Diving Deep into the Preimage Security of AES-Like Hashing

Shiyao Chen[4,5], Jian Guo[3], Eik List[6], Danping Shi[1,2(✉)], and Tianyu Zhang[3]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
shidanping@iie.ac.cn

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[3] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
guojian@ntu.edu.sg , tianyu005@e.ntu.edu.sg

[4] Strategic Centre for Research in Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore, Singapore
shiyao.chen@ntu.edu.sg

[5] Digital Trust Centre, Nanyang Technological University, Singapore, Singapore

[6] Weimar, Germany

Abstract. Since the seminal works by Sasaki and Aoki, Meet-in-the-Middle (MITM) attacks are recognized as an effective technique for preimage and collision attacks on hash functions. At Eurocrypt 2021, Bao *et al.* automated MITM attacks on `AES`-like hashing and improved upon the best manual result. The attack framework has been furnished by subsequent works, yet far from complete. This paper introduces three key contributions dedicated to further generalizing the idea of MITM and refining the automatic model on `AES`-like hashing. (1) We introduce *S-box linearization* to MITM pseudo-preimage attacks on `AES`-like hashing. The technique works well with superposition states to preserve information after S-boxes at affordable cost. (2) We propose *distributed initial structures*, an extension on the original concept of initial states, that selects initial degrees of freedom in a more versatile manner to enlarge the search space. (3) We exploit the *structural similarities* between encryption and key schedule in constructions (*e.g.*, `Whirlpool` and `Streebog`) to model propagations more accurately and avoid repeated costs. Weaponed with these innovative techniques, we further empower the MITM framework and improve the attack results on `AES`-like designs for preimage and collision. We obtain the first preimage attacks on 10-round `AES`-192, 10-round `Rijndael`-192/256, and 7.75-round `Whirlpool`, reduced time and/or memory complexities for preimage attacks on 5-, 6-round `Whirlpool` and 7.5-, 8.5-round `Streebog`, as well as improved collision attacks on 6- and 6.5-round `Whirlpool`.

Eik List - visiting independent researcher at Nanyang Technological University, Singapore.

M. Joye and G. Leander (Eds.): EUROCRYPT 2024, LNCS 14651, pp. 398–426, 2024.
https://doi.org/10.1007/978-3-031-58716-0_14

Keywords: Meet-in-the-Middle Attack · S-box Linearization · Distributed Initial Structures · Structural Similiarities · AES · Rijndael · Whirlpool · Streebog

1 Introduction

1.1 Hash Functions

Hash functions map arbitrary long inputs to fixed-length hash values and have been used in a myriad of applications. There are three fundamental security requirements for a cryptographic hash function to fulfill, namely preimage, second-preimage, and collision resistance. This work focuses on the notion of preimage resistance: given a hash function H and a random hash value t, it should be computationally infeasible to find a preimage x such that $H(x) = t$.

To make use of the coexistence of encryption and hashing in embedded systems, a conventional strategy is to construct a hash function from a secure block cipher to minimize hardware or software costs: the encryption function of a block cipher is first transformed into a one-way compression function and then iterated following the Merkle-Damgård design. In 1993, Preneel, Govaerts, and van de Walle [33] identified 12 secure modes for the encryption-compression-function conversion, later known as the PGV modes.

The strategy is highly practical if the underlying block cipher is widely used and has seen a long record of withstanding cryptanalysis, which makes `AES` the perfect candidate. The MMO mode (one of the PGV modes) instantiated with `AES-128` have been standardized by the Zigbee [2] protocol suite and ISO/IEC [24]. Given the high security of `AES`, several dedicated hash functions are designed with `AES`-like structures, *e.g.*, the ISO standards `Whirlpool` [9,23] or the ISO and GOST standard `Streebog` [1,15,25], which are collectively referred to as `AES`-like hashing.

1.2 Meet-in-the-Middle Attacks on Block-Cipher-Based Hashing

The Meet-in-the-Middle (MITM) attack is well-known for its effectiveness in cryptanalysis of Double-DES [14] and key recovery. In a series of pioneer works [4,5,36,37], Sasaki and Aoki enlightened the community with MITM attacks applied to the security analyzation of cryptographic hash functions. The core attack framework had been extended ever since by numerous techniques, such as splice-and-cut [4], initial structures [37], indirect and partial matching [4,37], biclique as a formalization of initial structures [26], sieve-in-the-middle [11] and match-boxes [17].

MITM attack on block-cipher-based hash functions is, in essence, a pseudo-preimage attack: the attack splits the computation of the compression function into two chucks, the forward and the backward chunk, so that two portions of input bits, called neutral bits, affect only one of the sub-functions. In such a setting, the chunks are computed independently and end at a common state

where their (partial) values are matched. Usually, a third set of bits is shared by both chunks, which is captured in the notion of a 3-subset MITM attack [10].

Sasaki was the first to apply this to a preimage attack on AES hashing modes [35]. However, to avoid the complex relations from the round keys, the key was still fixed to a constant. Sasaki *et al.* then introduced the guess-and-determine strategy [38]. Bao *et al.* [6] revisited the attacks by introducing the degree of freedom from the key space.

At Eurocrypt 2021, Bao *et al.* [7] automated the search for efficient MITM preimage attacks with Mixed-integer Linear Programming (MILP) and applied it to AES hashing modes and Haraka v2. Dong *et al.* [16] later extended this automation model to search for key-recovery and collision attacks and introduced nonlinear constraints for the neutral bits. Later in 2022, Bao *et al.* [8] brought up the concept of superposition bytes, which allowed forward and backward neutral words to propagate simultaneously and independently at a common byte through linear operations in the encryption and key schedule. They also proposed bi-directional attribute propagation and cancellation, *i.e.*, the known values in each chunk are propagated not only in the direction of the chunk but in both directions. Moreover, they integrated the guess-and-determine method into their models. Hua *et al.* [22] then combined guess-and-determine with nonlinearly constrained neutral words in their search for preimage attacks. More recently, Qin *et al.* [34] applied the new framework to Sponges. As a contrast to those very detailed frameworks, Schrottenloher and Stevens proposed a simpler MILP-modeling approach for preimage attacks against keyless permutations [39] and was later extended to ciphers with very light key schedule [40]. While their model was considerably more lightweight and applicable to AES-like permutations, its exclusion of the key schedule made it less effective against the AES than the detailed frameworks.

1.3 Gaps

While previous works on automating MITM on AES-like hashing already provided a groundlaying seminal framework [7,8,16,34], the complexity of the task has left several gaps. Among those, we identified three core challenges:

1. The preimage security evaluation on AES-like hashing has always been at byte-level as the S-box details are abstracted away. Recently, Zhang *et al.* [42] studied the field inversion S-box with algebraic properties and thus can consider the S-box details. However, quoting their words, *this linearizes the non-linear layer of AES, but unfortunately, no attacks better than the current state-of-the-art has been found based on this fact.*
2. The selection of initial states in previous works was limited to two full states, one of them in the encryption function and the other in the key schedule. Initial states could be more scattered, even across several intermediate states. Thus, the artificial limits on initial states had discarded a fraction of the solution pool.

3. It has been a long endeavor to address the dependencies in the model that lead to incorrect measures on the degree of freedom consumption. Particularly, the dependencies due to the structural similarity between the encryption and key schedule in some designs have been overlooked and may lead to duplicate costs.

1.4 Our Contributions

In this work, we have proposed and incorporated three techniques to fill the gaps and improve the state-of-the-art attacks. We would like to point out that the core ideas behind the techniques are fairly generic and expected to have more applications beyond this paper.

Linearizing S-Boxes. We introduce *S-box linearization* (**LIN**) to MITM attacks on AES-like hashing and efficiently incorporate it with the superposition structure. Both propagations at a superposition byte are preserved through the S-box at the cost of guessing over a small pool of *hints*. Making use of the linear relation between propagations, **LIN** checks if a guessed hint is correct efficiently using for- and backward values at S-box input.

In comparison, the study in [42] exploited the algebraic properties of field inversion S-boxes thus resulting in guess-and-determine and announced no improved result on AES. A similar conclusion on their work was also drawn by Liu *et al.* in [29]. The checking phase in plain guess-and-determine requires full information on forward and backward neutral bytes and leads to a cost on degrees of matching, while **LIN** in our proposal uses only local information and spares such cost. To conclude, **LIN** serves as a lightweight alternative to plain guess-and-determine and introduces a new trade-off rule. The technique enables us to mount the first bit-level preimage attacks on AES-like hashing and improve the state-of-the-art.

Distributed Initial Structures. In this work, we further generalize the concept of initial structures, originally proposed by Sasaki and Aoki [37]. We lift the artificial limitation on selecting two full states as initial states and introduce *distributed initial structures* (**DIS**). We now allow the initial states to be distributed in a combination of encryption states and round keys, as long as the total initial degrees of freedom remain the same. An important reflection of this idea is to assign more superposition bytes in the AES key schedule. As only a portion of bytes is propagated through the AES S-box in each key schedule round, more superposition information can be allowed in round keys by the introduction of **DIS**. This has expanded the solution pool by adding more alternatives to the invocation of constraints and allowing more valid propagation patterns.

Structural Similarities. The *structural similarities* (**SIM**) between encryption and key schedule may lead to dependencies across multiple rounds. We observe that values injected into an encryption state by round key addition may propagate through similar sets of operators in encryption and key schedule. Therefore, certain costs of degrees of freedom can be traced back to the same constraints

and previous attacks may be suboptimal due to double counting such costs. By modeling the degree consumption more accurately, we enlarged the search space by sparing unnecessary double costs of earlier approaches. Moreover, the approach potentially finds attacks with high concentrations of constraints around the starting points, which could help reduce the memory complexity of the attack.

1.5 Application Results

Our results are as summarized in Table 1. The effectiveness of our proposed techniques is well demonstrated through improved attacks on standards including `AES-192` hashing, `Whirlpool`, and `Streebog`, as well as the `Rijndael` hashing family. We argue that the techniques are significant and essential to the breakthroughs.

Both **LIN** and **DIS** are critical for attacking one additional round of `AES-192` hashing, i.e. excluding either technique would not yield an attack. Simply using guess-and-determine strategy combined with the `AES` S-box property also could not improve the attack on `Rijndael-192/256`. Moreover, the attack advantage on (pseudo-)preimage is non-trivial, *i.e.*, proportional to the size of a subset rather than a fixed constant.

Incorporating **SIM**, we improve the (pseudo-) preimage attacks on 5- and 6-round `Whirlpool` in terms of time and/or memory complexity. In particular, we achieve a memoryless attack on 5-round `Whirlpool`. Besides, we present the first preimage attack on 7.75-round `Whirlpool`, which extends the state-of-the-art by almost a full round while maintaining the same time complexity. What is more, our efficient MILP-based search model improves the 6-round collision attack on `Whirlpool` and extends it to 6.5 rounds. For `Streebog`, our approach reduces the time and/or memory complexity on 7.5- and 8.5-round `Streebog` compared to previous best (pseudo-)preimage attacks.

