Automated Meet-in-the-Middle Attack Goes to Feistel

Qingliang Hou[1,2], Xiaoyang Dong[3,6,7](✉), Lingyue Qin[4,6](✉), Guoyan Zhang[1,5,7](✉), and Xiaoyun Wang[1,3,5,6,7](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
qinglianghou@mail.sdu.edu.cn, guoyanzhang@sdu.edu.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
{xiaoyangdong,xiaoyunwang}@tsinghua.edu.cn
[4] BNRist, Tsinghua University, Beijing, China
qinly@tsinghua.edu.cn
[5] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[6] Zhongguancun Laboratory, Beijing, China
[7] Shandong Institute of Blockchain, Jinan, China

Abstract. Feistel network and its generalizations (GFN) are another important building blocks for constructing hash functions, e.g., `Simpira v2`, `Areion`, and the ISO standard `Lesamnta-LW`. The Meet-in-the-Middle (MitM) is a general paradigm to build preimage and collision attacks on hash functions, which has been automated in several papers. However, those automatic tools mostly focus on the hash function with Substitution-Permutation network (SPN) as building blocks, and only one for Feistel network by Schrottenloher and Stevens (at CRYPTO 2022).

In this paper, we introduce a new automatic model for MitM attacks on Feistel networks by generalizing the traditional *direct or indirect partial matching strategies* and also Sasaki's multi-round matching strategy. Besides, we find the equivalent transformations of Feistel and GFN can significantly simplify the MILP model. Based on our automatic model, we improve the preimage attacks on `Feistel-SP-MMO`, `Simpira-2/-4-DM`, `Areion-256/-512-DM` by 1–2 rounds or significantly reduce the complexities. Furthermore, we fill in the gap left by Schrottenloher and Stevens at CRYPTO 2022 on the large branch ($b > 4$) `Simpira`-b's attack and propose the first 11-round attack on `Simpira-6`. Besides, we significantly improve the collision attack on the ISO standard hash `Lesamnta-LW` by increasing the attacked round number from previous 11 to ours 17 rounds.

Keywords: MitM · Automatic Tool · Feistel · `Simpira v2` · `Lesamnta-LW` · `Areion`

1 Introduction

The cryptographic hash function is one of the most important primitives, playing a vital role in digital signatures, message integrity, passwords, and proof-of-work, etc. The collision resistance, preimage resistance, and second-preimage resistance are the three basic security requirements for cryptographic hash functions. Besides the well-known `SHA-3` [12], another crucial design strategy is to build hash functions on block ciphers [38,43]. Typical examples are PGV-modes [43], Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), and Miyaguchi-Preneel (MP), etc., instantiated with `AES` [19] or other `AES`-like constructions, e.g., `Whirlpool` [8], `Grøstl` [28], `ECHO` [11], `Haraka v2` [37]. Feistel network and generalized Feistel network (GFN) are important designs for block ciphers and permutations. To share the security proof and implementation benefit, building Feistel (or GFN) primitives with `AES` round function becomes popular in research communities, e.g., `Simpira v2` [29], `Areion` [35], and the ISO lightweight hash function standard `Lesamnta-LW` [31], etc., which are the main targets of this paper.

The Meet-in-the-Middle (MitM) Attack is a time-memory trade-off cryptanalysis technique introduced by Diffie and Hellman to attack block cipher [22]. At SAC 2008, Aumasson, Meier, and Mendel [4] proposed the MitM preimage attacks on reduced `MD5` and full 3-pass HAVAL. At ASIACRYPT 2008, Sasaki and Aoki formally combined the MitM and local-collision techniques to attack full 3, 4, and 5-pass HAVAL. Further, they proposed the *splice-and-cut* technique [3] and the *initial structure* [49] to strengthen MitM attack and successfully broke the preimage resistance of the full `MD5`. In the past decades, the MitM attack has been widely applied to the cryptanalysis on block ciphers [14,25,34,41] and hash functions [3,30,49]. Simultaneously, various techniques have been introduced to improve the framework of MitM attack, such as internal state guessing [25], splice-and-cut [3], initial structure [49], bicliques [13], 3-subset MitM [14], indirect-partial matching [3,49], sieve-in-the-middle [17], match-box [27], dissection [23], MitM with guess-and-determine [50], differential-aided MitM [16,26,36], algebraic MitM [40], two-stage MitM [5], quantum MitM [51], etc. Till now, the MitM attack and its variants have broken MD4 [30,39], MD5 [49], KeeLoq [33], HAVAL [4,48], GOST [34], GEA-1/2 [1,10], etc.

Automatic Tools are significantly boosting the MitM attacks, recently. At CRYPTO 2011 and 2016, several automatic tools [15,21] were proposed for MitM attacks on `AES`. At FSE 2012, Wu *et al.* [53] introduced a search algorithm for MitM attacks on `Grøstl`. In [45], Sasaki first programmed the MitM attack on `GIFT` into a dedicated Mixed-Integer-Linear-Programming (MILP) model. At EUROCRYPT 2021, Bao *et al.* [6] introduced the MILP-based automatic search framework for MitM preimage attacks on `AES`-like hashing, whose compression function is built from `AES`-like block cipher or permutation. At CRYPTO 2021, Dong *et al.* [24] further extended Bao *et al.*'s model into key-recovery and collision attacks. At CRYPTO 2022, Schrottenloher and Stevens [51] simplified the language of the automatic model and applied it in both classic and quantum settings. Bao *et al.* [7] considered the MitM attack in view of the superposition states. At EUROCRYPT 2023, Qin *et al.* [44] proposed MitM attacks and automatic tools on sponge-based hashing.

Most state-of-the-art automatic tools of MitM attacks are about `AES`-like substitution-permutation network (SPN) primitives [6,7,24]. For Feistel or GFN constructions, most MitM cryptanalysis results are achieved by hand, such as the attacks on MD-SHA hash functions [2,3,30,49]. At ACNS 2013, Sasaki *et al.* [47] studied the preimage attacks on hash functions based on Feistel constructions with substitution-permutation (SP) round function, i.e., Feistel-SP. At CRYPTO 2022, Schrottenloher and Stevens [51] introduced an efficient MitM automatic tool including the first application to Feistel constructions, e.g., `Simpira v2` [29].

Our Contributions

In this paper, we focus on building a new MILP-based MitM automatic tool on hash functions with Feistel or GFN constructions.

For the first contribution, we first generalize the matching strategy for MitM attack. The essential idea of MitM attack is to find two neutral states (represented by 🟦 and 🟥 bytes), which are computed along two independent paths ('forward' and 'backward') that are then linked in the middle by deterministic relations, i.e. the matching point. The deterministic relations are usually of the form $f_{\mathcal{B}} = g_{\mathcal{R}}$, where $f_{\mathcal{B}}$ and $g_{\mathcal{R}}$ are determined by 🟦 and 🟥, respectively. In [3,49], the matching equation $f_{\mathcal{B}} = g_{\mathcal{R}}$ is usually part of the full state, which is then named as *partial matching*. If $f_{\mathcal{B}} = g_{\mathcal{R}}$ is derived directly, then it is a *direct partial matching* [3]. However, if $f_{\mathcal{B}} = g_{\mathcal{R}}$ is computed by a linear transformation on the outputs of forward and backward computation, then it is named as *indirect partial matching* [2,49]. For both direct and indirect partial matching, the relation $f_{\mathcal{B}} = g_{\mathcal{R}}$ is essential for MitM attacks. Almost all the recent MitM attacks and automatic models [6,7,24,44] leverage these two traditional matching strategies.

However, in this paper, we find the relations $f'_{\mathcal{B}} = g'_{\mathcal{B}}$ (or $f'_{\mathcal{R}} = g'_{\mathcal{R}}$) can also be used for matching, where $f'_{\mathcal{B}}$ and $g'_{\mathcal{B}}$ are determined only by 🟦 bytes. Together with the direct and indirect partial matching strategies, we propose a generalized matching strategy. After programming the new matching strategy into our MILP model, we significantly reduce the 5-round preimage attack on `Areion-256` from 2^{248} [35] to 2^{193}, and improve the preimage attack on `Simpira-2` from previous 5 rounds [51] to ours 7 rounds.

For the second contribution, We first generalize Sasaki's multi-round matching strategy for Feistel [47] into full-round matching. At ACNS 2013, Sasaki [47] proposed a matching strategy for Feistel-SP and GFN. For the Feistel-SP structure, it is hard to find any matching at first glance, but two-byte matching obviously appeared after applying a linear transformation to 4 consecutive rounds. In this paper, we find Sasaki's multi-round matching can be further extended into full-round matching. Therefore, the states involved in matching come from all round functions from the matching point to the initial structure. The full-round matching strategy may discover more useful matching equations than the multi-round matching. The reason is that in the multi-round matching, the involved states are first computed along forward and backward from the known bytes in the initial structure, and many bytes become unknown (i.e., depending on both 🟥 and 🟦 bytes, denoted as □ bytes), and then it is hard to derive any matching equations through the □ bytes. In full-round matching, matches are constructed by directly considering the fresh states from the initial structure.

Since many internal states are considered in full-round matching, it becomes hard to build MILP constraints for matching. To solve this problem, we find an equivalent transformation of Feistel and GFN that can significantly simplify the MILP programming of the full-round matching, where each byte of the full state can be programmed individually to determine if it is a one-byte matching.

Based on the above techniques, the achievements in this paper are listed below and also in Table 1.

– Based on the above techniques, we improve Sasaki's 11-round MitM attack [47] on Feistel-SP to ours 12 rounds with almost the same time complexity.
– We improve Schrottenloher and Stevens's MitM preimage attacks at CRYPTO 2022 [51] on `Simpira v2` by improving the attack on `Simpira-2` from 5 rounds [51] to ours 7 rounds, and improving the attack on `Simpira-4` from 9 rounds [51] to ours 11 rounds. As stated by Schrottenloher and Stevens [52, Appendix B7], they can not attack on `Simpira-`b versions with $b \notin \{2, 3, 4\}$. We first fill the gap by introducing the 11-round MitM attack on `Simpira-6`.
– For the ISO standardized lightweight hash `Lesamnta-LW` [31], we significantly improve the collision attack from the previous 11-round attack to ours 17-round attack. Moreover, we also found a 20-round `Lesamnta-LW` MitM characteristic with time 2^{124} which is better than the generic birthday bound 2^{128}, but it's higher than the designers' security claim against collision attack, which is 2^{120}.
– For the hash function `Areion` [35] proposed at TCHES 2023, we improve the MitM preimage attack on `Areion256-DM` from the previous 5 rounds to ours 7 rounds, and improve the attack on `Areion512-DM` from previous 10 rounds to ours 11 rounds. For the source code, please refer to

https://github.com/Hql-code/MitM-Feistel

Comparison to Schrottenloher and Stevens's MitM Attack. At CRYPTO 2022, Schrottenloher and Stevens [51] introduced automatic MitM tools based on MILP, which are also applied to preimage attacks on Feistel constructions, i.e., `Simpira v2` [29] and `Sparkle` [9]. Their model is a top-down model with a greatly simplified attack representation excluding many details. While our model in this paper follows the bottom-up approach, which has been used by Bao *et al.* [6,7] and Dong *et al.* [24]. Therefore, our model inherits the advantages of previous works [6,7,24], which is easy to understand and use by only specifying the admissible coloring transitions at each stage and computing the parameters which give the time and memory complexities of the MitM attack. On `Simpira v2`'s attacks [51], to simplify the model, the attacks are of branch-level. However, in our model, all attacks are found at the byte-level, which is more fine-grained. Combined with our new model on the matching strategy, we can improve Schrottenloher and Stevens' attacks on `Simpira-2/-4` by up to 2 rounds. Also, we find an attack on 11-round `Simpira-6`, while Schrottenloher and Stevens stated that their attack can not apply to it [52, Appendix B7].

Table 1. A Summary of the Attacks.

