on PTP generators). *Let H be an n -bit narrow-pipe iterated hash function, and let $\mathcal{A}_{\mu,d}$ be a t -bit PTP generator for its compression function with amortized per-call complexity 2^s . Under the heuristic randomness assumption of unconstrained $n - t$ output bits, a CPC attack on H can be mounted with time complexity*

$$T \approx 2^{s+(n-t)/2}, \quad (12)$$

which beats the generic bound $2^{n/2}$ if and only if $s < t/2$.

4.3 Extension to DS Construction and Herding Attacks

Limitations of differential-based DS construction. Kelsey and Kohno [38] discussed accelerating the offline diamond construction via differential-based CPC attacks. Such attacks require fixed input differences, forcing the diamond to be built pairwise at a cost close to 2^k independent 2-collisions, each of cost 2^w for a differential CPC. Compared with the generic diamond construction with cost $\simeq \sqrt{k} \cdot 2^{(n+k)/2}$, this gives an improvement only when $w < (n - k)/2$. Since the generic herding attack requires $k \approx n/3$, this amounts to the condition $w < n/3$.

In this paper, we find that PTP generators avoid this pairwise restriction. The attack procedure of Section 4.2 fixes only an output-side target (μ, d) and imposes no constraint on the input chaining value. Hence the same target (μ, d) can be reused for all nodes in a diamond layer, with the same amortized per-output cost. The methodology for CPC attacks in Section 4.2 therefore naturally extends to this multi-prefix setting. Rather than solving 2^k CPC instances independently by a differential-based method, we generate samples from all 2^k chaining values in a layer and use cross-list collisions to build an auxiliary graph, in which a perfect matching gives all pairwise merges for the next layer at once.

Layer transition. Fix a mask μ with $\text{wt}(\mu) = t$ and a target $d \in \{0, 1\}^t$. Suppose layer j contains 2^j chaining values $CV_1, \dots, CV_{2^j}$, each inherited from a distinct prefix path. We construct layer $j - 1$ as follows.

1. For each $i \in [1, 2^j]$, query $\mathcal{A}_{\mu, d}(CV_i)$ a total of L_j times to build a list

$$\mathcal{L}_i = \{ (M_{i,j}, H(CV_i, M_{i,j})) : j \in [1, L_j] \}.$$

2. Concatenate all $\mathcal{L}_i$ and sort by H -value; for each pair (i, j) with $i \neq j$ that shares an H -value, record an edge in an auxiliary graph $\mathcal{G}$ on vertex set $\{CV_1, \dots, CV_{2^j}\}$.
3. Find a perfect matching $\mathcal{M}$ in $\mathcal{G}$. Each edge $(CV_i, CV_j) \in \mathcal{M}$ yields a colliding pair $H(CV_i, M_i) = H(CV_j, M_j)$, which contributes one node to layer $j - 1$.

Per-layer cost. Conditioned on the t -bit filter $\Pi_\mu(\cdot) = d$, the $n - t$ unconstrained output bits are uniform by Definition 1, so each list is drawn from an effective space of size 2^{n-t} . Following the perfect matching analysis of Blackburn, Stinson, and Upadhyay [8, Theorem 3], applied here in the filtered space of size 2^{n-t} with 2^k lists, the matching exists asymptotically almost surely when each list has size

$$L_j \approx \sqrt{j \ln 2} \cdot 2^{(n-t-j)/2} \approx 0.83 \sqrt{j} \cdot 2^{(n-t-j)/2}. \quad (13)$$

The cost of constructing layer $j - 1$ from layer j is therefore

$$2^s \cdot L_j \cdot 2^j \approx 0.83 \cdot 2^s \sqrt{j} \cdot 2^{(n-t+j)/2}.$$

Total offline cost. Summing over all k layers and using Equation (13), the j -th layer costs $2^s L_j 2^j \approx 2^s \sqrt{j \ln 2} \cdot 2^{(n-t+j)/2}$. More precisely,

$$\sqrt{\ln 2} \sum_{j=1}^k \sqrt{j} 2^{j/2} \approx \sqrt{\ln 2} (2 + \sqrt{2}) \sqrt{k} 2^{k/2},$$

where $\log_2(\sqrt{\ln 2} (2 + \sqrt{2})) \approx 1.51$. Hence the total offline cost is

$$2^s \sum_{j=1}^k L_j \cdot 2^j \leq 2^s \sqrt{k} \cdot 2^{(n-t+k)/2+1.51}. \quad (14)$$

Compared with the generic diamond construction cost $\sqrt{k} \cdot 2^{(n+k)/2+1.51}$ [8, Theorem 1], this is a speedup whenever $s < t/2$, matching the LIS12 condition.

Online linking. The online phase links the challenge prefix to one leaf of the diamond at cost $2^{n-k-\tau}$, where τ is the advantage of a PTP attack on H over the generic preimage bound (briefly introduced in Section 2.4). Combining Equation (14) with the online cost and optimizing over k yields the following.

Lemma 3 (Herding complexity based on PTP generators). *Let H be an n -bit narrow-pipe iterated hash function. Given a t -bit PTP generator for its compression function with amortized per-call complexity 2^s , and a partial-target preimage attack on H with advantage τ , a dedicated herding attack can be mounted with complexity*

$$\min_k 2^s \sqrt{k} \cdot 2^{(n-t+k)/2+1.51} + 2^{n-k-\tau} \approx \tilde{O}\left(2^{2s/3} \cdot 2^{(2n-t-\tau)/3}\right), \quad (15)$$

achieved at $k \simeq (n + t - 2\tau - 2s)/3$ after suppressing polynomial and constant factors. When $2^s = O(1)$, this simplifies to $2^{(2n-t-\tau)/3}$.

Remark 1. The proposed MITM-based CPC attack can also be applied to settings where differential-based attacks are applicable, such as multi-collisions [37], expandable messages [39], etc.

5 Applications to Preimage and Collision Attacks

To demonstrate the improved SCIS MITM attack framework presented in Section 3, in this section, we apply it to AES-like hashing, including the first 8-round classical collision attack on Kiasu-BC hashing, which succeeds in extending one more attacked round when compared to the result at Crypto 2022 [6], and the first 8-round classical (pseudo-)preimage results on Saturnin-hash, which improves one more round over the previous best classical and quantum results (both with 7 rounds) at FSE 2023 [54]. In the following, we will introduce Kiasu-BC and Saturnin-hash, respectively, followed by attack details on these targets. For more details on AES-like hashing, we refer the readers to Section A.

