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
`guojian@ntu.edu.sg, tianyu005@e.ntu.edu.sg`
[4] Strategic Centre for Research in Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore
`shiyao.chen@ntu.edu.sg`
[5] Digital Trust Centre, Nanyang Technological University, Singapore
[6] Independent Researcher
`elist@posteo.de`

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 elucidates three key contributions dedicated in 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 suits perfectly with superposition states to preserve information after S-box with an 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`.

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 function 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 common coexistence of encryption and hashing in embedded systems, a conventional strategy is to construct hash function from secure block ciphers to minimize hardware or software costs: the encryption function of a block cipher is first transformed into a one-way compression function, then iterate following the Merkle-Damgård design. In 1993, Preneel, Govaerts, and van de Walle [36] 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]. In view of 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` [25,1,15], 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 [4,5,39,40], 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 [40], indirect and partial matching [4,40], biclique as a formalization of initial structures [28], sieve-in-the-middle [12] 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 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 [11].

Sasaki was the first to apply this to a preimage attack on `AES` hashing modes [38]. Though, to avoid the complex relations from the round keys, the key was still fixed to a constant. Sasaki *et al.* then introduced guess-and-determine strategy [41]. 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 extend 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 and 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.* [37] 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 [42] and was later extended to ciphers with very light key schedule [43]. 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 [7,8,16,37], 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.* [45] 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 leads 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 is preserved when input to an S-box at a 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 [45] 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 [31]. 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 improved 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 [40]. 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 remains 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 leads to dependencies across multiple rounds. We observe that values mixed to 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 mapped to the same constraints and previous attacks may be suboptimal as duplicate costs persist. 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.

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*	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	[46]
	9/12	$\mathbf{2^{180}}$	$\mathbf{2^{187}}$	2^{180}	**LIN**, BiDir	App. C.1
`Rijndael-192/256`	9/12	2^{168}	2^{181}	–	BiDir	[46]
(Hash)	**10/12**	2^{188}	2^{191}	2^{180}	**LIN**, BiDir	App. C.2
`Whirlpool` (Hash)	5/10	2^{416}	2^{448}	2^{96}	Dedicated	[41]
	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*	[41]
	6/10	2^{440}	2^{477}	2^{192}	GnD	[8]
	6/10	$\mathbf{2^{416}}$	$\mathbf{2^{465}}$	2^{288}	**SIM**, BiDir, GnD	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	App. D.2
`Streebog-512` (Compression)	7.5/12	2^{496}	–	2^{64}	Dedicated method	[32]
	7.5/12	2^{441}	–	2^{192}	GnD, MulAK	[22]
	7.5/12	$\mathbf{2^{433}}$	–	$\mathbf{2^{177}}$	**SIM**, GnD	App. E.1
	8.5/12	2^{481}	–	2^{288}	GnD, MulAK	[22]
	8.5/12	2^{481}	–	$\mathbf{2^{129}}$	**SIM**, GnD	App. E.2
`Streebog-512` (Hash)	7.5/12	–	2^{496}	2^{64}	Dedicated method	[32]
	7.5/12	–	$2^{478.25}$	2^{256}	MITM + Multi-collision[⋆]	[22]
	7.5/12	–	$\mathbf{2^{474.25}}$	2^{256}	MITM + Multi-collision	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	App. E.2

Collision Attacks

Cipher (target)	#Rounds	Time	Memory	Essential technique(s)	References
`Whirlpool` (Hash)	4.5/10	2^{120}	2^{16}	Rebound	[34]
	4.5/10	2^{64}	2^{16}	Rebound	[30]
	5/10	2^{120}	2^{64}	Super-SBox	[29,18]
	5.5/10	2^{184-s}	2^{s}	Rebound	[30]
	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	App. A
	6.5/10	$\mathbf{2^{240}}$	$\mathbf{2^{240}}$	New MILP model, MITM	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 only list 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.

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

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 standardizations 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.

LIN and **DIS** are both critical to the attack one more round of `AES-192` hashing, excluding either the attack will not be possible. It is also worth mentioning that a simple guess-and-determine using `AES` S-box property cannot 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 rounds `Whirlpool`, which extends an almost full round against state-of-the-art while maintaining the same time complexity. What is more, our efficient MILP-based search model improves 6-round collision attack on `Whirlpool`, which can be extended 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 attack results.

1.6 Organization

The remainder of this work is structured as follows. In Section 2, we provide preliminaries on the MITM attacks and the target designs. Then we elaborate the proposed techniques and their significance in Section 3. Thereupon, we present the enhanced MITM framework and MILP modeling in Section 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 Section 5. Furthermore, we provide the attacks and details of `Whirlpool`, `Rijndael`, 8-round `AES-192` and `Streebog` in the appendix. Finally, we conclude and discuss in Section 6.

2 Preliminaries

2.1 MITM Attacks: Notations and Principle

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

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

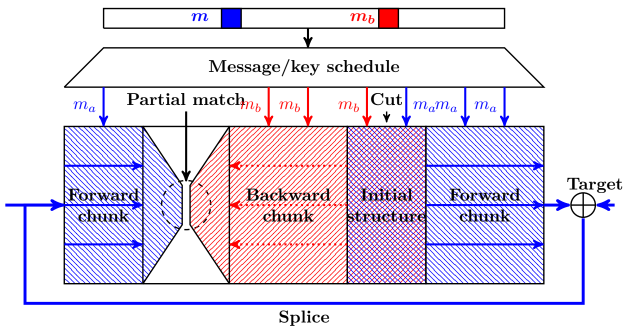


Fig. 1. A high-level overview of MITM attacks [38].

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.

Table 2. Common notations.

DoF	Degree(s) of freedom.
$S^{\mathtt{ENC}}$	Starting state in the encryption.
$S^{\mathtt{KSA}}$	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^-.
$\mathcal{B}^{\mathtt{KSA}}/\mathcal{R}^{\mathtt{KSA}}/\mathcal{G}^{\mathtt{KSA}}$	Sets of indices of forward neutral, the backward neutral, and constant bytes in $S^{\mathtt{KSA}}$, respectively.
$\mathcal{B}^{\mathtt{ENC}}/\mathcal{R}^{\mathtt{ENC}}/\mathcal{G}^{\mathtt{ENC}}$	Sets of indices of forward neutral, backward neutral, and constant bytes in $S^{\mathtt{ENC}}$, respectively.
$\overrightarrow{\iota}/\overleftarrow{\iota}$	Initial DoF of the forward and backward chunk, respectively, with $\overrightarrow{\iota} = \lvert\mathcal{B}^{\mathtt{ENC}}\rvert + \lvert\mathcal{B}^{\mathtt{KSA}}\rvert$ and $\overleftarrow{\iota} = \lvert\mathcal{R}^{\mathtt{ENC}}\rvert + \lvert\mathcal{R}^{\mathtt{KSA}}\rvert$.
$\overrightarrow{\sigma}/\overleftarrow{\sigma}$	The consumed DoF of the forward and backward chunk, respectively.

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, find all (v^+, g^+) in $T^+[m^-, g]$, check if (g^+, g^-, g) is compatible with v^+ and v^-.
5. For compatible (v^+, v^-), check for a full match.
6. 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

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

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 $2^{(n+l)/2+1}$ [35]. 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` $= 4$, `Nk` $\in \{4, 6, 8\}$, and `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 Figure 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.

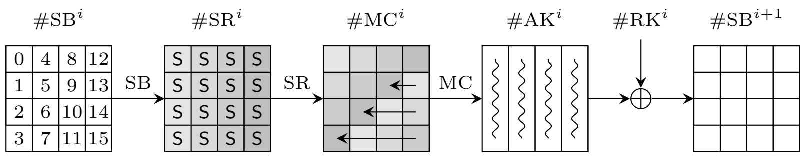


Fig. 2. `AES`-like round function.

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:

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

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 [36] (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 Figure 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.

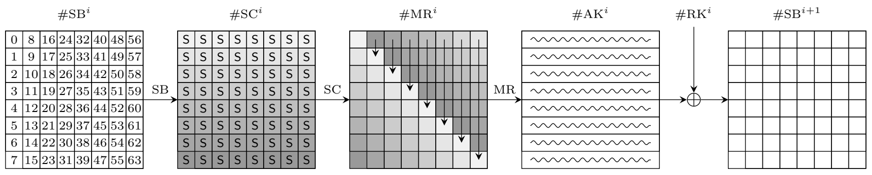


Fig. 3. The round function of `Whirlpool`.

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 mechanism of MP mode. 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 rest of paper, we will denote a state by the operation that is used as the direct inputs for, and upper-script the round index. For example, $\#SB^i$ denotes the state before the SB operation in Round i, as is shown in Figures 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 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 completely after the S-box or which consumed one byte as 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 in order 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.* [45] 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: $S^{\mathtt{ENC}}$ in the encryption function and $S^{\mathtt{KSA}}$ in the key schedule, as the generation of round keys is independently from 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 locates, as long as the values of those states could be independently chosen 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 respectively in encryption and key schedule. Previous models implicitly limited the key bytes to only depend on $S^{\texttt{KSA}}$ and not $S^{\texttt{ENC}}$, and consequently, shrunk the solution pool.

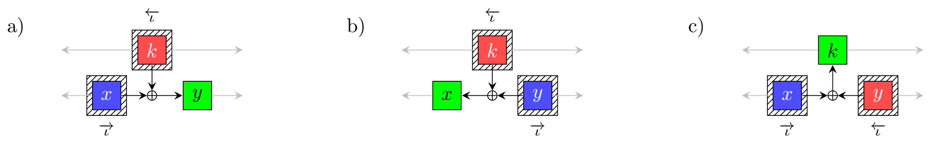


Fig. 4. Conceptual strategies of distributing initial states in two states and a key byte.

Consider an intuitive toy example in Figure 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 color green 🟩 to denote a byte in superposition. Without loss of generality, there are three possible scenarios as depicted in Figure 4. The previous models cover the first two while excludes the third, in which the key bytes is dependent on the initial DoF from encryption.

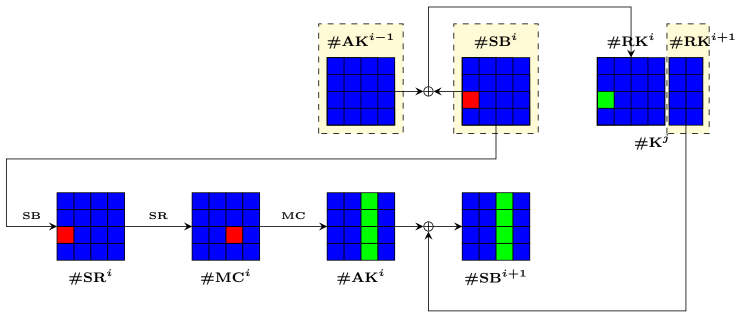


Fig. 5. Example of a distributed initial structure on `AES-192`.

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 are sufficient information to define all the round keys and all the intermediate encryption states. For example, Figure 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** in this paper, though heuristic, does include more key schedule behaviors and leads to round improvement in the cryptanalysis of AES-192.

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 [38] 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. Though, 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 Figure 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 the 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 into our MILP models.

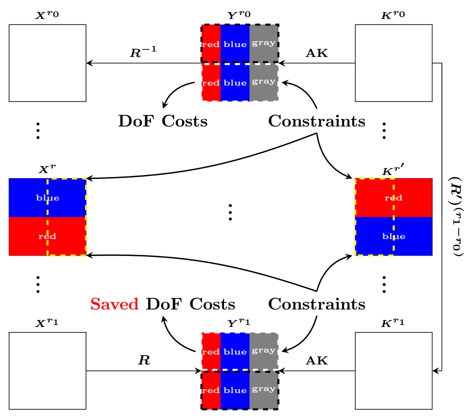


Fig. 6. High-level view of the different constraints connected by key schedule.

General Concept. For SPNs we can write the round function as composition of a nonlinear S-box layer SB, an affine layer $\mathbf{A}$, and a key addition. Assume, the 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 key and encryption state are equal before the S-box layer, then the same values will also be the results in both message 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 $\#\mathrm{AK}^i_a$.

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.

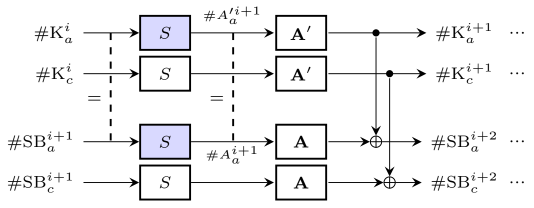


Fig. 7. Exploiting similar operations in key and message schedule.

4 Enhanced Attack Framework and MILP Model

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

4.1 Exceptional MITM Framework

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

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 $d_{\mathcal{B}} + g_{\mathcal{B}} \le d_{\mathcal{R}} + g_{\mathcal{R}}$. 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, find all (v^+, g^+) in $T^+[m^-, g, h^-]$, check if h^- is compatible with (v^+, v^-), and then check if (g^+, g^-, g) is compatible with (v^+, v^-).
5. For compatible (v^+, v^-), check for a full match.
6. 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:

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

4.2 MILP-based Search Model

Now we introduce a enhanced MILP model to integrate proposed techniques, including a 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/or $r = 1$ respectively denote the byte contains forward and/or backward neutral information. 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 is able to 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 seconds, and the optimization of MITM collision attack models of 6-round `Whirlpool` can be finished within just 300 seconds[7].

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** intergrated.

– 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 [44].

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

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 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 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 we name as the 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:

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

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 Equation (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 a 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:

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

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, $d_{\mathcal{M}}^{\mathtt{XOR}}$ results from:

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

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

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

Objective Function. The objective function of our search model is:

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

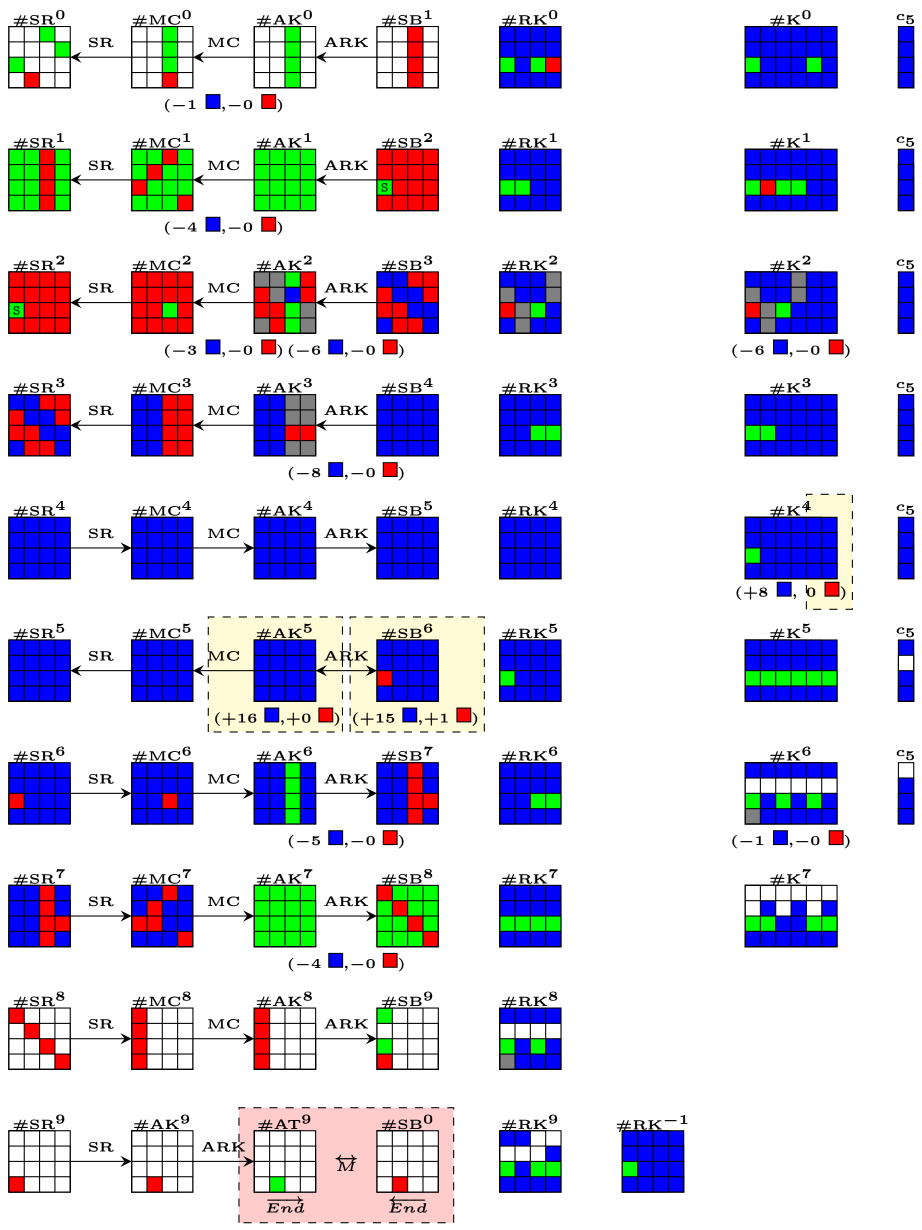
According to Equation (3), $\min\{\overrightarrow{d_b}, \overleftarrow{d_r}, \overrightarrow{\overleftarrow{m}}\}$ determines the complexity of an MITM attack. Thus, the search for the optimal MITM attack pattern of given *config* is converted to a maximization problem on objective $\tau_{\mathtt{Obj}}$:

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 Figure 8 and summarized as below:


#SR⁰ SR #MC⁰ MC #AK⁰ ARK #SB¹ #RK⁰ #K⁰ c5
(−1 🟦,−0 🟥)
#SR¹ SR #MC¹ MC #AK¹ ARK #SB² #RK¹ #K¹ c5
S
(−4 🟦,−0 🟥)
#SR² SR #MC² MC #AK² ARK #SB³ #RK² #K² c5
S
(−3 🟦,−0 🟥) (−6 🟦,−0 🟥) (−6 🟦,−0 🟥)
#SR³ SR #MC³ MC #AK³ ARK #SB⁴ #RK³ #K³ c5
(−8 🟦,−0 🟥)
#SR⁴ SR #MC⁴ MC #AK⁴ ARK #SB⁵ #RK⁴ #K⁴ c5
(+8 🟦, 0 🟥)
#SR⁵ SR #MC⁵ MC #AK⁵ ARK #SB⁶ #RK⁵ #K⁵ c5
(+16 🟦,+0 🟥) (+15 🟦,+1 🟥)
#SR⁶ SR #MC⁶ MC #AK⁶ ARK #SB⁷ #RK⁶ #K⁶ c5
(−5 🟦,−0 🟥) (−1 🟦,−0 🟥)
#SR⁷ SR #MC⁷ MC #AK⁷ ARK #SB⁸ #RK⁷ #K⁷
(−4 🟦,−0 🟥)
#SR⁸ SR #MC⁸ MC #AK⁸ ARK #SB⁹ #RK⁸
#SR⁹ SR #AK⁹ ARK #AT⁹ #SB⁰ #RK⁹ #RK⁻¹
M
End
End

- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (39$ 🟦, 1 🟥$)$ $\overleftrightarrow{S}^{\mathtt{ENC}}=(31$ 🟦, 1 🟥$)$ $\overleftrightarrow{S}^{\mathtt{KSA}}=(8$ 🟦, 0 🟥$)$
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, d_{\mathcal{L}}) = (1, 1, 1, 0.5)$
- $(d_{\mathcal{B}}-d_{\mathcal{L}}, d_{\mathcal{R}}-d_{\mathcal{L}}, d_{\mathcal{M}}) = (0.5, 0.5, 1)$

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

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

```
1  Initialize a table $T^{\text{neutral}}_{\text{blue}}$;
2  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.
3  for 4 🟦 blue cells $\#\text{K}^4[4,5,6,7] \in (\mathbb{F}_2^8)^4$ do
4      Use 4 constants $\#\text{K}^2[6,7,12,13]$ to derive $\#\text{K}^4[12,13], \#\text{K}^3[20,23]$;
5      for 3 🟦 blue cells $\#\text{K}^3[18,19,22] \in (\mathbb{F}_2^8)^3$ do
6          Use 2 constant $\#\text{K}^2[1,2]$ to derive $\#\text{K}^4[1,2]$;
7          for 6 🟦 blue cells $\#\text{K}^4[8,9,10,11,14,15] \in (\mathbb{F}_2^8)^6$ do
8              Derive $\#\text{RK}^2[14,15]$ from $\#\text{K}^4[6,7,14,15]$;
9              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
                 $\text{MC}^{-1} \cdot \#\text{K}^1[20..23] = (*,*,*,0)$;
10             Derive $\#\text{RK}^2[4,5]$ from $\#\text{K}^4[4,5], \#\text{K}^3[21,22]$;
11             Use 4 constant $\#\text{AK}^2[4,5,14,15]$ and $\#\text{RK}^2[4,5,14,15]$ to derive
                 $\#\text{SB}^3[4,5,14,15]$;
12             for 2 🟦 blue cells $\#\text{K}^4[3], \#\text{K}^3[16] \in (\mathbb{F}_2^8)^2$ do
13                 Derive $\#\text{RK}^2[3]$ from $\#\text{K}^3[16,20], \#\text{K}^4[3]$;
14                 Use 1 constant $\#\text{AK}^2[3]$ to derive $\#\text{SB}^3[3]$;
15                 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
                     $\text{MC}^{-1} \cdot \#\text{K}^1[16..19] = (0,*,*,*)$;
16                 Derive $\#\text{K}^4[16..23]$;
17                 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 $\#\text{K}^4$
18                 Derive the blue part of $\#\text{SR}^2[2]$;
                   // Linearization of S-boxes
19                 for $2^{8\times 0.5}$ values of $c_0 = (\#\text{SR}^2[2])^{17}$ do
20                     Compute 14 constants ($\#\text{MC}^0[11]$, $\#\text{MC}^1[2,5]$, $\#\text{SB}^6[2]$,
                         $\#\text{SB}^7[8..11,14]$, $\#\text{SB}^8[0,5,10,15]$, $\#\text{K}^6[3]) = (c_1,\ldots,c_{14})$;
21                     Update the table $T^{\text{neutral}}_{\text{blue}}[c_0,\ldots,c_{14}] \overset{+}{=}$ (🟦 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
22 return $T^{\text{neutral}}_{\text{blue}}$;
```

- Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 39 bytes (16 in $\#\mathrm{AK}^5$, 15 bytes in $\#\mathrm{SB}^6$, and 8 bytes $K^4[16\dots 23]$);
- 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 $d_{\mathcal{R}} - d_{\mathcal{L}} = 1 - 0.5 = 0.5$.