Target	Attacks	Settings	Rounds	Time	Memory	Generic	Ref
`Feistel-SP-128`	Preimage	Classical	11	2^{112}	2^{24}	2^{128}	[47]
		Classical	12	2^{113}	2^{48}	2^{128}	Sect. 5
`Simpira-2`	Preimage	Classical	5	2^{128}	-	2^{256}	[51]
		Quantum	5	2^{64}	-	2^{128}	[51]
		Classical	7	2^{225}	2^{96}	2^{256}	Sect. 6.1
`Simpira-4`	Preimage	Classical	9	2^{128}	-	2^{256}	[51]
		Quantum	9	2^{64}	-	2^{128}	[51]
		Classical	11	2^{225}	2^{160}	2^{256}	Sect. 6.2
`Simpira-6`	Preimage	Classical	11	$2^{193.6}$	2^{193}	2^{256}	Full Ver. [32]
`Lesamnta-LW`	Collision	Classical	11	2^{97}	2^{96}	2^{128}	[31]
		Classical	17	$2^{113.58}$	2^{112}	2^{128}	Sect. 7
		Classical	20	2^{124}	2^{124}	2^{128}	Full Ver. [32]
`Areion256-DM`	Preimage	Classical	5	2^{248}	2^{8}	2^{256}	[35]
		Classical	5	2^{193}	2^{88}	2^{256}	Sect. 8
		Classical	7	2^{240}	2^{64}	2^{256}	Sect. 8
`Areion512-DM`	Preimage	Classical	10	2^{248}	2^{8}	2^{256}	[35]
		Classical	11	2^{241}	2^{48}	2^{256}	Sect. 8

2 Preliminaries

In the section, we first introduce the main notations used in the following paper, and briefly describe the Meet-in-the-Middle attack, the specification of `AES`, (Generalized) Feistel Networks, `Areion`, `Lesamnta-LW`, and the idea of Sasaki's preimage attack on Feistel-SP.

2.1 Notations

$A_{\mathsf{SB}}^{(r)}$	: the internal state after operation `SB` in round r, $r \geq 0$
$A_{\mathsf{SB}}^{(r)}[i]$	: the i-th byte of the internal state $A_{\mathsf{SB}}^{(r)}$
🟥, $\mathcal{R}$	: known byte with backward computation, $(x, y) = (0, 1)$
🟦, $\mathcal{B}$	: known byte with forward computation, $(x, y) = (1, 0)$
⬛, $\mathcal{G}$	: known byte with forward and backward computations, $(x, y) = (1, 1)$
□, $\mathcal{W}$	: unknown byte in forward and backward computations, $(x, y) = (0, 0)$
$\lambda_{\mathcal{R}}$	: the byte number of the 🟥 bytes in the starting state
$\lambda_{\mathcal{B}}$	: the byte number of the 🟦 bytes in the starting state
DoF	: degree of freedom in bytes
$\text{DoF}_{\mathcal{R}}$	: the byte number of DoF of the 🟥 neutral words
$\text{DoF}_{\mathcal{B}}$	: the byte number of DoF of the 🟦 neutral words

$l_{\mathcal{B}}$	: the byte number of consumed DoF of the 🟦 bytes
$l_{\mathcal{R}}$	: the byte number of consumed DoF of the 🟥 bytes
DoM	: the byte number of DoF of the matching point
$End_{\mathcal{B}}$	: the matching point determined by 🟦 bytes
$End_{\mathcal{R}}$	: the matching point determined by 🟥 bytes

2.2 The Meet-in-the-Middle Attack

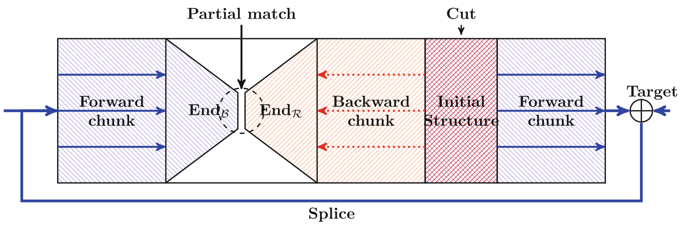


Fig. 1. The closed computation path of the MitM attack

Since the pioneering works on preimage attacks on Merkle-Damgård hashing, e.g. `MD4`, `MD5`, and `HAVAL` [3,30,39,49], techniques such as *splice-and-cut* [3], *initial structure* [49] and *(indirect-) partial matching* [2,49] have been invented to significantly improve the MitM approach. In Fig. 1, the compression function is divided at certain intermediate rounds (initial structure) into two chunks:

1. In the initial structure, a starting state is chosen with $\lambda_{\mathcal{R}}$ 🟥 bytes and $\lambda_{\mathcal{B}}$ 🟦 bytes, which are also denoted as the initial degree of freedom (DoF) of 🟥 and 🟦 bytes. The 🟥 and 🟦 bytes are then constrained linearly [46,47] or nonlinearly [24] by $l_{\mathcal{R}}$ and $l_{\mathcal{B}}$ byte equations, so that the two chunks can be computed independently on two distinct solution spaces of 🟥 and 🟦 derived by solving the constraint equations. The two solution spaces are named as *neutral space*. The DoFs of the 🟥 or 🟦 neutral space are denoted as $\text{DoF}_{\mathcal{R}}$ or $\text{DoF}_{\mathcal{B}}$.
2. The two *neutral space*s are computed along two independent paths ('forward chunk' and 'backward chunk').
3. One chunk is computed across the first and last rounds via the *feed-forward mechanism* of the hashing mode, and they end at a common intermediate round (partial matching point) to derive the deterministic relation '$End_{\mathcal{B}} = End_{\mathcal{R}}$' for matching. The number of bytes for matching is denoted as the degree of matching (DoM).

Thereafter, a closed computation path of the MitM attack is derived. After setting up the configurations, the basic attack procedure goes as follows:

1. Choose constants for the initial structure.
2. For all $2^{8\cdot\text{DoF}_{\mathcal{R}}}$ values of 🟥 neutral space, compute backward from the initial structure to the matching points $End_{\mathcal{R}}$ to generate a table $L_{\mathcal{R}}[End_{\mathcal{R}}]$.

3. Similarly, build $L_{\mathcal{B}}$ for $2^{8\cdot\mathrm{DoF}_{\mathcal{B}}}$ values of 🟦 neutral space with forward computation.
4. Check for the DoM bytes match on indices between $L_{\mathcal{R}}$ and $L_{\mathcal{B}}$.
5. For the pairs surviving the partial match, check for a full-state match.
6. Steps 1–5 form one MitM episode that will be repeated until a full match is found.

The attack complexity. An MitM episode is performed with time $2^{8\cdot\max(\mathrm{DoF}_{\mathcal{R}},\mathrm{DoF}_{\mathcal{B}})} + 2^{8\cdot(\mathrm{DoF}_{\mathcal{R}}+\mathrm{DoF}_{\mathcal{B}}-\mathrm{DoM})}$. To find an h-bit target preimage, $2^{h-8\cdot(\mathrm{DoF}_{\mathcal{R}}+\mathrm{DoF}_{\mathcal{B}})}$ MitM episodes are needed. The total time complexity of the attack is:

$$2^{h-8\cdot\min(\mathrm{DoF}_{\mathcal{R}},\mathrm{DoF}_{\mathcal{B}},\mathrm{DoM})}. \tag{1}$$

Nonlinearly Constrained Neutral Words [24]**.** In order to compute the allowable values for the neutral words, one has to solve certain systems of equations. In previous MitM preimage attacks [46,50], the systems of equations are usually linear, i.e., *linearly constrained neutral words*, which can be solved with ease. At CRYPTO 2021, Dong *et al.* [24] found that the systems of equations can be nonlinear, which can not be solved directly like linear system. Therefore, Dong *et al.* proposed a table-based method to solve those *nonlinearly constrained neutral words*. Suppose in the starting state, there are $\lambda_{\mathcal{R}}$ 🟥 bytes and $\lambda_{\mathcal{B}}$ 🟦 bytes, and the number of nonlinear constraints are $l_{\mathcal{R}}$ and $l_{\mathcal{B}}$ for 🟥 and 🟦 bytes.

1. Fix the ⬛ bytes for the initial structure,
2. For $2^{\lambda_{\mathcal{R}}}$ 🟥 values, compute the $l_{\mathcal{R}}$ bytes constraints (denoted as $\mathfrak{c}_{\mathcal{R}} \in \mathbb{F}_2^{8\cdot l_{\mathcal{R}}}$), and store the $\lambda_{\mathcal{R}}$ 🟥 bytes in table $U_{\mathcal{R}}[\mathfrak{c}_{\mathcal{R}}]$,
3. For $2^{\lambda_{\mathcal{B}}}$ 🟦 values, compute the $l_{\mathcal{B}}$ bytes constraints (denoted as $\mathfrak{c}_{\mathcal{B}} \in \mathbb{F}_2^{8\cdot l_{\mathcal{B}}}$), and store the $\lambda_{\mathcal{B}}$ 🟦 bytes in table $U_{\mathcal{B}}[\mathfrak{c}_{\mathcal{B}}]$.

Then, for given $\mathfrak{c}_{\mathcal{R}}$ and $\mathfrak{c}_{\mathcal{B}}$, the values in $U_{\mathcal{R}}[\mathfrak{c}_{\mathcal{R}}]$ and $U_{\mathcal{B}}[\mathfrak{c}_{\mathcal{B}}]$ can be computed independently (i.e., neutral) in one MitM episode. Therefore, we have $\mathrm{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}}$ and $\mathrm{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}}$. According to [24], both the time and memory complexities of one precomputation are $2^{\lambda_{\mathcal{R}}} + 2^{\lambda_{\mathcal{B}}}$. After the precomputation, $2^{l_{\mathcal{R}}+l_{\mathcal{B}}}$ MitM episodes are produced.

Automated MitM Based MILP. At EUROCRYPT 2021, Bao *et al.* [6] proposed the MILP-based automatic model for MitM preimage attacks on `AES`-like hashing. At CRYPTO 2021, Dong *et al.* extended the model into key-recovery and collision. At CRYPTO 2022, Bao *et al.* [7] proposed the superposition MitM attack, i.e., the 🟦 bytes and 🟥 bytes are handled independently in linear operations. A similar idea has been proposed and called indirect-partial matching in 2009 [2]. In the superposition MitM attack framework, each state involved in a linear operation is separated into two virtual states, which are also called superposition states. One state preserves the 🟦 bytes, ⬛ bytes, and □ bytes in the original state, while the positions where 🟥 bytes are located turn ⬛. The other state can be obtained similarly but exchanging the 🟦 and 🟥 bytes. Therefore, two superposition states can be propagated equally and independently along the forward or backward computation paths through linear operations. The initial DoFs can be consumed in both directions. Then, two superposition states are finally combined before the next nonlinear operation after a series of linear operations. The color patterns and how the states are separated and combined are visualized in Fig. 2.

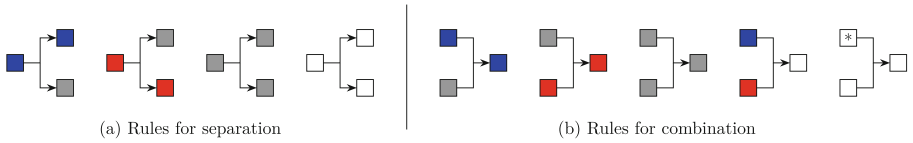


Fig. 2. Rules for separation and combination, where "$*$" means any color

The rules `MC-Rule` and `XOR-Rule` are first introduced in [6] to model the propagation rules of `MixColumn` and `AddRoundKey` in `AES`-like hashing. Since $\lambda_{\mathcal{B}}$ 🟦 bytes of the starting states are imposed $l_{\mathcal{B}}$ constraints (similar to 🟥), the rules `MC-Rule` and `XOR-Rule` are required to describe how the impacts from the neutral bytes in one chunk are limited on the opposite chunk. For more details on the two basic rules, please refer to [6] and also Supplementary Material A in our full version paper [32].