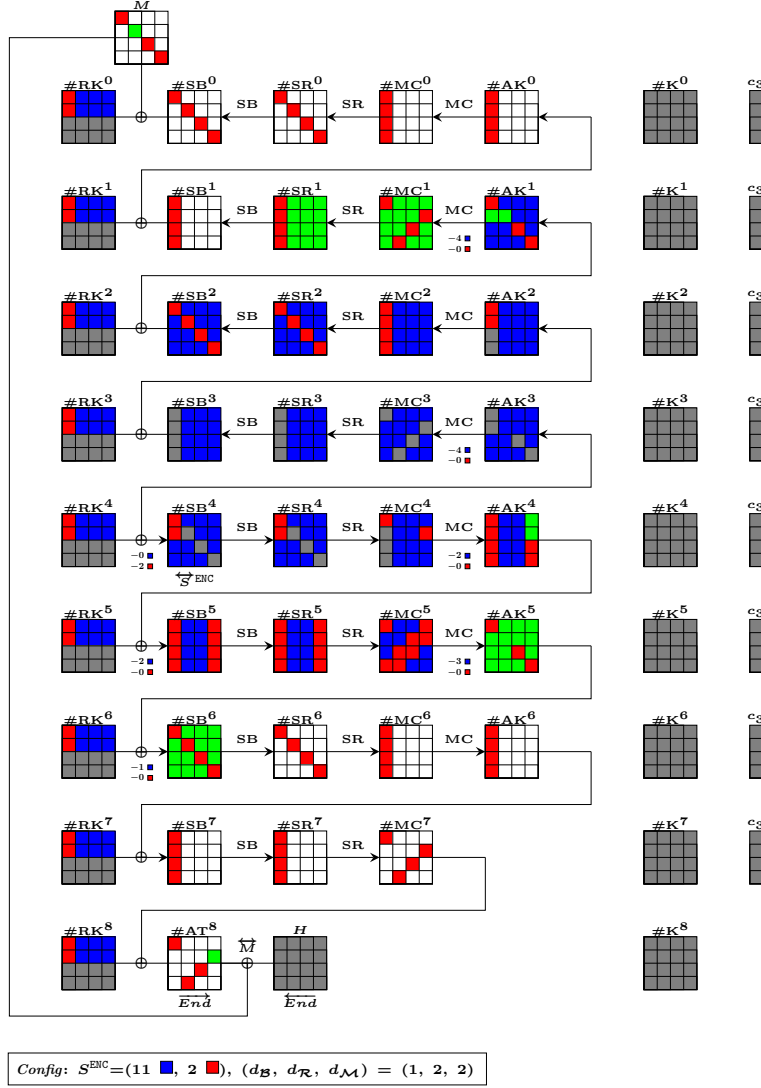


Fig. 4. 8-round collision attack on Kiasu-BC hashing.

5.1 8-round Collision Attack on Kiasu-BC Hashing

The tweakable block cipher Kiasu-BC strictly follows AES-128, which directly adopts the round function, key schedule, and number of rounds of AES-128. The only difference is that Kiasu-BC takes an additional 64-bit tweak added to the topmost two rows of the AES state after every round. This makes it identical to AES-128 when the tweak is set to all zero, which makes it an interesting primitive, as it can reuse existing components of AES implementations. All existing and

new results on AES-128 can usually be directly applied to Kiasu-BC. However, the additional tweak input introduces more freedom in attacks, *e.g.*, the MITM attack scenarios, which makes a comprehensive cryptanalysis of Kiasu-BC under different settings necessary to determine possible negative effects of such extra degree of freedom.

As shown in Figure 4, the first 8-round collision attack on Kiasu-BC hashing is identified. The master key can be fixed to any constant (*e.g.*, the initial value). For convenience, we denote the constants as below:

- Enumerating constants *ec*: the 3 cells $\#SB^4[5, 10, 15]$ in the starting point, 11 cells $\#AK^4[14, 15]$, $\#SB^5[12, 13]$, $\#AK^5[0, 10, 15]$, and $\#MC^3[0, 7, 10, 13]$ for the costs of degree of freedom in the blue neutral word propagation;
- Forward constants *fc*: 1 cell in $\#SB^6[5]$;
- Backward constants *bc*: 4 cells in $\#MC^1[0, 7, 10, 13]$.

Then, following the constant-as-neutral-word MITM framework, we can derive the attack process in Algorithm 4, where for each *ec* constant, one can obtain one red candidate on average for each *fc*, *bc* in Stage 1 and obtain 2^{48} candidates after the constant-as-neutral-word matching by (*fc*, *bc*). Thus, after looping 2^8 values of the enumerating constants *ec*, it can form 2^{48+8} candidates and expect a 112-bit collision on the output target (the other 16 bits are matched in Stage 1). Finally, the time and memory complexity of this 8-round collision attack on Kiasu-BC are both 2^{56} .

It should be noted that one needs an efficient method to generate the blue neutral word space on the fly in Stage 2 of Algorithm 4. We thus provide details on how to generate the blue candidates on the fly in Algorithm 5, which is depicted in Figure 5.

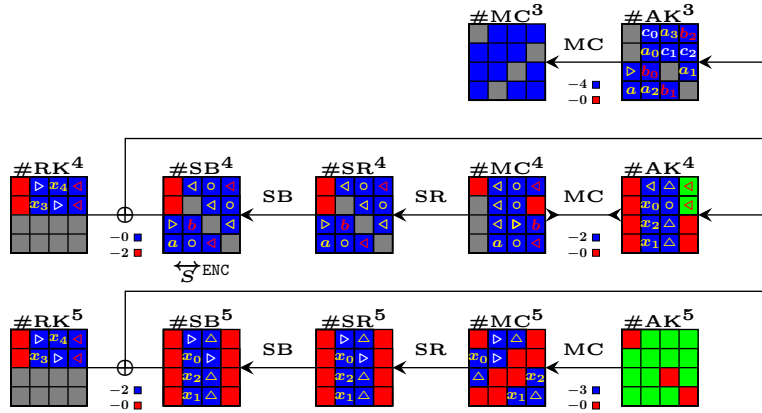


Fig. 5. Blue neutral word generation of 8-round collision on Kiasu-BC hashing.