1.6 Organization

The remainder of this work is structured as follows. In Sect. 2, we provide preliminaries on the MITM attacks and the target designs. Then we elaborate the proposed techniques and their significance in Sect. 3. Thereupon, we present the enhanced MITM framework and MILP modeling in Sect. 4. We detailed pseudo-preimage results of the first attack on 10-round `AES-192` hashing and the memoryless attack on 5-round `Whirlpool` in Sect. 5. Furthermore, other attacks and details of `Whirlpool`, `Rijndael`, 8-round `AES-192` and `Streebog` are provided in the full version [12]. Finally, we conclude and discuss in Sect. 6.

2 Preliminaries

2.1 MITM Attacks: Notations and Principle

We provide a high-level overview of MITM pseudo-preimage attacks in Fig. 1 and a list of notations common in all our attack descriptions in Table 2.

Table 1. Results of our improved attacks on AES-like Hashing.

Preimage Attacks

Cipher (target)	#Rounds	T_1 †	T_2 ‡	Memory	Essential technique(s)	References
AES-192 (Hash)	8/12	2^{112} §	2^{116}	2^{16}	MITM	[6]
	8/12	$\mathbf{2^{100}}$	$\mathbf{2^{115}}$	2^{96}	**LIN**, **DIS**, BiDir*	[12, App. B]¶
	9/12	2^{120}	2^{125}	–	MILP	[7]
	9/12	2^{112}	2^{121}	–	BiDir	[8]
	10/12	2^{124}	2^{127}	2^{124}	**LIN**, **DIS**, BiDir	Sect. 5.1
Rijndael-192/192 (Hash)	9/12	2^{184}	2^{189}	–	BiDir	[43]
	9/12	$\mathbf{2^{180}}$	$\mathbf{2^{187}}$	2^{180}	**LIN**, BiDir	[12, App. C.1]
Rijndael-192/256 (Hash)	9/12	2^{168}	2^{181}	–	BiDir	[43]
	10/12	2^{180}	2^{187}	2^{180}	**LIN**, BiDir	[12, App. C.2]
Whirlpool (Hash)	5/10	2^{416}	2^{448}	2^{96}	Dedicated	[38]
	5/10	2^{352}	2^{433}	2^{160}	BiDir, MulAK*	[8]
	5/10	$\mathbf{2^{320}}$	$\mathbf{2^{417}}$	$\boldsymbol{O(1)}$	**SIM**, BiDir	Sect. 5.2
	6/10	2^{448}	2^{481}	2^{256}	Dedicated, GnD*	[38]
	6/10	2^{440}	2^{477}	2^{192}	GnD	[8]
	6/10	$\mathbf{2^{416}}$	$\mathbf{2^{465}}$	2^{288}	**SIM**, BiDir, GnD	[12, App. D.1]
	7/10	2^{480}	2^{497}	2^{128}	GnD, MulAK	[8]
	7.75/10	2^{480}	2^{497}	2^{256}	**SIM**, BiDir, GnD	[12, App. D.2]
Streebog-512 (Compression)	7.5/12	2^{496}	–	2^{64}	Dedicated method	[30]
	7.5/12	2^{441}	–	2^{192}	GnD, MulAK	[22]
	7.5/12	$\mathbf{2^{433}}$	–	$\mathbf{2^{177}}$	**SIM**, GnD	[12, App. E.1]
	8.5/12	2^{481}	–	2^{288}	GnD, MulAK	[22]
	8.5/12	2^{481}	–	$\mathbf{2^{129}}$	**SIM**, GnD	[12, App. E.2]
Streebog-512 (Hash)	7.5/12	–	2^{496}	2^{64}	Dedicated method	[30]
	7.5/12	–	$2^{478.25}$	2^{256}	MITM + Multi-collision⋆	[22]
	7.5/12	–	$\mathbf{2^{474.25}}$	2^{256}	MITM + Multi-collision	[12, App. E.1]
	8.5/12	–	$2^{498.25}$	2^{288}	MITM + Multi-collision	[22]
	8.5/12	–	$2^{498.25}$	$\mathbf{2^{256}}$	MITM + Multi-collision	[12, App. E.2]

Collision Attacks

Cipher (target)	#Rounds	Time	Memory	Essential technique(s)	References
Whirlpool (Hash)	4.5/10	2^{120}	2^{16}	Rebound	[31]
	4.5/10	2^{64}	2^{16}	Rebound	[28]
	5/10	2^{120}	2^{64}	Super-SBox	[18,27]
	5.5/10	2^{184-s}	2^{s}	Rebound	[28]
	6/10	2^{228}	2^{228}	Quantum	[20]
	6/10	2^{248}	2^{248}	MILP, MITM	[16]
	6/10	$\mathbf{2^{240}}$	$\mathbf{2^{240}}$	New MILP model, MITM	[12, App. A]
	6.5/10	$\mathbf{2^{240}}$	$\mathbf{2^{240}}$	New MILP model, MITM	[12, App. A]

† T_1 represents the time complexity of the pseudo-preimage attack on compression function.
‡ T_2 represents the time complexity of the preimage attack on hash function.
§ We list only single-target result in [6] for comparison in this table.
* BiDir, MulAK and GnD are techniques introduced to the MITM framework in [8], respectively, short for bi-directional attribute propagation and cancellation, multiple ways of AddRoundKey and guess-and-determine.
¶ Please refer to the full version of this paper on ePrint [12].

⋆ The attack on the compression function of Streebog is converted into a preimage attack on its hash function using the technique from [3].

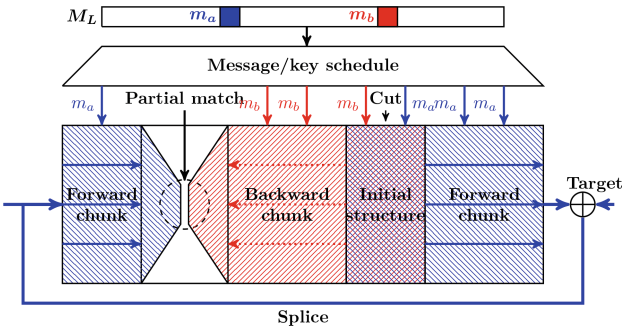

Fig. 1. A high-level overview of MITM attacks [35].

In block-cipher-based hashing, the MITM technique is often used to mount only a pseudo-preimage attack, *i.e.*, a preimage that uses a chaining value different from the fixed initial value of the hash function specification. The pseudo-preimage is later transformed into a preimage on the hash function.

The MITM attack divides the computations into two independent chunks, forward and backward. A byte is called neutral if its value is used only (*i.e.*, is known only) in one of the chunks and has a constant influence on the respective other. Both chunks end at E^+ and E^-, respectively, which are the input and output of a matching operation M. Note that the attack exploits the feed-forward of start and end values in block-cipher-based compression functions. Based on the properties of M, an MITM attack can invoke certain constraints to filter ineligible candidates, which are called partial-match constraints. Those pairs that satisfy these constraints are checked thereupon on larger parts of their states if their combination constitutes a valid pseudo-preimage.

The MITM attack framework [8] with guess-and-determine is described in the following. Without loss of generality, we assume $d_{\mathcal{B}} + g_{\mathcal{B}} \leq d_{\mathcal{R}} + g_{\mathcal{R}}$:

1. Assign arbitrary values to the constants in pre-defined constraints.
2. Compute V^+ and V^- based on the constants.
3. For all tuples $(v^+, g^+, g) \in V^+ \times G^+ \times G$, compute to E^+, obtain m^+ for matching, store (v^+, g^+) in $T^+[m^+, g]$. We have $|T^+| = 2^{d_{\mathcal{B}}+g_{\mathcal{B}}+g_{\mathcal{BR}}}$
4. For all tuples $(v^-, g^-, g) \in V^- \times G^- \times G$, compute to E^-, obtain m^- for matching.
5. For all (v^+, g^+) in $T^+[m^-, g]$, compute to check if (g^+, g^-, g) is compatible with v^+ and v^-.
6. For compatible (v^+, v^-), check for a full match.
7. If a full match is discovered, compute and return the preimage. Otherwise, revert to Step 1, change the arbitrary values, and repeat the rest.

The computational complexity of the MITM pseudo-preimage attacks is

$$\begin{aligned} & 2^{n-(d_{\mathcal{B}}+d_{\mathcal{R}})} \cdot \left(2^{d_{\mathcal{B}}+g_{\mathcal{B}}+g_{\mathcal{BR}}} + 2^{d_{\mathcal{R}}+g_{\mathcal{R}}+g_{\mathcal{BR}}} + 2^{d_{\mathcal{B}}+g_{\mathcal{B}}+d_{\mathcal{R}}+g_{\mathcal{R}}+g_{\mathcal{BR}}-d_{\mathcal{M}}}\right) \\ \simeq\ & 2^{n-min(d_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}}, d_{\mathcal{R}}-g_{\mathcal{B}}-g_{\mathcal{BR}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}})} := 2^l . \end{aligned} \tag{1}$$

Table 2. Common notations.

DoF	Degree(s) of freedom
$S^{\texttt{ENC}}$	Starting state in the encryption
$S^{\texttt{KSA}}$	Starting state in the key schedule
E^+/E^-	Ending states of forward and backward computations, respectively
M	Matching operation between E^+ and E^-
$d_{\mathcal{B}}/d_{\mathcal{R}}$	DoF of the forward and backward chunk, respectively
$g_{\mathcal{B}}/g_{\mathcal{R}}/g_{\mathcal{BR}}$	DoF of guessed values in the forward, backward, and in both chunks, respectively
$d_{\mathcal{M}}$	Degrees of matching
V^+/V^-	Sets of values for forward and backward neutral bytes satisfying the predefined constraints, with $\lvert V^+\rvert = 2^{d_{\mathcal{B}}}$ and $\lvert V^-\rvert = 2^{d_{\mathcal{R}}}$, respectively
$G^+/G^-/G$	Sets of guessed values in forward/backward/both chunk(s), with $\lvert G^+\rvert = 2^{g_{\mathcal{B}}}$, $\lvert G^-\rvert = 2^{g_{\mathcal{R}}}$, and $\lvert G\rvert = 2^{g_{\mathcal{BR}}}$
T^+/T^-	Lookup tables constructed at E^+/E^-
$\mathcal{B}^{\texttt{KSA}}/\mathcal{R}^{\texttt{KSA}}/\mathcal{G}^{\texttt{KSA}}$	Sets of indices of forward neutral, backward neutral, and constant bytes in $S^{\texttt{KSA}}$, respectively
$\mathcal{B}^{\texttt{ENC}}/\mathcal{R}^{\texttt{ENC}}/\mathcal{G}^{\texttt{ENC}}$	Sets of indices of forward neutral, backward neutral, and constant bytes in $S^{\texttt{ENC}}$, respectively
$\overrightarrow{\iota}/\overleftarrow{\iota}$	Initial DoF of the forward and backward chunk, respectively, with $\overrightarrow{\iota} = \lvert\mathcal{B}^{\texttt{ENC}}\rvert + \lvert\mathcal{B}^{\texttt{KSA}}\rvert$ and $\overleftarrow{\iota} = \lvert\mathcal{R}^{\texttt{ENC}}\rvert + \lvert\mathcal{R}^{\texttt{KSA}}\rvert$
$\overrightarrow{\sigma}/\overleftarrow{\sigma}$	The consumed DoF of the forward and backward chunk, respectively

A pseudo-preimage attack with a computational complexity of 2^l ($l < n - 2$) can be converted to a preimage attack with a computation complexity of $2^{(n+l)/2+1}$ [32]. First, a total of $2^{(n-l)/2}$ pseudo-preimages is obtained. Then, a total of $2^{(n+l)/2+1}$ random values are inserted after the initialization vector IV to obtain $2^{(n+l)/2+1}$ chaining values. Then, one can expect a match between a chaining value and a pseudo-preimage with non-negligible probability, which yields a preimage for the hash function.