2.3 `AES`

To be concrete, we first recall the round function of `AES-128` [19]. It operates on a 16-byte state arranged into a 4×4 matrix and contains four operations as illustrated in Fig. 3: `SubBytes` (`SB`), `ShiftRows` (`SR`), `MixColumns` (`MC`), and `AddRoundKey` (`AK`). The `MixColumns` is to multiply an MDS matrix to each column of the state. Embedding a block cipher into the PGV hashing modes [43], such as Davies-Meyer (DM, Fig. 4), Matyas-Meyer-Oseas (MMO, Fig. 5) and Miyaguchi-Preneel (MP), is a common way to build the compression functions for hashing.

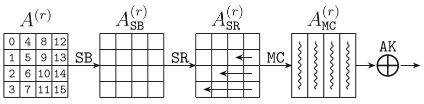


Fig. 3. One round `AES`

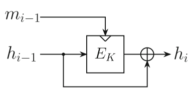


Fig. 4. DM

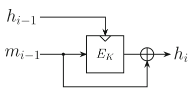


Fig. 5. MMO

2.4 (Generalized) Feistel Networks

Another widely used design approach is the Feistel network, which was first used in DES [18], and the generalized Feistel network (GFN) [54]. When the round function of Feistel adopts `AddRoundKey` (`AK`), `SubBytes` (`SB`), and a permutation layer, i.e., SP round function, the Feistel is named as Feistel-SP. In this paper, the permutation layer is a `MixColumns` (`MC`) with MDS, as shown in Fig. 6. Figure 7 is an equivalent transformation of Fig. 6, where $\tilde{A}^{(r)} = \texttt{MC}^{-1}(A^{(r)})$, $\tilde{B}^{(r)} = \texttt{MC}^{-1}(B^{(r)})$, $\tilde{A}^{(r+1)} = \texttt{MC}^{-1}(A^{(r+1)})$, and $\tilde{B}^{(r+1)} = \texttt{MC}^{-1}(B^{(r+1)})$. The round function of GFN adopts multiple branches, e.g., the round function of 4-branch `Simpira v2` in Fig. 8.

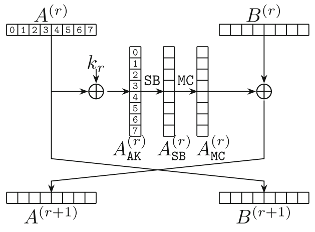
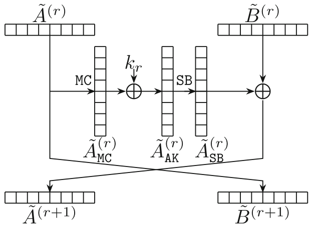


Fig. 6. One round Feistel-SP



$\tilde{A}^{(r)}$
$\tilde{B}^{(r)}$
MC
k_r
SB
$\tilde{A}^{(r)}_{\texttt{MC}}$
$\tilde{A}^{(r)}_{\texttt{AK}}$
$\tilde{A}^{(r)}_{\texttt{SB}}$
$\tilde{A}^{(r+1)}$
$\tilde{B}^{(r+1)}$

Fig. 7. Equivalent transform of Feistel-SP

2.5 `Simpira v2`

`Simpira v2` [29] is a family of cryptographic permutations that support inputs of $128 \times b$ bits, where b is the number of branches. When $b = 1$, `Simpira v2` consists of 12 rounds `AES` with different constants. When $b \geq 2$, `Simpira v2` is a Generalized Feistel Structure (GFS) with the F-function that consists of two rounds of `AES`. We denote `Simpira v2` family members with b branches as `Simpira`-b. The total number of rounds is 15 for $b = 2$, $b = 4$ and $b = 6$, 21 for $b = 3$, and 18 for $b = 8$. Figure 8 shows the round function of `Simpira`-4.

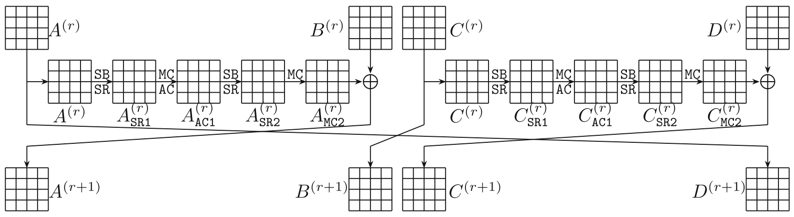


Fig. 8. The round function of `Simpira`-4

2.6 Areion

Areion [35] is a family of highly-efficient permutations based on AES instruction. It consists of two versions with 256-bit and 512-bit, named as Areion-256 (the round function is shown in Fig. 9) and Areion-512. Based on the two permutations, two hash functions with short input are designed with Davies-Meyer (DM) construction, i.e., Areion256-DM and Areion512-DM, which are our targets.

2.7 Lesamnta-LW

Lesamnta-LW is a lightweight 256-bit hash function proposed by Hirose *et al.* in 2010 [31], which has been specified in ISO/IEC 29192-5:2016. Lesamnta-LW is a Merkle-Damgård iterated hash function [20,42]. Figure 11 shows a hash with two message blocks, where the i-th compression function (CF) is $\texttt{CF}(h_{i-1}, m_i) = E(h^0_{i-1}, m_i \| h^1_{i-1}) = h_i$, with h^0_{i-1}, h^1_{i-1}, $m_i \in \mathbb{F}_2^{128}$, h_{i-1}, $h_i \in \mathbb{F}_2^{256}$, and $h_{i-1} = h^0_{i-1} \| h^1_{i-1}$. The initial h_0 is the initial vector and the last h_N is the 256-bit digest. The internal block cipher of CF is of 64 rounds with 256-bit plaintext and 32-bit round keys. Our attack is independent of the key schedule which is omitted. Figure 10 shows the round function, where $m_i = A^{(r)} \| B^{(r)}$, $h^1_{i-1} = C^{(r)} \| D^{(r)}$. Lesamnta-LW uses AES's components, i.e., SB and MC, while P just permutes the bytes. Lesamnta-LW claims at least 2^{120} security levels against both collision and preimage attacks, and we target the MitM collision attack on Lesamnta-LW.

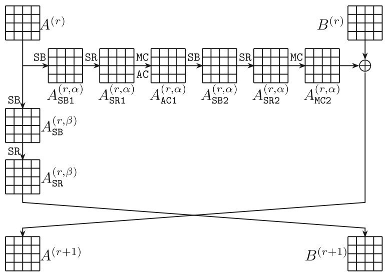

$A^{(r)}$
$B^{(r)}$
SB
SR
MC
AC
SB
SR
MC
$A^{(r,\alpha)}_{\texttt{SB1}}$
$A^{(r,\alpha)}_{\texttt{SR1}}$
$A^{(r,\alpha)}_{\texttt{AC1}}$
$A^{(r,\alpha)}_{\texttt{SB2}}$
$A^{(r,\alpha)}_{\texttt{SR2}}$
$A^{(r,\alpha)}_{\texttt{MC2}}$
SB
$A^{(r,\beta)}_{\texttt{SB}}$
SR
$A^{(r,\beta)}_{\texttt{SR}}$
$A^{(r+1)}$
$B^{(r+1)}$

Fig. 9. One round Areion-256

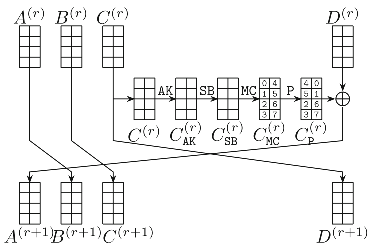

$A^{(r)}$ $B^{(r)}$ $C^{(r)}$
$D^{(r)}$
AK
SB
MC
P
$C^{(r)}$ $C^{(r)}_{\texttt{AK}}$ $C^{(r)}_{\texttt{SB}}$ $C^{(r)}_{\texttt{MC}}$ $C^{(r)}_{\texttt{P}}$
$A^{(r+1)}$ $B^{(r+1)}$ $C^{(r+1)}$
$D^{(r+1)}$

Fig. 10. One round Lesamnta-LW

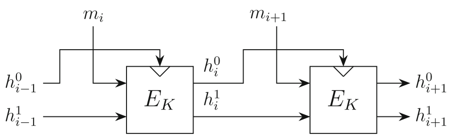


Fig. 11. Lesamnta-LW hash with two message blocks

2.8 Sasaki's Preimage Attack on Feistel-SP

At ACNS 2013, Sasaki [47] introduced the MitM preimage attacks on MMO hashing mode with Feistel-SP block ciphers by omitting the last network twist. In Fig. 12(a), $A^{(6)}_{\mathtt{AK}}$ and $A^{(7)}_{\mathtt{AK}}$ are chosen as the initial states with $\lambda_{\mathcal{R}} = 11$ and $\lambda_{\mathcal{B}} = 3$. The 🟩 just represents the linear combination of 🟥 and 🟦 bytes. From $B^{(7)}$ to $A^{(8)}$, the consumed DoF of 🟥 is $l_{\mathcal{R}} = 8$. Therefore, the remaining DoFs of 🟦 and 🟥 are $\mathrm{DoF}_{\mathcal{R}} = 11 - 8 = 3$ and $\mathrm{DoF}_{\mathcal{B}} = 3$, respectively. In Fig. 12(b), by assigning conditions $k_0 = k_{10} \oplus H_A$ and $k_1 = k_9 \oplus H_B$, we have $A^{(10)}_{\mathtt{MC}} = A^{(0)}_{\mathtt{MC}}$ and $A^{(9)}_{\mathtt{MC}} = A^{(1)}_{\mathtt{MC}}$. Therefore, $A^{(2)} = B^{(9)} \oplus H_A$ and $B^{(2)} = A^{(9)} \oplus H_B$. In Fig. 12(c), Sasaki applied a linear transformation in the computation from $A^{(3)}_{\mathtt{SB}}$ to $A^{(5)}_{\mathtt{SB}}$ to derive a multi-round matching with DoM = 2 as shown in Fig. 13. The time complexity is $2^{8\times(16-\min\{3,3,2\})} = 2^{112}$.