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 $\#\mathrm{K}^2[2]$ 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 $\#\mathrm{SB}^3, \#\mathrm{K}^4, \#\mathrm{K}^3$ 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 Figure 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 pre-computation table $T^{\mathrm{neutral}}_{\mathrm{blue}}$ 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 at 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.

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

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

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 Figure 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 are more concentrated at the staring point, *e.g.*, 48 red cells cancelled at Round 0 (48 bytes DoF cost will be compensated at Round 1), and another 24 red cells will be cancelled at the MC operation for #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 now elaborate the attack configuration as below:

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

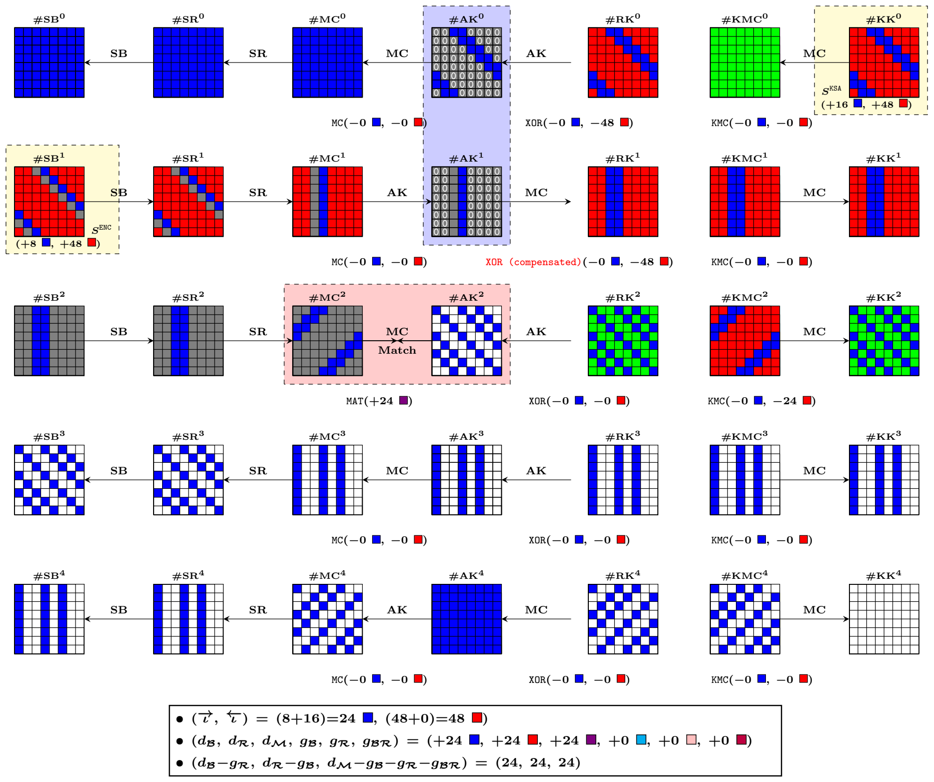


Fig. 9. An MITM pseudo-preimage attack of 5-round `Whirlpool`.

bytes can be cancelled to constants marked as zero constant[8] ▣ in $\#\mathrm{AK}^0$ and $\#\mathrm{AK}^1$, 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 $\#\mathrm{KMC}^2 \xrightarrow{\mathrm{MC}} \#\mathrm{KK}^2$;
– 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 $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$.

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 $\#\mathrm{KK}^0$ and $\#\mathrm{SB}^1$.

[8] The AddRoundConstant in 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 cancelled to all zero constants.

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 Figure 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

```
1  Fix 8 ⬛ constant cells in $\#\mathrm{SB}^1$ to all zero;
2  Fix 8 constants $(a_{16}, a_{17}, \cdots, a_{23})$ in Equation (13) to all zero;
3  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
4      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
5          Compute forward to the matching state $\#\mathrm{MC}^2$;
6          Compute backward to the matching state $\#\mathrm{AK}^2$;
           // Only ⬛ gray and 🟦 blue cells in matching states here
7          if $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$ pass the partial match then
               // $2^{8\times(16+24-24)} = 2^{8\times 16}$ candidates expected
8              Solve Equation (13) according to current constant $C$ and obtain
                $2^{8\times 24}$ $\#\mathrm{KMC}^2$ for red neutral words;
9              Compute forward and backward to match the rest 40 cells;
10             if the full match is found then
                   // $2^{8\times(16+24-40)} = 1$ candidate expected
11                 Output the preimage and stop;
```

The attack complexity. The time complexity of the above MITM pseudo-preimage attack on 5-round Whirlpool is about $2^{512-8\times\min(24,24,24)} = 2^{320}$.[9] 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 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 4-bit dimension. We also investigated the S-boxes of Whirlpool and Streebog and found that their S-boxes possess lower 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 [7,8,16,46]. 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.

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

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.
3. Riham AlTawy and Amr M. Youssef. Preimage Attacks on Reduced-Round Stribog. In *AFRICACRYPT*, pages 109–125, 2014.
4. Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2008.
5. Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2009.
6. Zhenzhen Bao, Lin Ding, Jian Guo, Haoyang Wang, and Wenying Zhang. Improved meet-in-the-middle preimage attacks against AES hashing modes. *IACR Trans. Symmetric Cryptol.*, 2019(4):318–347, 2019.
7. Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic Search of Meet-in-the-Middle Preimage Attacks on AES-like Hashing. In *EUROCRYPT I*, pages 771–804, 2021.
8. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. Superposition Meet-in-the-Middle Attacks: Updates on Fundamental Security of AES-like Hashing. In *CRYPTO I*, pages 64–93, 2022.
9. Paulo S.L.M. Barreto and Vincent Rijmen. The WHIRLPOOL hashing function. Available online at http://www.larc.usp.br/ pbarreto/WhirlpoolPage.html, September 2000. Revised in May 2003.
10. Eli Biham and Orr Dunkelman. A framework for iterative hash functions-haifa. In *Second NIST Cryptographic Hash Workshop*, volume 2006, page 2, 2006.
11. Andrey Bogdanov and Christian Rechberger. A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2010.
12. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-Middle: Improved MITM Attacks. In Ran Canetti and Juan A. Garay, editors, *CRYPTO I*, volume 8042 of *Lecture Notes in Computer Science*, pages 222–240. Springer, 2013.
13. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
14. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.
15. Vasily Dolmatov and Alexey Degtyarev. GOST R 34.11-2012: Hash Function. RFC 6986, August 2013.
16. Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-Middle Attacks Revisited: Key-Recovery, Collision, and Preimage Attacks. In *CRYPTO III*, pages 278–308, 2021.
17. Thomas Fuhr and Brice Minaud. Match Box Meet-in-the-Middle Attack Against KATAN. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2014.

18. Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In *FSE*, pages 365–383, 2010.
19. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In *ASIACRYPT*, pages 56–75, 2010.
20. Akinori Hosoyamada and Yu Sasaki. Finding Hash Collisions with Quantum Computers by Using Differential Trails with Smaller Probability than Birthday Bound. In *EUROCRYPT II*, pages 249–279, 2020.
21. Qingliang Hou, Xiaoyang Dong, Lingyue Qin, Guoyan Zhang, and Xiaoyun Wang. Automated meet-in-the-middle attack goes to feistel. In *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*, pages 370–404, 2023.
22. Jialiang Hua, Xiaoyang Dong, Siwei Sun, Zhiyu Zhang, Lei Hu, and Xiaoyun Wang. 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. Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.
27. Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In *CRYPTO*, pages 306–316, 2004.
28. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family. *IACR Cryptol. ePrint Arch.*, page 286, 2011.
29. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In *ASIACRYPT*, pages 126–143, 2009.
30. Mario Lamberger, Florian Mendel, Martin Schläffer, Christian Rechberger, and Vincent Rijmen. The rebound attack and subspace distinguishers: Application to whirlpool. *J. Cryptol.*, 28(2):257–296, 2015.
31. Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. Algebraic attacks on RAIN and AIM using equivalent representations. *IACR Trans. Symmetric Cryptol.*, 2023(4):166–186, 2023.
32. Bingke Ma, Bao Li, Ronglin Hao, and Xiaoqian Li. Improved (Pseudo) Preimage Attacks on Reduced-Round GOST and Grøstl-256 and Studies on Several Truncation Patterns for AES-like Compression Functions. In *IWSEC*, pages 79–96, 2015.
33. Florian Mendel, Norbert Pramstaller, and Christian Rechberger. A (Second) Preimage Attack on the GOST Hash Function. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 224–234. Springer, 2008.
34. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced whirlpool and grøstl. In Orr Dunkelman, editor, *Fast Software Encryption*, pages 260–276, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
35. Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

36. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In *CRYPTO*, pages 368–378, 1993.
37. Lingyue Qin, Jialiang Hua, Xiaoyang Dong, Hailun Yan, and Xiaoyun Wang. Meet-in-the-Middle Preimage Attacks on Sponge-Based Hashing. In *EUROCRYPT IV*, pages 158–188, 2023.
38. Yu Sasaki. Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 378–396. Springer, 2011.
39. Yu Sasaki and Kazumaro Aoki. Preimage Attacks on 3, 4, and 5-Pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 253–271. Springer, 2008.
40. Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2009.
41. Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating Fundamental Security Requirements on Whirlpool: Improved Preimage and Collision Attacks. In *ASIACRYPT*, pages 562–579, 2012.
42. André Schrottenloher and Marc Stevens. Simplified MITM Modeling for Permutations: New (Quantum) Attacks. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO III*, volume 13509 of *Lecture Notes in Computer Science*, pages 717–747. Springer, 2022.
43. André Schrottenloher and Marc Stevens. Simplified Modeling of MITM Attacks for Block Ciphers: new (Quantum) Attacks. *IACR Cryptol. ePrint Arch.*, page 816, 2023.
44. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.
45. Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic attacks on round-reduced rain and full AIM-III. In *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*, pages 285–310, 2023.
46. Tianyu Zhang. Comprehensive preimage security evaluations onrijndael-based hashing. In Jianying Zhou, Lejla Batina, Zengpeng Li, Jingqiang Lin, Eleonora Losiouk, Suryadipta Majumdar, Daisuke Mashima, Weizhi Meng, Stjepan Picek, Mohammad Ashiqur Rahman, Jun Shao, Masaki Shimaoka, Ezekiel Soremekun, Chunhua Su, Je Sen Teh, Aleksei Udovenko, Cong Wang, Leo Zhang, and Yury Zhauniarovich, editors, *Applied Cryptography and Network Security Workshops*, pages 23–42, Cham, 2023. Springer Nature Switzerland.

Appendix A Improved Collision Attacks of Whirlpool

Besides our results on preimage attacks, we also improve collision attacks on round-reduced `Whirlpool`. Our attack on 6-round is shown in Figure 10. Compared to the previous best collision attack on 6 rounds of `Whirlpool` presented by Dong *et al.* at Crypto 2021 [16], we reduced the time complexity from 2^{248} to 2^{240}. We also extended it to a collision attack on 6.5-round `Whirlpool`, which is presented in Figure 11 and has the same complexity of 2^{240} operations. When converting an MITM preimage attack into a collision attack, the input for key schedule will be fixed for `Whirlpool`, thus there will be no DoF introduced in key schedule in our search model. To demonstrate, we only provide the attack details for 6-round `Whirlpool` as below.

– Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 36 bytes (in $\#\mathrm{MC}^3$);
– Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 4 bytes (in $\#\mathrm{MC}^3$);
– Consumed DoF for forward $\overrightarrow{\sigma}$: 32 bytes (16 for $\#\mathrm{MC}^4 \xrightarrow{\mathrm{MC}} \#\mathrm{AK}^4$ and 16 for $\#\mathrm{AK}^1 \xrightarrow{\mathrm{MC}^{-1}} \#\mathrm{MC}^1$);
– Consumed DoF for backward $\overleftarrow{\sigma}$: zero bytes;
– Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: all zero bytes;
– Matching DoF $d_{\mathcal{M}}$: 4 bytes between $\#\mathrm{MC}^5$ and $\#\mathrm{AK}^5$.

Then, the remaining DoF is $d_{\mathcal{B}} = d_{\mathcal{R}} = d_{\mathcal{M}} = 4$.

Compute initial values for forward neutral words (Blue). To get the initial values of forward neutral words, one should focus on the constraints among states $\#\mathrm{MC}^4$ and $\#\mathrm{AK}^4$, $\#\mathrm{MC}^1$ and $\#\mathrm{AK}^1$ as below

– For $\#\mathrm{MC}^4$ and $\#\mathrm{AK}^4$, there are 16 bytes MC DoF cost for forward neutral words, which applies the constraints in Equation (14):

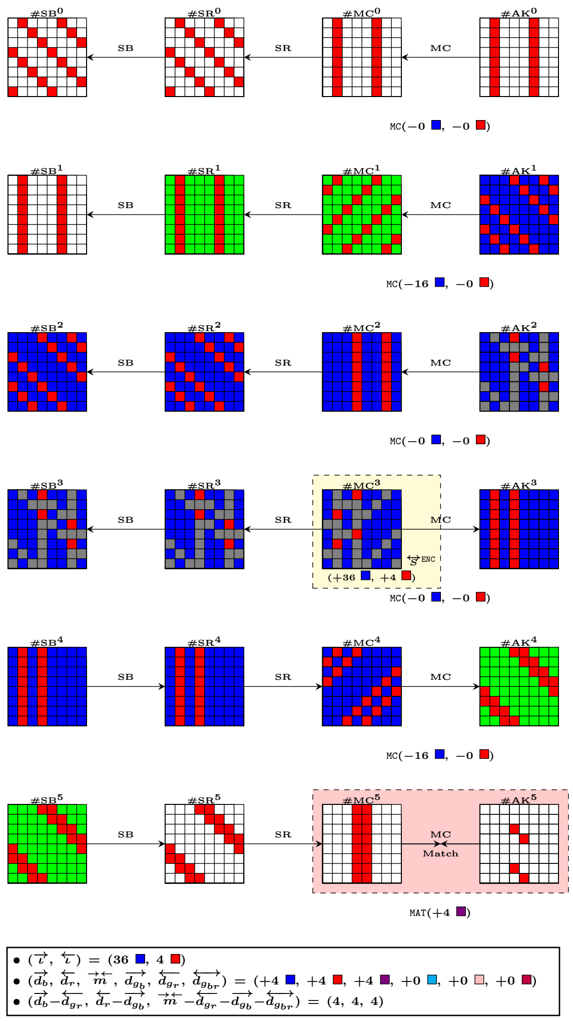
$$\begin{pmatrix} - & - & - & a_6 & a_8 & - & - & - \\ - & - & - & - & a_9 & a_{10} & - & - \\ - & - & - & - & - & a_{11} & a_{12} & - \\ - & - & - & - & - & - & a_{13} & a_{14} \\ a_0 & - & - & - & - & - & - & a_{15} \\ a_1 & a_2 & - & - & - & - & - & - \\ - & a_3 & a_4 & - & - & - & - & - \\ - & - & a_5 & a_7 & - & - & - & - \end{pmatrix} = \mathrm{MC} \cdot \begin{pmatrix} \#\mathrm{MC}^4_0 & \#\mathrm{MC}^4_8 & \cdots & \#\mathrm{MC}^4_{56} \\ \#\mathrm{MC}^4_1 & \#\mathrm{MC}^4_9 & \cdots & \#\mathrm{MC}^4_{57} \\ \#\mathrm{MC}^4_2 & \#\mathrm{MC}^4_{10} & \cdots & \#\mathrm{MC}^4_{58} \\ \#\mathrm{MC}^4_3 & \#\mathrm{MC}^4_{11} & \cdots & \#\mathrm{MC}^4_{59} \\ \#\mathrm{MC}^4_4 & \#\mathrm{MC}^4_{12} & \cdots & \#\mathrm{MC}^4_{60} \\ \#\mathrm{MC}^4_5 & \#\mathrm{MC}^4_{13} & \cdots & \#\mathrm{MC}^4_{61} \\ \#\mathrm{MC}^4_6 & \#\mathrm{MC}^4_{14} & \cdots & \#\mathrm{MC}^4_{62} \\ \#\mathrm{MC}^4_7 & \#\mathrm{MC}^4_{15} & \cdots & \#\mathrm{MC}^4_{63} \end{pmatrix}, \tag{14}$$

where a_i $(0 \le i \le 15)$ are the chosen constants for $\#\mathrm{AK}^4$.