2.2 AES-Like Hashing

To start with, we list some common notations in AES-like hashing:

- Nb/Nk: number of columns of a state in the encryption procedure or the secret key. When Nb and Nk are identical, we will denote both by NCOL.
- NROW: number of rows in an encryption or key state.

AES-like hashing refers to hash functions whose compression function follows an AES-like round structure. In this section, we will focus on recalling the necessary details of AES and Whirlpool that are used in this work.

AES. In 2001, the NIST selected a subset of the Rijndael family of block ciphers [13] with a block size of 128 bits and key sizes of 128, 192, or 256 bits (Nb = 4,

$\texttt{Nk} \in \{4, 6, 8\}$, and $\texttt{NROW} = 4$) as the Advanced Encryption Standard. Conventionally, the transposed of a column is referred to as a word, and the word size is thus fixed to $4 \times 8 = 32$ bits. As shown in Fig. 2, an AES round consists of the following operations:

- SubBytes (SB): A non-linear byte-wise substitution.
- ShiftRows (SR): A cyclic left shift on the i-th row by i bytes, for $i \in \{0, 1, 2, 3\}$.
- MixColumns (MC): A column-wise left multiplication of a 4-×-4 maximum-distance-separable matrix.
- AddRoundKey (AK): A bitwise XOR of the round key to the state.

The final round differs in the sense that it omits the MixColumns operation. Before the first round, a whitening key is added to the plaintext.

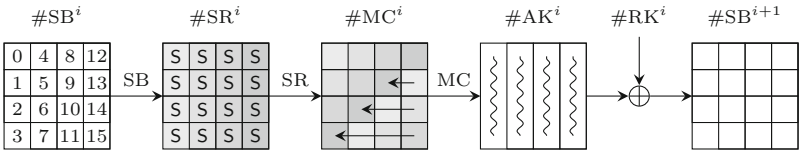

Fig. 2. AES-like round function.

The round keys are expanded from the master key *key*: Let w be an array of bytes, when $i < \texttt{Nk}$, the key words are derived directly from the secret key $w[i] = key[i]$. Otherwise, $w[i]$ is calculated as follows:

$$\begin{cases} w[i - \texttt{Nk}] \oplus \texttt{Rot}(\texttt{S}(w[i-1])) \oplus C[i/\texttt{Nk}] & i \bmod \texttt{Nk} \equiv 0 \text{ and } \texttt{Nk} < 8 \\ w[i - \texttt{Nk}] \oplus \texttt{S}(w[i-1]) & i \bmod \texttt{Nk} \equiv 4 \text{ and } \texttt{Nk} = 8 \\ w[i - \texttt{Nk}] \oplus w[i-1] & \text{otherwise,} \end{cases} \tag{2}$$

where S denotes the AES S-box, Rot is a left rotation of the input by one byte, and C represents the list of round constants.

The AES-128 in the Matyas-Meyer-Oseas (MMO) mode is used in the standards of the Zigbee protocol suite [2] and ISO/IEC [24]. The MMO mode is defined as the mapping $f : f(H_i, M_i) = E_{H_i}(M_i) \oplus M_i$, where H_i stands for the i-th chaining value, M_i as the i-th message block, and E_k stands for the block cipher encryption under key k.

Whirlpool. In 2000, Rijmen and Barreto [9] designed Whirlpool as a submission to the NESSIE competition that was later tweaked and adopted as an ISO/IEC standard [23]. Whirlpool is a block-cipher-based hash function with a 512-bit hash value, which adopts a 10-round AES-like block cipher with 8×8-byte ($\texttt{NROW} = \texttt{NCOL} = 8$) keys and plaintexts in Miyaguchi-Preneel mode [33] (MP mode) as its compression function (CF). The MP mode is defined as $f(H_i, M_i) = E_{H_i}(M_i) \oplus M_i \oplus H_i$. It takes the 512-bit chaining value H_i as the key and the 512-bit message

block M_i as its plaintext input. Encryption and key schedule essentially use the same round function, except for the fact that the key state has additions with round constants and the encryption state sees additions with the round keys. The round function is depicted in Fig. 3 and consists of:

- `SubBytes` (SB): applies the Substitution-Box to each byte.
- `ShiftColumns` (SC): cyclically shifts the j-column downwards by j bytes.
- `MixRows` (MR): multiplies each row of the state by an MDS matrix.
- `AddRoundKey` (AK): XORs the round key to the state.

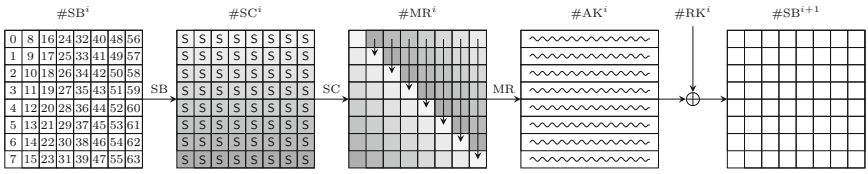

Fig. 3. The round function of `Whirlpool`.

Note that the final round is a complete round unlike that in the `AES`; a whitening key is added before the first round of encryption as in the `AES`. However, in the MP mode, the whitening key cancels in splice-and-cut MITM attacks due to the feed-forward operation. The key schedule shares the same operations, but replaces `AddRoundKey` by `AddRoundConstants` (AC), which XORs the round constants to the first row of the key state before the result of AC is used as the round key that is added to the state. For more details, we refer the readers to the design paper [9].

Remark 1. Given that the *transposition between row and column* has no impact on attack results, for convenience, we use `ShiftRows` and `MixColumns` instead of `ShiftColumns` and `MixRows` in the rest of paper for `Whirlpool` hereafter. Thus, the states will be *transposed* to correspond with the states of `Whirlpool`.

Remark 2. In the remainder, we will denote a state by the operation that it is used as the direct input for, and will superscript the round index. For example, $\#\mathrm{SB}^i$ denotes the state before the SB operation in Round i, as is shown in Figs. 2 and 3.

Remark 3. The Russian national standard `Streebog` follows a similar structure as `Whirlpool`, due to the space limit, we provide the specification of `Streebog` in the full version [12, Appendix E].

3 Advanced Techniques in MITM Attacks

We advance the existing automated MITM frameworks with three generic techniques. Here, we detail the ideas and integration into the augmented framework.

3.1 S-Box Linearization (LIN)

In MITM attacks, we have two sets of states $V^+ = (v_0^+, \ldots, v_{2^{d_b}-1}^+)$ and $V^- = (v_0^-, \ldots, v_{2^{d_r}-1}^-)$ propagating through the cipher. The superposition structure allows any cell of any state $v_{i,j}$ to be represented as the sum of its forward and backward neutral components: $v_{i,j} = v_i^+ \oplus v_j^-$, such that the components v_i^+ and v_j^- can be propagated independently through linear operations. Though, the nonlinear operations, *i.e.*, an S-box S in AES-like ciphers, prevent such trivial linear combinations.

Earlier works in the series of automated MITM attacks on AES-like ciphers had to define propagation rules which either lost knowledge about the cell after the S-box or which consumed one byte degree of freedom for forcing at least one neutral value to be constant before and after the nonlinear operation. We can linearize certain S-boxes partially or fully by restricting the input space or by guessing a hint from a set that is smaller than the input space.

In this work, we consider full linearization with a hint. Thus, we aim at finding a decomposition of S, more precisely, functions $F, G, H : \mathbb{F}_2^b \times \mathbb{F}_2^b \to \mathbb{F}_2^b$ with F and G being linear over $\mathbb{F}_2$ such that

$$S(v^+ \oplus v^-) = F(v^+, H(v^+, v^-)) \oplus G(v^-, H(v^+, v^-))\,.$$

The range of H is the set of space of hints. Assuming balancedness of H, *i.e.*, $d_{\mathcal{L}} = \dim(\mathsf{range}(H))$, then $2^{d_{\mathcal{L}}}$ elements have to be guessed at most to linearize S, which is beneficial if we find such a function H with $d_{\mathcal{L}} < b$. Then, we add a complexity term of $2^{d_{\mathcal{L}}}$ to the attack for guessing the hint for each combination of (v_i^+, v_j^-) but can propagate a superposition through the S-box.

The S-box of the AES is given by $\mathsf{S}(v) = \mathbf{A} \cdot v^{254} \oplus \texttt{0x63}$ for a fixed $\mathbf{A} \in \mathbb{F}_2^{8\times 8}$. The power map and the XOR is in the field $\mathbb{F}_{2^8}$ with a fixed irreducible polynomial; only the affine layer $\mathbf{A}$ is not defined over this field. At Asiacrypt 2023, Zhang *et al.* [42] observed that one can decompose 254 into $17 \cdot 14 + 16$ and obtain

$$\begin{aligned}(v^+ + v^-)^{254} &= ((v^+ + v^-)^{17})^{14} \cdot (v^+ + v^-)^{16} \\ &= (H(v^+, v^-))^{14} \cdot ((v^+)^{16} + (v^-)^{16})\,,\end{aligned}$$

where the last equality holds since exponentiation with any power of 2 is linear over $\mathbb{F}_2$. Then according to the following Theorem 1, the hint $H(v^+, v^-)$ can take $|\mathsf{range}(H)| = |\{v^{17} : v \in \mathbb{F}_{2^8}\}| = 16$ values (including the zero element). Thus, one can linearize the AES S-box by guessing at most 16 candidates.