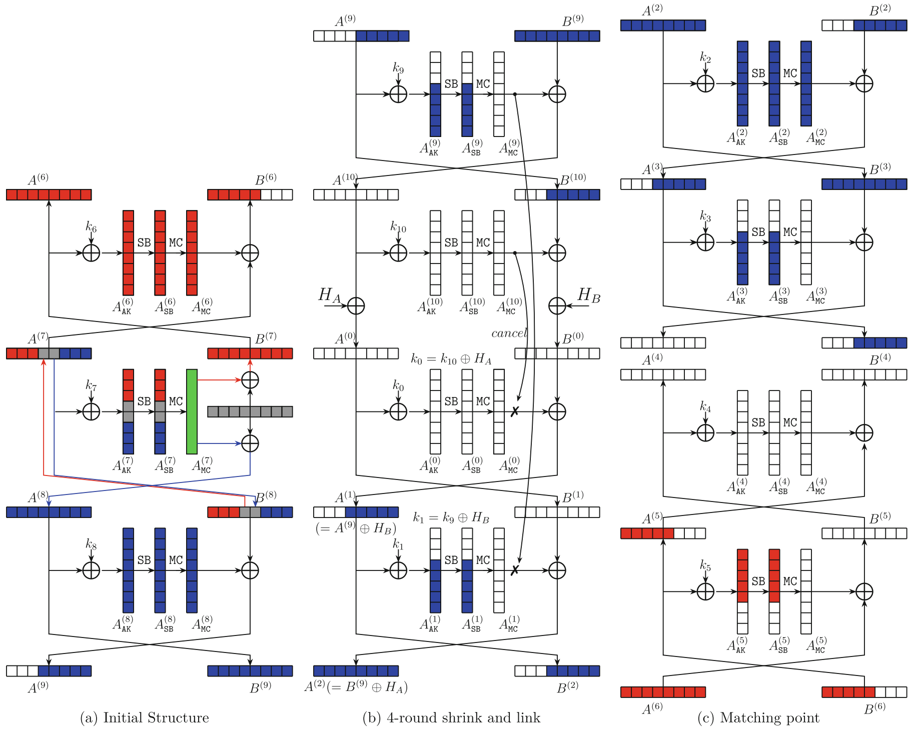
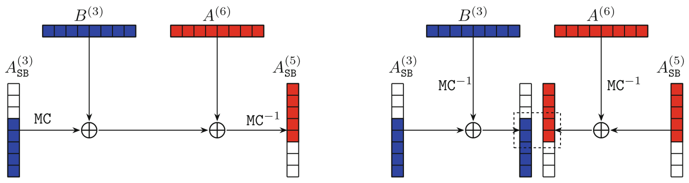


Fig. 12. Sasaki's attack



(a) Original (b) Transformed

Fig. 13. Matching in Sasaki's attack

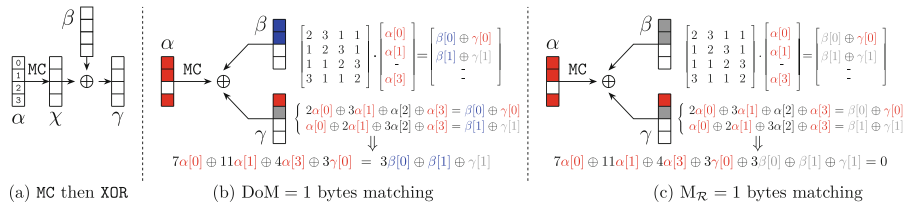
3 Generalization on Matching Strategy in MitM

In the matching point of the MitM attack, with forward and backward computations, if two matching states F^+ and F^- are determined only by the 🟦 and 🟥, respectively, then, the relation $F^+ = F^-$ acts as a *direct partial matching*. This simple matching strategy is frequently used in previous works [46,49]. In ASIACRYPT 2009, Aoki *et al.* introduced the *indirect partial matching technique* [2], where F^+ can be expressed as $\phi_{\mathcal{B}} + \phi_{\mathcal{R}}$, and $F^- = \Phi_{\mathcal{B}} + \Phi_{\mathcal{R}}$. $\phi_{\mathcal{B}}$ and $\Phi_{\mathcal{B}}$ are determined by the 🟦 and ⬛ bytes. $\phi_{\mathcal{R}}$ and $\Phi_{\mathcal{R}}$ are determined by the 🟥 and ⬛ bytes. Therefore, the DoM-byte equation $\phi_{\mathcal{B}} + \Phi_{\mathcal{B}} = \phi_{\mathcal{R}} + \Phi_{\mathcal{R}}$ can be built from $F^+ = F^-$, which acts as the matching. In this paper, we denote $End_{\mathcal{B}} = \phi_{\mathcal{B}} + \Phi_{\mathcal{B}}$ and $End_{\mathcal{R}} = \phi_{\mathcal{R}} + \Phi_{\mathcal{R}}$.

In addition to the above two common matching strategies, we find that the byte equation determined only by one of the two colors (🟦, 🟥) can also be used in the MitM attack. Taking the matching by combining `MixColumn` and `XOR` operations at `MixColumns` and `AddRoundKey` for `AES` as an example as shown in Fig. 14(a). Suppose from the matching states, there exist $\text{M}_{\mathcal{R}}$ byte-equations $\pi_{\mathcal{R}} = 0$, $\text{M}_{\mathcal{B}}$ byte-equations $\pi_{\mathcal{B}} = 0$, and DoM byte-equations $End_{\mathcal{B}} = End_{\mathcal{R}}$, where $End_{\mathcal{R}}$ and $\pi_{\mathcal{R}}$ are determined by 🟥 and ⬛, $End_{\mathcal{B}}$ and $\pi_{\mathcal{B}}$ are determined by 🟦 and ⬛. Figure 14(b) is a commonly used matching strategy (indirect partial matching) in previous MitM attacks [46,47], where there exists DoM = 1 byte matching equation $End_{\mathcal{B}} = End_{\mathcal{R}}$. Figure 14(c) is the new matching strategy, where there exists $\text{M}_{\mathcal{R}} = 1$ byte matching equation:

$$\pi_{\mathcal{R}} = 7\alpha[0] \oplus 11\alpha[1] \oplus 4\alpha[3] \oplus 3\gamma[0] \oplus 3\beta[0] \oplus \beta[1] \oplus \gamma[1] = 0.$$

This matching method in Fig. 14(c) can not be included in any of the two common matching strategies (direct or indirect partial matching), but can still lead to valid MitM attacks. With the new matching strategy, we introduce the new MitM procedures in the following:



(a) MC then XOR
(b) DoM = 1 bytes matching
(c) M$_{\mathcal{R}}$ = 1 bytes matching
$7\alpha[0] \oplus 11\alpha[1] \oplus 4\alpha[3] \oplus 3\gamma[0] = 3\beta[0] \oplus \beta[1] \oplus \gamma[1]$
$7\alpha[0] \oplus 11\alpha[1] \oplus 4\alpha[3] \oplus 3\gamma[0] \oplus 3\beta[0] \oplus \beta[1] \oplus \gamma[1] = 0$

Fig. 14. Examples in Generalized Matching Strategy

1. Choose constants for the initial structure.
2. For all $2^{8 \cdot \mathrm{DoF}_{\mathcal{R}}}$ values of 🟥 neutral space, compute from the initial structure to the matching points. If $\pi_{\mathcal{R}} = 0$ holds, store the $\mathrm{DoF}_{\mathcal{R}}$ 🟥 bytes in table $L_{\mathcal{R}}[End_{\mathcal{R}}]$.
3. For all $2^{8 \cdot \mathrm{DoF}_{\mathcal{B}}}$ values of 🟦 neutral space, compute from the initial structure to the matching points. If $\pi_{\mathcal{B}} = 0$ holds, store the $\mathrm{DoF}_{\mathcal{B}}$ 🟦 bytes in table $L_{\mathcal{B}}[End_{\mathcal{B}}]$.
4. Check for the DoM bytes matching with $End_{\mathcal{R}} = End_{\mathcal{B}}$ on indices between $L_{\mathcal{R}}$ and $L_{\mathcal{B}}$.
5. For the pairs surviving the partial matching, check for a full-state match.
6. Steps 1–5 form one MitM episode that will be repeated until a full match is found.

The Complexity. In one MitM episode, the time complexities of Step 2 and 3 are $2^{8 \cdot \mathrm{DoF}_{\mathcal{R}}}$ and $2^{8 \cdot \mathrm{DoF}_{\mathcal{B}}}$, respectively. The memory costs of Step 2 and 3 are $2^{8(\mathrm{DoF}_{\mathcal{R}} - \mathrm{M}_{\mathcal{R}})}$ and $2^{8(\mathrm{DoF}_{\mathcal{B}} - \mathrm{M}_{\mathcal{B}})}$. In Step 4 and 5, there expect $2^{8(\mathrm{DoF}_{\mathcal{R}} - \mathrm{M}_{\mathcal{R}})} \cdot 2^{8(\mathrm{DoF}_{\mathcal{B}} - \mathrm{M}_{\mathcal{B}}) - 8 \cdot \mathrm{DoM}}$ surviving pairs to check for a full-state match. Therefore, the time complexity of one MitM episode is

$$2^{8 \cdot \mathrm{DoF}_{\mathcal{R}}} + 2^{8 \cdot \mathrm{DoF}_{\mathcal{B}}} + 2^{8(\mathrm{DoF}_{\mathcal{R}} + \mathrm{DoF}_{\mathcal{B}} - \mathrm{M}_{\mathcal{R}} - \mathrm{M}_{\mathcal{B}} - \mathrm{DoM})}.$$

For a given h-bit target, $2^{h - 8(\mathrm{DoF}_{\mathcal{R}} + \mathrm{DoF}_{\mathcal{B}})}$ MitM episodes are needed to perform, and the total time complexity is

$$2^{h - 8 \cdot \min(\mathrm{DoF}_{\mathcal{R}}, \mathrm{DoF}_{\mathcal{B}}, \mathrm{M}_{\mathcal{R}} + \mathrm{M}_{\mathcal{B}} + \mathrm{DoM})}. \tag{2}$$

Remark 1. Compared with the attack framework proposed by Bao *et al.* [6], steps 2–3 in our framework will first filter the states that do not satisfy the matching equations containing only one color, and then store the remaining states in tables. The overall memory is $2^{8 \times \min\{\mathrm{DoF}_{\mathcal{R}} - \mathrm{M}_{\mathcal{R}}, \mathrm{DoF}_{\mathcal{B}} - \mathrm{M}_{\mathcal{B}}\}}$ which may be lower than the main memory cost in [6], i.e. $2^{8 \times \min\{\mathrm{DoF}_{\mathcal{R}}, \mathrm{DoF}_{\mathcal{B}}\}}$.

Modelling the Matching Point. For a given byte in Fig. 14, we introduce a Boolean variable ω, that $\omega = 1$ means this byte is □, otherwise $\omega = 0$. ω_i^{α}, ω_i^{β}, and ω_i^{γ} indicate whether the i-th byte in α, β, and γ is white respectively, and $\omega_i^{(\beta,\gamma)}$ is defined by $\texttt{OR}(\omega_i^{\beta}, \omega_i^{\gamma})$, i.e., $\omega_i^{(\beta,\gamma)} = 1$ if ω_i^{β} or ω_i^{γ} is 1. Besides, an auxiliary state χ is introduced in Fig. 14, where $\chi = \beta \oplus \gamma$. The rule to generate χ follows the `XOR-Rule` in [6], (i.e. 🟦 $\oplus$ 🟥 $=$ □, 🟦 $\oplus$ 🟦 $=$ 🟦, □ $\oplus$ 🟦 $=$ □, etc.). Moreover, we introduce 4 general variables $n_{\mathcal{B}}^{\alpha}$, $n_{\mathcal{R}}^{\alpha}$, $n_{\mathcal{B}}^{\chi}$ and $n_{\mathcal{R}}^{\chi}$ to count the numbers of 🟦 cells and ▨ cells or the number of 🟥 cells and ▨ cells in α or χ. For example, $n_{\mathcal{B}}^{\alpha}$ is the number of 🟦 cells and ▨ cells in α. Another general variable $n_{\mathcal{G}}$ is introduced to count the total number of ▨ cells in α and χ. Suppose $(x_i^{\alpha}, y_i^{\alpha})$ and (x_i^{χ}, y_i^{χ}) denote the i-th cell in α and χ respectively, then we have

$$\begin{cases} n_{\mathcal{B}}^{\alpha} = \sum_{i=0}^{3} x_i^{\alpha}; \\ n_{\mathcal{R}}^{\alpha} = \sum_{i=0}^{3} y_i^{\alpha}; \end{cases} \qquad \begin{cases} n_{\mathcal{B}}^{\chi} = \sum_{i=0}^{3} x_i^{\chi}; \\ n_{\mathcal{R}}^{\chi} = \sum_{i=0}^{3} y_i^{\chi}; \end{cases} \qquad n_{\mathcal{G}} = \sum_{i=0}^{3} \texttt{AND}(x_i^{\alpha}, y_i^{\alpha}) + \texttt{AND}(x_i^{\chi}, y_i^{\chi}).$$

where $\texttt{AND}(x_i, y_i) = 1$ if and only if $x_i = y_i = 1$. To avoid double counting the number of equations derived only by ▨, let $\mathrm{M}_{\mathcal{G}} = \max\{0, n_{\mathcal{G}} - 4\}$ and exclude $\mathrm{M}_{\mathcal{G}}$ equations from $\pi_{\mathcal{R}} = 0$. Then, the number of equations in $\pi_{\mathcal{B}} = 0$ and $\pi_{\mathcal{R}} = 0$ can be calculated by

$$\mathrm{M}_{\mathcal{B}} = \max\{0, \ \ n_{\mathcal{B}}^{\alpha} + n_{\mathcal{B}}^{\chi} - 4\}, \qquad \mathrm{M}_{\mathcal{R}} = \max\{0, \ \ n_{\mathcal{R}}^{\alpha} + n_{\mathcal{R}}^{\chi} - \mathrm{M}_{\mathcal{G}} - 4\}. \quad (3)$$

For the `MC` then `XOR` operations in Fig. 14, we can build $4 - \sum_{i=0}^{3}(\omega_i^{(\beta,\gamma)} + \omega_i^{\alpha})$ linear equations which are determined by only known cells (🟦, 🟥, ▨). Therefore, the number of byte equations $End_{\mathcal{B}} = End_{\mathcal{R}}$ is equal to the total linear equations minus $M_{\mathcal{B}}$ and $M_{\mathcal{R}}$ equations. We get

$$\mathrm{DoM} = \max\left\{0, \ \ 4 - \sum_{i=0}^{3}(\omega_i^{(\beta,\gamma)} + \omega_i^{\alpha}) - \mathrm{M}_{\mathcal{B}} - \mathrm{M}_{\mathcal{R}}\right\}. \quad (4)$$

4 Automatic Model for Transformed Feistel Structure

In this section, we first generalize Sasaki's multi-round matching strategy into full-round matching. Then, we introduce an equivalent transformation of Feistel and GFN, which is very friendly with the new proposed full-round matching strategy. At last, we construct the MILP constraints to describe the attributes propagation through transformed Feistel and how the full-round match is deployed. Combining the equivalent transformation and full-round match, the MILP model can be simplified and easy to program.