– For $\#\mathrm{MC}^1$ and $\#\mathrm{AK}^1$, there are also 16 bytes MC DoF cost for forward neutral words, which applies the constraints in Equation (15) as below

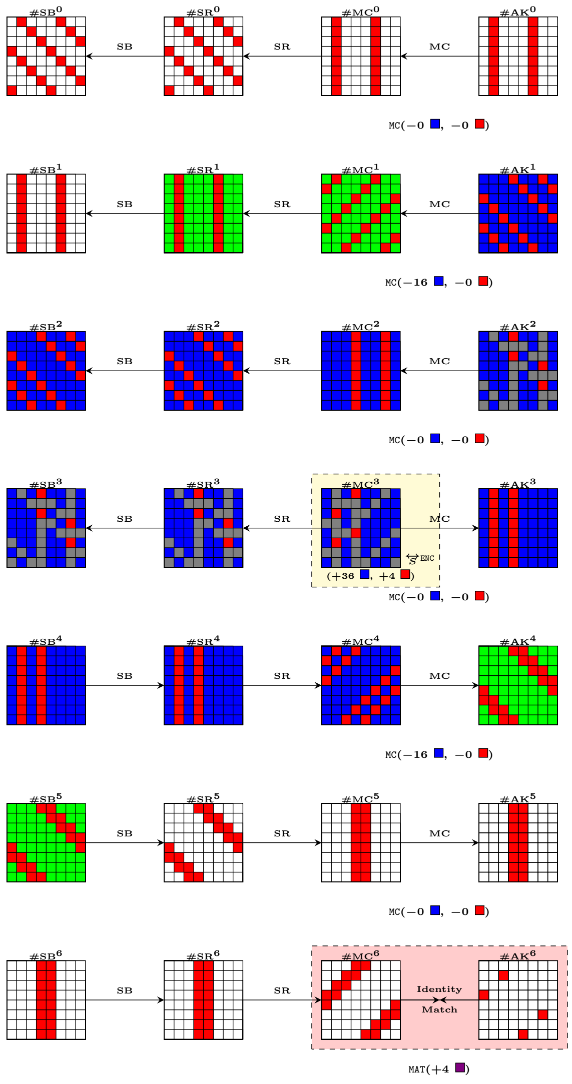
$$\begin{pmatrix} - & b_2 & - & - & - & b_{10} & - & - \\ b_0 & - & - & - & b_8 & - & - & - \\ - & - & - & b_6 & - & - & - & b_{14} \\ - & - & b_4 & - & - & - & b_{12} & - \\ - & b_3 & - & - & - & b_{11} & - & - \\ b_1 & - & - & - & b_9 & - & - & - \\ - & - & - & b_7 & - & - & - & b_{15} \\ - & - & b_5 & - & - & - & b_{12} & - \end{pmatrix} = \mathrm{MC}^{-1} \cdot \begin{pmatrix} \#\mathrm{AK}^1_0 & \#\mathrm{AK}^1_8 & \cdots & \#\mathrm{AK}^1_{56} \\ \#\mathrm{AK}^1_1 & \#\mathrm{AK}^1_9 & \cdots & \#\mathrm{AK}^1_{57} \\ \#\mathrm{AK}^1_2 & \#\mathrm{AK}^1_{10} & \cdots & \#\mathrm{AK}^1_{58} \\ \#\mathrm{AK}^1_3 & \#\mathrm{AK}^1_{11} & \cdots & \#\mathrm{AK}^1_{59} \\ \#\mathrm{AK}^1_4 & \#\mathrm{AK}^1_{12} & \cdots & \#\mathrm{AK}^1_{60} \\ \#\mathrm{AK}^1_5 & \#\mathrm{AK}^1_{13} & \cdots & \#\mathrm{AK}^1_{61} \\ \#\mathrm{AK}^1_6 & \#\mathrm{AK}^1_{14} & \cdots & \#\mathrm{AK}^1_{62} \\ \#\mathrm{AK}^1_7 & \#\mathrm{AK}^1_{15} & \cdots & \#\mathrm{AK}^1_{63} \end{pmatrix}, \tag{15}$$

where b_i $(0 \le i \le 15)$ are the chosen constants for $\#\mathrm{MC}^1$.



- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (36$ 🟦, 4 🟥$)$
- $(\overrightarrow{d_b}, \overleftarrow{d_r}, \overrightarrow{\overleftarrow{m}}, \overrightarrow{d_{g_b}}, \overleftarrow{d_{g_r}}, \overleftrightarrow{d_{g_{br}}}) = (+4$ 🟦, $+4$ 🟥, $+4$ 🟪, $+0$ 🟦, $+0$ 🟥, $+0$ 🟥$)$
- $(\overrightarrow{d_b} - \overleftarrow{d_{g_r}}, \overleftarrow{d_r} - \overrightarrow{d_{g_b}}, \overrightarrow{\overleftarrow{m}} - \overleftarrow{d_{g_r}} - \overrightarrow{d_{g_b}} - \overleftrightarrow{d_{g_{br}}}) = (4, 4, 4)$

Fig. 10. An MITM collision attack of 6-round `Whirlpool`.



- $(\overrightarrow{\iota}, \overleftarrow{\iota})$ = (36 🟦, 4 🟥)
- $(\overrightarrow{d_b}, \overleftarrow{d_r}, \overrightarrow{\overleftarrow{m}}, \overrightarrow{d_{g_b}}, \overleftarrow{d_{g_r}}, \overleftrightarrow{d_{g_{br}}})$ = (+4 🟦, +4 🟥, +4 🟪, +0 🟦, +0 🟥, +0 🟥)
- $(\overrightarrow{d_b}-\overleftarrow{d_{g_r}}, \overleftarrow{d_r}-\overrightarrow{d_{g_b}}, \overrightarrow{\overleftarrow{m}}-\overleftarrow{d_{g_r}}-\overrightarrow{d_{g_b}}-\overleftrightarrow{d_{g_{br}}}) = (4, 4, 4)$

Fig. 11. An MITM collision attack of 6.5-round `Whirlpool`.

In Algorithm 4, we provide an efficient method to obtain blue neutral words satisfying the constraints (in Equation (14) and (15)) in a non-linear system, which has the time complexity and memory complexity both with $2^{8\times 30} = 2^{240}$.

Algorithm 4: Compute forward neutral words (blue) for 6-round collision attack on `Whirlpool`

```
1  Initiate two tables $T'^{\text{neutral}}_{\text{blue}}$ and $T^{\text{neutral}}_{\text{blue}}$;
   // The index list $I$ corresponds with 12 constant cells in columns 0,
      2, 3, 5, 6 and 7 of $\#\text{MC}^3$
2  Set list $I = [4, 6, 17, 20, 34, 39, 41, 46, 48, 53, 60, 63]$;
3  Fix 6 ■ constant cells $(a_0, \cdots, a_5)$ in Equation (14) to all zero;
4  Fix 12 ■ constant cells $\#\text{MC}^3[I]$ to all zero;
5  Fix 12 ■ constant cells in columns 1 and 3 of $\#\text{MC}^3$ to all zero;
6  for any 24 ■ blue cells in $\#\text{SB}^4 = X \in (\mathbb{F}_2^8)^{24}$ do
7      Compute to 18 middle values denoted by $m^+$, which correspond with
         $(a_0, \cdots, a_5)$ and $\#\text{MC}^3[I]$ when passing MC or $\text{MC}^{-1}$;
8      Update to table with $T'^{\text{neutral}}_{\text{blue}}[m^+][0] \mathrel{+}= X$ (with size $2^{8\times 24}$);
9  for left 24 ■ blue cells in $\#\text{SB}^4 = Y \in (\mathbb{F}_2^8)^{24}$ do
10     Compute to 18 middle values denoted by $m^-$, which correspond with
         $(a_0, \cdots, a_5)$ and $\#\text{MC}^3[I]$ when passing MC or $\text{MC}^{-1}$;
11     Update to table with $T'^{\text{neutral}}_{\text{blue}}[m^-][1] \mathrel{+}= Y$ (with size $2^{8\times(48-18)} = 2^{8\times 30}$);
12 for each $(x, y)$ in $T'^{\text{neutral}}_{\text{blue}}$ do
13     Compute to rest 10 constants cells $(a_6, \cdots, a_{15})$ in Equation (14);
14     Compute to 16 constants cells $(b_0, \cdots, b_{15})$ in Equation (15);
       // Each index has $2^{8\times(30-26)} = 2^{8\times 4}$ values on average
15     Update to table with $T^{\text{neutral}}_{\text{blue}}[(a_6, \cdots, a_{15}, b_0, \cdots, b_{15})] \mathrel{+}= (x, y)$;
16 return $T^{\text{neutral}}_{\text{blue}}$;
```

Compute initial values backward neutral bytes (Red). As there is no DoF cost for backward neutral words, to obtain the corresponding initial values, one only needs to enumerate all possible values of 4 red cells in $\#\text{MC}^3$.

The MITM collision attack procedure for 6-round `Whirlpool`. The main attack procedure derived from Figure 10 is provided in Algorithm 5.

The attack complexity. The time complexity of the above MITM collision attack of 6-round `Whirlpool` is about 2^{240}. The table $T^{\text{neutral}}_{\text{blue}}$ dominates the memory complexity with $2^{30\times 8} = 2^{240}$.

Appendix B Improved Pseudo-preimage Attack on 8-round `AES-192`

The attack configuration is as below:

Algorithm 5: MITM collision attack of 6-round `Whirlpool`

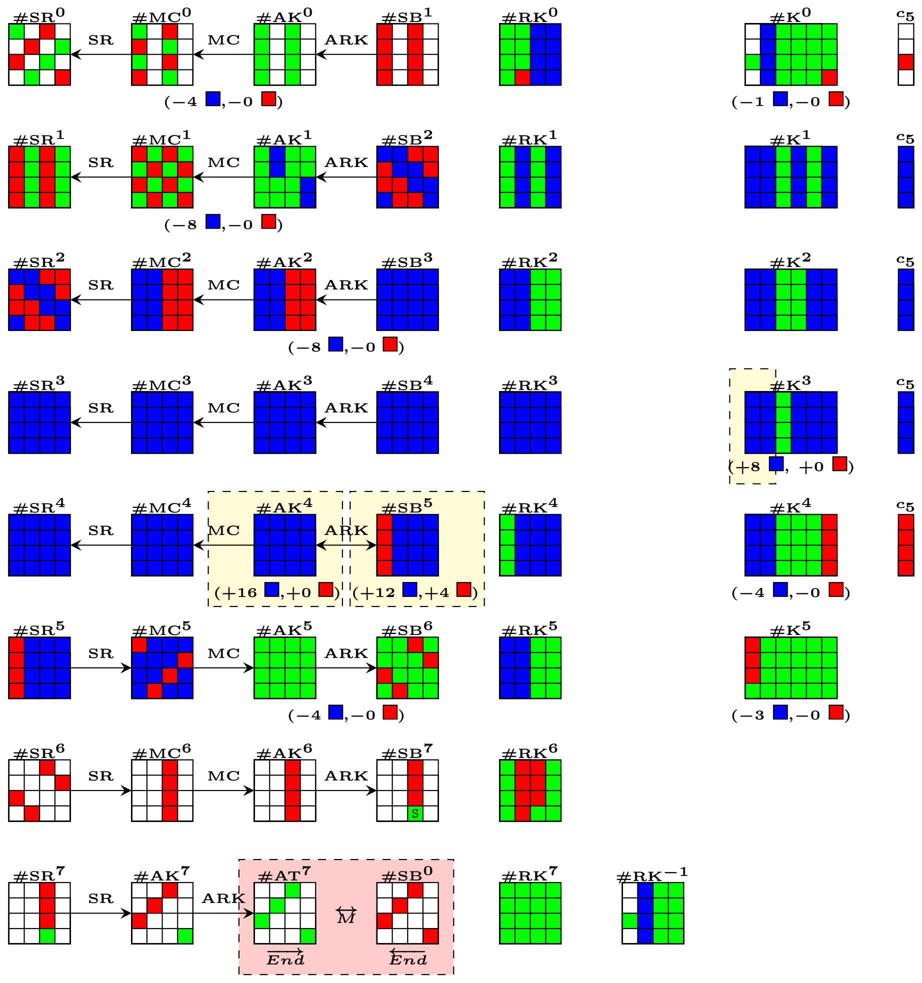
```
1  Let 10 ■ constant cells $(a_6, \cdots, a_{15})$ in Equation (14) and 16 ■ constant cells
     $(b_0, \cdots, b_{15})$ in Equation (15) be $C_G$;
2  Call Algorithm 4 to build $T^{\text{neutral}}_{\text{blue}}$ (with size $2^{8\times 30}$);
3  Initiate a table $T_{\text{collision}}$;
4  for $C_G \in (\mathbb{F}_2^8)^{26}$ do
       // For red neutral words
5      for 4 🟥 red cells in $\#\text{MC}^3 \in (\mathbb{F}_2^8)^4$ do
           // As there is no any 🟦 blue cell in matching states $\#\text{AK}^5$
              and $\#\text{MC}^5$, the same color match can be used here
6          Compute forward to the matching state $\#\text{MC}^5$;
7          Compute backward to the matching state $\#\text{AK}^5$;
8          if $\#\text{MC}^5$ and $\#\text{AK}^5$ pass the partial match then
               // $2^{8\times(26+4+4-4)} = 2^{8\times 30}$ candidates expected
9              Lookup in table with $T^{\text{neutral}}_{\text{blue}}[C_G]$;
10             Reconstruct the message $X$ from neutral words and constants;
11             Insert $X$ into $T_{\text{collision}}$ indexed by the hash value of $X$;
12             if The full collision is found then
                   // One candidate expected
13                 Output the collision and stop;
```

- The initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 36 bytes (16 in $\#\text{AK}^4$, 12 in $\#\text{SB}^5[4..15]$ and eight in $\#\text{K}^3[0..7]$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 4 bytes ($\#\text{SB}^5[0..3]$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: 32 bytes;
- Consumed DoF for backward $\overleftarrow{\sigma}$: zero bytes;
- Guessed bytes for blue, red, and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{B}} = g_{\mathcal{BR}} = 0$ bytes;
- Guessed byte equivalents for linearization: $d_{\mathcal{L}} = 0.5$ bytes.
- Matching DoF $d_{\mathcal{M}}$: 4 byte.

The remaining DoF are composed of $d_{\mathcal{B}} - d_{\mathcal{L}} = 4 - 0.5 = 3.5$, $d_{\mathcal{R}} - d_{\mathcal{L}} = 4 - 0.5 = 3.5$ and $d_{\mathcal{M}} = 4$.

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.

Fix $\#\text{K}^0[23], \#\text{K}^4[20..23], \#\text{K}^5[0,1,2], \#\text{MC}^0[1,3,8,10], \#\text{MC}^1[1, 3, 5, 7, 8, 10, 13, 15]$ be all zero, and the following constraints will be utilized by to get


#SR⁰ SR #MC⁰ MC #AK⁰ ARK #SB¹ #RK⁰ #K⁰ c5
(−4 🟦,−0 🟥)
(−1 🟦,−0 🟥)
#SR¹ SR #MC¹ MC #AK¹ ARK #SB² #RK¹ #K¹ c5
(−8 🟦,−0 🟥)
#SR² SR #MC² MC #AK² ARK #SB³ #RK² #K² c5
(−8 🟦,−0 🟥)
#SR³ SR #MC³ MC #AK³ ARK #SB⁴ #RK³ #K³ c5
(+8 🟦, +0 🟥)
#SR⁴ SR #MC⁴ MC #AK⁴ ARK #SB⁵ #RK⁴ #K⁴ c5
(+16 🟦,+0 🟥)
(+12 🟦,+4 🟥)
(−4 🟦,−0 🟥)
#SR⁵ SR #MC⁵ MC #AK⁵ ARK #SB⁶ #RK⁵ #K⁵
(−4 🟦,−0 🟥)
(−3 🟦,−0 🟥)
#SR⁶ SR #MC⁶ MC #AK⁶ ARK #SB⁷ #RK⁶
S
#SR⁷ SR #AK⁷ ARK #AT⁷ #SB⁰ #RK⁷ #RK⁻¹
M
End
End

- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (36$ 🟦, 4 🟥) $\overleftrightarrow{S}^{\text{ENC}}=(28$ 🟦, 4 🟥) $\overleftrightarrow{S}^{\text{KSA}}=(8$ 🟦, 0 🟥)
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, d_{\mathcal{L}}) = (4, 4, 4, 0.5)$
- $(d_{\mathcal{B}}-d_{\mathcal{L}}, d_{\mathcal{R}}-d_{\mathcal{L}}, d_{\mathcal{M}}) = (3.5, 3.5, 4)$

Fig. 12. An MITM pseudo-preimage attack of 8-round `AES-192`.

the initial values of blue neutral bytes.

$$\begin{cases} \bigoplus_{i=0}^{5} \#\mathrm{K}^3[4\cdot i] \oplus \mathsf{S}(\#\mathrm{K}^3[21]) = 0\,, \\ \bigoplus_{i=0}^{5} \#\mathrm{K}^3[4\cdot i+1] \oplus \mathsf{S}(\#\mathrm{K}^3[22]) = 0\,, \\ \bigoplus_{i=0}^{5} \#\mathrm{K}^3[4\cdot i+2] \oplus \mathsf{S}(\#\mathrm{K}^3[23]) = 0\,, \\ \bigoplus_{i=0}^{5} \#\mathrm{K}^3[4\cdot i+3] \oplus \mathsf{S}(\#\mathrm{K}^3[20]) = 0\,, \\ \#\mathrm{K}^3[0] \oplus \mathsf{S}(\#\mathrm{K}^3[21]) = 0, \\ \#\mathrm{K}^3[1] \oplus \mathsf{S}(\#\mathrm{K}^3[22]) = 0, \\ \#\mathrm{K}^3[2] \oplus \mathsf{S}(\#\mathrm{K}^3[23]) = 0 \\ \bigoplus_{i=2}^{5} \#\mathrm{K}^3[4\cdot i+3] = 0, \\ \mathrm{MC}^{-1}\cdot \begin{pmatrix} \bigoplus_{i=1}^{4} \#\mathrm{K}^3[4\cdot i] \\ \bigoplus_{i=1}^{4} \#\mathrm{K}^3[4\cdot i+1] \\ \bigoplus_{i=1}^{4} \#\mathrm{K}^3[4\cdot i+2] \\ \bigoplus_{i=1}^{4} \#\mathrm{K}^3[4\cdot i+3] \end{pmatrix} = \begin{pmatrix} * \\ 0 \\ * \\ 0 \end{pmatrix}, \\ \mathrm{MC}^{-1}\cdot \begin{pmatrix} \#\mathrm{K}^3[0] \oplus \mathsf{S}(\#\mathrm{K}^3[17] \oplus \#\mathrm{K}^3[21]) \oplus \mathsf{S}(\#\mathrm{K}^3[13] \oplus \#\mathrm{K}^3[21]) \\ \#\mathrm{K}^3[1] \oplus \mathsf{S}(\#\mathrm{K}^3[18] \oplus \#\mathrm{K}^3[22]) \oplus \mathsf{S}(\#\mathrm{K}^3[14] \oplus \#\mathrm{K}^3[23]) \\ \#\mathrm{K}^3[2] \oplus \mathsf{S}(\#\mathrm{K}^3[19] \oplus \#\mathrm{K}^3[23]) \oplus \mathsf{S}(\#\mathrm{K}^3[15] \oplus \#\mathrm{K}^3[23]) \\ \#\mathrm{K}^3[3] \oplus \mathsf{S}(\#\mathrm{K}^3[16] \oplus \#\mathrm{K}^3[20]) \oplus \mathsf{S}(\#\mathrm{K}^3[12] \oplus \#\mathrm{K}^3[20]) \end{pmatrix} = \begin{pmatrix} 0 \\ * \\ 0 \\ * \end{pmatrix}. \end{cases} \tag{16}$$

In the following, let $\#X[i] = \#\mathrm{SB}^2[i] \oplus \#\mathrm{K}^3[i]$ for all $i \in \{0, \ldots, 15\}$.

$$\mathrm{MC}^{-1}\cdot \begin{pmatrix} \#X[0] \oplus \#\mathrm{K}^3[8] & \#X[4] \oplus \#\mathrm{K}^3[12] & \#\mathrm{K}^3[8] \oplus \#\mathrm{K}^3[16] & \#\mathrm{K}^3[12] \oplus \#\mathrm{K}^3[20] \\ \#\mathrm{K}^3[1] \oplus \#\mathrm{K}^3[9] & \#X[5] \oplus \#\mathrm{K}^3[13] & \#X[9] \oplus \#\mathrm{K}^3[17] & \#\mathrm{K}^3[13] \oplus \#\mathrm{K}^3[21] \\ \#\mathrm{K}^3[2] \oplus \#\mathrm{K}^3[10] & \#\mathrm{K}^3[6] \oplus \#\mathrm{K}^3[14] & \#\mathrm{SB}^2[10] \oplus \#\mathrm{K}^4[10] \oplus \#\mathrm{K}^4[18] & \#X[14] \oplus \#\mathrm{K}^3[22] \\ \#X[3] \oplus \#\mathrm{K}^3[11] & \#\mathrm{K}^3[7] \oplus \#\mathrm{K}^3[15] & \#\mathrm{K}^3[11] \oplus \#\mathrm{K}^3[19] & \#X[15] \oplus \#\mathrm{K}^3[23] \end{pmatrix} = \begin{pmatrix} 0 & * & 0 & * \\ * & 0 & * & 0 \\ 0 & * & 0 & * \\ * & 0 & * & 0 \end{pmatrix}. \tag{17}$$

Algorithm 6: Compute forward neutral words (blue) for 8-round `AES-192`.

```
1  Initiate two tables $T^{\text{neutral}}_{\text{middle}}$ and $T^{\text{neutral}}_{\text{blue}}$;
2  Fix $\#\mathrm{K}^0[23]$, $\#\mathrm{K}^4[20..23]$, $\#\mathrm{K}^5[0,1,2]$, $\#\mathrm{MC}^0[1,3,8,10]$, and
     $\#\mathrm{MC}^1[1,3,5,7,8,10,13,15]$ to zeroes.
3  for 12 ■ blue cells $\#\mathrm{K}^3[0..11] \in (\mathbb{F}_2^8)^{12}$ do
4      Compute to 18 middle values denoted by $m^+$, which is the part of
         Equation (16) that involves $\#\mathrm{K}^3[0..11]$;
5      Update table with $T^{\text{neutral}}_{\text{middle}}[m^+] \overset{+}{=} \#\mathrm{K}^3[0..11]$
6  for 12 ■ blue cells $\#\mathrm{K}^3[12..23] \in (\mathbb{F}_2^8)^{12}$ do
7      Compute to 18 middle values denoted by $m^-$, which is part of
         Equation (16) that involves $\#\mathrm{K}^3[0..11]$;
8      Derive entries in $T^{\text{neutral}}_{\text{middle}}[m^-]$ to get $\#\mathrm{K}^3[0..11]$;
9      Derieve 8 ■ blue bytes in $\#\mathrm{SB}^2$ by solving 8 linear equations in
         Equation (17);
       // Now we know all ■ blue cells in $\#\mathrm{SB}^2$ and $\#\mathrm{K}^3$.
10     Compute 8 constants $(\#\mathrm{SB}^5[0..3], \#\mathrm{SB}^6[2,7,8,13]) = (c_0, \ldots, c_7)$ ;
11     Update the table $T^{\text{neutral}}_{\text{blue}}[c_0, \ldots, c_7] \overset{+}{=}$ (■ in $\#\mathrm{K}^3$ and $\#\mathrm{SB}^2$);
       // $2^{8\times 4}$ candidates expected
12 return $T^{\text{neutral}}_{\text{blue}}$;
```

Algorithm 6 generates the solution space of blue neutral words. The time complexity is upper bounded by $2^{8\times 12} = 2^{96}$ simple operations.