Algorithm 4: MITM collision attack on 8-round on Kiasu-BC

```

1 foreach  $ec$  in  $2^8$  values do
2   Initialize an empty table  $W$ ;
   // Stage 1:  $2^{16-16} = 2^0$  red candidate for each  $fc, bc$ 
3   foreach  $r$  in  $2^{16}$  red values do
4     // Forward computation over  $fc$ 
     Initialize an empty temporary table  $U$ ;
5     foreach  $fc$  in  $2^8$  values do
6       Compute value for partial-match  $m^+$ ;
7        $U[m^+] \leftarrow fc$ ;
     // Backward computation over  $bc$ 
8     foreach  $bc$  in  $2^{32}$  values do
9       Compute value for partial-match  $m^-$ ;
10      foreach  $fc$  in  $U[m^-]$  do
11         $W[(fc, bc)] \leftarrow r$ ;
    // Stage 2:  $2^{48}$  blue candidates for each  $ec$ 
12  foreach  $b$  in blue space  $B$  generated from Algorithm 5 with  $ec$  do
13    Compute from  $b$  to the constant values of  $(fc, bc)$ ;
14    foreach  $r$  in  $W[(fc, bc)]$  do
15      Get the candidate  $(ec, fc, bc, b, r)$ ;
16      Check if a collision pair has been found;

```

Verification. We performed experiments on this 8-round PTP attack of Kiasu-BC hashing to verify the correctness of the above attack procedures. For practical tests, we fixed the master key to all zero bytes and randomly fixed all enumerating constants ec , backward constants bc , and 1 blue neutral byte $\#SB^5[6]$. This corresponds to the time complexity of $2^{48-8} = 2^{40}$ (1 fixed blue neutral byte), which finally expects about $2^{40+16-32-16} = 2^8$ candidates (2 free red neutral bytes and 4 fixed bc constants) with 2-byte partial matching on the output. The codes and results are available via <https://github.com/csy1234/kiasu-mitm>, which verified the complexity.

5.2 8-round Preimage Attack on Saturnin-Hash

Saturnin is a suite of lightweight symmetric algorithms proposed by Canteaut *et al.* [12] to provide a high security level even in the presence of quantum computers, which runs as the 2nd round candidates of the NIST LWC. Its underlying hash function Saturnin-hash is based on a 256-bit AES-like block cipher with the 256-bit key, which adopts MMO mode. The round function of Saturnin-hash consists of AK, SB layers and uses linear layers MR (even rounds) and MC (odd rounds) alternately. Its key schedule is linear: the master key K is used in even rounds, which rotates by 5 cells (each cell has 16 bits) to get $\tilde{K}$ used in odd rounds.

Algorithm 5: Blue neutral word generation for Kiasu-BC hashing

```

// Enumerating 5 free bytes in outer loop
1 foreach  $(x_0, \dots, x_4)$  in  $\#MC^5, \#RK^5$  of  $2^{40}$  values do
    // Enumerate 1 free byte  $(a)$  in  $\#MC^4$ 
    2 Initialize an empty temporary table  $U$ ;
    3 foreach  $(a)$  in  $\#MC^4$  of  $2^8$  values do
        4 Compute 3 bytes  $\triangleleft$  according to  $ec$  in  $\#AK^5$  and  $(x_0, x_1, x_2)$ ;
        5 Compute backwards to  $\#AK^4$  to obtain 4 bytes  $\triangleleft$  (MDS property);
        6 Compute backwards to  $\#AK^3$  to obtain 1 byte  $\triangleright$  (MDS property)
           and  $ec$  in  $\#AK^5$ ;
        7 Compute forwards to  $\#MC^4$  to obtain 3 bytes  $\triangleright$  (MDS property);
        8 Compute backwards to  $\#SB^4$  for the match value
            $m^a = MC^{-1}(0, \#SB^4[13] \triangleright, \#SB^4[14] \triangleleft, \#SB^4[15] \blacksquare)[1]$ ;
        9  $U[m^a] \leftarrow$  related known and deduced cells  $(a, a_0, a_1, a_2, a_3, \triangleleft, \triangleright, \triangleright)$ ;
    // Enumerate 1 free byte  $(b)$  in  $\#MC^4$ 
    10 foreach  $(b)$  in  $\#MC^4$  of  $2^8$  values do
        11 Similarly, compute related cells  $(b_0, b_1, b_2, \blacksquare)$  according to  $ec$ 
           constants and the MDS property;
        12 Then, compute the match value  $m^b =$ 
            $MC^{-1}(\#SB^4[12] \#RK^4[12] \blacksquare, \#RK^4[13] \blacksquare, \#RK^4[14] \blacksquare, \#RK^4[15] \blacksquare)[1]$ ;
           // There will be  $2^{40+8+8-8} = 2^{48}$  solutions on average
    13 Lookup table by  $U[m^b + \#MC^3[13]]$  to derive the blue neutral space;

```

As shown in Figure 6, we present the 8-round pseudo preimage result on Saturnin-hash in the classical setting with the time complexity 2^{224} and memory complexity 2^{176} , which can lead to the preimage attack with the time complexity 2^{240} and memory complexity 2^{176} by the generic conversion [45]. We note that the authors of Saturnin-hash [12] only claimed a preimage security of 2^{224} and a collision security of 2^{112} under the classical setting, so our attacks could provide better understandings of Saturnin-hash against MITM attacks, and the designers' recommendation of 16 super-rounds for hashing ensures a large security margin.

To explain the attack process, we take $\#AK^1$ as the blue starting point. The bytes $\#AK^1[6, 10]$ and $\#AK^2[5]$ are in fact constants during the attack (the XOR operations in Rounds 1 and 2 consume 7 degrees of freedom each). In this attack, we fix 18 bytes $\#AK^0[1, 5]$ and gray impact on $\#MC^1[1, 5]$, $\#SB^2[0-5, 7]$, $\#SB^3[4, 6-11]$ by blue neutral words to zero. We denote the constants as below:

- Enumerating constants ec : 2 cells $\#AK^1[6, 10]$;
- Forward constants fc : gray impact on 1 cell in $\#AK^2[5]$ by blue neutral words denoted as (c_0) ;
- Backward constants bc : gray impact on 10 cells in $\#MC^4[6, 10]$, $\#SB^7[1, 5]$, $\#AK^6[9, 13]$, $\#MC^5[2, 6, 10, 14]$ by blue neutral words denoted as $(c_1, \dots, c_{10})$.