4.1 The Generalization of Sasaki's Matching Strategy for Feistel

In [47], Sasaki proposed a matching strategy for Feistel with a linear transformation. As shown in Fig. 13, it is hard to see any matching in the original Fig. 13(a). However, after a linear transformation in Fig. 13(b), the two-byte matching is obviously obtained. Besides the attack on balanced Feistel-SP, Sasaki [47] also built MitM attacks on GFN with SP round function, where the matching point covers 7 consecutive rounds. A similar linear transformation as in Fig. 13(b) is also applied, but involves more internal states.

Inspired by Sasaki's matching strategy [47], we generalize the matching strategy to full-round matching, i.e., the matching can happen by writing down the internal states involved from the *matching point* to the *initial structure*. For example, we can further extend Fig. 13(a) by replacing $B^{(3)}$ by $\texttt{MC}(A_{\texttt{SB}}^{(7)}) \oplus B^{(7)} \oplus H_A$ and replacing $A^{(6)}$ by $B^{(7)}$, where the internal states $A_{\texttt{SB}}^{(7)}$ and $B^{(7)}$ come from the initial structure. Therefore, Fig. 13 becomes Fig. 15. The advantages of the generalized full-round matching are summarized below:

I Since the internal states from the initial structure preserve more useful information than other internal states (there are usually no □ bytes in the initial structure), a full-round matching may be more likely to produce a valid match than a local-round matching (e.g., 3 or 4 rounds). An example is found for `Simpira`-4 in Fig. 18, where the matching obviously exists for the full-round case, but disappears for certain local-round case.

II Also a linear transformation is applied to Fig. 15(a) to obtain Fig. 15(b). This is essential and can not be replaced by Bao *et al.*'s superposition MitM technique [7]. If we apply the superposition MitM technique in Fig. 15(a), $A_{\texttt{SB}}^{(3)}$ will be separated into two states following the rules in Fig. 2, then one of the two states will be all □ after `MC`. Therefore, an unknown state will be XORed into the matching path, which leads to no matching at all.

If we apply a linear transformation to obtain Fig. 15(b), each byte of $A_{\texttt{SB}}^{(3)}$ will be involved in the matching path individually. For example, considering the 4-th byte, there is a one-byte equation

$$\texttt{MC}^{-1}\left(B^{(7)}\right)[3] \oplus A_{\texttt{SB}}^{(7)}[3] \oplus A_{\texttt{SB}}^{(3)}[3] \oplus A_{\texttt{SB}}^{(5)}[3] = \texttt{MC}^{-1}\left(B^{(7)} \oplus H_A\right)[3], \quad (5)$$

which is obviously a matching equation (no □ byte is involved).

III The transformed structure in Fig. 15(b) is easy to program in the automatic tool. As shown in Eq. (5), each byte can be individually considered, which is very friendly than the untransformed case in Fig. 15(a). As a matter of fact, this is very important when building the automatic tool, since for many (generalized) Feistel networks, the situation is much more complex than the very easy case for Feistel-SP. For example, in our 11-round attack on `Simpira`-4 (Fig. 23), there are more states involved in matching than that in Fig. 15(a). Therefore, if we do not apply the linear transformation, we have to program many `MC` operations into a whole matching rule, which is very complex or even infeasible for many ciphers like `Simpira`-4.

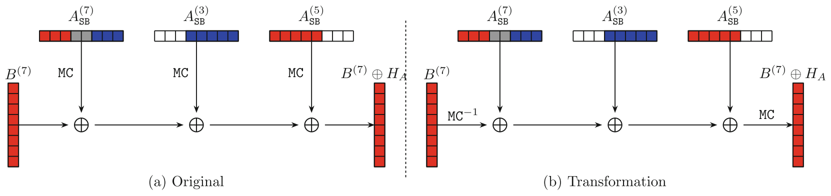


Fig. 15. Full-match in Feistel-SP

We find that the transformation in Fig. 15(b) can be directly obtained if we consider MitM attacks on an equivalent transformation of Feistel-SP, i.e., Fig. 6(b). To better understand this fact, we take the MILP-based MitM attack on transformed `Simpira`-4 as an example in the following part.

4.2 MILP-Based MitM Attack on Transformed Feistel

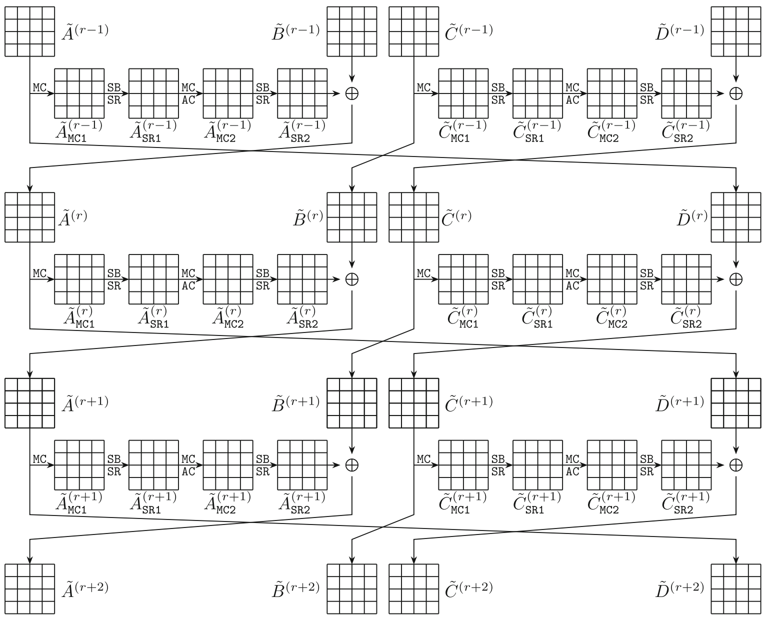
As shown in Fig. 8, the output $A^{(r+1)}$ is equivalent to $B^{(r)} \oplus \mathtt{MC}(A_{\mathtt{SR2}}^{(r)})$. With a linear transformation on $A^{(r+1)}$, we have $\mathtt{MC}^{-1}(A^{(r+1)}) = \mathtt{MC}^{-1}(B^{(r)}) \oplus A_{\mathtt{SR2}}^{(r)}$. Similarly, $B^{(r+1)}, C^{(r+1)}$ and $D^{(r+1)}$ can be handled in the same way. For the sake of simplicity and intuition, we transform the Feistel network by putting the last `MixColumn` operation first in each round like Fig. 6(b). Then the output of each round is the state after the above linear transformation in the original structure. Therefore, we propose the following property.

Property 1. `Simpira`-4 is equivalent to the permutation with a round function

$$\mathcal{R}'_i = \mathtt{SR} \circ \mathtt{SB} \circ \mathtt{AC} \circ \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB} \circ \mathtt{MC},$$

except for replacing the input $\left(A^{(r)}, B^{(r)}, C^{(r)}, D^{(r)}\right)$ by $\Big(\tilde{A}^{(r)}, \tilde{B}^{(r)}, \tilde{C}^{(r)}, \tilde{D}^{(r)}\Big) = \left(\mathtt{MC}^{-1}(A^{(r)}), \mathtt{MC}^{-1}(B^{(r)}), \mathtt{MC}^{-1}(C^{(r)}), \mathtt{MC}^{-1}(D^{(r)})\right)$, and the final output becomes $\left(\tilde{A}^{(r+1)}, \tilde{B}^{(r+1)}, \tilde{C}^{(r+1)}, \tilde{D}^{(r+1)}\right)$.

Following Property 1, we represent the 3-round transformed `Simpira`-4 in Fig. 16, where $\tilde{A}^{(r+1)} = \tilde{B}^{(r)} \oplus \tilde{A}_{\mathtt{SR2}}^{(r)}$. In this way, $\tilde{A}_{\mathtt{MC1}}^{(r)} = \mathtt{MC}(\tilde{A}^{(r)}) = A^{(r)}$, then $\tilde{A}_{\mathtt{SR2}}^{(r)} = A_{\mathtt{SR2}}^{(r)}$. According to the predefined $\tilde{B}^{(r)} = \mathtt{MC}^{-1}(B^{(r)})$, $\tilde{A}^{(r+1)}$ is equivalent to $\mathtt{MC}^{-1}(B^{(r)}) \oplus A_{\mathtt{SR2}}^{(r)}$. Therefore, the output $\tilde{A}^{(r+1)}$ in the transformed `Simpira`-4 is actual the state $\mathtt{MC}^{-1}(A^{(r+1)})$ in the original `Simpira`-4 (Fig. 8). This is also true for $\tilde{B}^{(r+1)}$, $\tilde{C}^{(r+1)}$ and $\tilde{D}^{(r+1)}$.