The MITM attack procedure for 8-round ***AES-192***. We provide the main attack procedure derived from Figure 12 in Algorithm 7.

The attack complexity. The time complexity of the above MITM pseudo-preimage attack of 8-round AES-192 is about $2^{128-8\times\min(3.5,3.5,4)} = 2^{100}$. The table T^+ dominates the memory complexity with approximately $2^{8\times 4.5} = 2^{36}$. The pre-computation table $T^{\text{neutral}}_{\text{blue}}$ dominates the memory complexity with $2^{8\times 12} = 2^{96}$.

Algorithm 7: MITM attack on 8 rounds of the AES-192 compression function.

```
   // For constants
1  for $(c_0, \ldots, c_7) \in (\mathbb{F}_2^8)^8$ do
2      Initialize $T^+$.
       // For blue neutral words
3      for $2^{8\times 0.5}$ values $c_8$ of the hint pool do
4          Lookup the table $T^{\text{neutral}}_{\text{blue}}[c_0, \cdots, c_7]$ to get candidates of ■ blue cells in
            $\#\text{K}^3$ and $\#\text{SB}^2$;
5          for the $2^{8\times 4}$ values in $\#\text{K}^3$ and $\#\text{SB}^2$ do
6              Compute $m^+$ (the blue part of $\#\text{AT}^7[2,5,8,15]$) and update the
                table $T^+[m^+, c_8] \overset{\pm}{=}$ [blue neutral bytes];
               // $2^{8\times 0}$ entries for each index in $T^+$.
       // For red neutral words
7      for red bytes in $\#\text{SB}^5 \in (\mathbb{F}_2^8)^4$ and the $2^{8\times 0.5}$ values of $c_8$ do
8          Compute $m^-$ (the red part of $\#\text{AT}^7[2,5,8,15] \oplus \#\text{SB}^0[2,5,8,15]$), ,
            which equals $m^+$;
9          Check for entries in $T^+[m^-, c_8]$ to derive 1 blue neutral byte;
10         Compute the blue and red parts of $\#\text{SB}^7[11]$;
11         Check if the red and blue parts of $\#\text{SB}^7[11]$ really produce $c_8$;
           // $2^{8\times(8+4.5-0.5)} = 2^{8\times 12}$ candidates expected
12         Compute the full $\#\text{AT}^7$ and $\#\text{SB}^0$ in both colors to check the
            remaining 12 cells;
13         if The full match is found then
               // $2^{8\times(12-12)} = 1$ candidate expected
14             Output the preimage and stop;
```

Appendix C Improved Pseudo-preimage Attack on 9-round Rijndael-192/192

C.1 Improved Pseudo-preimage Attack on 9-round Rijndael-192/192

The attack configuration is as below:

- The initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 2 bytes (2 in $\#\mathrm{AK}^3$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 44 bytes ($\#\mathrm{K}^7$ and $\#\mathrm{SR}^3$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: zero bytes;
- Consumed DoF for backward $\overleftarrow{\sigma}$: 42 bytes;
- Guessed bytes for blue, red, and both colors $g_{\mathcal{B}}$, $g_{\mathcal{R}}$, $g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{BR}} = 0$ byte and $g_{\mathcal{B}} = g_{\mathcal{BR}} = 0$ bytes;
- Guessed byte equivalents for linearization: $d_{\mathcal{L}} = 0.5$ bytes.
- Matching DoF $d_{\mathcal{M}}$: 2 bytes between $\#\mathrm{AT}^8$ and $\#\mathrm{SB}^0$.

The remaining DoF are composed of

- $d_{\mathcal{B}} - g_{\mathcal{BR}} - d_{\mathcal{L}} = 2 - 0 - 0.5 = 1.5$,
- $d_{\mathcal{R}} - g_{\mathcal{BR}} - d_{\mathcal{L}} = 2 - 0 - 0.5 = 1.5$,
- $d_{\mathcal{M}} - g_{\mathcal{BR}} = 2 - 0 = 2$.

Compute initial values forward neutral bytes (Blue). We choose constants as follows:

- Fix 1 constant for the red parts of $\#\mathrm{AK}^3[19]$.
- Fix 1 constant for the red part of $\#\mathrm{SB}^4[19]$ that defines the red part of $\#\mathrm{MC}^4[7]$.
- Fix 6 constants for the red parts of $\#\mathrm{SB}^5[4, 5, 6, 7, 10, 11]$.
- Fix 12 constants for the red parts of $\#\mathrm{SB}^6[0, 3, 4, 5, 8, 9, 10, 17, 18, 19, 22, 23]$.
- Fix 4 constants for the red parts of $\#\mathrm{SB}^7[0, 5, 10, 15]$.
- Fix 1 constant for $\#\mathrm{K}^4[19]$.
- Fix 2 constants for $\#\mathrm{K}^8[0, 3]$.

Moreover, we will iterate over 20 values:

- Define 2 variables for the red parts of $\#\mathrm{MC}^0[0, 3] = (a_0, a_1)$.
- Define 4 variables for the red parts of $\#\mathrm{MC}^1[0, 15, 18, 21] = (a_2, a_3, a_4, a_5)$.
- Define 2 variables for $\#\mathrm{AK}^3[16, 17] = (a_6, a_7)$.
- Define 1 variable for the red part of $\#\mathrm{AK}^3[18] = a_8$.
- Define 9 variables for the red parts of $\#\mathrm{SB}^2[0, 1, 2, 14, 15, 16, 19, 20, 21]$ that define $\#\mathrm{MC}^2[0, 3, 6, 7, 16, 17, 18, 20, 21] = (a_9, \ldots, a_{17})$.
- Define 2 variables for the red part of $\#\mathrm{MC}^5[5, 8] = (a_{18}, a_{19})$.

Algorithm 8 generates the solution space of red neutral words. The time complexity is upper bounded by $2^{8\times 22.5} = 2^{180}$ simple operations.

The MITM attack procedure for 9-round `Rijndael-192/192`. For forward neutral words (🟦 cells), we will iterate over $\#\mathrm{AK}^3[18, 19]$. We provide the main attack procedure derived from Figure 13 in Algorithm 9.

The attack complexity. The time complexity of the above MITM pseudo-preimage attack of 9-round `Rijndael-192/192` is about $2^{192-8\times\min(1.5,1.5,2)} = 2^{180}$. The table $T^{\mathrm{neutral}}_{\mathrm{red}}$ needs memory for $2^{8\times 22.5} = 2^{180}$ entries.

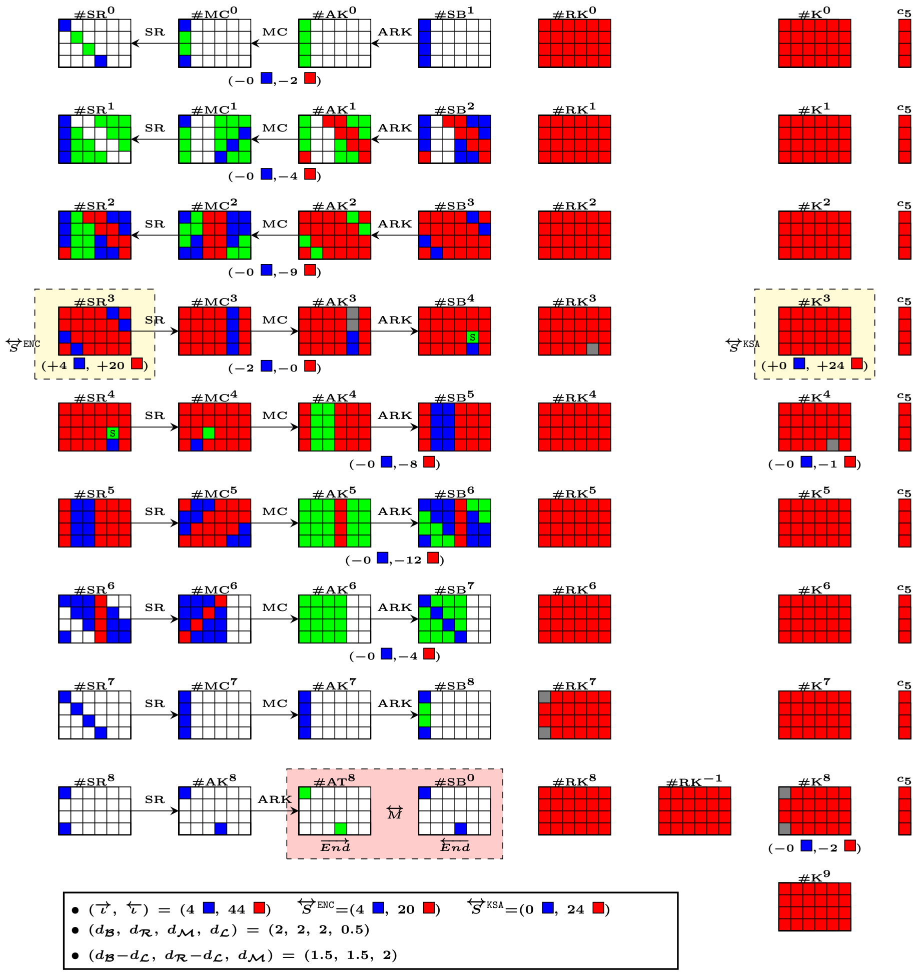

• $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (4$ 🟦, 44 🟥) $\overleftrightarrow{S}^{\texttt{ENC}} = (4$ 🟦, 20 🟥) $\overleftrightarrow{S}^{\texttt{KSA}} = (0$ 🟦, 24 🟥)
• $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, d_{\mathcal{L}}) = (2, 2, 2, 0.5)$
• $(d_{\mathcal{B}} - d_{\mathcal{L}}, d_{\mathcal{R}} - d_{\mathcal{L}}, d_{\mathcal{M}}) = (1.5, 1.5, 2)$

Fig. 13. An MITM pseudo-preimage attack of 9-round `Rijndael-192/192`.

Algorithm 8: Compute backward neutral words (red) for 9-round Rijndael-192/192.

```
1  Initiate a table $T_{\text{red}}^{\text{neutral}}$;
2  for all values of $\#\text{K}^7[0,3]$ do
3      Derive $\#\text{K}^7[20,21]$ such that $\#\text{K}^7[0,3] \oplus S(\#\text{K}^7[21]), S(\#\text{K}^7[20])$ produce
         the expected constants of $\#\text{K}^8[0,3]$.
       // $2^{8\times(2-0)}$ candidates expected
4  for all values of $\#\text{K}^7[7,11,15]$ do
5      Derive $\#\text{K}^7[19]$ such that $\#\text{K}^7[7] \oplus \#\text{K}^7[11] \oplus \#\text{K}^7[15] \oplus \#\text{K}^7[19]$ yields the
         constant of $\#\text{K}^4[19]$.
       // $2^{8\times(3-0)}$ candidates expected
6  Construct $\#\text{K}^7$ by combining the candidates with all possible values for the
     remaining 16 bytes. At this stage, we know the full blue part of $\#\text{K}^4$ and can
     derive the red parts of all key words, including $\#\text{RK}^{-1}[0,15]$ and $\#\text{RK}^8[0,15]$.
   // $2^{8\times(3+2+16)} = 2^{8\times21}$ candidates expected
7  Compute $\#\text{AK}^4[4..7]$ from $\#\text{RK}^4[4..7]$ and the constants of the red parts of
     $\#\text{SB}^5[4..7]$. Derive $\#\text{MC}^4[4..7]$ and keep only those values that yield the
     expected constant of the red part of $\#\text{MC}^4[7]$.
   // $2^{8\times(21-1)} = 2^{8\times20}$ candidates expected
8  Compute $\#\text{AK}^6[0,5,10,15]$ from $\#\text{RK}^6[0,5,10,15]$ and the constants of the red
     parts of $\#\text{SB}^7[0,5,10,15]$. With the 12 fixed constants of the red parts of
     $\#\text{MC}^6[0,1,2,4,5,7,8,10,11,13,14,15]$ (defined by the constants in $\#\text{SB}^6$),
     derive the red parts of $\#\text{MC}^6[0..15]$, $\#\text{AK}^6[0..15]$, and $\#\text{SB}^7[0..15]$ column by
     column.
   // $2^{8\times20}$ candidates expected
9  for those values $2^{8\times20}$ candidates do
10     Compute backwards to $\#\text{SB}^6[12..15]$. Combine them with
         $\#\text{RK}^5[0,3,4,5,8,9,10,12..15,17,18,19,22,23]$ and the fixed red parts of
         $\#\text{SB}^6$, derive $\#\text{AK}^5[0,3,4,5,8,9,10,12..15,17,18,19,22,23]$.
11     for all possible values of $\#\text{MC}^5[5,8] = (a_{18}, a_{19})$ do
12         Combine the knowledge of partial $\#\text{AK}^5$ with the constants of
             $\#\text{MC}^5[1,2,4,19,22,23]$ defined from $\#\text{SB}^5[4,5,6,7,10,11]$ to derive
             the full red part of $\#\text{MC}^5$ column by column.
           // $2^{8\times(20+2)} = 2^{8\times22}$ candidates expected
13         Compute backwards to $\#\text{SR}^4$. Guess the 16 possible values of $a_{20}$ of
             $\#\text{SR}^4[18]$ to compute to $\#\text{AK}^3$.
           // $2^{8\times(22+0.5)} = 2^{8\times22.5}$ candidates expected
14         Define $\#\text{AK}^3[16,17] = (a_6, a_7)$ and the red part of $\#\text{AK}^3[18] = a_8$.
15         Compute backwards to $\#\text{MC}^2$. Define the red parts of
             $\#\text{MC}^2[0,3,6,7,16,17,18,20,21] = (a_9, \ldots, a_{17})$.
16         Compute backwards to $\#\text{MC}^1[0..3,12..23]$. Define the red parts of
             $\#\text{MC}^1[0,15,18,21] = (a_2, a_3, a_4, a_5)$.
17         Compute backwards to $\#\text{MC}^0[0..3]$. Define the red parts of
             $\#\text{MC}^0[0,3] = (a_0, a_1)$.
18         Update the table $T_{\text{red}}^{\text{neutral}}[a_0, \ldots, a_{19}, a_{20}] \overset{\pm}{=} (K^7, \#\text{RK}^8[0,15],$
             $\#\text{RK}^{-1}[0,15])$;
           // For each of the $2^{8\times(20+0.5)}$ buckets for each value of
               $(a_0, \ldots, a_{19}, a_{20})$, $2^{8\times(22.5-20.5)} = 2^{8\times2}$ candidates expected
19 return $T_{\text{red}}^{\text{neutral}}$;
```

Algorithm 9: MITM attack on 9 rounds of the `Rijndael-192/192` compression function.

```
     // For constants
1    for $(a_0, \ldots, a_{19}) \in (\mathbb{F}_2^8)^{20}$ combined with the 16 values $a_{20}$ of the hint pool do
         // For red neutral words
2        for all $2^{8\times 2}$ red candidates in $T_{red}^{neutral}[a_0, \cdots, a_{19}, a_{20}]$ do
3            Retrieve the red parts of $\#\mathrm{AT}^8[0,15] = \#\mathrm{RK}^8[0,15] \oplus \#\mathrm{RK}^{-1}[0,15]$;
             // $2^{8\times(20.5+2)} = 2^{8\times 22.5}$ candidates expected
         // For blue neutral words
4        for $\#\mathrm{AK}^3[18,19] \in (\mathbb{F}_2^8)^2$ do
             // $2^{8\times(20.5+2)} = 2^{8\times 22.5}$ candidates expected
5            Compute forwards to $\#\mathrm{AK}^8[0,15]$;
6            Compute backwards to $\#\mathrm{SB}^0[0,15]$;
7            Check for the candidates whose red part of $\#\mathrm{RK}^8[0,15] \oplus \#\mathrm{RK}^{-1}[0,15]$
               equals the red part of
               $\#\mathrm{RK}^8[0,15] \oplus \#\mathrm{RK}^{-1}[0,15] \oplus \#\mathrm{AT}^8[0,15] \oplus \#\mathrm{SB}^0[0,15]$;
8            if the two-byte partial match is passed then
                 // $2^{8\times(20+0.5+2+2-2)} = 2^{8\times 22.5}$ candidates expected
9                Check if the red and red parts of $\#\mathrm{SR}^4[18]$ really produce $a_{20}$;
                 // $2^{8\times(22.5-0.5)} = 2^{8\times 22}$ candidates expected
10               Compute the full $\#\mathrm{AT}^9$ and $\#\mathrm{SB}^0$ in both colors to check the
                   remaining 22 cells;
11               if The full match is found then
                     // $2^{8\times(22-22)} = 1$ candidate expected
12                   Output the preimage and stop;
```

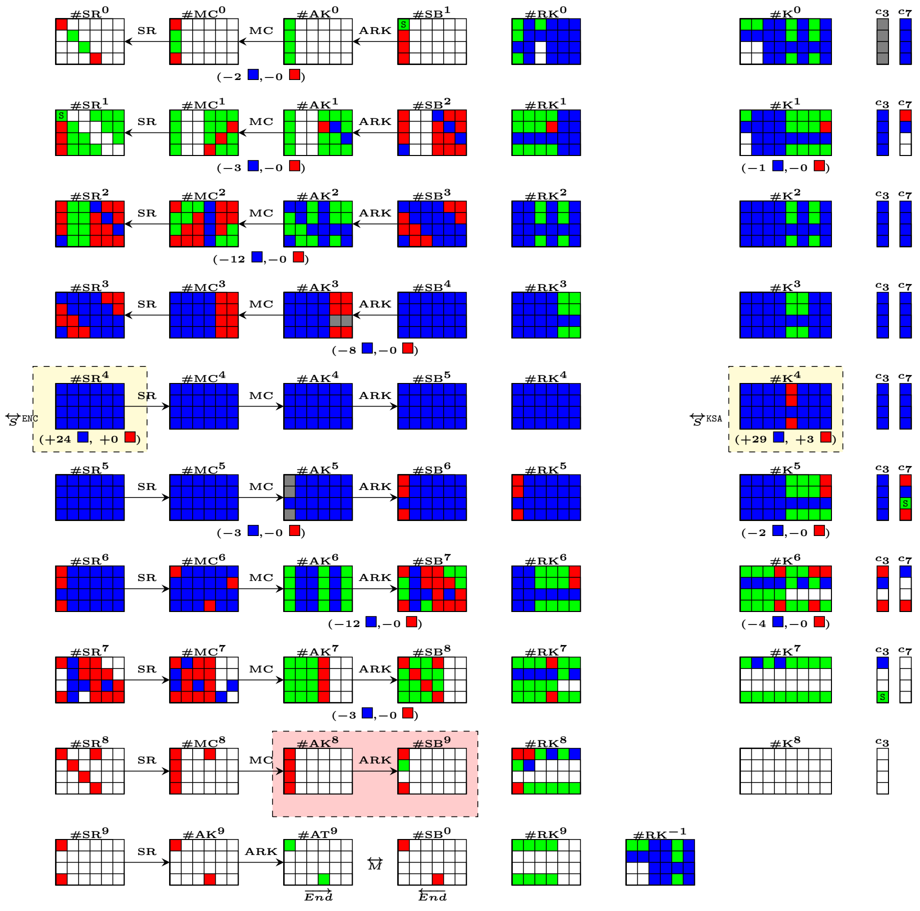
C.2 Improved Pseudo-preimage Attack on 10-round `Rijndael-192/256`

The attack configuration is as below:

- The initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 53 bytes (29 in K^4 and 24 in $\#\mathrm{SR}^4$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 3 bytes ($K^4[2]$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: 50 bytes;
- Consumed DoF for backward $\overleftarrow{\sigma}$: 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 $g_{\mathcal{B}} = g_{\mathcal{BR}} = 0$ bytes;
- Guessed byte equivalents for linearization: $d_{\mathcal{L}} = 1.5$ bytes;
- Matching DoF $d_{\mathcal{M}}$: 2 bytes between $\#\mathrm{AT}^9$ and $\#\mathrm{SB}^0$.