Theorem 1. *Let d be a divisor of $|\mathbb{F}_q^*| = p^n - 1$ where $q = p^n$, and let $X = \{x^d : x \in \mathbb{F}_q^*\}$. The size of X is $|X| = |\mathbb{F}_q^*|/d = (p^n - 1)/d$.*

3.2 Distributed Initial Structures (DIS)

It has been a common approach in MILP-based MITM models to select two independent initial states: $S^{\texttt{ENC}}$ in the encryption function and $S^{\texttt{KSA}}$ in the key

schedule, as the generation of round keys is independent of the encryption in most designs. In this work, we generalize the selection of initial states.

In essence, the initial states in MITM attacks are composed of some intermediate bytes in the compression function where we distribute initial DoFs for forward and backward computations. There should be no limitations on where the initial DoF is located, as long as the values of those states can be chosen independently of each other in the actual attack. In other words, the initial DoFs can be distributed to several scattered intermediate states, rather than rigidly selecting two full states in encryption and key schedule, respectively. Previous models implicitly limited the key bytes to depend only on $S^{\texttt{KSA}}$ and not on $S^{\texttt{ENC}}$, and consequently, shrunk the solution pool.

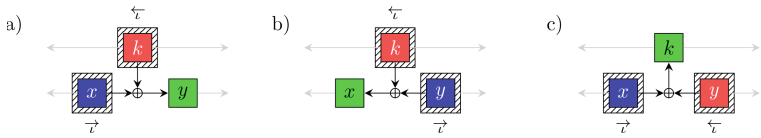

Fig. 4. Conceptual strategies of distributing initial states in two states and a key byte.

Consider an intuitive toy example in Fig. 4. Assume x and y denotes bytes in the encryption and k denote a byte in the key, and we distribute initial DoF in this system for forward and backward computation. The whole system has a total initial DoF of 2. We use $\overrightarrow{\iota}$ and $\overleftarrow{\iota}$ to denote the initial DoF for forward and backward respectively, and the color green ■ to denote a byte in superposition. Without loss of generality, there are three possible scenarios as depicted in Fig. 4. The previous models covered the first two cases while excluding the third one, wherein the key byte is dependent on the initial DoF from the encryption.

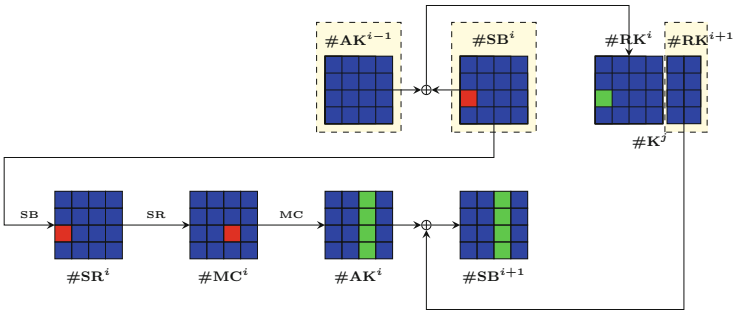

Fig. 5. Example of a distributed initial structure on `AES-192`.

We extend the above insight to MITM attacks by the introduction of **DIS**. We now distribute the initial DoF to several intermediate states in the compression function, provided that the bytes are independent and there is sufficient

information to define all the round keys and all the intermediate encryption states. For example, Fig. 5 describes an example to distribute initial DoF in AES-192. We will distribute initial DoFs in $\#\mathrm{AK}^i$, $\#\mathrm{SB}^{i+1}$ and the rightmost two columns of $\#\mathrm{K}^j$. In this way, we have straightforwardly defined a full intermediate encryption state $\#\mathrm{SB}^{i+1}$, and we can squeeze out a full $\#\mathrm{K}^j$ for key schedule propagations.

The technique is useful in AES, since the key schedule has relatively low confusion and more linear relations can be preserved. In this work, we realize **DIS** in a heuristic manner. We still select $S^{\mathtt{ENC}}$ and $S^{\mathtt{KSA}}$ respectively to search for attack configurations, but make the exception that superposition bytes are now allowed in $S^{\mathtt{KSA}}$ and remain refrained from $S^{\mathtt{ENC}}$. When a configuration is obtained, we check if the initial states can be equivalently chosen to properly define the superposition bytes in $S^{\mathtt{KSA}}$ within the maximum available DoF of the target cipher. Our realization of **DIS**, though heuristic, is more tailored to the key schedule and helps extend the analysis of AES-192 by one round.

3.3 Structural Similarities (SIM)

The models by Bao *et al.* and Dong *et al.* could find longer attacks than the manual attacks *e.g.*, by Sasaki [35] since the former effectively used the degree of freedom in key space to obtain reductions in the encryption state. From the XOR of the state with a round key, state bytes in superposition could become single-colored, and forward or backward neutral bytes could become constants. Such concessions are useful and often necessary before and after non-linear operations so that the knowledge of a byte can be propagated further. However, they come at the price of consuming a degree of freedom from the possible solution space.

In previous works, the effects of multiple round-key additions on the state have been usually modeled to be *independent* from each other and the state values. However, constraints from some consecutive rounds may stem from the same source of the neutral words in the key and state, *e.g.*, as depicted in Fig. 6, constraints in states Y^{r_0} and Y^{r_1} are set on the same neutral words in states X^r and $K^{r'}$. Thus, tracing constraints back to such shared sources may enlarge the search space by avoiding duplicate DoF consumption. However, modeling all such dependent constraint relations can become challenging since the relations between all state and key bytes would have to be considered, which can render models infeasible to compute. Nevertheless, we can efficiently model certain special cases, and consider here the *structural similarities* of encryption and key schedule, *e.g.*, Whirlpool and Streebog use almost the same functions for updating key and message, differing only in the usage of round constants. Considering previous best MITM attacks on Whirlpool [8], Bao *et al.* already observed such dependency between encryption and key schedule, however, in their 7-round attacks on Whirlpool, they only used it in a post-processing step to generate the solution space of neutral words. In this work, we include this structural similarity explicitly in our MILP models.

General Concept. For SPNs, we can write the round function as a composition of a nonlinear S-box layer SB, an affine layer **A**, and a key addition. Assume, the

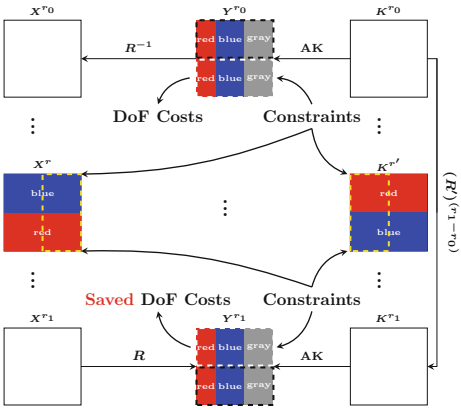

Fig. 6. High-level view of the different constraints connected by key schedule.

key schedule employs the same S-box layer SB, an affine layer $\mathbf{A}'$, and a constant addition. Consider an interval of rounds from i to $i+1$ and assume that we can split the key $\#\mathrm{K}^i$ and the state in the encryption $\#\mathrm{AK}^i$ into an active part, subscripted by a, and a constant part subscripted by c, each: $\#\mathrm{K}^i = \#\mathrm{K}^i_a \| \#\mathrm{K}^i_c$ and $\#\mathrm{AK}^i = \#\mathrm{AK}^i_a \| \#\mathrm{AK}^i_c$. Figure 7 illustrates this setting. If the active parts of the key and encryption state are equal before the S-box layer, then the same values will also be the results in both the encryption and key schedule:

$$\#\mathrm{SB}^{i+1}_a = \#\mathrm{K}^{i+1}_a \Leftrightarrow \#A^{i+1}_a = \#{A'}^{i+1}_a .$$

If $\#A^{i+1}_a$ and $\#{A'}^{i+1}_a$ are mapped to the same positions of the state after $\mathbf{A}$ and $\mathbf{A}'$, respectively, then the nonlinear contributions will cancel. Thus, we can define a nonlinear function G and a linear function H such that the active part of the message after the round is given by

$$\#\mathrm{SB}^{i+1}_a = G(\#\mathrm{K}^i_a) \oplus H(\#\mathrm{K}^i_c, \#\mathrm{K}^i_a, \#\mathrm{AK}^i_c) .$$

Note that it does not depend on $\#\mathrm{AK}^i_a$.

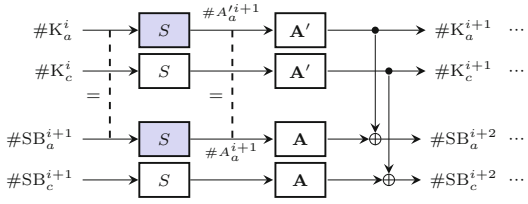

Fig. 7. Exploiting similar operations in the encryption and key schedule.

We can generalize this observation to multiple rounds if the similarities between key schedule and message schedule allow. Given that the diffusion of

AES-like ciphers is usually strong, *i.e.*, an MDS matrix, the number of subsequent rounds where this approach can be employed seems limited to two or three rounds, depending on the round constants and the linear layers. However, it will enlarge the search space and lead to high concentrations of constraints around the starting points, which could help reduce the memory complexity of the attack, as we will demonstrate later in our application results.

4 Enhanced Attack Framework and MILP Model

This section demonstrates our enhanced attack framework, that we call the exceptional MITM framework, and the equipped MILP-based search model.

4.1 Exceptional MITM Framework

We append the following two new notations to Table 2 to reflect the use of **LIN** (Table 3):

Table 3. Additional notations.

$d_{\mathcal{L}}$	DoF consumed by S-Box linearizations
H	Space or set of hints from linearized S-boxes, with $\|H\| = 2^{d_{\mathcal{L}}}$

Again, without loss of generality, we assume $d_{\mathcal{B}} + g_{\mathcal{B}} \leq d_{\mathcal{R}} + g_{\mathcal{R}}$. The exceptional MITM attack framework is formulated as follows:

1. Assign arbitrary values to the constants in pre-defined constraints.
2. Compute V^+ and V^- based on the constants.
3. For all tuples $(v^+, g^+, g, h^+) \in V^+ \times G^+ \times G \times H$, compute to E^+, obtain m^+ for matching, store (v^+, g^+) in $T^+[m^+, g, h^+]$.
4. For all tuples $(v^-, g^-, g, h^-) \in V^- \times G^- \times G \times H$, compute to E^-, obtain m^- for matching.
5. For all (v^+, g^+) in $T^+[m^-, g, h^-]$ and (v^+, v^-) consistent with h^-, compute to check if (g^+, g^-, g) is compatible with (v^+, v^-).
6. For compatible (v^+, v^-), check for a full match.
7. If a full match is discovered, compute and return the preimage. Otherwise, revert to step 1, change the arbitrary values, and repeat the rest.