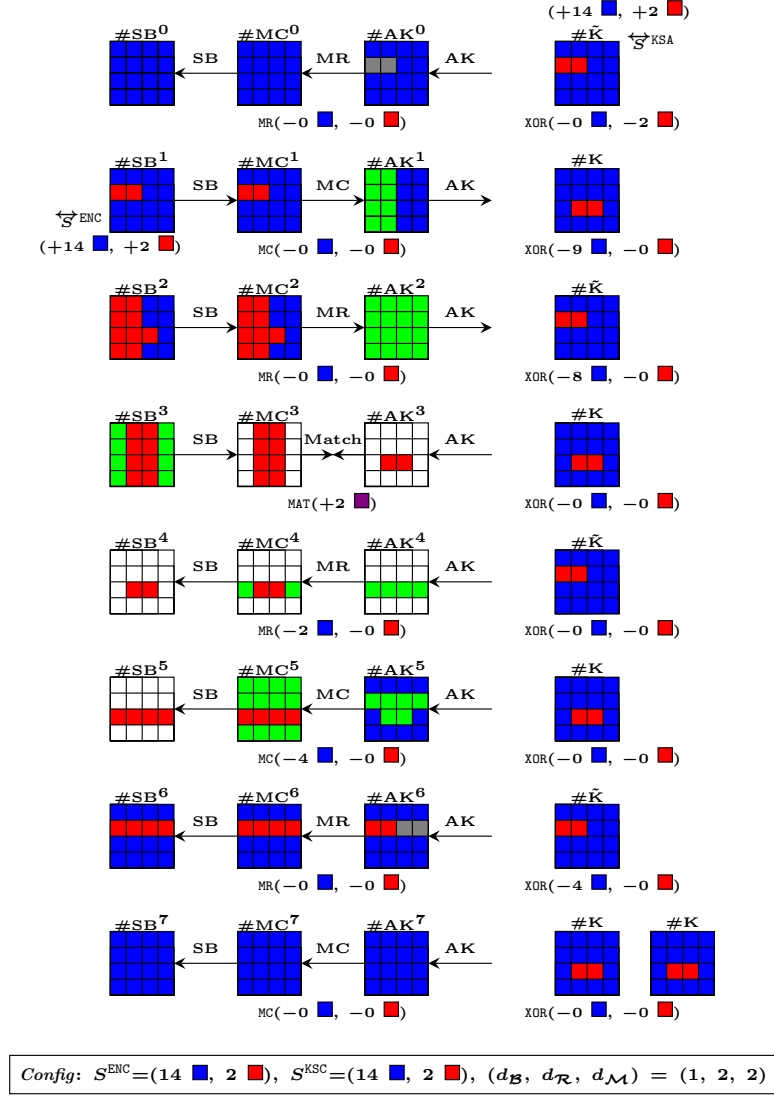


Fig. 6. 8-round pseudo-preimage attack on Saturnin-hash.

Then, following the constant-as-neutral-word MITM framework, the attack process is given in [Algorithm 6](#), where for a given ec constant, one can obtain one red candidate on average for each fc, bc in Stage 1. Thus, it can form 2^{224} candidates and expect a 256-bit full match in the state (the other 32 bits are matched in Stage 1). Finally, the time and memory complexity of this 8-round pseudo preimage attack on Saturnin-hash are 2^{224} and 2^{176} (*i.e.*, the table to index red candidates), respectively.

Algorithm 6: MITM pseudo preimage attack on 8-round Saturnin-hash

```
1 Initialize an empty table  $W$ ;  
2 Use the constants  $\#AK^0[1, 5]$  to derive  $\#K[6, 10]$ ;  
   // Stage 1:  $2^{32-32} = 2^0$  red candidate for each  $fc, bc$   
3 foreach  $r$  in  $2^{32}$  red values do  
   // Forward computation over  $fc$   
4   Initialize an empty temporary table  $U$ ;  
5   foreach  $fc$  in  $2^{16}$  values of  $c_0$  do  
6     Compute value for partial-match  $m^+$ ;  
7      $U[m^+] \leftarrow fc$ ;  
   // Backward computation over  $bc$   
8   foreach  $bc$  in  $2^{160}$  values of  $(c_1, \dots, c_{10})$  do  
9     Compute value for partial-match  $m^-$ ;  
10    foreach  $fc$  in  $U[m^-]$  do  
11      // Table  $W$  has the size of  $2^{160+16+32-32} = 2^{176}$   
       $W[(fc, bc)] \leftarrow r$ ;  
   // Stage 2:  $2^{192}$  blue candidates for each  $ec$   
12 Generate blue neutral word space  $B$  that complies with  $ec$  by Algorithm 7;  
13 foreach  $b$  in  $B$  do  
14   Compute from  $b$  to the constant values of  $(fc, bc)$ ;  
15   foreach  $r$  in  $W[(fc, bc)]$  do  
16     // Form  $2^{224}$  candidates by enumerating  $2^{32}$  values for  $ec$   
     Get the candidate  $(ec, fc, bc, b, r)$ ;  
17     Check if a preimage has been found;
```

Algorithm 7: Blue neutral word generation for 8-round Saturnin-hash

```
1 foreach assignment of the  $2^{64}$  values of  $\#AK^1[8, 12], \#K[8, 12]$  do  
2   Use two constants  $\#SB^3[4, 8]$  to derive  $\#K[9, 13]$ ;  
3   foreach assignment of the  $2^{32}$  values of  $\#AK^1[9, 13]$  do  
4     Use constant  $\#SB^3[9]$  to derive  $\#K[14]$ ;  
5     foreach assignment of the  $2^{16}$  values of  $\#AK^1[14]$  do  
6       Use two constants  $\#SB^2[6, 10]$  to derive  $\#K[11, 15]$ ;  
7       Use two constants  $\#SB^3[7, 11]$  to derive  $\#AK^1[11, 15]$ ;  
8       foreach assignment of the  $2^{80}$  values of  $\#AK^1[0-2, 4-5]$  do  
9         Use two constants  $\#MC^1[1, 5]$  to derive  $\#AK^1[3, 7]$ ;  
10        Use seven constants  $\#SB^2[0-5, 7]$  to derive  $\#K[0-5, 7]$ ;  
11        Derive 11 constants  $\#AK^2[5], \#MC^4[6, 10], \#MC^5[2, 6, 10, 14],$   
          $\#SB^7[1, 5], \#AK^6[9, 13]$ ;
```

6 Applications to Chosen-Prefix Collision and Herding Attacks

In this section, we present CPC attacks and double-phase accelerated herding attacks on AES-MMO/MP and Whirlpool using the extended LIS12 conversion developed in [Section 4](#).