The remaining DoF are composed of

- $d_{\mathcal{B}} - g_{\mathcal{BR}} - d_{\mathcal{L}} = 3 - 0 - 3 \times 0.5 = 1.5$,
- $d_{\mathcal{R}} - g_{\mathcal{BR}} - d_{\mathcal{L}} = 3 - 0 - 3 \times 0.5 = 1.5$,
- $d_{\mathcal{M}} - g_{\mathcal{BR}} = 2 - 0 = 2$.



- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (53$ 🟦, 3 🟥$)$ $\overleftrightarrow{S}^{\mathrm{ENC}}=(24$ 🟦, 0 🟥$)$ $\overleftrightarrow{S}^{\mathrm{KSA}}=(29$ 🟦, 3 🟥$)$
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, d_{\mathcal{L}}) = (3, 3, 2, 1.5)$
- $(d_{\mathcal{B}}-d_{\mathcal{L}}, d_{\mathcal{R}}-d_{\mathcal{L}}, d_{\mathcal{M}}) = (1.5, 1.5, 2)$

Fig. 14. An MITM pseudo-preimage attack of 10-round `Rijndael-192/256`.

Compute initial values forward neutral bytes (Blue). We choose constants as follows:

- Fix 3 constants for the blue parts of $\#\mathrm{MC}^1[15, 18, 21]$
- Fix 12 constants for the blue parts of $\#\mathrm{MC}^2[0, 3, 6, 7, 9, 10, 11, 16, 17, 18, 20, 21]$
- Fix 6 constants for the blue parts of $\#\mathrm{AK}^3[16, 17, 19, 20, 21, 23]$
- Fix 4 constants for the blue parts of $\#\mathrm{SB}^7[12, 13, 14, 15]$
- Fix 1 constant for the blue part of $K^1[29]$.
- Fix 2 constants for the blue parts of $K^5[28, 29]$.
- Fix 4 constants for the blue parts of $K^6[12, 15, 24, 27]$.

Moreover, we will iterate over 18 values:

- Define 2 variables for the blue parts of $\#\mathrm{MC}^0[0, 3] = (a_0, a_1)$.
- Define 2 variables for $\#\mathrm{AK}^3[18, 22] = (a_2, a_3)$.
- Define 3 variables for $\#\mathrm{AK}^5[0, 1, 3] = (a_4, a_5, a_6)$.
- Define 8 variables for the blue parts of $\#\mathrm{SB}^7[0, 3, 8, 9, 17, 19, 22, 23] = (a_7, \ldots, a_{14})$.
- Define 3 variables for the blue parts of $\#\mathrm{SB}^8[0, 5, 10] = (a_{15}, a_{16}, a_{17})$.

Algorithm 10 generates the solution space of blue neutral words. The time complexity is upper bounded by $2^{8\times 23} = 2^{184}$ simple operations since Line 7 needs to consider that amount of candidates. Note that this is slightly suboptimal and there may be an algorithm that achieves a time complexity of $2^{8\times 22.5} = 2^{180}$ operations. Though, we consider the theoretical time complexity, which will be dominated by conducting $2^{8\times 23.5}$ matching operations. Thus, the solution-space construction has relatively little impact on the final time complexity.

The MITM attack procedure for 10-round ***`Rijndael-192/256`****.* For backward neutral words (🟥 cells), we will iterate over $K^4[16, 17, 19]$. We provide the main attack procedure derived from Figure 14 in Algorithm 11.

The attack complexity. The time complexity of the above MITM pseudo-preimage attack of 10-round `Rijndael-192/256` is about $2^{192-8\times\min(0.5,0.5,0.5)} = 2^{188}$. The table $T^{\mathrm{neutral}}_{\mathrm{blue}}$ needs memory for $2^{8\times 22.5} = 2^{184}$ entries.

Appendix D Improved Preimage Attacks of Whirlpool

D.1 Improved Preimage Attack of 6-round Whirlpool

The improved result on 6-round `Whirlpool` is presented in Figure 15. By using the **SIM** technique (at Round 4 and 5), the XOR DoF costs can be compensated at Round 5 with 29 🟦 blue bytes DoF cost, which reduces the time complexity by a factor of 2^{24} compared to previous best MITM pseudo-preimage attack. It leads to the attack configuration as below:

Algorithm 10: Compute forward neutral words (blue) for 10-round `Rijndael-192/256`.

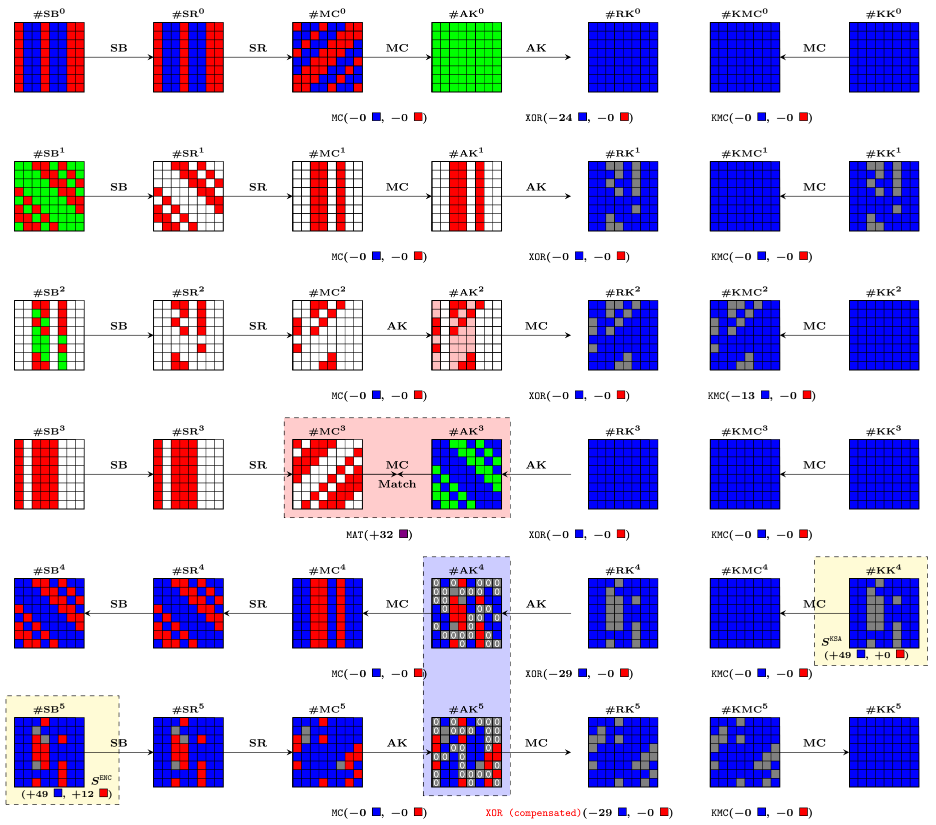
```
1   Initiate a table $T_{\text{blue}}^{\text{neutral}}$;
    for all bytes of the topmost row of $\#K^4$ and $\#K^4[29]$, $\#K^4[0,4,8,12,20,24,28,29]$ do
        Derive $\#K^5[28]$, $\#K^6[12]$, $\#K^6[24]$. Keep only those values which lead to the fixed constant blue parts of $\#K^5[28]$, $\#K^6[12]$, $\#K^6[24]$;
        // $2^{8\times(9-3)}$ candidates expected
    for all bytes of the second row of $\#K^4$ and $\#K^4[30]$ do
        Derive $\#K^5[29]$ and $\#K^1[29]$. Keep only those values which lead to the fixed constant blue parts of $\#K^1[29]$ and $\#K^5[29]$;
        // $2^{8\times(9-2)}$ candidates expected
    for all bytes of the bottommost row of $\#K^4$ and $\#K^4[28]$ do
        Keep only those values which lead to the fixed constant blue parts of $\#K^6[15]$, $\#K^6[27]$;
        // $2^{8\times(9-2)}$ candidates expected
    Merge all surviving blue parts of $\#K^4$, including Row 3. At this stage, we know the full blue part of $\#K^4$ and can derive the blue parts of all key words backwards to $\#K^0$ and forwards to the full $\#K^5$;
    // $2^{8\times(29-7)} = 2^{8\times 22}$ candidates expected
2   Compute $\#AK^6[12..15]$ and $\#MC[12..15]$. Keep only those values that lead to constant blue part of $\#MC^6[15]$;
    // $2^{8\times(22-1)} = 2^{8\times 21}$ candidates expected
3   Derive the blue parts of $\#AK^2[1,2,6,7,11,16,20,21]$ from $\#RK^2$. Combine them with the constant blue parts of $\#MC^2[0,3,6,7,9,10,11,16,17,18,20,21]$, derive the blue parts of $\#MC^2[1,2,4,5,8,19,22,23]$ and $\#SB^3[0,3,4,5,8,9,10,17,18,19,22,23]$ column by column. For each column, we have exactly four bytes in $\#MC^2$ and $\#AK^2$ together;
    // $2^{8\times(21-0)} = 2^{8\times 21}$ candidates expected
4   Derive the blue parts of $\#AK^1[13,14,15,16,18,19,20,21,23]$ from $\#RK^1$. Combine them with the constant blue parts of $\#MC^1[15,18,21]$ and derive the blue parts of $\#MC^1[12..23]$, $\#AK^1[12..25]$ and thus $\#SB^2[12,17,22]$ column by column. For each column, we have exactly four bytes in $\#MC^2$ and $\#AK^2$ together;
    // $2^{8\times(21-0)} = 2^{8\times 21}$ candidates expected
5   for those values $2^{8\times 21}$ candidates of the blue parts of $\#K^2$ and $\#SB^3$ do
6       Compute from $\#SB^3$ to $\#AK^3$;
7       for the blue parts of $\#AK^3[18,22] = (a_0, a_1)$ do
            // $2^{8\times(21+2)} = 2^{8\times 23}$ candidates expected
8           Combine those with the constants for the blue parts of $\#AK^3[16,17,19..21,23]$ and $\#RK^3$ to derive $\#SB^4$ and compute forwards to $\#AK^5$. Denote $\#AK^5[0,1,3] = (a_2, a_3, a_4)$.
9           With the earlier derived $\#MC^6[12..15]$, compute from them to $\#AK^5[12,17,22]$. Keep only those values that match in $\#AK^5[12,17,22]$;
            // $2^{8\times(23-3)} = 2^{8\times 20}$ candidates expected
10          Guess $\#SB^2[3]$ and the four bits of $z^1[0]$. Together with $\#RK^0[0..3]$, compute $\#MC^0[0..3]$. Denote $MC^0[0,3] = (a_0, a_1)$.
            // $2^{8\times(20+1+0.5)} = 2^{8\times 21.5}$ candidates expected
11          Compute from $\#AK^5$ to $\#AK^6$ and the blue parts of $\#SB^7$. Denote the blue parts of $\#SB^7[0,3,8,9,17,19,22,23]$ as $(a_7,\ldots,a_{14})$;
12          Compute to $\#AK^7[0..15]$ and derive $\#SB^8[0..15]$. Denote the blue parts of $\#SB^8[0,5,10] = (a_{15}, a_{16}, a_{17})$;
13          Guess the four bits of $z^5[31]$ in the key schedule for the S-box of $K^5[31]$, which is necessary to derive $\#RK^7[2,6,10,14]$. Guess the four bits of $z^7[15]$ in the key schedule for the S-box of $K^7[15]$, which is necessary to compute $\#RK^9[15]$;
            // $2^{8\times(21.5+0.5+0.5)} = 2^{8\times 22.5}$ candidates expected
14          Compute $\#RK^9[0,15]$ and $\#RK^1[0,15]$;
15          Update the table $T_{\text{blue}}^{\text{neutral}}[a_0,\ldots,a_{17},z^1[0],z^5[31],z^7[15]] \overset{+}{=} (K^4, \#RK^9[0,15], \#RK^{-1}[0,15])$;
            // For each of the $2^{8\times(18+1.5)}$ buckets for each value of $(a_0,\ldots,a_{17},z^1[0],z^5[31],z^7[15])$, $2^{8\times(22.5-19.5)} = 2^{8\times 3}$ candidates expected
16  return $T_{\text{blue}}^{\text{neutral}}$;
```

Algorithm 11: MITM attack on 10 rounds of the `Rijndael-192/256` compression function.

```
   // For constants
1  for $(a_0, \ldots, a_{17}) \in (\mathbb{F}_2^8)^{18}$ combined with the $16^3$ guesses of $z^1[0], z^5[31], z^7[15]$ do
       // For blue neutral words
2      for all $2^{8\times3}$ blue candidates do
3          Retrieve the blue part of $\#\mathrm{AT}^9[0,15] = \#\mathrm{RK}^9[0,15] \oplus \#\mathrm{RK}^{-1}[0,15]$;
           // $2^{8\times(19.5+3)} = 2^{8\times22.5}$ candidates expected
       // For red neutral words
4      for $K^4[16,17,19] \in (\mathbb{F}_2^8)^3$ do
           // $2^{8\times(19.5+3)} = 2^{8\times22.5}$ candidates expected
5          Derive the red parts in the key schedule including $\#\mathrm{RK}^9[0,15]$ and $\#\mathrm{RK}^{-1}[0,15]$;
6          Derive $\#\mathrm{SB}^6[0,1,3]$ and compute forwards to $\#\mathrm{AK}^9[0,15]$;
7          Derive $\#\mathrm{AK}^3[16..23]$ and compute backwards to $\#\mathrm{SB}^0[0,15]$;
8          Check for the blue candidates in $T_{\mathrm{blue}}^{\mathrm{neutral}}$ whose blue part of $\#\mathrm{RK}^9[0,15] \oplus \#\mathrm{RK}^{-1}[0,15]$ equals the red part of $\#\mathrm{RK}^9[0,15] \oplus \#\mathrm{RK}^{-1}[0,15] \oplus \#\mathrm{AT}^9[0,15] \oplus \#\mathrm{SB}^0[0,15]$;
9          if the two-byte partial match is passed then
               // $2^{8\times(18+1.5+3+3-2)} = 2^{8\times23.5}$ candidates expected
10             Check if the red and blue parts of $\#\mathrm{SR}^1[0]$ really produce $z^1[0]$ and similarly if the red and blue parts of $\#\mathrm{K}^5[31]$ and $\#\mathrm{K}^7[15]$ produce $z^5[31]$ and $z^7[15]$ from $T_{\mathrm{blue}}^{\mathrm{neutral}}$;
               // $2^{8\times(23.5-1.5)} = 2^{8\times22}$ candidates expected
11             Compute the full $\#\mathrm{AT}^9$ and $\#\mathrm{SB}^0$ in both colors to check the remaining 22 cells;
12             if The full match is found then
                   // $2^{8\times(22-22)} = 1$ candidate expected
13                 Output the preimage and stop;
```

- Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 69 bytes (among 49 blue cells in $\#\mathrm{SB}^5$, 29 out of 49 cells are *set to be equal to* the corresponding cells in $\#\mathrm{KK}^4$, and thus can be cancelled to zero constant ⓪ in $\#\mathrm{AK}^4$ and $\#\mathrm{AK}^5$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 12 bytes (in $\#\mathrm{SB}^5$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: 37 bytes (13 for $\#\mathrm{KK}^2 \xrightarrow{\mathrm{MC}^{-1}} \#\mathrm{KMC}^2$ and 24 for $\#\mathrm{AK}^0 \oplus \#\mathrm{RK}^0 = \#\mathrm{SB}^1$);
- Consumed DoF for backward $\overleftarrow{\sigma}$: zero bytes;
- Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}} : g_{\mathcal{B}} = g_{\mathcal{BR}} = 0$ byte and $g_{\mathcal{R}} = 20$ bytes;
- Matching DoF $d_{\mathcal{M}}$: 32 bytes between $\#\mathrm{MC}^3$ and $\#\mathrm{AK}^3$.

Then, the remaining DoF is $d_{\mathcal{B}} - g_{\mathcal{R}} = 12, d_{\mathcal{R}} = 12, d_{\mathcal{M}} - d_{\mathcal{R}} = 12$.



- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (49+20)=69$ 🟦, $(12+0)=12$ 🟥)
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}) = (+32$ 🟦, $+12$ 🟥, $+32$ 🟪, $+0$ 🟦, $+20$ 🟥, $+0$ 🟥)
- $(d_{\mathcal{B}}-g_{\mathcal{R}}, d_{\mathcal{R}}-g_{\mathcal{B}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}}) = (12, 12, 12)$

Fig. 15. An MITM pseudo-preimage attack of 6-round `Whirlpool`.

Compute initial values for forward neutral words (Blue). To get the initial values of forward neutral words, one should focus on the constraints among states $\#\mathrm{KK}^2$ and $\#\mathrm{KMC}^2$, $\#\mathrm{RK}^0$, $\#\mathrm{AK}^0$ and $\#\mathrm{SB}^1$ as below

– For $\#\mathrm{KK}^2$ and $\#\mathrm{KMC}^2$, there are 13 bytes MC DoF cost for forward neutral words, which applies the constraints in Equation (18) as below

$$\begin{pmatrix} - & - & b_3 & b_6 & - & b_{12} & - & - \\ - & - & b_4 & - & b_9 & - & - & - \\ b_0 & - & - & b_7 & - & - & - & - \\ b_1 & - & b_5 & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ b_2 & - & - & - & - & - & - & - \\ - & - & - & - & b_{10} & - & - & - \\ - & - & - & b_8 & b_{11} & - & - & - \end{pmatrix} = \mathrm{MC}^{-1} \cdot \begin{pmatrix} \#\mathrm{KK}^2_0 & \#\mathrm{KK}^2_8 & \cdots & \#\mathrm{KK}^2_{56} \\ \#\mathrm{KK}^2_1 & \#\mathrm{KK}^2_9 & \cdots & \#\mathrm{KK}^2_{57} \\ \#\mathrm{KK}^2_2 & \#\mathrm{KK}^2_{10} & \cdots & \#\mathrm{KK}^2_{58} \\ \#\mathrm{KK}^2_3 & \#\mathrm{KK}^2_{11} & \cdots & \#\mathrm{KK}^2_{59} \\ \#\mathrm{KK}^2_4 & \#\mathrm{KK}^2_{12} & \cdots & \#\mathrm{KK}^2_{60} \\ \#\mathrm{KK}^2_5 & \#\mathrm{KK}^2_{13} & \cdots & \#\mathrm{KK}^2_{61} \\ \#\mathrm{KK}^2_6 & \#\mathrm{KK}^2_{14} & \cdots & \#\mathrm{KK}^2_{62} \\ \#\mathrm{KK}^2_7 & \#\mathrm{KK}^2_{15} & \cdots & \#\mathrm{KK}^2_{63} \end{pmatrix}, \tag{18}$$

where b_i $(0 \le i \le 12)$ are the chosen constants for $\#\mathrm{KMC}^2$.