The computational complexity of the above attack is evaluated as follows:

$$\begin{aligned} & 2^{n-(d_{\mathcal{B}}+d_{\mathcal{R}})} \cdot \left(2^{d_{\mathcal{B}}+g_{\mathcal{B}}+g_{\mathcal{B}\mathcal{R}}+d_{\mathcal{L}}} + 2^{d_{\mathcal{R}}+g_{\mathcal{R}}+g_{\mathcal{B}\mathcal{R}}+d_{\mathcal{L}}} + 2^{d_{\mathcal{B}}+g_{\mathcal{B}}+d_{\mathcal{R}}+g_{\mathcal{R}}+g_{\mathcal{B}\mathcal{R}}-d_{\mathcal{M}}}\right) \\ \simeq\ & 2^{n-min(d_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{B}\mathcal{R}}-d_{\mathcal{L}}, d_{\mathcal{R}}-g_{\mathcal{B}}-g_{\mathcal{B}\mathcal{R}}-d_{\mathcal{L}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{B}\mathcal{R}})}. \end{aligned} \tag{3}$$

4.2 MILP-Based Search Model

Now we introduce an enhanced MILP model to integrate proposed techniques, including new coloring schemes and corresponding propagation rules in detail.

Color-Encoding Scheme of Neutral Words. We encode different byte types in the superposition structure with three binary variables b, r, and w:.

- A blue cell ■ denotes a forward neutral byte, encoded as $(b, r, w) = (1, 0, 0)$.
- A red cell ■ denotes a backward neutral byte, encoded as $(b, r, w) = (0, 1, 0)$.
- A white cell □ denotes an arbitrary byte, encoded as $(b, r, w) = (0, 0, 1)$.
- A gray cell ■ denotes a constant byte, encoded as $(b, r, w) = (0, 0, 0)$.
- A green cell ■ denotes a superposition byte, encoded as $(b, r, w) = (1, 1, 0)$.

In the encoding system, $b = 1$ and $r = 1$ denote that a byte contains forward and backward neutral information, respectively. And $w = 1$ uniquely identifies an arbitrary byte, whose value is unknown, against other byte types. Such construction has high clarity and interpretability, rather than a simple enumeration of the five possible byte types in the superposition structure , and allows a more straightforward realization of propagation rules and efficient counting of DoF. For example, we can easily obtain the initial DoFs $|\mathcal{B}^{\texttt{ENC}}|$, $|\mathcal{R}^{\texttt{ENC}}|$, $|\mathcal{B}^{\texttt{KSA}}|$, and $|\mathcal{R}^{\texttt{KSA}}|$ by simply summing up b, r encoders in corresponding states.

Our model achieves high efficiencies and makes it possible for better attacks on designs with large state sizes. The new model can formulate attacks on `Whirlpool` and `Streebog` in full-sized versions (8×8), while Bao *et al.* 's attack on `Whirlpool` is limited to 4×4 versions with symmetry patterns [8]. Specifically, our improved MITM attack configurations for `Whirlpool` can be found within 200 s, and the optimization of MITM collision attack models of 6-round `Whirlpool` can be finished within just 300 s[1].

In the rest of this chapter, notions b_α, r_α, and w_α are used to represent the encoders of a byte α.

Propagations Through `SubBytes`. We formulate the SB-rule, a byte-wise propagation rule for `SubBytes`, with **LIN** integrated.

- When the input byte is not green, the color of the output byte is identical to that of the input byte.
- When the input byte is green, the output byte is either white by default or green with one cost of linearization ($d_{\mathcal{L}}$ incremented by one).

Modeling the SB-rule requires both encoders of the input and output byte as well as one additional encoder to indicate linearization cost. The rule can be converted to MILP constraints with the convex-hull method [41].

Propagations Through `MixColumns`. The `MixColumns` operation takes a column as input and outputs a column. Assume that the input is a mix of n_b blue

[1] We ran our MILP models with Gurobi 9.5.2 on a desktop computer with 3.6 GHz Intel Core i9 and 16GB 2667 MHz DDR4.

bytes, n_r red bytes, n_c gray bytes, n_g green bytes, and n_w white bytes, the basic rule for the `MixColumns` operation, MC-rule in short, is formulated as follows:

- When $n_w > 0$, the output contains only white bytes.
- When $n_w = 0$ and $n_b + n_r + n_g = 0$, the output contains only gray bytes.
- When $n_w = n_r = 0$ and $n_b + n_g > 0$, the output contains n'_b blue bytes and n'_c gray bytes, with n'_c consumed DoF from the forward chunk.
- When $n_w = n_b = 0$ and $n_r + n_g > 0$, the output contains n'_r red bytes and n'_c gray bytes, with n'_c consumed DoF from the backward chunk.
- Otherwise, the output is a mix of n'_b blue bytes, n'_r red bytes, n'_c gray bytes, and n'_g green bytes, with $n'_b + n'_c$ consumed DoF from the backward chunk and $n'_r + n'_c$ consumed DoF from the forward chunk.

We use α to denote a byte in the input column and β in the output. To realize the above functionality, we introduce three column-wise encoders Eb, Er, and Ew, which is constructed based on the encoding of input bytes:

$$Eb = \max_{\alpha} b_\alpha,\ Er = \max_{\alpha} r_\alpha,\ \text{and}\ Ew = \max_{\alpha} w_\alpha\,. \tag{4}$$

Let further $\overrightarrow{\sigma_\beta}$ and $\overleftarrow{\sigma_\beta}$ be binary variables that respectively track the DoF consumption at byte β (in the output column) for the forward and backward chunk. Then, the MC-rule can be formulated as:

$$\begin{cases} \sum_\beta w_\beta = \texttt{NROW} \cdot Ew \\ \sum_\alpha b_\alpha + \sum_\beta b_\beta = \texttt{NROW} \cdot (Eb - Ew) \\ \sum_\alpha r_\alpha + \sum_\beta r_\beta = \texttt{NROW} \cdot (Er - Ew) \\ \texttt{NROW} \cdot (Eb - Ew) \le \sum_\beta b_\beta + \sum_\beta \overrightarrow{\sigma_\beta} \le \texttt{NROW} \cdot \min(Eb, 1 - Ew) \\ \texttt{NROW} \cdot (Er - Ew) \le \sum_\beta r_\beta + \sum_\beta \overleftarrow{\sigma_\beta} \le \texttt{NROW} \cdot \min(Er, 1 - Ew) \end{cases}. \tag{5}$$

Integrating Guess-and-Determine Into `MixColumns`. We introduce a lightweight realization of GnD by integrating its functionality into MC-rule, which is named GnD-MC-rule. We introduce four GnD encoders for an input byte α, g^w_α, g^b_α, g^r_α, and g^{br}_α, which satisfy:

$$w_\alpha = g^w_\alpha + g^b_\alpha + g^r_\alpha + g^{br}_\alpha\,. \tag{6}$$

The simple constraint ensures that, when an input byte α is non-white, all GnD encoders are 0, meaning no GnD is incurred. Otherwise, when α is white, exactly one GnD encoder equals to 1 with the following meaning:

- $g^w_\alpha = 1$: GnD is not activated and byte α remains unknown,
- $g^b_\alpha = 1$: α is guessed as blue for forward propagation,
- $g^r_\alpha = 1$: α is guessed as red for backward propagation,
- $g^{br}_\alpha = 1$: α is guessed as green for both forward and backward propagations.

The GnD-MC-rule is formulated based on MC-rule by a simple tweak on the construction of the column-wise encoders:

$$Eb' = \max_{\alpha}\{b_\alpha, g^b_\alpha, g^{br}_\alpha\},\ Er' = \max_{\alpha}\{r_\alpha, g^r_\alpha, g^{br}_\alpha\},\ \text{and}\ Ew' = \max_{\alpha} g^w_\alpha. \quad (7)$$

We count the guessed DoF by summing up the GnD encoders. Moreover, GnD can be turned off easily for efficiency by adding a simple constraint $w_\alpha = g^w_\alpha$.

`XOR` with Two Inputs. The XOR-rule models the propagation of variables through a simple `XOR` operation with two inputs:

- When the input involves a white byte, the output is white.
- When the input contains only gray bytes, the output is gray.
- When the input contains only blue bytes, the output is either blue with no consumption of DoF or gray consuming one DoF from the forward chunk.
- When the input contains only red bytes, the output is either red with no DoF consumption or gray consuming one DoF from the backward chunk.
- When the input is a mixture of red and blue bytes or involves green bytes, the output is green.

Generating the constraints to account for the XOR-rule in MILP is well understood and therefore omitted here.

`XOR` with Multiple Inputs. In addition to sequentially deriving the round keys following Eq. (2), we propose a new approach to model the `AES` key schedule. We find the expression of intermediate bytes in terms of bytes in `KSA` by invoking a sourcing function, which is designed to recursively obtain the parents of an intermediate byte and cancels whenever a byte is XORed an even number of times. Then we propose the n-XOR-rule to determine the coloring of an intermediate byte and the consumed DoF, detailed as follows:

- When the input involves a white byte, the output is white.
- When the input contains only gray bytes, the output is gray.
- When the input contains only blue bytes, the output is either blue with no DoF consumed or gray with 1 DoF consumed from the forward chunk.
- When the input contains only red bytes, the output is either red with no DoF consumed or gray with 1 DoF consumed from the backward chunk.
- When the input contains both red and blue bytes, the output is one of the following:
  - green, with no consumption of DoF,
  - blue, with 1 DoF consumed from the backward chunk,
  - red, with 1 DoF consumed from the forward chunk, or
  - gray, with 1 DoF consumed from each forward and backward chunk.

We denote a byte in the expression of an intermediate byte as γ and introduce three encoders Pb, Pr, and Pw for the intermediate byte that satisfy:

$$Pb = \max_{\gamma} b_\gamma,\ Pr = \max_{\gamma} r_\gamma,\ \text{and}\ Pw = \max_{\gamma} w_\gamma. \quad (8)$$

The constraints for the n-input XOR rule can be obtained by using the convex-hull method on Pb, Pr, Pw, the encoders of the intermediate byte, and two encoders for DoF costs.