Remark 2. In this paper, the complexity analysis for herding attacks intentionally omits polynomial and constant factors (e.g., 1.51 , $\sqrt{k}$) to keep the balancing condition and overall complexity computation clean. This simplification does not affect our claimed improvement over [\[26\]](#): the speedup originates from applying the extended LIS12 conversion to accelerate the offline diamond construction, a gain that persists under a more rigorous analysis in [\[26\]](#).

6.1 7-round CPC and Herding Attacks on AES-MMO/MP

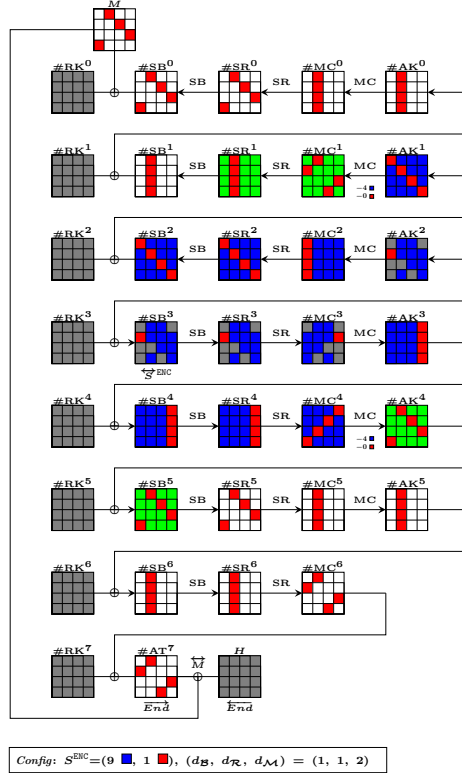


Fig. 7. 7-round (chosen-prefix) collision attack on AES-MMO/MP [\[16, Fig.6\]](#).

6.2 6-round CPC and Herding Attacks on Whirlpool

Similarly to [Section 6.1](#), by [Lemma 2](#), we can convert an MITM-based collision attack into a CPC attack. In particular, by letting $s = 0$ and $t = 32$, the state-of-the-art collision attack on 6-round Whirlpool can be converted to a CPC attack with complexity 2^{240} . This fills an evident gap, as no CPC attack on Whirlpool has been proposed in the classical setting using the current rebound-based framework [\[14\]](#), and matches the best attacked round in the quantum setting.

We give an overview of the double-phase accelerated herding attack on 6-round Whirlpool. For the offline phase, we use the PTP generator in the collision attack on 6-round Whirlpool from [\[16, Fig.10\]](#), with 32-bit partial match and amortized cost $O(1)$ per output (*i.e.*, $s = 0$, $t = 32$). For the online phase, the same trail on 6-round Whirlpool proposed in [\[16, Fig.10\]](#) can be used as the PTP attack, with $\tau = 32$. The total attack complexity is approximately $2^{(2 \cdot 512 - 32 - 32)/3} = 2^{320}$.

7 Conclusion

In this paper, we revisited the algorithmic structure of meet-in-the-middle attacks, with a focus on single-color initial structures and their applications to hash-function cryptanalysis. First, we showed that the constant space in MITM attacks can be used as an additional source of degrees of freedom, leading to an improved SCIS MITM framework that accelerates the red-side computation and relaxes the usual balancing condition in automatic MITM search. Second, we extended the conversion in [\[44\]](#) into the setting of chosen-prefix collision attacks, diamond structure construction, and herding attacks. We demonstrated the impact of these ideas on AES-like hash functions, obtaining improved preimage, collision, chosen-prefix collision, and herding attacks on several reduced-round targets, including Kiasu-BC hashing, Saturnin-hash, AES-MMO/MP, and Whirlpool. These results suggest that revisiting the algorithmic structure of MITM attacks can reveal useful degrees of freedom and broaden the applicability of MITM-based cryptanalysis beyond its traditional settings.

Acknowledgements. We would like to thank all anonymous reviewers for their comments. This research is supported by the National Cryptologic Science Fund of China (Grant No. 2025NCSF01012), the National Natural Science Foundation of China (Grants No. 62422214, 62472421), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB0690200), the Youth Innovation Promotion Association of Chinese Academy of Sciences, the International Partnership Program of Chinese Academy of Sciences (Grant No. 205GJHZ2024005FN), the Shandong Provincial Natural Science Foundation (ZR2026QC1032), the National Natural Science Foundation of China (Grant No. U2336207), the National Cryptologic Science Fund of China (Grant No. 2025NCSF01013), Key R&D Program of Shandong Province, China (Grant No. 2026CXPT037), Fundamental and Interdisciplinary Disciplines Breakthrough