– For $\#\mathrm{AK}^0, \#\mathrm{RK}^0$ and $\#\mathrm{SB}^1$, there are 24 bytes XOR DoF cost for forward neutral words, which applies the constraints in Equation (19) as below

$$\begin{pmatrix} \#\mathrm{AK}^0_0 & \#\mathrm{AK}^0_8 & \cdots & \#\mathrm{AK}^0_{56} \\ \#\mathrm{AK}^0_1 & \#\mathrm{AK}^0_9 & \cdots & \#\mathrm{AK}^0_{57} \\ \#\mathrm{AK}^0_2 & \#\mathrm{AK}^0_{10} & \cdots & \#\mathrm{AK}^0_{58} \\ \#\mathrm{AK}^0_3 & \#\mathrm{AK}^0_{11} & \cdots & \#\mathrm{AK}^0_{59} \\ \#\mathrm{AK}^0_4 & \#\mathrm{AK}^0_{12} & \cdots & \#\mathrm{AK}^0_{60} \\ \#\mathrm{AK}^0_5 & \#\mathrm{AK}^0_{13} & \cdots & \#\mathrm{AK}^0_{61} \\ \#\mathrm{AK}^0_6 & \#\mathrm{AK}^0_{14} & \cdots & \#\mathrm{AK}^0_{62} \\ \#\mathrm{AK}^0_7 & \#\mathrm{AK}^0_{15} & \cdots & \#\mathrm{AK}^0_{63} \end{pmatrix} \oplus \begin{pmatrix} \#\mathrm{RK}^0_0 & \#\mathrm{RK}^0_8 & \cdots & \#\mathrm{RK}^0_{56} \\ \#\mathrm{RK}^0_1 & \#\mathrm{RK}^0_9 & \cdots & \#\mathrm{RK}^0_{57} \\ \#\mathrm{RK}^0_2 & \#\mathrm{RK}^0_{10} & \cdots & \#\mathrm{RK}^0_{58} \\ \#\mathrm{RK}^0_3 & \#\mathrm{RK}^0_{11} & \cdots & \#\mathrm{RK}^0_{59} \\ \#\mathrm{RK}^0_4 & \#\mathrm{RK}^0_{12} & \cdots & \#\mathrm{RK}^0_{60} \\ \#\mathrm{RK}^0_5 & \#\mathrm{RK}^0_{13} & \cdots & \#\mathrm{RK}^0_{61} \\ \#\mathrm{RK}^0_6 & \#\mathrm{RK}^0_{14} & \cdots & \#\mathrm{RK}^0_{62} \\ \#\mathrm{RK}^0_7 & \#\mathrm{RK}^0_{15} & \cdots & \#\mathrm{RK}^0_{63} \end{pmatrix} = \begin{pmatrix} - & - & c_6 & c_9 & - & c_{15} & - & - \\ - & - & - & c_{10} & c_{12} & - & c_{18} & - \\ - & - & - & - & c_{13} & c_{16} & - & c_{21} \\ c_0 & - & - & - & - & c_{17} & c_{19} & - \\ - & c_3 & - & - & - & - & c_{20} & c_{22} \\ c_1 & - & c_7 & - & - & - & - & c_{23} \\ c_2 & c_4 & - & c_{11} & - & - & - & - \\ - & c_5 & c_8 & - & c_{14} & - & - & - \end{pmatrix}, \quad (19)$$

where c_i $(0 \leq i \leq 23)$ are the chosen constants for $\#\mathrm{SB}^1$.

However, one still needs an efficient method to obtain blue neutral words satisfying the constraints (in Equation (18) and (19)) in a non-linear system. We provide Algorithm 12 to efficiently obtain the solution space of blue neutral words with the time complexity $2^{8\times 49} = 2^{392}$ and memory complexity $2^{8\times 36} = 2^{288}$.

Algorithm 12: Compute forward neutral words (blue) for 6-round `Whirlpool`

```
1 Initiate a table $T^{\text{neutral}}_{\text{blue}}$;
  // The index list $I$ corresponds with constant cells in Equation (19)
2 Set list $I =$
    $[3, 5, 6, 12, 14, 15, 16, 21, 23, 24, 25, 30, 33, 34, 39, 40, 42, 43, 49, 51, 52, 58, 60, 61]$;
3 Fix 15 ▨ constant cells in $\#\mathrm{KK}^4$ to zero;
4 for 49 ■ blue cells in $\#\mathrm{KK}^4 \in (\mathbb{F}_2^8)^{49}$ do
5     Compute backwards to $\#\mathrm{KMC}^2$;
6     if 13 ▨ constants cells in $\#\mathrm{KMC}^2$ are all zero then
          // $2^{8\times(49-13)} = 2^{8\times 36}$ candidates expected
7         Compute backwards to $\#\mathrm{KK}^0$;
          // $2^{8\times(36-24)} = 2^{8\times 12}$ candidates expected for each row
8         Update to table with $T^{\text{neutral}}_{\text{blue}}[\#\mathrm{KK}^0[I]] \mathrel{+}= \#\mathrm{KK}^4$;
9 return $T^{\text{neutral}}_{\text{blue}}$;
```

Compute initial values backward neutral bytes (Red). As there is no DoF cost for backward neutral words, to obtain the corresponding initial values, one only needs to enumerate all possible values of 12 red cells in $\#\mathrm{SB}^5$.

The MITM attack procedure for 6-round `Whirlpool`. The main attack procedure derived from Figure 15 is provided in Algorithm 13 as below.

The attack complexity. The time complexity of the above MITM pseudo-preimage attack of 6-round `Whirlpool` is about $2^{512-8\times\min(12,12,12)} = 2^{416}$. The table $T^{\text{neutral}}_{\text{blue}}$ dominates the memory complexity with $2^{36\times 8} = 2^{288}$.

Algorithm 13: MITM attack on 6-round `Whirlpool` compression function

```
1  Fix 3 ■ constant cells in $\#\mathrm{AK}^5$ and 15 ■ constant cells in $\#\mathrm{KK}^4$ to all zero;
2  Fix 13 constants $(b_0, \cdots, b_{12})$ in Equation (18) to all zero;
3  Fix 4 constants $(c_{20}, c_{21}, c_{22}, c_{23})$ in Equation (19) to all zero;
4  Set list $I =$
     $[3,5,6,12,14,15,16,21,23,24,25,30,33,34,39,40,42,43,49,51,52,58,60,61]$;
5  Call Algorithm 12 to build $T^{\mathrm{neutral}}_{\mathrm{blue}}$;
6  for $C = (c_0, \cdots, c_{19}, 0, 0, 0, 0) \in (\mathbb{F}_2^8)^{20}$ do
       // For red neutral words
7      for 20 🟥 guessed red cells in $\#\mathrm{AK}^2 \in (\mathbb{F}_2^8)^{20}$ do
8          for 12 🟥 red cells in $\#\mathrm{SB}^5 \in (\mathbb{F}_2^8)^{12}$ do
9              Compute forwards and backwards to cells with red information in
                 $\#\mathrm{MC}^3, \#\mathrm{AK}^3$ and store in a table $T^-$ (with size $2^{8\times 32}$);
       // For blue neutral words
10     for 20 🟦 blue cells in $\#\mathrm{AK}^5 \in (\mathbb{F}_2^8)^{20}$ do
11         Compute forward to $\#\mathrm{AK}^0$;
12         Search table with $T^{\mathrm{neutral}}_{\mathrm{blue}}[C \oplus \#\mathrm{AK}^0[I]]$ and obtain $\#\mathrm{KK}^4$;
13         Compute backward to $\#\mathrm{AK}^3$ according to $2^{8\times 12}$ $\#\mathrm{KK}^4$ on average;
14         Check in the table $T^-$;
15         if $\#\mathrm{MC}^3$ and $\#\mathrm{AK}^3$ pass the partial match then
               // $2^{8\times(20+32+32-32)} = 2^{8\times 52}$ candidates expected
16             Compute to check 20 🟥 guessed red cells in $\#\mathrm{AK}^2$;
17             if Guess values are correct then
                   // $2^{8\times(52-20)} = 2^{8\times 32}$ candidates expected
18                 Compute to match the rest 32 cells;
19                 if The full match is found then
                       // $2^{8\times(32-32)} = 1$ candidate expected
20                     Output the preimage and stop;
```

D.2 Preimage Attack of 7.75-round Whirlpool

The previous best preimage attacks on 7-round `Whirlpool` in [8], could not be extended by another partial round due to the AK operation in the final round. We can use the **SIM** technique at the penultimate round $\#\mathrm{AK}^6$, so that more constant cells can be guaranteed to pass the final MC operation. More precisely, we want $\#\mathrm{AK}^6$ to have *equally colored cells*[10]. Finally, for the first time, we obtain an MITM attack of 7.75-round `Whirlpool`, which is presented in Figure 16 and has the attack configuration as below

– Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 16 bytes (16 blue cells in $\#\mathrm{SB}^6$ are *set to be equal to* the corresponding cells in $\#\mathrm{KK}^5$, and thus can be cancelled to zero constant ▣ in $\#\mathrm{AK}^5$ and $\#\mathrm{AK}^6$);

[10] That is with only blue and gray cells or only red and gray cells.

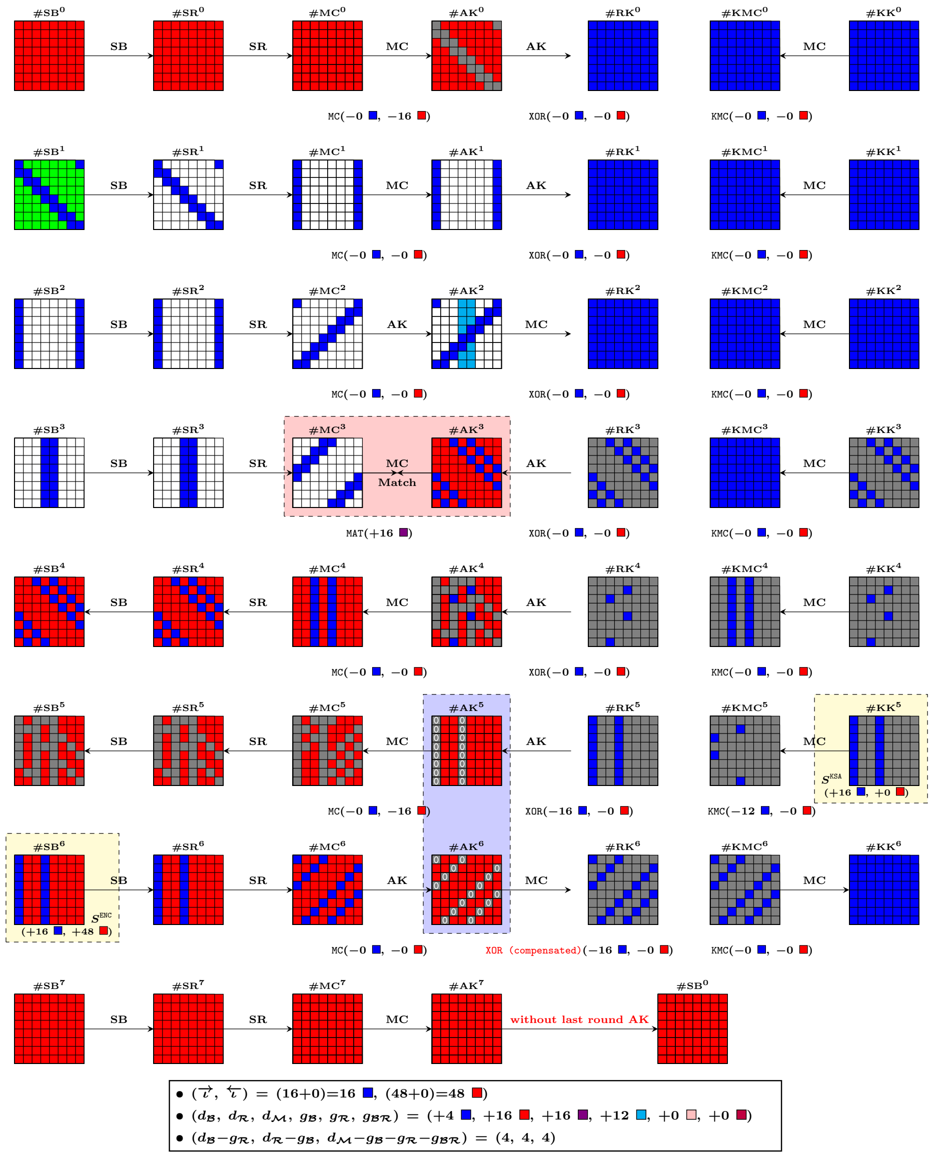

#SB⁰ SB #SR⁰ SR #MC⁰ MC #AK⁰ AK #RK⁰ #KMC⁰ MC #KK⁰
MC(−0 🟦, −16 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
MAT(+16 🟪)
XOR (compensated)(−16 🟦, −0 🟥)
KMC(−12 🟦, −0 🟥)
S^KSA (+16 🟦, +0 🟥)
S^ENC (+16 🟦, +48 🟥)
Match
without last round AK
• $(\overrightarrow{\iota}, \overleftarrow{\iota})$ = (16+0)=16 🟦, (48+0)=48 🟥)
• $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}})$ = (+4 🟦, +16 🟥, +16 🟪, +12 🟦, +0 🟥, +0 🟪)
• $(d_{\mathcal{B}}-g_{\mathcal{R}}, d_{\mathcal{R}}-g_{\mathcal{B}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}})$ = (4, 4, 4)

Fig. 16. An MITM pseudo-preimage attack of 7.75-round `Whirlpool`.

- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 48 bytes (in $\#\mathrm{SB}^6$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: 12 bytes (12 for $\#\mathrm{KK}^5 \xrightarrow{\mathrm{MC}^{-1}} \#\mathrm{KMC}^5$);
- Consumed DoF for backward $\overleftarrow{\sigma}$: 32 bytes (16 for $\#\mathrm{AK}^5 \xrightarrow{\mathrm{MC}^{-1}} \#\mathrm{MC}^5$ and 16 for $\#\mathrm{MC}^0 \xrightarrow{\mathrm{MC}} \#\mathrm{AK}^0$);
- Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{BR}} = 0$ byte and $g_{\mathcal{B}} = 12$ bytes;
- Matching DoF $d_{\mathcal{M}}$: 16 bytes between $\#\mathrm{MC}^3$ and $\#\mathrm{AK}^3$.

Then, the remaining DoF is $d_{\mathcal{B}} = 4, d_{\mathcal{R}} - g_{\mathcal{B}} = 4, d_{\mathcal{M}} - g_{\mathcal{B}} = 4$.

Compute initial values forward neutral bytes (Blue). Considering that the constraints on forward neutral words are linear, to obtain the corresponding initial values, one can choose and enumerate all possible values of 4 🟦 blue cells in $\#\mathrm{KMC}^5$ for forward neutral words.

Compute initial values for backward neutral words (Red). To get the initial values of backward neutral words, one should focus on the constraints among states $\#\mathrm{AK}^5$ and $\#\mathrm{MC}^5$, $\#\mathrm{MC}^0$ and $\#\mathrm{AK}^0$.

- For $\#\mathrm{MC}^0$ and $\#\mathrm{AK}^0$, there are also 16 bytes MC DoF cost for backward neutral words, which applies the constraints in Equation (20) as below

$$\begin{pmatrix} g_0 & - & - & - & - & - & - & g_{14} \\ g_1 & g_2 & - & - & - & - & - & - \\ - & g_3 & g_4 & - & - & - & - & - \\ - & - & g_5 & g_6 & - & - & - & - \\ - & - & - & g_7 & g_8 & - & - & - \\ - & - & - & - & g_9 & g_{10} & - & - \\ - & - & - & - & - & g_{11} & g_{12} & - \\ - & - & - & - & - & - & g_{13} & g_{15} \end{pmatrix} = \mathrm{MC} \cdot \begin{pmatrix} \#\mathrm{MC}^0_0 & \#\mathrm{MC}^0_8 & \cdots & \#\mathrm{MC}^0_{56} \\ \#\mathrm{MC}^0_1 & \#\mathrm{MC}^0_9 & \cdots & \#\mathrm{MC}^0_{57} \\ \#\mathrm{MC}^0_2 & \#\mathrm{MC}^0_{10} & \cdots & \#\mathrm{MC}^0_{58} \\ \#\mathrm{MC}^0_3 & \#\mathrm{MC}^0_{11} & \cdots & \#\mathrm{MC}^0_{59} \\ \#\mathrm{MC}^0_4 & \#\mathrm{MC}^0_{12} & \cdots & \#\mathrm{MC}^0_{60} \\ \#\mathrm{MC}^0_5 & \#\mathrm{MC}^0_{13} & \cdots & \#\mathrm{MC}^0_{61} \\ \#\mathrm{MC}^0_6 & \#\mathrm{MC}^0_{14} & \cdots & \#\mathrm{MC}^0_{62} \\ \#\mathrm{MC}^0_7 & \#\mathrm{MC}^0_{15} & \cdots & \#\mathrm{MC}^0_{63} \end{pmatrix}, \tag{20}$$

  where g_i $(0 \le i \le 15)$ are the chosen constants for $\#\mathrm{MC}^5$.

- For $\#\mathrm{AK}^5$ and $\#\mathrm{MC}^5$, there are 16 bytes MC DoF cost for backward neutral words, which applies the constraints in Equation (21) as below

$$\begin{pmatrix} - & - & e_2 & - & e_4 & - & - & - \\ - & e_0 & - & - & - & - & - & e_{13} \\ - & - & e_3 & - & - & - & e_{10} & - \\ - & e_1 & - & - & - & e_7 & - & e_{14} \\ - & - & - & - & e_5 & - & e_{11} & - \\ - & - & - & - & - & e_8 & - & e_{15} \\ - & - & - & - & e_6 & - & e_{12} & - \\ - & - & - & - & - & e_9 & - & - \end{pmatrix} = \mathrm{MC}^{-1} \cdot \begin{pmatrix} \#\mathrm{AK}^5_0 & \#\mathrm{AK}^5_8 & \cdots & \#\mathrm{AK}^5_{56} \\ \#\mathrm{AK}^5_1 & \#\mathrm{AK}^5_9 & \cdots & \#\mathrm{AK}^5_{57} \\ \#\mathrm{AK}^5_2 & \#\mathrm{AK}^5_{10} & \cdots & \#\mathrm{AK}^5_{58} \\ \#\mathrm{AK}^5_3 & \#\mathrm{AK}^5_{11} & \cdots & \#\mathrm{AK}^5_{59} \\ \#\mathrm{AK}^5_4 & \#\mathrm{AK}^5_{12} & \cdots & \#\mathrm{AK}^5_{60} \\ \#\mathrm{AK}^5_5 & \#\mathrm{AK}^5_{13} & \cdots & \#\mathrm{AK}^5_{61} \\ \#\mathrm{AK}^5_6 & \#\mathrm{AK}^5_{14} & \cdots & \#\mathrm{AK}^5_{62} \\ \#\mathrm{AK}^5_7 & \#\mathrm{AK}^5_{15} & \cdots & \#\mathrm{AK}^5_{63} \end{pmatrix}, \tag{21}$$

  where e_i $(0 \le i \le 15)$ are the chosen constants for $\#\mathrm{MC}^5$.

The MITM attack procedure for 7.75-round `Whirlpool`. We provide the main attack procedure derived from Figure 16 in Algorithm 14 as below.

The attack complexity. The time complexity of the above MITM pseudo-preimage attack of 7.75-round `Whirlpool` is about $2^{512-8\times\min(4,4,4)} = 2^{480}$. The table $T^{\mathrm{neutral}}_{\mathrm{red}}$ dominates the memory complexity with $2^{32\times 8} = 2^{256}$.