Matching. We deploy two types of matching in our attacks: the XOR-match and the MC-match. The XOR-match is used at the feed-forward that checks E^+, E^-, and $\#\mathrm{RK}^{-1}$ byte by byte. If $w_\alpha = 0$ holds for position α in E^+, E^-, and $\#\mathrm{RK}^{-1}$, then $m_\alpha = 1$. Otherwise, $m_\alpha = 1$. Then, $d_{\mathcal{M}}^{\texttt{XOR}}$ results from:

$$d_{\mathcal{M}}^{\texttt{XOR}} = \sum_{\alpha} m_\alpha \,. \tag{9}$$

Besides, the MC-match takes the input and output of a `MixColumns` operation as E^+ and E^- and counts the cumulative non-white bytes at a common column index. Let Δ be a column index and denote the cumulative non-white bytes in E^+_Δ and E^-_Δ as t_Δ. If there exist $t_\Delta >$ `NROW`, then we have a $t_\Delta -$ `NROW` degrees for matching at column Δ. Otherwise, there are no degrees of matching at column Δ. Then $d_{\mathcal{M}}^{\texttt{MC}}$ is given by the sum over all columns:

$$d_{\mathcal{M}}^{\texttt{MC}} = \sum_{\Delta} \max(0, t_\Delta - \texttt{NROW}) \,. \tag{10}$$

Objective Function. Our search model aims to maximize:

$$\min\{d_{\mathcal{B}} - g_{\mathcal{R}} - g_{\mathcal{BR}} - d_{\mathcal{L}}, d_{\mathcal{R}} - g_{\mathcal{B}} - g_{\mathcal{BR}} - d_{\mathcal{L}}, d_{\mathcal{M}} - g_{\mathcal{B}} - g_{\mathcal{R}} - g_{\mathcal{BR}}\} \tag{11}$$

According to Eq. (3), $\min\{\overrightarrow{d_b}, \overleftarrow{d_r}, \overrightarrow{m}^{\leftarrow}\}$ determines the complexity of an MITM attack. Thus, the search for the optimal MITM attack pattern of given *config* is converted to a maximization problem on objective $\tau_{\texttt{Obj}}$:

5 Applications to AES and Whirlpool

In this section, we briefly describe the *first* 10-round MITM pseudo-preimage attack on the compression function of `AES-192` and the *memoryless* 5-round MITM pseudo-preimage attack on the compression function of `Whirlpool`.

5.1 First MITM Pseudo-preimage Attack on 10-Round `AES-192`

The previous best MITM pseudo-preimage attack on `AES-192` reaches 9 rounds [8]. Adopting **LIN** and **DIS**, we obtain the first MITM pseudo-preimage attack on 10-round `AES-192`, which is provided in Fig. 8 and summarized below:

- Initial DoF for forward neutral words $\overrightarrow{\iota}$ (■): 39 bytes (16 in $\#\mathrm{AK}^5$, 15 bytes in $\#\mathrm{SB}^6$, and 8 bytes $\#\mathrm{K}^4[16\ldots23]$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (■): 1 byte ($\#\mathrm{SB}^6[2]$);
- Consumed DoF in forward computation $\overrightarrow{\sigma}$: 38 bytes;
- Consumed DoF in backward computation $\overleftarrow{\sigma}$: zero bytes;
- Guessed bytes for blue, red, and both colors $g_{\mathcal{B}}$, $g_{\mathcal{R}}$, $g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{BR}} = 0$ bytes and $g_{\mathcal{B}} = 0$ bytes;
- Guessed byte equivalents for linearization: $d_{\mathcal{L}} = 0.5$ bytes.
- Matching DoF $d_{\mathcal{M}}$: 1 byte between $\#\mathrm{AT}^9$ and $\#\mathrm{SB}^0$.
- Remaining DoF: $d_{\mathcal{B}} - d_{\mathcal{L}} = 1 - 0.5 = 0.5$ and $d_{\mathcal{R}} - d_{\mathcal{L}} = 1 - 0.5 = 0.5$.

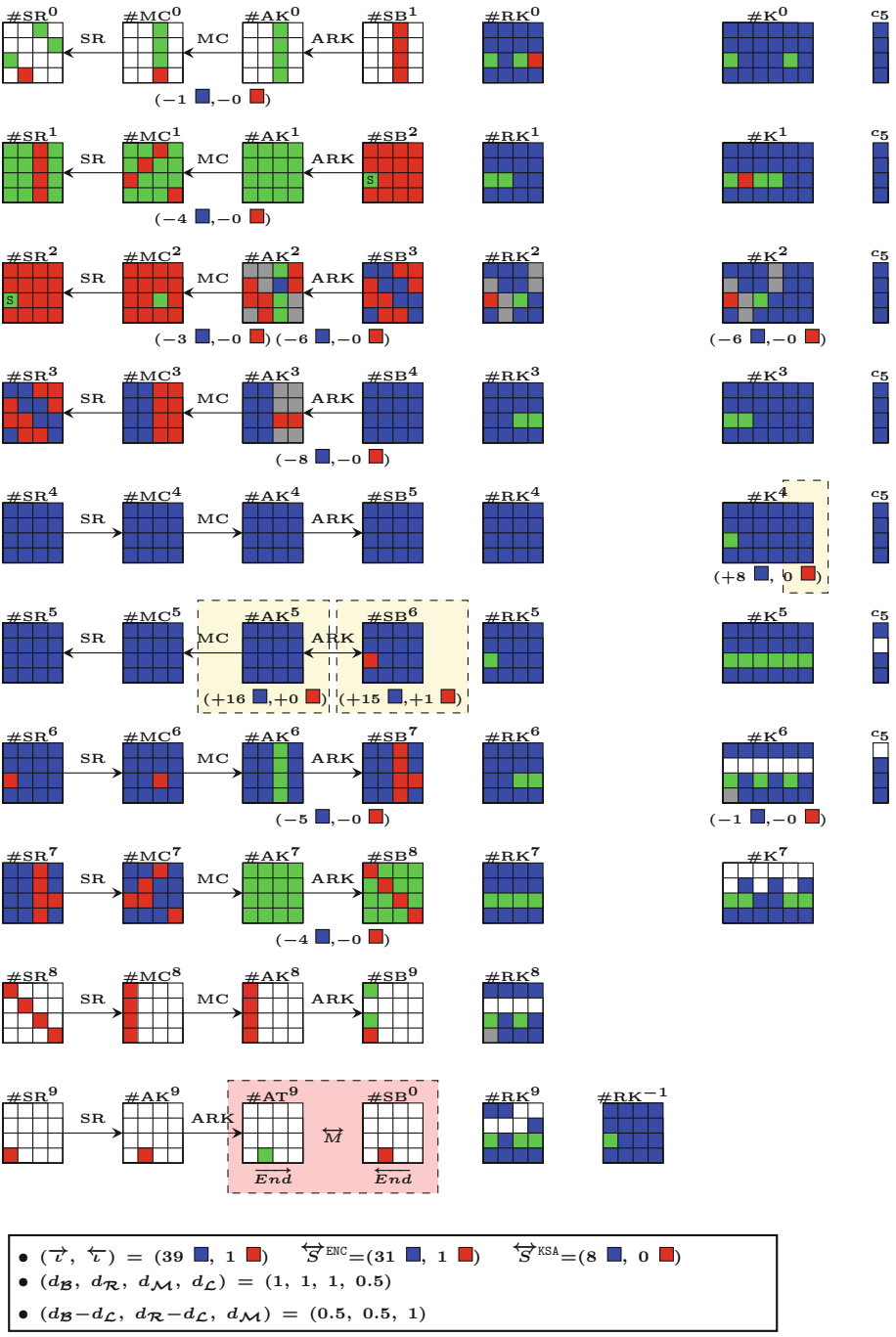

Fig. 8. An MITM pseudo-preimage attack of 10-round AES-192.

Algorithm 1: Computing forward neutral words (blue) for 10-round AES-192.

Initialize a table $T_{\text{blue}}^{\text{neutral}}$;
Fix 6 bytes $\#\text{K}^2[1,2,6,7,12,13]$, 6 ■ cells in $\#\text{AK}^2$, 3 bytes $\#\text{MC}^2[8,9,11]$, 2 bytes $\#\text{MC}^1[8,15]$ and $\#\text{AK}^3[8..15]$ be all zero.
for *4* ■ *blue cells* $\#\text{K}^4[4,5,6,7] \in (\mathbb{F}_2^8)^4$ **do**
  Use 4 constants $\#\text{K}^2[6,7,12,13]$ to derive $\#\text{K}^4[12,13], \#\text{K}^3[20,23]$;
  for *3* ■ *blue cells* $\#\text{K}^3[18,19,22] \in (\mathbb{F}_2^8)^3$ **do**
    Use 2 constant $\#\text{K}^2[1,2]$ to derive $\#\text{K}^4[1,2]$;
    for *6* ■ *blue cells* $\#\text{K}^4[8,9,10,11,14,15] \in (\mathbb{F}_2^8)^6$ **do**
      Derive $\#\text{RK}^2[14,15]$ from $\#\text{K}^4[6,7,14,15]$;
      Use 1 constant $\#\text{MC}^1[15]$ to derive $\#\text{K}^3[21]$ from $\#\text{K}^1[20..23] = \#\text{K}^4[12..15] \oplus \#\text{K}^4[8..11] \oplus \#\text{K}^3[20..23]$ and $\text{MC}^{-1} \cdot \#\text{K}^1[20..23] = (*,*,*,0)$;
      Derive $\#\text{RK}^2[4,5]$ from $\#\text{K}^4[4,5], \#\text{K}^3[21,22]$;
      Use 4 constant $\#\text{AK}^2[4,5,14,15]$ and $\#\text{RK}^2[4,5,14,15]$ to derive $\#\text{SB}^3[4,5,14,15]$;
      for *2* ■ *blue cells* $\#\text{K}^4[3], \#\text{K}^3[16] \in (\mathbb{F}_2^8)^2$ **do**
        Derive $\#\text{RK}^2[3]$ from $\#\text{K}^3[16,20], \#\text{K}^4[3]$;
        Use 1 constant $\#\text{AK}^2[3]$ to derive $\#\text{SB}^3[3]$;
        Use 1 constant $\#\text{MC}^1[8]$ to derive $\#\text{K}^3[17]$ from $\#\text{K}^1[16..19] = \#\text{K}^3[16..19] \oplus \#\text{K}^4[4..7] \oplus \#\text{K}^4[8..11]$ and $\text{MC}^{-1} \cdot \#\text{K}^1[16..19] = (0,*,*,*)$;
        Derive $\#\text{K}^4[16..23]$;
        Use 4 constants $\#\text{AK}^2[0], \#\text{MC}^2[8,9,11]$ to derive $\#\text{K}^4[0], \#\text{SB}^3[0,9,10]$ by solving 4 linear Equation (12);
        `// Now we know all` ■ `blue cells in` $\#\text{SB}^3$ `and` $\#\text{K}^4$
        Derive the blue part of $\#\text{SR}^2[2]$;
        `// Linearization of S-boxes`
        for $2^{8\times 0.5}$ *values of* $c_0 = (\#\text{SR}^2[2])^{17}$ **do**
          Compute 14 constants $(\#\text{MC}^0[11], \#\text{MC}^1[2,5], \#\text{SB}^6[2], \#\text{SB}^7[8..11,14], \#\text{SB}^8[0,5,10,15], \#\text{K}^6[3]) = (c_1, \ldots, c_{14})$;
          Update the table $T_{\text{blue}}^{\text{neutral}}[c_0, \ldots, c_{14}] \stackrel{+}{=}$ (■ in $\#\text{K}^4$ and $\#\text{SB}^3$);
          `// For each value of` $(c_0, \ldots, c_{14})$`,` $2^{8\times(15.5-14.5)} = 2^{8\times 1}$ `candidates expected`
return $T_{\text{blue}}^{\text{neutral}}$;