Plan of the Ministry of Education of China (Grant No. JYB2025XDXM114),
and the Ministry of Education in Singapore under Grant RG102/24.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for step-reduced SHA-2. In: Matsui, M. (ed.) *Advances in Cryptology - ASIACRYPT 2009*, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 578–597. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_34, https://doi.org/10.1007/978-3-642-10366-7_34
2. Aoki, K., Sasaki, Y.: Preimage attacks on one-block md4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) *Selected Areas in Cryptography*, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5381, pp. 103–119. Springer (2008). https://doi.org/10.1007/978-3-642-04159-4_7, https://doi.org/10.1007/978-3-642-04159-4_7
3. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) *Advances in Cryptology - CRYPTO 2009*, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 70–89. Springer (2009). https://doi.org/10.1007/978-3-642-03356-8_5, https://doi.org/10.1007/978-3-642-03356-8_5
4. Bao, Z., Ding, L., Guo, J., Wang, H., Zhang, W.: Improved meet-in-the-middle preimage attacks against AES hashing modes. *IACR Trans. Symmetric Cryptol.* **2019**(4), 318–347 (2019). <https://doi.org/10.13154/TOSC.V2019.I4.318-347>, <https://doi.org/10.13154/tosc.v2019.i4.318-347>
5. Bao, Z., Dong, X., Guo, J., Li, Z., Shi, D., Sun, S., Wang, X.: Automatic search of meet-in-the-middle preimage attacks on aes-like hashing. In: Canteaut, A., Standaert, F. (eds.) *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 771–804. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_27, https://doi.org/10.1007/978-3-030-77870-5_27
6. Bao, Z., Guo, J., Shi, D., Tu, Y.: Superposition meet-in-the-middle attacks: Updates on fundamental security of aes-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 64–93. Springer (2022). https://doi.org/10.1007/978-3-031-15802-5_3, https://doi.org/10.1007/978-3-031-15802-5_3
7. Biham, E., Dunkelman, O.: A framework for iterative hash functions - HAIFA. *IACR Cryptol. ePrint Arch.* **2007**, 278 (2007), <http://eprint.iacr.org/2007/278>
8. Blackburn, S.R., Stinson, D.R., Upadhyay, J.: On the complexity of the herding attack and some related attacks on hash functions. *Des. Codes Cryptogr.* **64**(1-2), 171–193 (2012). <https://doi.org/10.1007/S10623-010-9481-X>, <https://doi.org/10.1007/s10623-010-9481-x>
9. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) *Selected Areas in Cryptography - 17th International*

- Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6544, pp. 229–240. Springer (2010). https://doi.org/10.1007/978-3-642-19574-7_16, https://doi.org/10.1007/978-3-642-19574-7_16
10. Bouillaguet, C., Derbez, P., Fouque, P.: Automatic search of attacks on round-reduced AES and applications. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 169–187. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_10, https://doi.org/10.1007/978-3-642-22792-9_10
 11. Cannière, C.D., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4284, pp. 1–20. Springer (2006). https://doi.org/10.1007/11935230_1, https://doi.org/10.1007/11935230_1
 12. Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. IACR Trans. Symmetric Cryptol. **2020**(S1), 160–207 (2020). <https://doi.org/10.13154/TOSC.V2020.IS1.160-207>, <https://doi.org/10.13154/tosc.v2020.iS1.160-207>
 13. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: Improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 222–240. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_13, https://doi.org/10.1007/978-3-642-40041-4_13
 14. Chen, S., Dong, X., Guo, J., Zhang, T.: Chosen-prefix collisions on aes-like hashing. IACR Trans. Symmetric Cryptol. **2024**(4), 64–96 (2024). <https://doi.org/10.46586/TOSC.V2024.I4.64-96>, <https://doi.org/10.46586/tosc.v2024.i4.64-96>
 15. Chen, S., Guo, J., List, E., Shi, D., Zhang, T.: Diving deep into the preimage security of aes-like hashing. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 398–426. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_14, https://doi.org/10.1007/978-3-031-58716-0_14
 16. Chen, S., Guo, J., List, E., Shi, D., Zhang, T.: Scrutinizing the security of aes-based hashing and one-way functions. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16245, pp. 157–188. Springer (2025). https://doi.org/10.1007/978-981-95-5018-0_6, https://doi.org/10.1007/978-981-95-5018-0_6
 17. Connectivity Standards Alliance: Zigbee Core R23.2 Specification. Connectivity Standards Alliance (2023), <https://csa-iot.org/wp-content/uploads/2023/04/05-3474-23-csg-zigbee-specification-compressed.pdf>, document 05-3474-23

18. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). <https://doi.org/10.1007/978-3-662-04722-4>, <https://doi.org/10.1007/978-3-662-04722-4>
19. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer (1989). https://doi.org/10.1007/0-387-34805-0_39, https://doi.org/10.1007/0-387-34805-0_39
20. Degré, M., Derbez, P., Schrottenloher, A.: Simplified meet-in-the-middle preimage attacks on aes-based hashing. IACR Commun. Cryptol. **2**(4), 9 (2026). <https://doi.org/10.62056/AKP2TX4E->, <https://doi.org/10.62056/akp2tx4e->
21. Derbez, P., Fouque, P.: Exhausting demirci-selçuk meet-in-the-middle attacks against reduced-round AES. In: Fast Software Encryption - 20th International Workshop, FSE 2013. pp. 541–560. Lecture Notes in Computer Science, Springer (2013)
22. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977). <https://doi.org/10.1109/C-M.1977.217750>, <https://doi.org/10.1109/C-M.1977.217750>
23. Dinur, I., Jean, J.: Cryptanalysis of FIDES. In: Fast Software Encryption - 21st International Workshop, FSE 2014. pp. 224–240. Lecture Notes in Computer Science, Springer (2014)
24. Dong, X., Guo, J., Li, S., Pham, P.: Triangulating rebound attack on aes-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 94–124. Springer (2022). https://doi.org/10.1007/978-3-031-15802-5_4
25. Dong, X., Guo, J., Li, S., Pham, P.: Chosen-key distinguishing attacks on full aes-192, aes-256, kiasu-bc, and more. IACR Trans. Symmetric Cryptol. **2025**(3), 400–443 (2025). <https://doi.org/10.46586/TOSC.V2025.I3.400-443>, <https://doi.org/10.46586/tosc.v2025.i3.400-443>
26. Dong, X., Guo, J., Li, S., Pham, P., Zhang, T.: Improved meet-in-the-middle nostradamus attacks on aes-like hashing. IACR Trans. Symmetric Cryptol. **2024**(1), 158–187 (2024). <https://doi.org/10.46586/TOSC.V2024.I1.158-187>, <https://doi.org/10.46586/tosc.v2024.i1.158-187>
27. Dong, X., Hua, J., Sun, S., Li, Z., Wang, X., Hu, L.: Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 278–308. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_10, https://doi.org/10.1007/978-3-030-84252-9_10
28. Dong, X., Zhang, Z., Sun, S., Wei, C., Wang, X., Hu, L.: Automatic classical and quantum rebound attacks on aes-like hashing by exploiting related-key differentials. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13090, pp. 241–271. Springer