Algorithm 14: MITM attack on 7.75-round `Whirlpool` compression function

```
1  Fix 48 ■ constant cells in $\#\mathrm{KMC}^5$ (columns 1, 2, 4, 5, 6, 7) to all zero;
2  Let constant $C_g = (g_0, \cdots, g_{15})$ in Equation (20);
3  Let constant $C_e = (e_0, \cdots, e_{15})$ in Equation (21);
4  Let the remaining 12 ■ constant cells in $\#\mathrm{KMC}^5$ (columns 0, 3) be $C_G$;
   // Enumerate $C_e$ because starting from $\#\mathrm{MC}^5$ for red neutral words
5  for $C_e \in (\mathbb{F}_2^8)^{16}$ do
       // For red neutral words
6      for 32 🟥 red cells in $\#\mathrm{MC}^5 \in (\mathbb{F}_2^8)^{32}$ do
7          Initiate table $T^{\mathrm{neutral}}_{\mathrm{red}}$;
8          Compute forwards to 16 constant cells in $\#\mathrm{AK}^0$, i.e., $C_g$;
           // $2^{8\times(32-16)} = 2^{8\times16}$ candidates expected for each row
9          Update to table $T^{\mathrm{neutral}}_{\mathrm{red}}[C_g]$ += $\#\mathrm{MC}^5$;
       // For blue neutral words
10     for $C_G \in (\mathbb{F}_2^8)^{12}$ do
11         for $C_g \in (\mathbb{F}_2^8)^{16}$ do
12             Search table with $T^{\mathrm{neutral}}_{\mathrm{red}}[C_g]$ and obtain $\#\mathrm{MC}^5$ ($2^{8\times16}$ on average);
13             Compute backwards to red cells in $\#\mathrm{AK}^3$ and store in a table $T^-$;
14             for 4 🟦 blue cells in $\#\mathrm{KMC}^5 \in (\mathbb{F}_2^8)^4$ do
15                 for 12 🟦 guessed blue cells in $\#\mathrm{AK}^2 \in (\mathbb{F}_2^8)^{12}$ do
16                     Compute to the blue cells in $\#\mathrm{MC}^3$ and $\#\mathrm{AK}^3$ and check in
                         $T^-$;
17                     if $\#\mathrm{MC}^3$ and $\#\mathrm{AK}^3$ pass the partial match then
                           // $2^{8\times(16+12+16+16+16-16)} = 2^{480}$ candidates expected
18                         Compute to check 12 🟦 guessed cells in $\#\mathrm{AK}^2$;
19                         if Guess values are correct then
                               // $2^{8\times(60-12)} = 2^{8\times48}$ candidates expected
20                             Compute to match the rest 48 cells;
21                             if The full match is found then
                                   // $2^{8\times(48-48)} = 1$ candidate expected
22                                 Output the preimage and stop;
```

Appendix E Improved Preimage Attacks of Streebog

`Streebog` is a block-cipher based hash function in the Russian national standard GOST R 34.11-2012 [1] and has been an ISO/IEC standard [25] since 2018. It has two versions `Streebog-512` and `Streebog-256`, which produce 512-bit and 256-bit hash values respectively. *Similar* to `Whirlpool`, `Streebog` adopts MP mode compression function g, defined as $g(N_i, H_i, M_i) = E_{\mathrm{MR}\circ\mathrm{TP}\circ\mathrm{SB}(N_i \oplus H_i)}(M_i) \oplus M_i \oplus H_i$, where E is a 12-round AES-like block cipher with 8×8-byte internal states, N_i is the counter and MR, TP, SB are operations in one round of E. Note that the compression function g of `Streebog` is adopted in the HAIFA framework [10], as shown in Figure 17. The initial state of E is $S_0 = M_i$, for each round, the state is updated by AddRoundKey (AK), SubBytes(SB), Transposition (TP) and MixRows (MR), *i.e.*, $S_{j+1} = \mathrm{MR}\circ\mathrm{TP}\circ\mathrm{SB}(S_j \oplus K_j)$, $j = 0, \cdots, 11$. The output of the compression function g is finally computed by $S_{12} \oplus K_{12}$. Key state is initialized by $K_0 = \mathrm{MR} \circ \mathrm{TP} \circ \mathrm{SB}(H_i \oplus N_i)$, then the round key is updated by $K_{j+1} = \mathrm{MR} \circ \mathrm{TP} \circ \mathrm{SB}(K_j \oplus C_j)$, where $j = 0, \cdots, 11$ and C_j is the round constant. Still for convenience, we transpose the row and column for the representation of `Streebog`. For more details, please refer to the original documents [1,15].

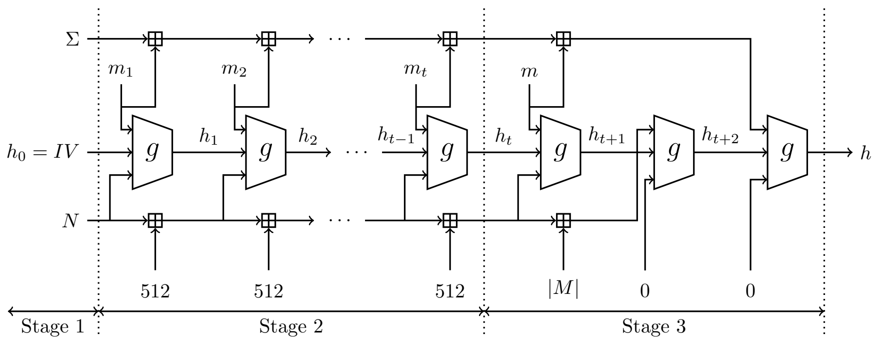


Fig. 17. `Streebog` hash function [26].

E.1 Improved Preimage Attack of 7.5-round `Streebog-512`

Using the **SIM** technique, we first search MITM attacks on 7.5-round compression function of `Streebog-512`, and our improved search result is given in Figure 18. Compared to the previous best 7.5-round pseudo-preimage attacks on `Streebog-512` found by Hua *et al.* [22] at ToSC 2022, we can improve the time complexity from 2^{441} to 2^{433} and memory complexity from 2^{192} to 2^{177}, our attack configuration is as below

- Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 10 bytes (10 blue cells in $\#\mathrm{SB}^4$ are *set to be equal to* the corresponding cells in $\#\mathrm{KK}^4$, and thus can be cancelled to zero constant ▣ in $\#\mathrm{AK}^3$ and some fixed constants[11] ▣ related to the round constants of `Streebog` in $\#\mathrm{AK}^4$);
- Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 54 bytes (in $\#\mathrm{SB}^4$);
- Consumed DoF for forward $\overrightarrow{\sigma}$: zero byte;
- Consumed DoF for backward $\overleftarrow{\sigma}$: 32 bytes (32 for $\#\mathrm{MC}^5 \xrightarrow{\mathrm{MC}} \#\mathrm{AK}^5$);
- Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{BR}} = 0$ byte and $g_{\mathcal{B}} = 12$ bytes;
- Matching DoF $d_{\mathcal{M}}$: 24 bytes between $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$.

Then, the remaining DoF is $d_{\mathcal{B}} = 10, d_{\mathcal{R}} - g_{\mathcal{B}} = 10, d_{\mathcal{M}} - g_{\mathcal{B}} = 12$.

Compute initial values forward neutral bytes (Blue). As there is no DoF cost for forward neutral words, to obtain the corresponding initial values, one only needs to enumerate all possible values of 10 blue cells in $\#\mathrm{SB}^4$.

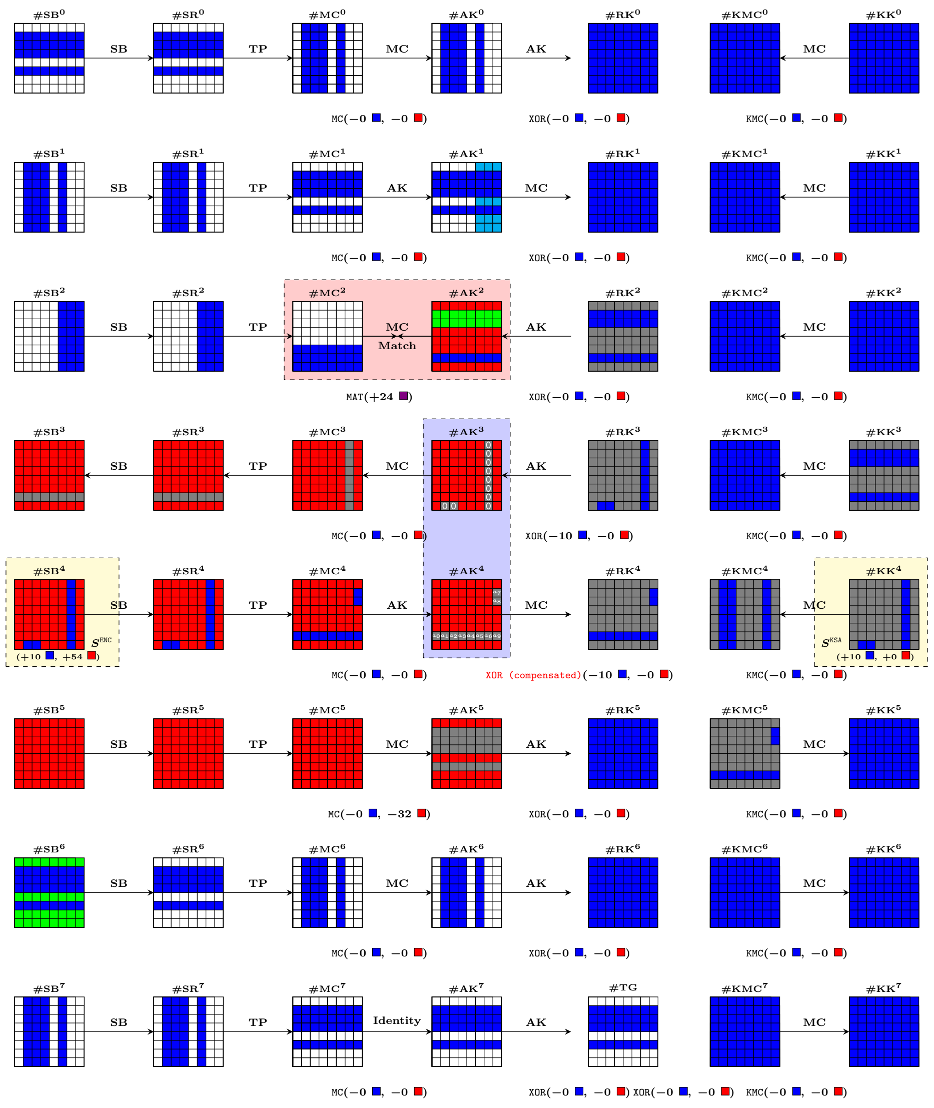
Compute initial values for backward neutral words (Red). To get the initial values of backward neutral words, we provide Algorithm 15 to obtain the solution space of red neutral words with the time complexity $2^{8\times 26} = 2^{208}$ and memory complexity $2^{8\times 22} = 2^{176}$.

Algorithm 15: Compute backward neutral words (red) for 7.5-round `Streebog`

```
1  Initiate a table $T_{\mathrm{red}}^{\mathrm{neutral}}$;
2  Fix 32 ■ constant cells in $\#\mathrm{AK}^5$ to all zero;
3  for 26 🟥 cells in $\#\mathrm{AK}^4 \in (\mathbb{F}_2^8)^{26}$ in the four rightmost columns (columns 4-7) do
4      Compute forward to the four bottom rows (rows 4-7) of $\#\mathrm{MC}^5$;
5      With 32 ■ fixed constants in $\#\mathrm{AK}^5$, derive the remaining red cells of $\#\mathrm{AK}^5$;
6      Derive the four topmost rows of $\#\mathrm{MC}^5$;
7      Compute backward to the four leftmost columns of $\#\mathrm{AK}^4$;
8      if 4 constants cells in $\#\mathrm{AK}^4$ in the leftmost columns are $(a_0, \ldots, a_3)$ then
           // $2^{8\times(26-4)} = 2^{8\times 22}$ candidates expected
9          Update table with $T_{\mathrm{red}}^{\mathrm{neutral}}$ += $\#\mathrm{AK}^4$;
10 return $T_{\mathrm{red}}^{\mathrm{neutral}}$;
```

The MITM attack procedure for 7.5-round ***`Streebog-512`***. We provide the main attack procedure derived from Figure 18 in Algorithm 16 as below.

[11] Because the round constant of the key schedule of `Streebog` is introduced at the input of the SB operation, which is different from `Whirlpool`. Thus, when considering forward computation by using the **SIM** technique, all zero constants will be changed to some fixed constants related to the round constants of `Streebog`, and this essentially has no impact on the attack.


#SB⁰ SB #SR⁰ TP #MC⁰ MC #AK⁰ AK #RK⁰ #KMC⁰ MC #KK⁰
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB¹ SB #SR¹ TP #MC¹ AK #AK¹ MC #RK¹ #KMC¹ MC #KK¹
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB² SB #SR² TP #MC² MC Match #AK² AK #RK² #KMC² MC #KK²
MAT(+24 🟪) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB³ SB #SR³ TP #MC³ MC #AK³ AK #RK³ #KMC³ MC #KK³
MC(−0 🟦, −0 🟥) XOR(−10 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁴ SB S^{ENC} (+10 🟦, +54 🟥) #SR⁴ TP #MC⁴ AK #AK⁴ MC #RK⁴ #KMC⁴ MC #KK⁴ S^{KSA} (+10 🟦, +0 🟥)
MC(−0 🟦, −0 🟥) XOR (compensated)(−10 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁵ SB #SR⁵ TP #MC⁵ MC #AK⁵ AK #RK⁵ #KMC⁵ MC #KK⁵
MC(−0 🟦, −32 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁶ SB #SR⁶ TP #MC⁶ MC #AK⁶ AK #RK⁶ #KMC⁶ MC #KK⁶
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁷ SB #SR⁷ TP #MC⁷ Identity #AK⁷ AK #TG #KMC⁷ MC #KK⁷
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)

- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (10+0)=10$ 🟦, $(54+0)=54$ 🟥)
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}) = (+10$ 🟦, $+22$ 🟥, $+24$ 🟪, $+12$ 🟦, $+0$ 🟥, $+0$ 🟥)
- $(d_{\mathcal{B}}-g_{\mathcal{R}}, d_{\mathcal{R}}-g_{\mathcal{B}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}}) = (10, 10, 12)$

Fig. 18. An MITM attack on 7.5-round compression function of `Streebog-512`.

Algorithm 16: MITM attack on 7.5-round `Streebog` compression function

```
Fix 32 ■ constant cells in $\#\mathrm{AK}^5$ to all zero;
Let 22 ■ constant cells in columns 0, 1 and 2 of $\#\mathrm{KK}^4$ be all zero;
Let 32 ■ constant cells in columns 3, 4, 5 and 7 of $\#\mathrm{KK}^4$ be $C_G$;
Call Algorithm 15 to build $T_{\mathrm{red}}^{\mathrm{neutral}}$ (with size $2^{8\times 22}$);
for $C_G \in (\mathbb{F}_2^8)^{32}$ do
    // For blue neutral words
    for 10 🟦 blue cells in $\#\mathrm{SB}^4 \in (\mathbb{F}_2^8)^{10}$ do
        for 12 🟦 guessed blue cells in $\#\mathrm{AK}^1 \in (\mathbb{F}_2^8)^{12}$ do
            Compute forward to the matching state $\#\mathrm{MC}^2$;
            Compute backward to the matching state $\#\mathrm{AK}^2$;
            Store in a table $T^+$ (with size $2^{8\times 22}$);
    // For red neutral words
    for $\#\mathrm{AK}^4 \in T_{red}^{neutral}$ do
        Compute backward to $\#\mathrm{AK}^2$;
        Check in the table $T^+$;
        if $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$ pass the partial match then
            // $2^{8\times(32+22+22-24)} = 2^{8\times 52}$ candidates expected
            Compute to check 12 🟦 guessed cells in $\#\mathrm{AK}^1$;
            if Guess values are correct then
                // $2^{8\times(52-12)} = 2^{8\times 40}$ candidates expected
                Compute forward and backward to match the rest 40 cells;
                if The full match is found then
                    // $2^{8\times(40-40)} = 1$ candidate expected
                    Output the preimage and stop;
```

The attack complexity. According to Algorithm 16, the time complexity of the above MITM pseudo-preimage attack on the compression function of 7.5-round `Streebog` is about $2^{512-8\times 10} + 2^{512-8\times 10} + 2^{512-8\times 12} \approx 2^{433}$. The tables T^+ and $T_{\mathrm{red}}^{\mathrm{neutral}}$ dominate the memory complexity with $2 \cdot 2^{8\times 22} = 2^{177}$.

We use the conversion given by AlTawy *et al.* [3, Section 6], who employed Joux' multi-collision technique [27] and another MITM attack [33] on the HAIFA framework [10] in their work. By using the conversion, the attack on the compression function of 7.5-round `Streebog` can be converted into a preimage attack on its hash function with the steps below:

1. Given an output $H(M)$ of `Streebog-512`, we produce 2^k preimages (Σ, h_{515}) for the last compression function and store them in a table T.
2. Using Joux's multi-collisions [27], 2^{512} messages can be constructed with a length of 512 message blocks, which all lead to the same value of h_{512}. That is $M_i = m_1^j \| m_2^j \| \cdots \| m_{512}^j$ $(i \in \{1, 2, \cdots, 2^{512}\}, j \in \{1, 2\})$, then we have 2^{512} candidates for Σ_{M_i}.
3. Assume the message has 513 complete blocks, then m_{pad} and $|M|$ are known constants. By randomly choosing 2^{512-k} m_{513}, together with h_{512} produced

in step 2 and the known values N_{513}, N_{514}, we can compute h_{515}. Then a right m_{513} is expected such that $h_{515} \in T$. Once we find a matching, Σ is known, so we can compute the sum $\Sigma_{M_i} = \Sigma - m_{\text{pad}} - m_{513}$.

4. We compute all the 2^{256} sums of all the 2^{256} 256-block message $\Sigma_{M_1} = m_1^j + m_2^j + \cdots + m_{256}^j$ and store them in a table T_1. Then, compute the sum of other 256-block messages $\Sigma_{M_2} = m_{256}^j + \cdots + m_{512}^j$ and check if $\Sigma_{M_i} - \Sigma_{M_2}$ is in table T_1. Once we find a matching, we know that the full 513-block message $M = m_1^j \| m_2^j \| \cdots \| m_{512}^j \| m^{513}$ is the preimage of $H(M)$.

For 7.5-round `Streebog-512` hash function, $k = 40.25$ is an almost optimal solution, so the time complexity is about $2^{40.25} \cdot 2^{433} + 512 \times 2^{256} + 3 \times 2^{512-40.25} + 2^{256} \approx 2^{474.25}$ and the memory complexity is bounded by 2^{256}.

E.2 Improved Preimage Attack of 8.5-round `Streebog-512`

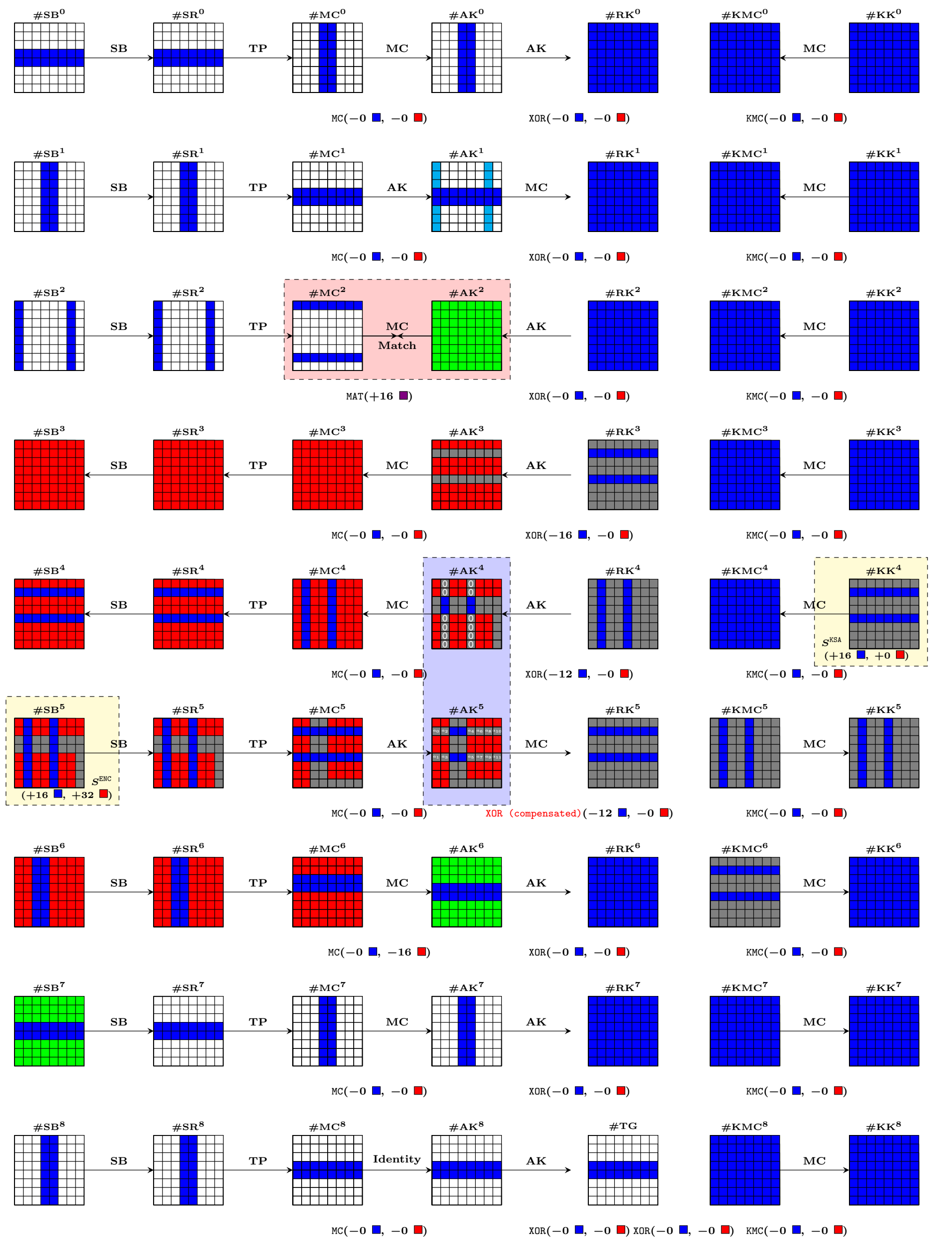
The previous best preimage attack on `Streebog-512` is also given by Hua *et al.* [22], which reaches 8.5-round and has the time complexity $2^{481}/2^{498.25}$ (compression function/hash function) and memory complexity 2^{288}. By using the **SIM** technique, obvious memory improvements can be achieved from 2^{288} to 2^{129} for the attack on compression function and to 2^{256} for the attack on hash function. Our better search results is provided in Figure 19, to gain the huge memory improvements, it firstly should be transformed into an equivalent configuration, which is given in Figure 20 and has the following attack configuration.