$\tilde{A}^{(r-1)}$
$\tilde{B}^{(r-1)}$
$\tilde{C}^{(r-1)}$
$\tilde{D}^{(r-1)}$
MC
SB SR
MC AC
$\tilde{A}^{(r-1)}_{\mathtt{MC1}}$
$\tilde{A}^{(r-1)}_{\mathtt{SR1}}$
$\tilde{A}^{(r-1)}_{\mathtt{MC2}}$
$\tilde{A}^{(r-1)}_{\mathtt{SR2}}$
$\tilde{C}^{(r-1)}_{\mathtt{MC1}}$
$\tilde{C}^{(r-1)}_{\mathtt{SR1}}$
$\tilde{C}^{(r-1)}_{\mathtt{MC2}}$
$\tilde{C}^{(r-1)}_{\mathtt{SR2}}$
$\tilde{A}^{(r)}$
$\tilde{B}^{(r)}$
$\tilde{C}^{(r)}$
$\tilde{D}^{(r)}$
$\tilde{A}^{(r)}_{\mathtt{MC1}}$
$\tilde{A}^{(r)}_{\mathtt{SR1}}$
$\tilde{A}^{(r)}_{\mathtt{MC2}}$
$\tilde{A}^{(r)}_{\mathtt{SR2}}$
$\tilde{C}^{(r)}_{\mathtt{MC1}}$
$\tilde{C}^{(r)}_{\mathtt{SR1}}$
$\tilde{C}^{(r)}_{\mathtt{MC2}}$
$\tilde{C}^{(r)}_{\mathtt{SR2}}$
$\tilde{A}^{(r+1)}$
$\tilde{B}^{(r+1)}$
$\tilde{C}^{(r+1)}$
$\tilde{D}^{(r+1)}$
$\tilde{A}^{(r+1)}_{\mathtt{MC1}}$
$\tilde{A}^{(r+1)}_{\mathtt{SR1}}$
$\tilde{A}^{(r+1)}_{\mathtt{MC2}}$
$\tilde{A}^{(r+1)}_{\mathtt{SR2}}$
$\tilde{C}^{(r+1)}_{\mathtt{MC1}}$
$\tilde{C}^{(r+1)}_{\mathtt{SR1}}$
$\tilde{C}^{(r+1)}_{\mathtt{MC2}}$
$\tilde{C}^{(r+1)}_{\mathtt{SR2}}$
$\tilde{A}^{(r+2)}$
$\tilde{B}^{(r+2)}$
$\tilde{C}^{(r+2)}$
$\tilde{D}^{(r+2)}$

Fig. 16. Equivalent transform of `Simpira`-4

MILP Constraints for the Computation Paths. As shown in Fig. 16, $\tilde{A}^{(r+1)}_{\mathtt{MC1}}$ can be computed by $\mathtt{MC}\left(\tilde{A}^{(r)}_{\mathtt{SR2}} \oplus \tilde{B}^{(r)}\right)$, where $\tilde{B}^{(r)}$ can be replaced by $\mathtt{MC}^{-1}\left(\tilde{C}^{(r-1)}_{\mathtt{MC1}}\right)$. Therefore, $\tilde{A}^{(r+1)}_{\mathtt{MC1}} = \mathtt{MC}\left(\tilde{A}^{(r)}_{\mathtt{SR2}}\right) \oplus \tilde{C}^{(r-1)}_{\mathtt{MC1}}$, which is also named as `MC-then-XOR-Rule`. In fact, if we sequentially compute the colors of $\tilde{A}^{(r+1)}_{\mathtt{MC1}}$ by computing $\tilde{B}^{(r)} = \mathtt{MC}^{-1}\left(\tilde{C}^{(r-1)}_{\mathtt{MC1}}\right)$ and then $\tilde{A}^{(r+1)}_{\mathtt{MC1}} = \mathtt{MC}\left(\tilde{A}^{(r)}_{\mathtt{SR2}} \oplus \tilde{B}^{(r)}\right)$, i.e., first apply `MC-Rule`, and then `XOR-Rule`, and then `MC-Rule`, we may lose many possible and useful color schemes even in the most advanced superposition MitM framework. An example is given in Fig. 17(a), when applying `MC-Rule` on the superposition states of $\tilde{C}^{(r-1)}_{\mathtt{MC1}}$, it will lead to all □ cells. Subsequently, $\tilde{A}^{(r+1)}_{\mathtt{MC1}}$ will end up with a full column of □ cells. However, if we apply the `MC-then-XOR-Rule` with superposition framework as shown in Fig. 17(b), three 🟥 cells will be preserved by consuming three 🟦 cells. This also fits our intuition, i.e. more linear operations yield a higher possibility of generating unknown cells.

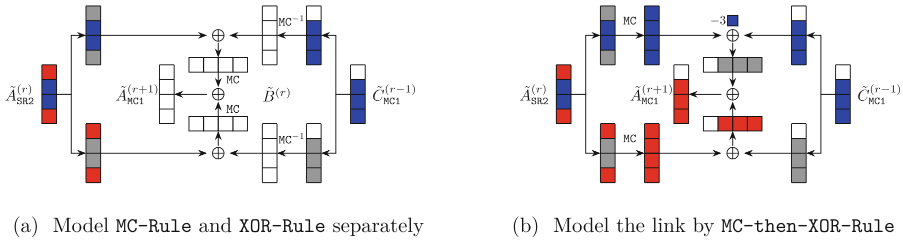

$\tilde{A}^{(r)}_{\texttt{SR2}}$
$\tilde{A}^{(r+1)}_{\texttt{MC1}}$
MC
MC
MC^{-1}
MC^{-1}
$\tilde{B}^{(r)}$
$\tilde{C}^{(r-1)}_{\texttt{MC1}}$
(a) Model `MC-Rule` and `XOR-Rule` separately
$\tilde{A}^{(r)}_{\texttt{SR2}}$
MC
MC
-3
$\tilde{A}^{(r+1)}_{\texttt{MC1}}$
$\tilde{C}^{(r-1)}_{\texttt{MC1}}$
(b) Model the link by `MC-then-XOR-Rule`

Fig. 17. The advantage of modeling link by applying `MC-then-XOR-Rule`

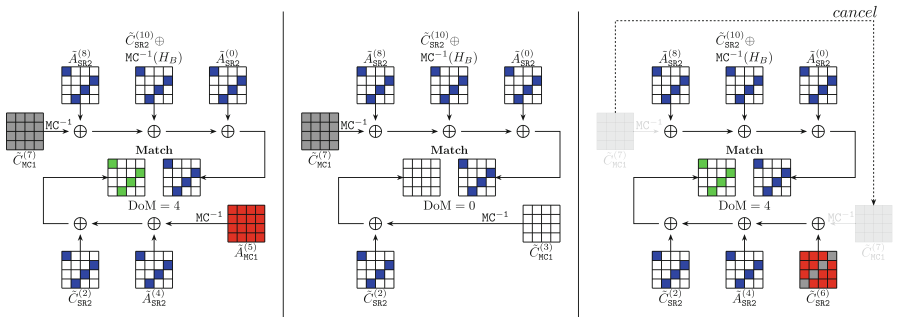
MILP Constraints for the Full-Round Match. In Fig. 12(c), the ending states are $(A^{(4)}, B^{(4)})$ computed from two opposite directions. With a linear transformation, two-byte partial matching is deduced as shown in Fig. 13. The matching phase involves two rounds of forward and two rounds of backward, respectively. So we denote such multi-round matching as *(2+2)-round match*. Taking the transformed `Simpira`-4 as an example, assume that the output state $\tilde{A}^{(r+1)}$ is chosen to be the ending states in Fig. 16. We have

$$\tilde{A}^{(r+1)} = \tilde{A}^{(r)}_{\texttt{SR2}} \oplus \tilde{B}^{(r)}, \text{ where } \tilde{B}^{(r)} = \texttt{MC}^{-1}\left(\tilde{C}^{(r-1)}_{\texttt{MC1}}\right). \tag{6}$$

As mentioned above, $\tilde{C}^{(r-1)}_{\texttt{MC1}}$ can be computed directly by $\texttt{MC}\left(\tilde{C}^{(r-2)}_{\texttt{SR2}}\right) \oplus \tilde{A}^{(r-3)}_{\texttt{MC1}}$ in the transformed `Simpira`-4 model. Hence, $\tilde{B}^{(r)}$ can be replaced by $\tilde{C}^{(r-2)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}\left(\tilde{A}^{(r-3)}_{\texttt{MC1}}\right)$ in Eq. (6). Immediately, $\tilde{A}^{(r-3)}_{\texttt{MC1}}$ can also be replaced in the same way. Subsequently, this replacement is done round by round until the initial structure to build the so-called *full-round matching*. Take our 11-round attack (Fig. 23) on transformed `Simpira`-4 in Sect. 6.2 as an example. The ending state $\tilde{D}^{(2)}$ is computed forward and backward to the initial structure. The shortest round that a matching exists is the $(6, 4)$-round matching given in Fig. 18(a). If a shorter round is considered for matching, e.g., $(6, 2)$-round in Fig. 18(b), there will be no matching, since the state $\tilde{C}^{(3)}_{\texttt{MC1}}$ will be all □. If we extend the $(6, 4)$-round matching to the full-round matching, we get Fig. 18(c), where the two states applied $\texttt{MC}^{-1}$ in both directions will eventually converge to an identical state $\tilde{C}^{(7)}_{\texttt{MC1}}$ in the initial structure. Figure 18(c) can also be displayed with the following full-round matching Eq. (7):

$$\texttt{MC}^{-1}\left(\tilde{C}^{(7)}_{\texttt{MC1}}\right) \oplus \tilde{A}^{(8)}_{\texttt{SR2}} \oplus \tilde{C}^{(10)}_{\texttt{SR2}} \oplus \tilde{A}^{(0)}_{\texttt{SR2}} \oplus \tilde{C}^{(2)}_{\texttt{SR2}} \oplus \tilde{A}^{(4)}_{\texttt{SR2}} \oplus \tilde{C}^{(6)}_{\texttt{SR2}} = \texttt{MC}^{-1}\left(\tilde{C}^{(7)}_{\texttt{MC1}} \oplus H_B\right), \tag{7}$$

where $\texttt{MC}^{-1}\left(\tilde{C}^{(7)}_{\texttt{MC1}}\right)$ can be cancelled in both sides. The reason follows the fact that the initial degrees of freedom of 🟦 and 🟥 cells will be consumed along the forward or backward computation path. The number of □ cells only becomes bigger through some linear or nonlinear operations. If the matching happens within shorter rounds, there will only be more matching cases after elongation. But on the contrary, while considering to find a shorter-round match from a longer one, there may be cases where the state in the shorter rounds will be □ after applying linear operations.



$\tilde{A}^{(8)}_{\texttt{SR2}}$
$\tilde{C}^{(10)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B)$
$\tilde{A}^{(0)}_{\texttt{SR2}}$
$\texttt{MC}^{-1}$
$\tilde{C}^{(7)}_{\texttt{MC1}}$
Match
DoM = 4
DoM = 0
$\tilde{A}^{(5)}_{\texttt{MC1}}$
$\tilde{C}^{(3)}_{\texttt{MC1}}$
$\tilde{C}^{(2)}_{\texttt{SR2}}$
$\tilde{A}^{(4)}_{\texttt{SR2}}$
$\tilde{C}^{(6)}_{\texttt{SR2}}$
cancel

(a) (6,4)-round matching **(b)** No matching in (6,2)-round **(c)** Matching through full-round

– In 18(a), 🟩 cell is the linear combination of 🟦 cells and 🟥 cells.
– In 18(b), $\tilde{C}^{(3)}_{\texttt{MC1}}$ is computed by $\texttt{MC}^{-1}\left(\tilde{A}^{(4)}_{\texttt{SR2}}\right) \oplus \tilde{A}^{(5)}_{\texttt{MC1}}$ in 18(a). Since there are □ cells in each column of $\tilde{A}^{(4)}_{\texttt{SR2}}$, the cells in $\tilde{C}^{(3)}_{\texttt{MC1}}$ become all unknown.
– In 18(c), $\texttt{MC}^{-1}\left(\tilde{A}^{(5)}_{\texttt{MC1}}\right)$ is replaced by $\texttt{MC}^{-1}\left(\tilde{C}^{(7)}_{\texttt{MC1}}\right) \oplus \tilde{C}^{(6)}_{\texttt{SR2}}$. The two states to perform $\texttt{MC}^{-1}$ converge to $\tilde{C}^{(7)}_{\texttt{MC1}}$, so both of them can be canceled in two directions.