- (2021). https://doi.org/10.1007/978-3-030-92062-3_9, https://doi.org/10.1007/978-3-030-92062-3_9
29. Dong, X., Zhao, B., Qin, L., Hou, Q., Zhang, S., Wang, X.: Generic mitm attack frameworks on sponge constructions. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IV. *Lecture Notes in Computer Science*, vol. 14923, pp. 3–37. Springer (2024). https://doi.org/10.1007/978-3-031-68385-5_1, https://doi.org/10.1007/978-3-031-68385-5_1
 30. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schl  ffer, M., Thomsen, S.S.: Gr  stl - a SHA-3 candidate. In: *Symmetric Cryptography* (2009)
 31. Gauravaram, P., McCullagh, A., Dawson, E.: Collision attacks on md5 and sha-1: Is this the 'sword of damocles' for electronic commerce? In: *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference*. pp. 73–88 (2006)
 32. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for aes-like permutations. In: Hong, S., Iwata, T. (eds.) *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers. Lecture Notes in Computer Science*, vol. 6147, pp. 365–383. Springer (2010). https://doi.org/10.1007/978-3-642-13858-4_21, https://doi.org/10.1007/978-3-642-13858-4_21
 33. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced meet-in-the-middle preimage attacks: First results on full tiger, and improved results on MD4 and SHA-2. In: Abe, M. (ed.) *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December 5-9, 2010. Proceedings. *Lecture Notes in Computer Science*, vol. 6477, pp. 56–75. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_4, https://doi.org/10.1007/978-3-642-17373-8_4
 34. Hou, Q., Dong, X., Qin, L., Zhang, G., Wang, X.: Automated meet-in-the-middle attack goes to feistel. In: Guo, J., Steinfeld, R. (eds.) *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security*, Guangzhou, China, December 4-8, 2023, Proceedings, Part III. *Lecture Notes in Computer Science*, vol. 14440, pp. 370–404. Springer (2023). https://doi.org/10.1007/978-981-99-8727-6_13, https://doi.org/10.1007/978-981-99-8727-6_13
 35. ISO/IEC: ISO/IEC 10118-2: 2010. IT Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher (2010)
 36. ISO/IEC: ISO/IEC 10118-3: 2018. IT Security techniques - Hash-functions - Part 3: Dedicated hash-functions (2018)
 37. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M.K. (ed.) *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. *Lecture Notes in Computer Science*, vol. 3152, pp. 306–316. Springer (2004). https://doi.org/10.1007/978-3-540-28628-8_19, https://doi.org/10.1007/978-3-540-28628-8_19
 38. Kelsey, J., Kohno, T.: Herding hash functions and the nostradamus attack. In: Vaudenay, S. (ed.) *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, St. Petersburg, Russia, May 28 - June 1, 2006, Proceed-

- ings. Lecture Notes in Computer Science, vol. 4004, pp. 183–200. Springer (2006). https://doi.org/10.1007/11761679_12, https://doi.org/10.1007/11761679_12
39. Kelsey, J., Schneier, B.: Second preimages on n -bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 474–490. Springer (2005). https://doi.org/10.1007/11426639_28, https://doi.org/10.1007/11426639_28
 40. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19–21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 244–263. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_15, https://doi.org/10.1007/978-3-642-34047-5_15
 41. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schl  ffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6–10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 126–143. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_8, https://doi.org/10.1007/978-3-642-10366-7_8
 42. Leurent, G., Peyrin, T.: From collisions to chosen-prefix collisions application to full SHA-1. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 527–555. Springer (2019). https://doi.org/10.1007/978-3-030-17659-4_18, https://doi.org/10.1007/978-3-030-17659-4_18
 43. Leurent, G., Peyrin, T.: SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12–14, 2020. pp. 1839–1856. USENIX Association (2020), <https://www.usenix.org/conference/usenixsecurity20/presentation/leurent>
 44. Li, J., Isobe, T., Shibutani, K.: Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to SHA-2. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19–21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 264–286. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_16, https://doi.org/10.1007/978-3-642-34047-5_16
 45. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996). <https://doi.org/10.1201/9781439821916>, <http://cacr.uwaterloo.ca/hac/>
 46. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer (1989). https://doi.org/10.1007/0-387-34805-0_40, https://doi.org/10.1007/0-387-34805-0_40
 47. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) Advances in Cryptology - CRYPTO

- '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings. Lecture Notes in Computer Science, vol. 773, pp. 368–378. Springer (1993). https://doi.org/10.1007/3-540-48329-2_31, https://doi.org/10.1007/3-540-48329-2_31
48. Qin, L., Hua, J., Dong, X., Yan, H., Wang, X.: Meet-in-the-middle preimage attacks on sponge-based hashing. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 158–188. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_6, https://doi.org/10.1007/978-3-031-30634-1_6
 49. Qin, L., Zhao, B., Hou, Q., Jia, K., Wang, G., Dong, X.: Linear cancellations in the mitm attacks on sponge functions. *IACR Trans. Symmetric Cryptol.* **2025**(3), 633–692 (2025). <https://doi.org/10.46586/TOSC.V2025.I3.633-692>, <https://doi.org/10.46586/tosc.v2025.i3.633-692>
 50. Sasaki, Y.: Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In: Joux, A. (ed.) *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*. Lecture Notes in Computer Science, vol. 6733, pp. 378–396. Springer (2011). https://doi.org/10.1007/978-3-642-21702-9_22, https://doi.org/10.1007/978-3-642-21702-9_22
 51. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, December 7-11, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5350, pp. 253–271. Springer (2008). https://doi.org/10.1007/978-3-540-89255-7_16, https://doi.org/10.1007/978-3-540-89255-7_16
 52. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 134–152. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_8, https://doi.org/10.1007/978-3-642-01001-9_8
 53. Schrottenloher, A., Stevens, M.: Simplified MITM modeling for permutations: New (quantum) attacks. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*. Lecture Notes in Computer Science, vol. 13509, pp. 717–747. Springer (2022). https://doi.org/10.1007/978-3-031-15982-4_24, https://doi.org/10.1007/978-3-031-15982-4_24
 54. Schrottenloher, A., Stevens, M.: Simplified modeling of MITM attacks for block ciphers: New (quantum) attacks. *IACR Trans. Symmetric Cryptol.* **2023**(3), 146–183 (2023). <https://doi.org/10.46586/TOSC.V2023.I3.146-183>, <https://doi.org/10.46586/tosc.v2023.i3.146-183>
 55. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Barcelona, Spain, May 20-24, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4515, pp. 1–22.