– Initial DoF for forward neutral words $\overrightarrow{\iota}$ (🟦): 4 bytes (4 blue cells in #AK^4 are *set to be equal to* the corresponding cells in #KK^5, and thus can be some fixed constants ▨ in #AK^3 and some fixed constants ▨ in #AK^5, these fixed constants are related to the round constants of `Streebog`.);
– Initial DoF for backward neutral words $\overleftarrow{\iota}$ (in 🟥): 32 bytes (in #AK^5);
– Consumed DoF for forward $\overrightarrow{\sigma}$: zero byte (16 XOR DoF cost for #RK^3 cancels blue cells in #SB^4 can be compensated by the **SIM** technique);
– Consumed DoF for backward $\overleftarrow{\sigma}$: 16 bytes (16 for #MC$^6 \xrightarrow{\text{MC}}$ #AK^6);
– Guessed bytes for blue, red and both colors $g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}$: $g_{\mathcal{R}} = g_{\mathcal{BR}} = 0$ byte and $g_{\mathcal{B}} = 12$ bytes;
– Matching DoF $d_{\mathcal{M}}$: 16 bytes between #MC^2 and #AK^2.

Then, the remaining DoF is $d_{\mathcal{B}} = 4, d_{\mathcal{R}} - g_{\mathcal{B}} = 4, d_{\mathcal{M}} - g_{\mathcal{B}} = 4$.

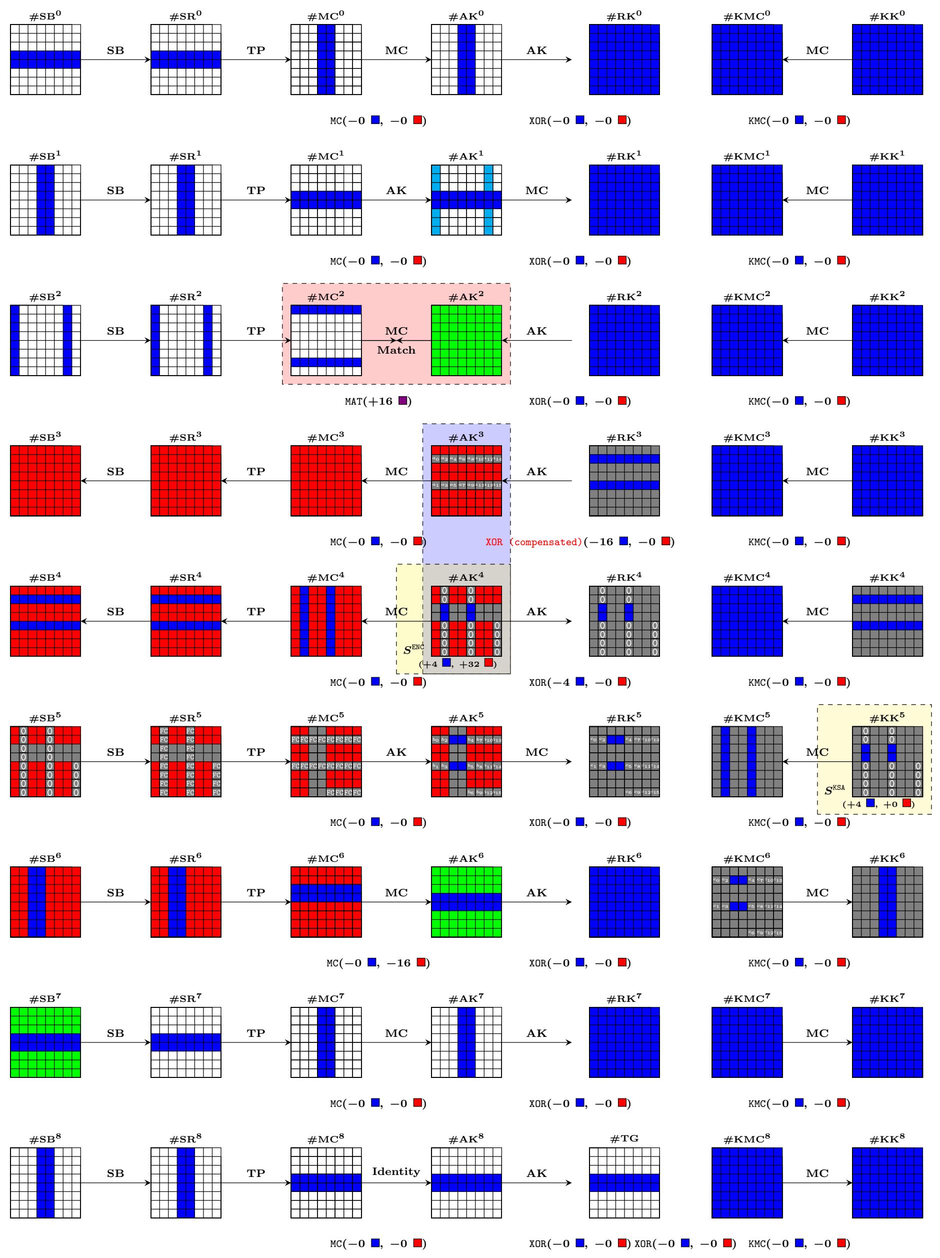
Compute initial values for forward neutral words (Blue). To obtain the initial values of forward neutral words, one should focus on the constraints among states #RK^3 (#KK^4), #AK^3 and #SB^4 as below

$$\begin{pmatrix} \#\text{SB}_0^4 & \#\text{SB}_8^4 & \cdots & \#\text{SB}_{56}^4 \\ \#\text{SB}_1^4 & \#\text{SB}_9^4 & \cdots & \#\text{SB}_{57}^4 \\ \#\text{SB}_2^4 & \#\text{SB}_{10}^4 & \cdots & \#\text{SB}_{58}^4 \\ \#\text{SB}_3^4 & \#\text{SB}_{11}^4 & \cdots & \#\text{SB}_{59}^4 \\ \#\text{SB}_4^4 & \#\text{SB}_{12}^4 & \cdots & \#\text{SB}_{60}^4 \\ \#\text{SB}_5^4 & \#\text{SB}_{13}^4 & \cdots & \#\text{SB}_{61}^4 \\ \#\text{SB}_6^4 & \#\text{SB}_{14}^4 & \cdots & \#\text{SB}_{62}^4 \\ \#\text{SB}_7^4 & \#\text{SB}_{15}^4 & \cdots & \#\text{SB}_{63}^4 \end{pmatrix} \oplus \begin{pmatrix} \#\text{RK}_0^3 & \#\text{RK}_8^3 & \cdots & \#\text{RK}_{56}^3 \\ \#\text{RK}_1^3 & \#\text{RK}_9^3 & \cdots & \#\text{RK}_{57}^3 \\ \#\text{RK}_2^3 & \#\text{RK}_{10}^3 & \cdots & \#\text{RK}_{58}^3 \\ \#\text{RK}_3^3 & \#\text{RK}_{11}^3 & \cdots & \#\text{RK}_{59}^3 \\ \#\text{RK}_4^3 & \#\text{RK}_{12}^3 & \cdots & \#\text{RK}_{60}^3 \\ \#\text{RK}_5^3 & \#\text{RK}_{13}^3 & \cdots & \#\text{RK}_{61}^3 \\ \#\text{RK}_6^3 & \#\text{RK}_{14}^3 & \cdots & \#\text{RK}_{62}^3 \\ \#\text{RK}_7^3 & \#\text{RK}_{15}^3 & \cdots & \#\text{RK}_{63}^3 \end{pmatrix} = \begin{pmatrix} - & - & - & - & - & - & - & - \\ a_0 & a_2 & a_4 & a_6 & a_8 & a_{10} & a_{12} & c_{14} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ a_1 & a_3 & a_5 & a_7 & a_9 & a_{11} & a_{13} & a_{15} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{pmatrix}, \tag{22}$$


#SB⁰ SB #SR⁰ TP #MC⁰ MC #AK⁰ AK #RK⁰ #KMC⁰ MC #KK⁰
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB¹ SB #SR¹ TP #MC¹ AK #AK¹ MC #RK¹ #KMC¹ MC #KK¹
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB² SB #SR² TP #MC² MC Match #AK² AK #RK² #KMC² MC #KK²
MAT(+16 🟪) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB³ SB #SR³ TP #MC³ MC #AK³ AK #RK³ #KMC³ MC #KK³
MC(−0 🟦, −0 🟥) XOR(−16 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁴ SB #SR⁴ TP #MC⁴ MC #AK⁴ AK #RK⁴ #KMC⁴ MC #KK⁴
S^KSA (+16 🟦, +0 🟥)
MC(−0 🟦, −0 🟥) XOR(−12 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁵ SB #SR⁵ TP #MC⁵ AK #AK⁵ MC #RK⁵ #KMC⁵ MC #KK⁵
S^ENC (+16 🟦, +32 🟥)
MC(−0 🟦, −0 🟥) XOR (compensated)(−12 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁶ SB #SR⁶ TP #MC⁶ MC #AK⁶ AK #RK⁶ #KMC⁶ MC #KK⁶
MC(−0 🟦, −16 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁷ SB #SR⁷ TP #MC⁷ MC #AK⁷ AK #RK⁷ #KMC⁷ MC #KK⁷
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)
#SB⁸ SB #SR⁸ TP #MC⁸ Identity #AK⁸ AK #TG #KMC⁸ MC #KK⁸
MC(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) XOR(−0 🟦, −0 🟥) KMC(−0 🟦, −0 🟥)

- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (16{+}4){=}20$ 🟦, $(32{+}0){=}32$ 🟥)
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}) = (+4$ 🟦, $+16$ 🟥, $+16$ 🟪, $+12$ 🟦, $+0$ 🟥, $+0$ 🟥)
- $(d_{\mathcal{B}}-g_{\mathcal{R}}, d_{\mathcal{R}}-g_{\mathcal{B}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}}) = (4, 4, 4)$

Fig. 19. An MITM attack on 8.5-round compression function of `Streebog-512`.



- $(\overrightarrow{\iota}, \overleftarrow{\iota}) = (4+0)=4$ 🟦, $(32+0)=32$ 🟥
- $(d_{\mathcal{B}}, d_{\mathcal{R}}, d_{\mathcal{M}}, g_{\mathcal{B}}, g_{\mathcal{R}}, g_{\mathcal{BR}}) = (+4$ 🟦, $+16$ 🟥, $+16$ 🟪, $+12$ 🟦, $+0$ 🟥, $+0$ 🟥$)$
- $(d_{\mathcal{B}}-g_{\mathcal{R}}, d_{\mathcal{R}}-g_{\mathcal{B}}, d_{\mathcal{M}}-g_{\mathcal{B}}-g_{\mathcal{R}}-g_{\mathcal{BR}}) = (4, 4, 4)$

Fig. 20. Equivalent MITM attack on 8.5-round compression function of `Streebog-512`.

where a_i $(0 \leq i \leq 15)$ are the chosen constants for $\#\mathrm{AK}^3$.

Thanks to the **SIM** technique, the conditions on Equation (22) can be naturally held if we properly set the states $\#\mathrm{AK}^4$ and $\#\mathrm{KK}^5$ as below

- Set 16 $\boxed{0}$ zero constant cells as shown in $\#\mathrm{AK}^4$ and $\#\mathrm{KK}^5$;
- Set 4 🟦 blue cells in $\#\mathrm{AK}^4$ to be equal to the corresponding cells in $\#\mathrm{KK}^5$;

Then for backward computation, as $\#\mathrm{AK}^4$ and $\#\mathrm{KK}^5$ follow almost the same operations in encryption and key schedule except the AddRoundConstant after passing the SB^{-1} operation in key schedule, thus it will cancel to 16 fixed constants $\boxed{a_i}$ in $\#\mathrm{AK}^3$, which is related to round constants of `Streebog`. Similarly, for forward computation[12], it will cancel to 16 fixed constant cells $\boxed{b_i}$ in $\#\mathrm{AK}^5$. We note that this usage of the **SIM** technique is different to previous results on `Whirlpool` and 7.5-round `Streebog`, which is asymmetric in terms of the DoF compensation and covers one more round to three rounds, *i.e.*, $\#\mathrm{AK}^3, \#\mathrm{AK}^4$ and $\#\mathrm{AK}^5$. For $\#\mathrm{AK}^3$ and $\#\mathrm{AK}^4$, it can be used to efficiently obtain forward (blue) neutral words, and the 16 fixed constant cells $\boxed{b_i}$ in $\#\mathrm{AK}^5$ can be used to efficiently obtain backward (red) neutral words.

Compute initial values for backward neutral words (Red). To obtain the initial values of red neutral words, the constraints among $\#\mathrm{AK}^6$ and $\#\mathrm{MC}^6$ are set as below

$$\begin{pmatrix} - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ e_0 & e_2 & e_4 & e_6 & e_8 & e_{10} & e_{12} & e_{14} \\ e_1 & e_3 & e_5 & e_7 & e_9 & e_{11} & e_{13} & e_{15} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{pmatrix} = \mathrm{MC} \cdot \begin{pmatrix} \#\mathrm{MC}^6_0 & \#\mathrm{MC}^6_8 & \cdots & \#\mathrm{MC}^6_{56} \\ \#\mathrm{MC}^6_1 & \#\mathrm{MC}^6_9 & \cdots & \#\mathrm{MC}^6_{57} \\ - & - & \cdots & - \\ - & - & \cdots & - \\ \#\mathrm{MC}^6_4 & \#\mathrm{MC}^6_{12} & \cdots & \#\mathrm{MC}^6_{60} \\ \#\mathrm{MC}^6_5 & \#\mathrm{MC}^6_{13} & \cdots & \#\mathrm{MC}^6_{61} \\ \#\mathrm{MC}^6_6 & \#\mathrm{MC}^6_{14} & \cdots & \#\mathrm{MC}^6_{62} \\ \#\mathrm{MC}^6_7 & \#\mathrm{MC}^6_{15} & \cdots & \#\mathrm{MC}^6_{63} \end{pmatrix}, \tag{23}$$

where e_i $(0 \leq i \leq 15)$ are the chosen constants for $\#\mathrm{AK}^6$. Then the following Algorithm 17 is provided to obtain the solution space of red neutral words with the time complexity $2^{8\times 32} = 2^{256}$ and memory complexity $2^{8\times 16} = 2^{128}$.

Algorithm 17: Compute backward neutral words (red) for 8.5-round `Streebog`

```
1 Initiate a table $T^{\mathrm{neutral}}_{\mathrm{red}}$;
2 Fix 16 ■ constants $(e_0, \cdots, e_{15})$ in Equation (23) to all zero;
3 Solve Equation (23) to get the solution space $S_{\mathrm{red}}$ (with dimension 32 bytes);
4 for each solution in $S_{red}$ do
5     Compute backward to $\#\mathrm{AK}^5$;
6     if 16 $\boxed{b_i}$ fixed constant cells in $\#\mathrm{AK}^5$ are matched then
          // $2^{8\times(32-16)} = 2^{8\times 16}$ candidates expected
7         Update to table with $T^{\mathrm{neutral}}_{\mathrm{red}}$ += $\#\mathrm{AK}^5$;
8 return $T^{\mathrm{neutral}}_{\mathrm{red}}$;
```

[12] According to `Streebog`'s S-box π, $\pi(\mathtt{0}) = \mathtt{FC}$, thus it has $\boxed{0} \xrightarrow{\pi} \boxed{FC}$ for $\#\mathrm{SB}^5 \xrightarrow{\mathrm{SB}} \#\mathrm{SR}^5$.

The MITM attack procedure for 8.5-round `Streebog-512`. We provide the main attack procedure derived from Figure 19 in Algorithm 18 as below.

Algorithm 18: MITM attack on 8.5-round `Streebog` compression function

```
1  Fix 16 $\boxed{a_i}$ constant cells in $\#\mathrm{AK}^3$;
2  Fix 16 $\boxed{b_i}$ constant cells in $\#\mathrm{AK}^5$;
3  Fix 16 $\boxed{0}$ constant cells in $\#\mathrm{AK}^4$;
4  Fix the rest 12 ■ constant cells in $\#\mathrm{AK}^4$to all zero;
5  Fix 16 $\boxed{0}$ constant cells in $\#\mathrm{KK}^5$;
6  Fix 16 ■ constants $(e_0, \cdots, e_{15})$ in Equation (23) to all zero;
7  Fix any 4 ■ constant cells in $\#\mathrm{KK}^5$ ($\#\mathrm{RK}^4$) be all zero;
8  Let the rest 44 ■ constant cells in $\#\mathrm{KK}^5$ be $C_G$;
9  Call Algorithm 17 to build $T^{\mathrm{neutral}}_{\mathrm{red}}$ (with size $2^{8\times 16}$);
10 for $C_G \in (\mathbb{F}_2^8)^{44}$ do
       // For red neutral words
11     Compute backward to $\#\mathrm{AK}^2$ by $T^{\mathrm{neutral}}_{\mathrm{red}}$ and store in a table $T^-$;
       // For blue neutral words
12     for 4 🟦 blue cells in $\#\mathrm{AK}^4 \in (\mathbb{F}_2^8)^4$ do
13         Set 4 🟦 corresponding blue cells in $\#\mathrm{AK}^5$ to be equal to that in $\#\mathrm{AK}^4$;
14         for 12 🟦 guessed blue cells in $\#\mathrm{MC}^1 \in (\mathbb{F}_2^8)^{12}$ do
15             Compute to $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$ and check in the table $T^-$;
16             if $\#\mathrm{MC}^2$ and $\#\mathrm{AK}^2$ pass the partial match then
                   // $2^{8\times(44+16+12+4-16)} = 2^{8\times 60}$ candidates expected
17                 Compute to check 12 🟦 guessed cells in $\#\mathrm{AK}^1$;
18                 if Guess values are correct then
                       // $2^{8\times(60-12)} = 2^{8\times 48}$ candidates expected
19                     Compute forward and backward to match the rest 48 cells;
20                     if The full match is found then
                           // $2^{8\times(48-48)} = 1$ candidate expected
21                         Output the preimage and stop;
```

The attack complexity. According to Algorithm 18, the time complexity of the above MITM pseudo-preimage attack on the compression function of 8.5-round `Streebog` is about $2^{512-8\times 4} + 2^{512-8\times 4} + 2^{512-8\times 4} \approx 2^{481}$. The table T^- and $T^{\mathrm{neutral}}_{\mathrm{red}}$ dominate the memory complexity with $2 \times 2^{8\times 16} = 2^{129}$.

For 8.5-round `Streebog-512` hash function, same to the choice in [22], $k = 16.25$, so the time complexity is about $2^{16.25} \cdot 2^{481} + 512 \times 2^{256} + 3 \times 2^{512-16.25} + 2^{256} \approx 2^{498.25}$ and the memory complexity is bounded by 2^{256}.