Fig. 18. The (6,4)-round match in `Simpira`-4, and its impacts on the match after being shortened or elongated

Following the above study, we only need to consider whether there exist match cells in the full-round matching. The two states to perform $\texttt{MC}^{-1}$ will eventually converge into the starting states in the initial structure, or even can be canceled in both matching directions as shown in Fig. 18(c). For the general case, assume the matching phase consists of two starting states I_1 and I_2, e.g., in Fig. 18(c) $I_1 = I_2 = \tilde{C}^{(7)}_{\texttt{MC1}}$, and assume t internal states $X_1, X_2, \cdots, X_t$ are involved in the full-round matching equation. Similar to Eq. (7), the generic full-round matching equation can be written as

$$\texttt{MC}^{-1}(I_1) \oplus X_1 \oplus \cdots \oplus X_t = \texttt{MC}^{-1}(I_2). \tag{8}$$

The matching equation can be computed for each byte individually. In the i-th column and j-th row $(i, j = 0, 1, 2, 3)$, the byte matching equation is linearly computed from $X_k[4i + j]$ $(k = 1, \cdots, t)$ and $I_1[4i, 4i + 1, 4i + 2, 4i + 3]$ and $I_2[4i, 4i + 1, 4i + 2, 4i + 3]$. From our analysis on the generalization of matching in Sect. 3, if all these involved bytes are not □ bytes, there will be valid matching for MitM attack. For j-th byte of X_k, we introduce a Boolean variable $\omega_j^{X_k}$, where $\omega_j^{X_k} = 1$ means this byte is $\square$, otherwise $\omega_j^{X_k} = 0$. Let

$$\omega_{4i+j} = \texttt{OR}\left(\omega_{4i+j}^{X_1}, \cdots, \omega_{4i+j}^{X_t}, \omega_{4i}^{I_1}, \cdots, \omega_{4i+3}^{I_1}, \omega_{4i}^{I_2}, \cdots, \omega_{4i+3}^{I_2}\right).$$

If $\omega_{4i+j} = 0$, then we get one valid matching byte for MitM in the i-th column and j-th row.

5 Meet-in-the-Middle Attack on Reduced Feistel-SP

With our new model, we find a 12-round preimage attack of `Feistel-SP-MMO` as shown in Fig. 19, which improves Sasaki's attack [47] by 1 round. The starting states are $\tilde{A}_{\texttt{MC}}^{(7)}$ and $\tilde{A}_{\texttt{MC}}^{(8)}$. The initial DoFs for 🟦 and 🟥 are $\lambda_{\mathcal{B}} = 14$, $\lambda_{\mathcal{R}} = 2$, respectively.

From $\tilde{A}_{\texttt{MC}}^{(9)}$, $\tilde{A}_{\texttt{MC}}^{(6)}$ and $\tilde{A}_{\texttt{MC}}^{(5)}$, we get 12 constraints on forward neutral words and 0 constraints on backward neutral words, i.e. $l_{\mathcal{B}} = 12$, $l_{\mathcal{R}} = 0$. Then we have $\text{DoF}_{\mathcal{B}} = 2$ and $\text{DoF}_{\mathcal{R}} = 2$. The matching points are $\tilde{A}^{(5)}$ and $\tilde{B}^{(5)}$. But only a full-round match is found through $\tilde{B}^{(5)}$, which is

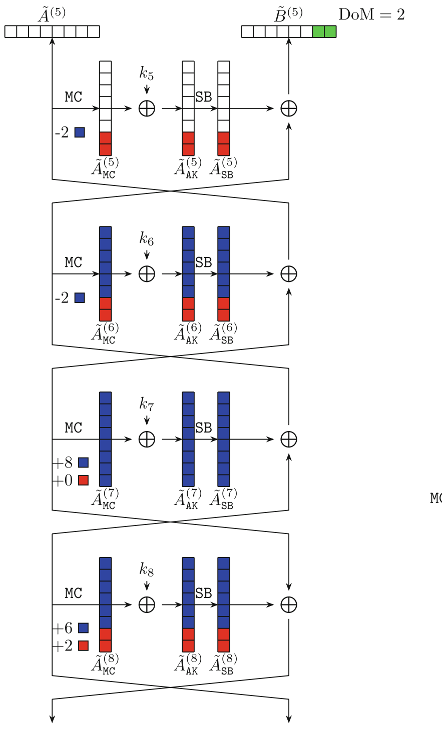
$$\texttt{MC}^{-1}\left(\tilde{A}_{\texttt{MC}}^{(7)}\right) \oplus \tilde{A}_{\texttt{SB}}^{(8)} \oplus \texttt{MC}^{-1}(H_A) \oplus \tilde{A}_{\texttt{SB}}^{(3)} \oplus \tilde{A}_{\texttt{SB}}^{(5)} \oplus \tilde{A}_{\texttt{SB}}^{(7)} = \texttt{MC}^{-1}\left(\tilde{A}_{\texttt{MC}}^{(8)}\right), \qquad (9)$$

with $\tilde{A}_{\texttt{SB}}^{(1)} = \tilde{A}_{\texttt{SB}}^{(10)}$ by assigning the same assumption to Sasaki's attack [47], i.e., $k_0 = k_{11} \oplus H_A$ and $k_1 = k_{10} \oplus H_B$. From Eq. (9), 2 bytes degree of match indexed by $[6, 7]$ are derived, i.e. DoM = 2. The 12-round MitM attack is given in Algorithm 1. The time complexity to precompute U is $2^{8\cdot\lambda_{\mathcal{B}}} = 2^{112}$. The memory to store U is $2^{8\cdot(\lambda_{\mathcal{B}}-8)} = 2^{48}$. The final time complexity is

$$2^{64+48} + 2^{8\times(16-\min\{14-12,\ 2,\ 2\})} \approx 2^{113}.$$

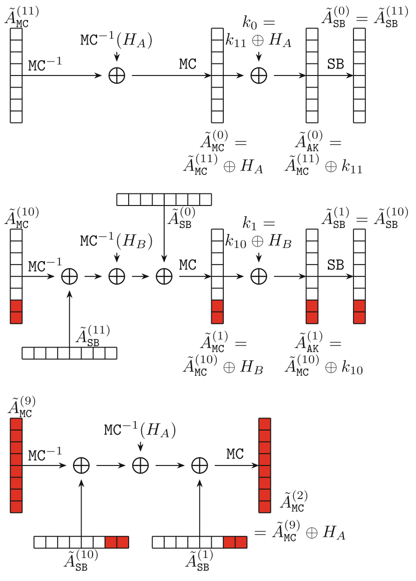
6 Meet-in-the-Middle Attack on Reduced `Simpira V2`

For `Simpira v2` [29] with branch number $b > 2$, the designers suggested the permutation-based hashing based on Davies-Meyer (DM) construction: $\pi(x) \oplus x$, where π is `Simpira v2` permutation. For the common size of digest, i.e., 256 bits, the output of `Simpira v2` has to be truncated. For a fair comparison with Schrottenloher and Stevens' attacks [51], we follow the same way of truncation for `Simpira v2`. We introduce the first 7-round attack on `Simpira-2` and 11-round attack on `Simpira-4`. To fill a gap left by Schrottenloher and Stevens [51], we introduce the first attack on reduced `Simpira-6` in Supplementary Material C in our full version paper [32]. We also give an experiment based on a new 7-round MitM characteristic of `Simpira-2` in Supplementary Material F in [32].



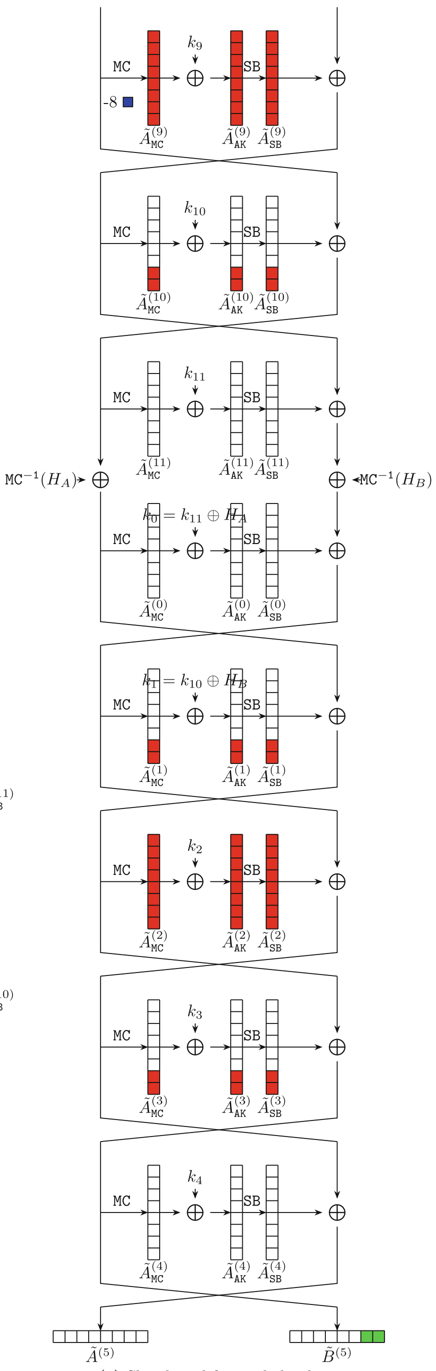
$\tilde{A}^{(5)}$
$\tilde{B}^{(5)}$
DoM = 2
MC
k_5
SB
-2
$\tilde{A}^{(5)}_{\mathtt{MC}}$ $\tilde{A}^{(5)}_{\mathtt{AK}}$ $\tilde{A}^{(5)}_{\mathtt{SB}}$
k_6
$\tilde{A}^{(6)}_{\mathtt{MC}}$ $\tilde{A}^{(6)}_{\mathtt{AK}}$ $\tilde{A}^{(6)}_{\mathtt{SB}}$
k_7
+8
+0
$\tilde{A}^{(7)}_{\mathtt{MC}}$ $\tilde{A}^{(7)}_{\mathtt{AK}}$ $\tilde{A}^{(7)}_{\mathtt{SB}}$
k_8
+6
+2
$\tilde{A}^{(8)}_{\mathtt{MC}}$ $\tilde{A}^{(8)}_{\mathtt{AK}}$ $\tilde{A}^{(8)}_{\mathtt{SB}}$

(a) Initial states and backward chunk



$\tilde{A}^{(11)}_{\mathtt{MC}}$
$\mathtt{MC}^{-1}(H_A)$
$\mathtt{MC}^{-1}$
MC
$k_0 = k_{11} \oplus H_A$
$\tilde{A}^{(0)}_{\mathtt{SB}} = \tilde{A}^{(11)}_{\mathtt{SB}}$
SB
$\tilde{A}^{(0)}_{\mathtt{MC}} = \tilde{A}^{(11)}_{\mathtt{MC}} \oplus H_A$
$\tilde{A}^{(0)}_{\mathtt{AK}} = \tilde{A}^{(11)}_{\mathtt{MC}} \oplus k_{11}$
$\tilde{A}^{(10)}_{\mathtt{MC}}$
$\tilde{A}^{(0)}_{\mathtt{SB}}$
$\mathtt{MC}^{-1}(H_B)$
$k_1 = k_{10} \oplus H_B$
$\tilde{A}^{(1)}_{\mathtt{SB}} = \tilde{A}^{(10)}_{\mathtt{SB}}$
$\tilde{A}^{(11)}_{\mathtt{SB}}$
$\tilde{A}^{(1)}_{\mathtt{MC}} = \tilde{A}^{(10)}_{\mathtt{MC}} \oplus H_B$
$\tilde{A}^{(1)}_{\mathtt{AK}} = \tilde{A}^{(10)}_{\mathtt{MC}} \oplus k_{10}$
$\tilde{A}^{(9)}_{\mathtt{MC}}$
$\mathtt{MC}^{-1}(H_A)$
$\tilde{A}^{(2)}_{\mathtt{MC}}$
$= \tilde{A}^{(9)}_{\mathtt{MC}} \oplus H_A$
$\tilde{A}^{(10)}_{\mathtt{SB}}$
$\tilde{A}^{(1)}_{\mathtt{SB}}$

(b) Details of shrink over 6 rounds in (c)