Compute Initial Values Forward Neutral Bytes (Blue). Note that the value of a byte in this phase represents the value computed from the blue parts. For example, fixing $\#\text{K}^2[2]$ to zero means that the blue initial bytes have a zero impact on this byte. To get the initial values of the blue neutral bytes, the following constraints among states $\#\text{SB}^3, \#\text{K}^4, \#\text{K}^3$ will be enforced.

$$\begin{cases} \#\mathrm{SB}^3[0] \oplus \#\mathrm{K}^4[0] \oplus \mathsf{S}(\#\mathrm{K}^3[21]) \oplus \mathsf{S}(\#\mathrm{K}^3[17] \oplus \#\mathrm{K}^3[21]) = \#\mathrm{AK}^2[0] \\ \mathrm{MC}^{-1} \cdot \begin{pmatrix} \#\mathrm{K}^4[0] \oplus \#\mathrm{K}^4[8] \\ \#\mathrm{SB}^3[9] \oplus \#\mathrm{K}^4[1] \oplus \#\mathrm{K}^4[9] \\ \#\mathrm{SB}^3[10] \oplus \#\mathrm{K}^4[2] \oplus \#\mathrm{K}^4[10] \\ \#\mathrm{K}^4[3] \oplus \#\mathrm{K}^4[11] \end{pmatrix} = \begin{pmatrix} \#\mathrm{MC}^2[8] \\ \#\mathrm{MC}^2[9] \\ * \\ \#\mathrm{MC}^2[11] \end{pmatrix} \end{cases}, \quad (12)$$

where $\#\mathrm{AK}^2[0]$ and $\#\mathrm{MC}^2[8, 9, 11]$ are fixed as zeroes. Algorithm 1 generates the solution space of blue neutral words. The time complexity is upper bounded by $2^{8\times 15.5} = 2^{124}$ operations.

The MITM Attack Procedure for 10*-Round* `AES-192`*.* For backward neutral words (■ cells), we will iterate over $\#\mathrm{SB}^6[2]$. We provide the main attack procedure derived from Fig. 8 in Algorithm 2.

The Attack Complexity. The time complexity of the above MITM pseudo-preimage attack of 10-round `AES-192` is about $2^{128-8\times\min(0.5,0.5,1)} = 2^{124}$. The table T^+ dominates the memory complexity with $2^{8\times 1.5} \approx 2^{12}$. The precomputation table $T_{\text{blue}}^{\text{neutral}}$ dominates the memory complexity with $2^{8\times 15.5} = 2^{124}$.

5.2 Improved MITM Preimage Attacks of `Whirlpool`

We use the **SIM** technique to search for attack configurations of `Whirlpool`, improved MITM (pseudo-)preimage attacks are obtained for both 5- and 6-round `Whirlpool`. In particular, we could find an attack on 5-round `Whirlpool` with $O(1)$ memory and present the first MITM attack on 7.75-round `Whirlpool`, including the SB, SR, and MC operations in the last round. The previous best (pseudo-)preimage attacks are the 7-round MITM attacks on `Whirlpool` presented by Bao *et al.* [8] at Crypto 2022.

Improved and Memoryless Preimage Attack of 5**-Round** `Whirlpool`**.** We directly perform search on the full-size 8×8 version for `Whirlpool`, and our improved search result for 5-round `Whirlpool` is given in Fig. 9. Compared to the previous best result of 5-round `Whirlpool` found by Bao *et al.* [8, Figure 13], GnD is still not required but BiDir is utilized (48 ■ cost at Round 0). However, the DoF cost for neutral words is more concentrated at the starting point, *e.g.*, 48 red cells canceled at Round 0 (48 bytes DoF cost will be compensated at Round 1), and another 24 red cells will be canceled at the MC operation for $\#\mathrm{KMC}^2$, which makes it more efficient to generate the red neutral words and can improve our attack on 5-round `Whirlpool` further to *memoryless* when combined with the *same color match*. We elaborate on the attack configuration below.

- Initial DoF for forward neutral words $\overrightarrow{\iota}$ (■): 24 bytes (8 blue cells in $\#\mathrm{SB}^1$ and 16 blue cells in $\#\mathrm{KK}^0$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (■): 48 bytes (48 red cells in $\#\mathrm{KK}^0$ are *set to be equal to the corresponding cells* in $\#\mathrm{SB}^1$, then all red bytes can

Algorithm 2: MITM attack on 10 rounds of the `AES-192` compression function.

for $(c_1, \ldots, c_{14}) \in (\mathbb{F}_2^8)^{14}$ **do**
  Initialize T^+;
  `// For blue neutral words`
  for *the* $2^{8\times 0.5}$ *values* c_0 *of the hint pool* **do**
    Lookup the table $T^{\text{neutral}}_{\text{blue}}[c_0, \cdots, c_{14}]$ to get candidates of ■ blue cells in $\#\text{K}^4$ and $\#\text{SB}^3$;
    for *the values of* $2^{8\times 1}$ *in* $\#\text{K}^4$ *and* $\#\text{SB}^3$ **do**
      Derive m^+ (the blue part of $\#\text{AT}^9[7] = \#\text{RK}^9[7] \oplus \#\text{RK}^{-1}[7]$) and update the table $T^+[m^+, c_0] \stackrel{+}{=}$ [blue neutral bytes];
      `//` $2^{8\times 0}$ `entries for each index in` T^+
  `// For red neutral words`
  for *the* $2^{8\times 0.5}$ *values of* c_0 **do**
    for $\#\text{SB}^6[2] \in \mathbb{F}_2^8$ **do**
      Compute m^- (the red part of $\#\text{AT}^9[7] \oplus \#\text{SB}^0[7]$), which equals the blue part of $\#\text{AT}^9[7]$;
      Check for entries in $T^+[m^-, c_0]$ to derive 1 blue neutral byte;
      Check if the red and blue parts of $\#\text{SR}^2[2]$ produce c_0;
      `//` $2^{8\times(14+0.5+1+1-1-0.5)} = 2^{8\times 15}$ `candidates expected`
      Compute the full $\#\text{AT}^9$ and $\#\text{SB}^0$ in both colors to check the remaining 15 cells;
      if *the full match is found* **then**
        `//` $2^{8\times(15-15)} = 1$ `candidate expected`
        Output the preimage and stop;

be canceled to constants marked as zero constant[2] ▣ in $\#\text{AK}^0$ and $\#\text{AK}^1$, with just 48 bytes DoF cost for XOR operation);
- Consumed DoF for forward $\overrightarrow{\sigma}$: zero;
- Consumed DoF for backward $\overleftarrow{\sigma}$: 24 bytes for $\#\text{KMC}^2 \xrightarrow{\text{MC}} \#\text{KK}^2$;
- Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: all zero byte;
- Matching DoF $d_{\mathcal{M}}$: 24 bytes between $\#\text{MC}^2$ and $\#\text{AK}^2$.

Then, the remaining DoF for the MITM attack is $d_{\mathcal{B}} = 24, d_{\mathcal{R}} = 24, d_{\mathcal{M}} = 24$.

Compute Initial Values for Forward Neutral Words (Blue). As there is no DoF cost for blue neutral words, to obtain the corresponding initial values, one only needs to enumerate values of $24(16 + 8)$ blue cells in $\#\text{KK}^0$ and $\#\text{SB}^1$.

[2] The AddRoundConstant in the key schedule of `Whirlpool` is the last operation for each round (SB, SR, MC, AC), that is the subkey added into the encryption already involved with the round constant. When using the **SIM** technique, the corresponding cells related to the XOR compensation here will be canceled to all zero constants.

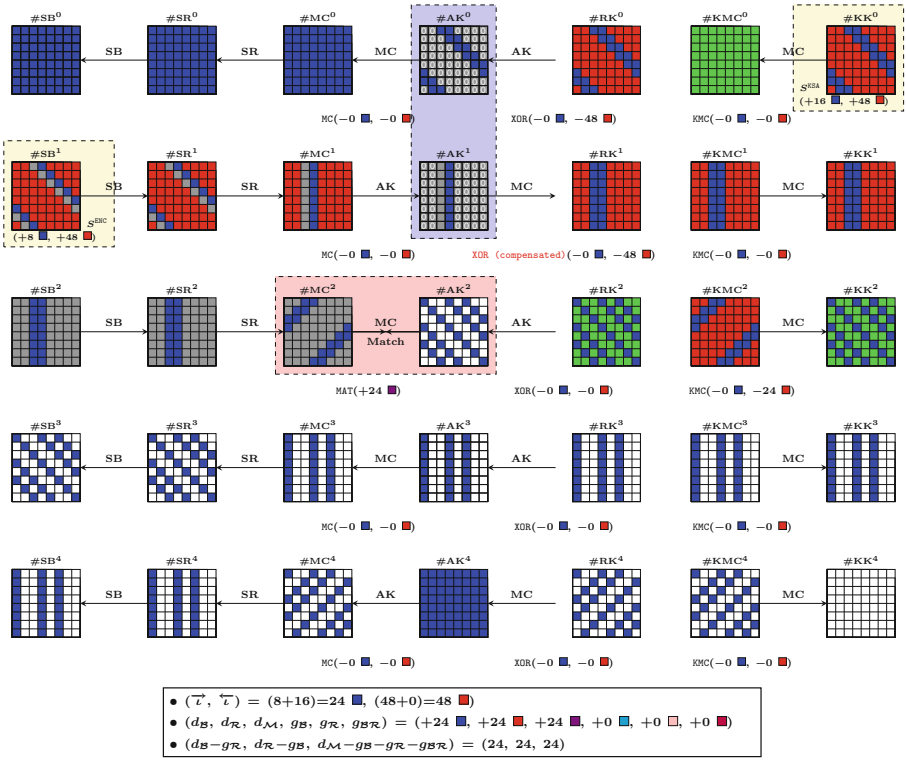

Fig. 9. An MITM pseudo-preimage attack of 5-round `Whirlpool`.