- Springer (2007). https://doi.org/10.1007/978-3-540-72540-4_1, https://doi.org/10.1007/978-3-540-72540-4_1
56. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and applications. *Int. J. Appl. Cryptogr.* **2**(4), 322–359 (2012). <https://doi.org/10.1504/IJACT.2012.048084>, <https://doi.org/10.1504/IJACT.2012.048084>
 57. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S. (ed.) *Advances in Cryptology - CRYPTO 2009*, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2009. *Proceedings. Lecture Notes in Computer Science*, vol. 5677, pp. 55–69. Springer (2009). https://doi.org/10.1007/978-3-642-03356-8_4, https://doi.org/10.1007/978-3-642-03356-8_4
 58. Zhang, Z., Sun, S., Wang, C., Hu, L.: Classical and quantum meet-in-the-middle nostradamus attacks on aes-like hashing. *IACR Trans. Symmetric Cryptol.* **2023**(2), 224–252 (2023). <https://doi.org/10.46586/TOSC.V2023.I2.224-252>, <https://doi.org/10.46586/tosc.v2023.i2.224-252>
 59. Zhao, B., Hou, Q., Qin, L., Dong, X.: Triangulating meet-in-the-middle attack. In: Kalai, Y.T., Kamara, S.F. (eds.) *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 17–21, 2025, *Proceedings, Part V. Lecture Notes in Computer Science*, vol. 16004, pp. 260–292. Springer (2025). https://doi.org/10.1007/978-3-032-01901-1_9, https://doi.org/10.1007/978-3-032-01901-1_9

Supplementary Material

Appendix A AES-like Hashing

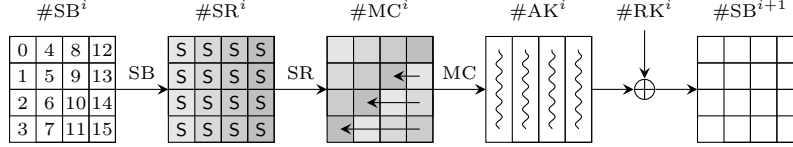


Fig. 9. AES-like round function.

In 2001, the NIST selected a subset of the Rijndael family of block ciphers [18] with 128-bit block size as the Advanced Encryption Standard (AES). As shown in Figure 9, an internal state of AES can be viewed as a 2-dimensional array, and an AES round consists of the following operations:

- SubBytes (SB): Applies the S-box to each byte of the state.
- ShiftRows (SR): Applies a left rotation on each row.
- MixColumns (MC): Left multiply a constant matrix to each column.
- AddRoundKey (AK): XOR the round key to the state.

In 2010, ISO and IEC standardized the usage of block-cipher-based hash functions [35]. A common paradigm to this end is to first construct a compression function by instantiating a secure block cipher with a PGV mode [47], then iterate it following Merkle-Damgård [46,19] or HAIFA [7] to process message with arbitrary length. AES-like hashing thus refers to hash functions built with an AES-like block cipher following the paradigm. Typical examples include Whirlpool [36], AES-MMO-128 [17], Grøstl [30], and many others.

Appendix B Collision Attacks on Saturnin-hash

B.1 7-round Collision Result on Saturnin-hash

The 7-round MITM collision attack trail of Saturnin-hash is presented in Figure 10, similar to Section 5.1, we can mount a collision attack on 7-round Saturnin-hash, which has the time and memory complexity of both with 2^{120} .

B.2 7-round CPC and Herding Results on Saturnin-hash

Similar to Section 6.1 and following the Lemma 2, we transform the MITM-based collision attack into a CPC attack. By letting $s = 0, t = 16$, the state-of-the-art collision attack on 7-round Saturnin-hash can be converted to CPC attack with complexity 2^{120} .

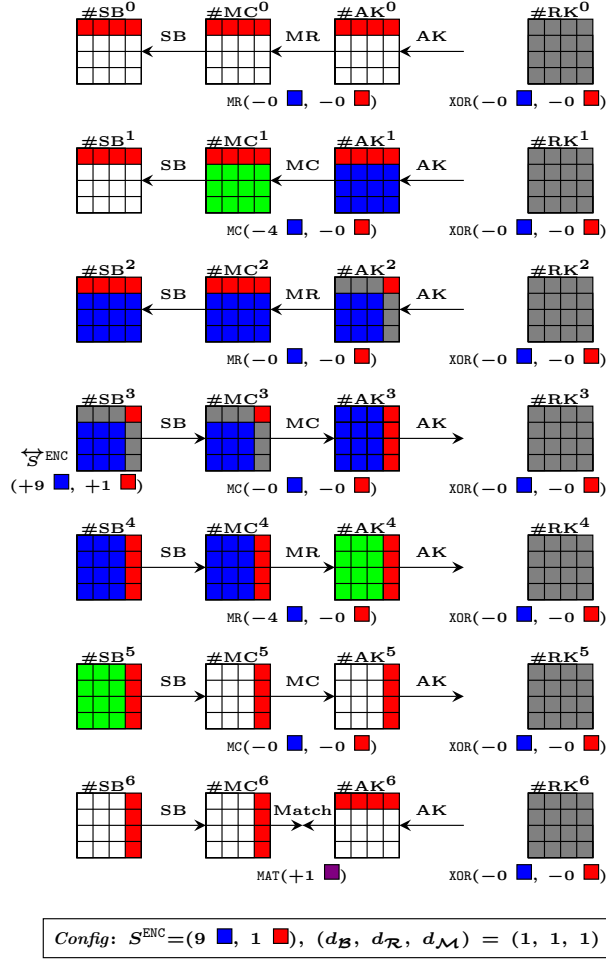


Fig. 10. 7-round collision attack on Saturnin-hash.

We now give a brief overview and complexity analysis of herding attack on 7-round Saturnin-hash. For the offline phase, we will leverage the proposed attack on 7-round Saturnin-hash (Section B.1), which is an efficient partial-target preimage generator with 16-bit condition and amortized cost $O(1)$ (i.e., $s = 0$, $t = 16$). For the online phase, the same 7-round trail of Saturnin-hash in Section B.1 can be used as the partial-target preimage attack, with $\tau = 16$. We now have the balancing condition $k \approx (n + s - 2\tau)/3 = 80$. The leaves of the diamond structure has the following format: the last 5 cells are enumerated, while the rest bits are fixed to zeros. And the online MITM attacks thus find a partial-target preimage for the fixed-zero bits. According to Equation (15), the total attack complexity is approximately $2^{(2 \cdot 256 - 16 - 16)/3} = 2^{160}$.