Compute Initial Values for Backward Neutral Bytes (Red). To get the initial values of red neutral words, the constraints among $\#\mathrm{KMC}^2$ and $\#\mathrm{KK}^2$ are as below

$$\begin{pmatrix} a_0 & - & - & a_9 & - & a_{15} & - & - \\ - & a_3 & - & - & a_{12} & - & a_{18} & - \\ - & - & a_6 & - & - & a_{16} & - & a_{21} \\ a_1 & - & - & a_{10} & - & - & a_{19} & - \\ - & a_4 & - & - & a_{13} & - & - & a_{22} \\ a_2 & - & a_7 & - & - & a_{18} & - & - \\ - & a_5 & - & a_{11} & - & - & a_{21} & - \\ - & - & a_8 & - & a_{14} & - & - & a_{23} \end{pmatrix} = \mathrm{MC} \cdot \begin{pmatrix} \#\mathrm{KMC}^2_0 & \#\mathrm{KMC}^2_8 & \cdots & \#\mathrm{KMC}^2_{48} & \#\mathrm{KMC}^2_{56} \\ \#\mathrm{KMC}^2_1 & - & \cdots & \#\mathrm{KMC}^2_{49} & \#\mathrm{KMC}^2_{57} \\ - & - & \cdots & \#\mathrm{KMC}^2_{50} & \#\mathrm{KMC}^2_{58} \\ - & \#\mathrm{KMC}^2_{11} & \cdots & \#\mathrm{KMC}^2_{51} & - \\ \#\mathrm{KMC}^2_4 & \#\mathrm{KMC}^2_{12} & \cdots & - & - \\ \#\mathrm{KMC}^2_5 & \#\mathrm{KMC}^2_{13} & \cdots & - & \#\mathrm{KMC}^2_{61} \\ \#\mathrm{KMC}^2_6 & \#\mathrm{KMC}^2_{14} & \cdots & \#\mathrm{KMC}^2_{54} & \#\mathrm{KMC}^2_{62} \\ \#\mathrm{KMC}^2_7 & \#\mathrm{KMC}^2_{15} & \cdots & \#\mathrm{KMC}^2_{55} & \#\mathrm{KMC}^2_{63} \end{pmatrix}, \tag{13}$$

where a_i $(0 \le i \le 23)$ are the chosen constants for $\#\mathrm{KK}^2$.

The MITM Attack Procedure for 5*-Round* `Whirlpool`. We provide the *memoryless* attack procedure derived from Fig. 9 in Algorithm 3. The *same-color match* is employed, which is firstly observed by Guo *et al.* [19] in MITM preimage attacks and recently also utilized by Hou *et al.* [21] for MITM attacks on Feistel constructions. It means a match with only blue ■ and gray ■ (or only red ■ and gray ■) at the matching point, rather than a mixture of red ■ and blue ■ cells. While gray ■ cells are fixed as constants, and thus are known in both forward

(blue) or backward (red) computations, a same-color match, *e.g.*, only blue or gray, can be performed independently of the red neutral words.

Algorithm 3: MITM attack on 5-round `Whirlpool` compression function

Fix 8 ■ constant cells in $\#\mathrm{SB}^1$ to all zero;
Fix 8 constants $(a_{16}, a_{17}, \cdots, a_{23})$ in Equation (13) to all zero;
for $C = (a_0, a_1, \cdots, a_{15}, a_{16}, \cdots, a_{23}) \in (\mathbb{F}_2^8)^{16}$ **do**
    // For red neutral words
    // As there is no ■ red cell in matching states $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$, one does not need a store table T^+ and thus does not need to solve the Equation (13) here for back neutral words, which can be done after the partial match
    // For blue neutral words
    for *8* ■ *blue cells in* $\#\mathrm{SB}^1 \in (\mathbb{F}_2^8)^8$ *and 16* ■ *blue cells in* $\#\mathrm{KK}^0 \in (\mathbb{F}_2^8)^{16}$ **do**
        Compute forward to the matching state $\#\mathrm{MC}^2$;
        Compute backward to the matching state $\#\mathrm{AK}^2$;
        // Only ■ gray and ■ blue cells in matching states here
        if $\#\mathrm{MC}^2$ *and* $\#\mathrm{AK}^2$ *pass the partial match* **then**
            // $2^{8\times(16+24-24)} = 2^{8\times16}$ candidates expected
            Solve Equation (13) according to current constant C and obtain $2^{8\times24}$ $\#\mathrm{KMC}^2$ for red neutral words;
            Compute forward and backward to match the rest 40 cells;
            if *the full match is found* **then**
                // $2^{8\times(16+24-40)} = 1$ candidate expected
                Output the preimage and stop;

The Attack Complexity. The time complexity of the above MITM pseudo-preimage attack on 5-round `Whirlpool` is about $2^{512-8\times\min(24,24,24)} = 2^{320}$.[3] Thanks to the *same color match* between $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$, the partial-matching is independent of backward red neutral words, and the constraints on backward neutral words are *linear*, then the store tables for the partial-matching can be *saved* and thus the memory complexity is $O(1)$.

6 Conclusion

In this work, we advanced the state-of-the-art MITM attacks on `AES`-like hashing. Among the new techniques, **LIN** and **DIS** contributed to the first 10-round preimage attack on `AES-192` and assorted improved results on `Rijndael`-based hashing, while **SIM** better addressed the dependencies and reduced the attack

[3] For comparisons, we directly use the calculation method in [8] to provide the time complexity for `Whirlpool`.

complexities on `Whirlpool` and `Streebog`. We argued that the ideas behind the new techniques are generic and expected them to have more applications in other attack scenarios.

Open Problems. The fact that the non-linear part of the `AES` S-box is a single monomial x^{254} allows us to obtain the full output from guessing a space with dimension four. We also investigated the S-boxes of `Whirlpool` and `Streebog` and found that their S-boxes possess fewer-dimensional subspaces (yielding parts of the output thus less powerful) for which we could not find better attacks. If more properties of S-boxes are revealed for `AES`-like hashing, *i.e.*, algebraic properties, better MITM attacks exploiting the **LIN** technique on these target ciphers can be expected. Furthermore, we applied the techniques to all `AES`/`Rijndael` variants but could not find other improvements compared to [7,8,16,43]. The relation between the security margin and block-key ratio remains to be further investigated.

Acknowledgements. We would like to thank all anonymous reviewers for their detailed and valuable comments.

This research is supported by the National Key R&D Program of China (Grants No. 2022YFB2701900), the National Natural Science Foundation of China (Grants No. 62172410), the Youth Innovation Promotion Association of Chinese Academy of Sciences, the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative and Strategic Capability Research Centres Funding Initiative, the Ministry of Education in Singapore under Grant RG93/23, and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – LI 4223/1-1. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

1. Federal Agency on Technical Regulation and Metrology, Information technology - Cryptographic data security - Hash-function, National Standard of the Russian Federation, GOST R 34.11-2012 (2012)
2. ZigBee Alliance. zigbee Specification Revision 22 1.0. Technical report, ZigBee Alliance, April 19 2017 (2017)
3. AlTawy, R., Youssef, A.M.: Preimage attacks on reduced-round Stribog. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 109–125. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06734-6_7
4. Aoki, K., Sasaki, Yu.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_7
5. Aoki, K., Sasaki, Yu.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_5
6. 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)

7. Bao, Z., et al.: Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 771–804. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_27
8. Bao, Z., Guo, J., Shi, D., Yi, T.: Superposition meet-in-the-middle attacks: updates on fundamental security of AES-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13507. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_3
9. Barreto, P.S.L.M., Rijmen, V.: The WHIRLPOOL hashing function. In: First open NESSIE Workshop, Leuven, Belgium, vol. 13, p. 14. Citeseer (2000)
10. 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.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_16
11. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_13
12. Chen, S., Guo, J., List, E., Shi, D., Zhang, T.: Diving deep into the preimage security of aes-like hashing. Cryptology ePrint Archive, Paper 2024/300 (2024). https://eprint.iacr.org/2024/300
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard Information Security and Cryptography. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-04722-4
14. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. IEEE Comput. **10**(6), 74–84 (1977)
15. Dolmatov, V., Degtyarev, A.: GOST R 34.11-2012: Hash Function. RFC 6986, August 2013 (2013)
16. 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.) CRYPTO 2021. LNCS, vol. 12827, pp. 278–308. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_10
17. Fuhr, T., Minaud, B.: Match box meet-in-the-middle attack against KATAN. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 61–81. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_4
18. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: improved attacks for AES-like permutations. In: FSE, pp. 365–383 (2010)
19. 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.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_4
20. Hosoyamada, A., Sasaki, Yu.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 249–279. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_9
21. 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.) ASIACRYPT 2023, Part III. LNCS, pp. 370–404. Springer, Cham (2023). https://doi.org/10.1007/978-981-99-8727-6_13
22. Hua, J., Dong, X., Sun, S., Zhang, Z., Lei, H., Wang, X.: Improved MITM cryptanalysis on streebog. IACR Trans. Symmetric Cryptol. **2022**(2), 63–91 (2022)

23. ISO/IEC. ISO/IEC 10118-3: 2004. IT Security techniques - Hash-functions - Part 3: Dedicated hash-functions (2004)
24. ISO/IEC. ISO/IEC 10118-2: 2010. IT Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher (2010)
25. ISO/IEC. ISO/IEC 10118-3: 2018. IT Security techniques - Hash-functions - Part 3: Dedicated hash-functions (2018)
26. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: attacks on skein-512 and the SHA-2 family. In: Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19–21, 2012. Revised Selected Papers, pp. 244–263 (2012)
27. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: results on the full whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_8
28. Lamberger, M., Mendel, F., Schläffer, M., Rechberger, C., Rijmen, V.: The rebound attack and subspace distinguishers: application to whirlpool. J. Cryptol. **28**(2), 257–296 (2015)
29. Liu, F., Mahzoun, M., Øygarden, M., Meier, W.: Algebraic attacks on RAIN and AIM using equivalent representations. IACR Trans. Symmetric Cryptol. **2023**(4), 166–186 (2023)
30. Ma, B., Li, B., Hao, R., Li, X.: Improved (Pseudo) preimage attacks on reduced-round GOST and Grøstl-256 and studies on several truncation patterns for AES-like compression functions. In: IWSEC, pp. 79–96 (2015)
31. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: cryptanalysis of reduced whirlpool and grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_16
32. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
33. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: a synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_31
34. 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.) EUROCRYPT 2023, IV. LNCS, vol. 14007, pp. 158–188. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30634-1_6
35. Sasaki, Yu.: Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 378–396. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_22
36. Sasaki, Yu., Aoki, K.: Preimage Attacks on 3, 4, and 5-Pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_16
37. Sasaki, Yu., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_8
38. Sasaki, Yu., Wang, L., Wu, S., Wu, W.: Investigating fundamental security requirements on whirlpool: improved preimage and collision attacks. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 562–579. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_34

39. Schrottenloher, A., Stevens, M.: Simplified MITM modeling for permutations: new (quantum) attacks. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. Lecture Notes in Computer Science, vol. 13509, pp. 717–747. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15982-4_24
40. Schrottenloher, A., Stevens, M.: Simplified modeling of MITM attacks for block ciphers: New (quantum) attacks. IACR Trans. Symmetric Cryptol. **2023**(3), 146–183 (2023)
41. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_9
42. Zhang, K., Qingju Wang, Y.Y., Guo, C., Cui, H.: Algebraic attacks on round-reduced rain and full AIM-III. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part III. LNCS, vol. 14440, pp. 285–310. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8727-6_10
43. Zhang, T.: Comprehensive preimage security evaluations on rijndael-based hashing. In: Zhou, J., et al. (eds.) ACNS 2023. LNCS, vol. 13907, pp. 23–42. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-41181-6_2