MC
k_9
SB
-8
$\tilde{A}^{(9)}_{\mathtt{MC}}$ $\tilde{A}^{(9)}_{\mathtt{AK}}$ $\tilde{A}^{(9)}_{\mathtt{SB}}$
k_{10}
$\tilde{A}^{(10)}_{\mathtt{MC}}$ $\tilde{A}^{(10)}_{\mathtt{AK}}$ $\tilde{A}^{(10)}_{\mathtt{SB}}$
k_{11}
$\tilde{A}^{(11)}_{\mathtt{MC}}$ $\tilde{A}^{(11)}_{\mathtt{AK}}$ $\tilde{A}^{(11)}_{\mathtt{SB}}$
$\mathtt{MC}^{-1}(H_A)$
$\mathtt{MC}^{-1}(H_B)$
$k_0 = k_{11} \oplus H_A$
$\tilde{A}^{(0)}_{\mathtt{MC}}$ $\tilde{A}^{(0)}_{\mathtt{AK}}$ $\tilde{A}^{(0)}_{\mathtt{SB}}$
$k_1 = k_{10} \oplus H_B$
$\tilde{A}^{(1)}_{\mathtt{MC}}$ $\tilde{A}^{(1)}_{\mathtt{AK}}$ $\tilde{A}^{(1)}_{\mathtt{SB}}$
k_2
$\tilde{A}^{(2)}_{\mathtt{MC}}$ $\tilde{A}^{(2)}_{\mathtt{AK}}$ $\tilde{A}^{(2)}_{\mathtt{SB}}$
k_3
$\tilde{A}^{(3)}_{\mathtt{MC}}$ $\tilde{A}^{(3)}_{\mathtt{AK}}$ $\tilde{A}^{(3)}_{\mathtt{SB}}$
k_4
$\tilde{A}^{(4)}_{\mathtt{MC}}$ $\tilde{A}^{(4)}_{\mathtt{AK}}$ $\tilde{A}^{(4)}_{\mathtt{SB}}$
$\tilde{A}^{(5)}$
$\tilde{B}^{(5)}$

(c) Shrink and forward chunk

Fig. 19. MitM attack on 12-round `Feistel-SP`

Algorithm 1: Preimage Attack on 12-round `Feistel-SP`

```
1  Set constraints on key schedule $k_0 = k_{11} \oplus H_A$ and $k_1 = k_{10} \oplus H_B$
2  for $g_b \in \mathbb{F}_2^{64}$        /* $\texttt{MC}(\tilde{A}^{(8)}_{\texttt{SB}}[0\text{-}5]\|0\|0) \oplus \tilde{A}^{(7)}_{\texttt{MC}} = g_b$ */
3  do
4      $U \leftarrow [\,]$
5      for $v_{\mathcal{B}} \in \mathbb{F}_2^{6\times 8}$ in $\tilde{A}^{(8)}_{\texttt{SB}}[0\text{-}5]$ do
6          $\tilde{A}^{(7)}_{\texttt{MC}} \leftarrow \texttt{MC}(v_{\mathcal{B}}\|0\|0) \oplus g_b$
7          Compute through AK and SB to get the values of 🟦 cells in $\tilde{A}^{(7)}_{\texttt{SB}}$
8          $c_0\|c_1 \leftarrow \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SB}})[6,7]$   /* $\tilde{A}^{(6)}_{\texttt{MC}} = \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SB}}) \oplus \tilde{A}^{(8)}_{\texttt{MC}}$ */
9          Compute 🟦 cells in $\tilde{A}^{(6)}_{\texttt{SB}}$
10         $c_2\|c_3 \leftarrow \texttt{MC}(\tilde{A}^{(6)}_{\texttt{SB}}[0\text{-}5]\|0\|0)[6,7] \oplus \tilde{A}^{(7)}_{\texttt{MC}}[6,7]$
11         $\mathfrak{c}_{\mathcal{B}} \leftarrow c_0\|c_1\|c_2\|c_3$
12         $U[\mathfrak{c}_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$   /* There are $2^{16}$ elements in $U[\mathfrak{c}_{\mathcal{B}}]$ given $\mathfrak{c}_{\mathcal{B}}$ */
13     end
14     for $\mathfrak{c}_{\mathcal{B}} \in \mathbb{F}_2^{4\times 8}$ do
15         $L \leftarrow [\,]$
16         for $v_{\mathcal{B}} \in U[\mathfrak{c}_{\mathcal{B}}]$ do
17             Compute backward to the 🟦 cells in $\tilde{A}^{(6)}_{\texttt{MC}}$. According to Fig. 19, derive 2 bytes
               $End_{\mathcal{B}}$ for matching by
18                 $$End_{\mathcal{B}} \leftarrow \texttt{MC}^{-1}\left(\tilde{A}^{(6)}_{\texttt{MC}}[0-5]\|0\|0\right)[6,7]$$
               $L[End_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$
19         end
20         for $2^{8\lambda_{\mathcal{R}}}$ values $v_{\mathcal{R}}$ of the 🟥 bytes in $\tilde{A}^{(8)}_{\texttt{MC}}$, $\lambda_{\mathcal{R}} = 2$ do
21             Compute backward to the 🟥 cells in $\tilde{A}^{(5)}_{\texttt{SB}}$
22             Due to the predefined constraints on key schedule, there always be
               $\tilde{A}^{(1)}_{\texttt{MC}} = \tilde{A}^{(10)}_{\texttt{MC}} \oplus H_B$ and $\tilde{A}^{(2)}_{\texttt{MC}} = \tilde{A}^{(9)}_{\texttt{MC}} \oplus H_A$
23             With $\tilde{A}^{(1)}_{\texttt{MC}}$ and $\tilde{A}^{(2)}_{\texttt{MC}}$, compute forward to the 🟥 cells in $\tilde{A}^{(3)}_{\texttt{SB}}$
24             From $\tilde{A}^{(2)}_{\texttt{MC}}$, $\tilde{A}^{(3)}_{\texttt{SB}}$ and $\tilde{A}^{(5)}_{\texttt{SB}}[6,7]$, derive 2 bytes $End_{\mathcal{R}}$ for matching by
                 $$End_{\mathcal{R}} \leftarrow \texttt{MC}^{-1}\left(\tilde{A}^{(2)}_{\texttt{MC}}\right)[6,7] \oplus \tilde{A}^{(3)}_{\texttt{SB}}[6,7] \oplus \tilde{A}^{(5)}_{\texttt{SB}}[6,7] \oplus \texttt{MC}^{-1}\left(0\|0\|0\|0\|0\|\tilde{A}^{(6)}_{\texttt{MC}}[6,7]\right)[6,7]$$
25             for $v_{\mathcal{B}} \in L[End_{\mathcal{R}}]$ do
26                 Reconstruct the (candidate) message $X$
27                 if $X$ is a preimage then
28                     Output $X$ and stop
29                 end
30             end
31         end
32     end
33 end
```

6.1 Meet-in-the-Middle Attack on 7-Round `Simpira-2`

As shown in Fig. 20, we give a 7-round preimage attack on `Simpira`-2. The starting states are $\tilde{A}^{(3)}_{\texttt{MC1}}$ and $\tilde{A}^{(4)}_{\texttt{MC1}}$, where $\lambda_{\mathcal{R}} = 4$ and $\lambda_{\mathcal{B}} = 28$. Along the forward and backward computation paths, there are 0 constraints on 🟥 and 20 constraints on 🟦, i.e. $l_{\mathcal{R}} = 0$ and $l_{\mathcal{B}} = 20$ as shown in Fig. 21. Then, we have $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 4$ and $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 8$. The matching points are $\tilde{A}^{(2)}$ and $\tilde{B}^{(2)}$ and the full-round matching equation is (10). Due to $\texttt{MC}^{-1}(\tilde{A}^{(3)}_{\texttt{MC1}})$ appears in both directions, $\mathtt{MC}^{-1}(\tilde{A}^{(3)}_{\mathtt{MC1}})$ makes no contribution to the match and can be canceled without influence as shown in Fig. 22.

$$\tilde{A}^{(2)}_{\mathtt{SR2}} \oplus \tilde{A}^{(4)}_{\mathtt{SR2}} \oplus \tilde{A}^{(6)}_{\mathtt{SR2}} \oplus \mathtt{MC}^{-1}(H_B) = \tilde{A}^{(0)}_{\mathtt{SR2}}. \qquad (10)$$

Then, 4 bytes for matching in the Eq. (10) indexed by $[3, 6, 9, 12]$ are only determined by the 🟥 bytes, i.e. $\mathrm{M}_{\mathcal{R}} = 4$. The detailed attack procedure is shown in Algorithm 2. The time to construct U is $2^{8\cdot\lambda_{\mathcal{B}}} = 2^{224}$. The memory cost to store U is $2^{8\cdot(\lambda_{\mathcal{B}}-16)} \approx 2^{96}$. According to Eq. (2), the overall time complexity to mount a MitM attack is

$$2^{224} + 2^{8\times(32-\min\{8,4,4\})} \approx 2^{225}.$$

The memory cost is about 2^{96} to store hash table U.

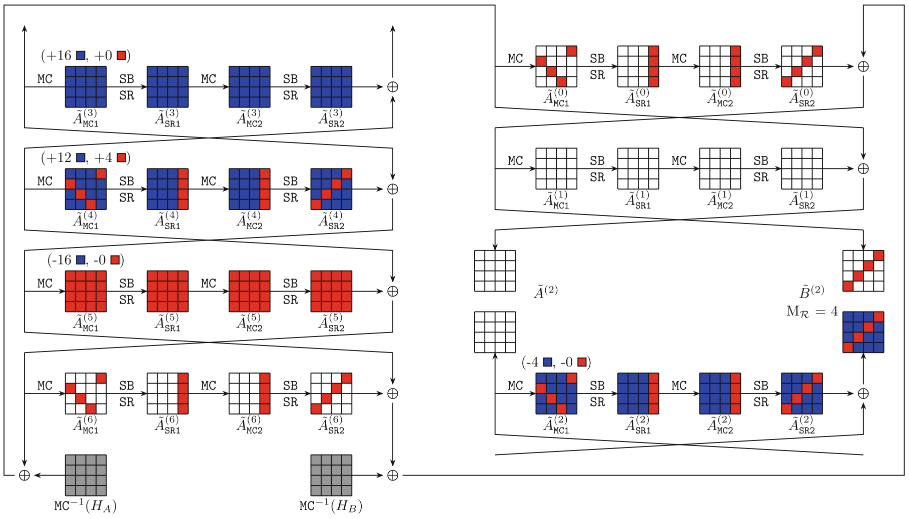

(+16 🟦, +0 🟥)
(+12 🟦, +4 🟥)
(-16 🟦, -0 🟥)
(-4 🟦, -0 🟥)
MC
SB
SR
$\tilde{A}^{(3)}_{\mathtt{MC1}}$
$\tilde{A}^{(3)}_{\mathtt{SR1}}$
$\tilde{A}^{(3)}_{\mathtt{MC2}}$
$\tilde{A}^{(3)}_{\mathtt{SR2}}$
$\tilde{A}^{(4)}_{\mathtt{MC1}}$
$\tilde{A}^{(4)}_{\mathtt{SR1}}$
$\tilde{A}^{(4)}_{\mathtt{MC2}}$
$\tilde{A}^{(4)}_{\mathtt{SR2}}$
$\tilde{A}^{(5)}_{\mathtt{MC1}}$
$\tilde{A}^{(5)}_{\mathtt{SR1}}$
$\tilde{A}^{(5)}_{\mathtt{MC2}}$
$\tilde{A}^{(5)}_{\mathtt{SR2}}$
$\tilde{A}^{(6)}_{\mathtt{MC1}}$
$\tilde{A}^{(6)}_{\mathtt{SR1}}$
$\tilde{A}^{(6)}_{\mathtt{MC2}}$
$\tilde{A}^{(6)}_{\mathtt{SR2}}$
$\tilde{A}^{(0)}_{\mathtt{MC1}}$
$\tilde{A}^{(0)}_{\mathtt{SR1}}$
$\tilde{A}^{(0)}_{\mathtt{MC2}}$
$\tilde{A}^{(0)}_{\mathtt{SR2}}$
$\tilde{A}^{(1)}_{\mathtt{MC1}}$
$\tilde{A}^{(1)}_{\mathtt{SR1}}$
$\tilde{A}^{(1)}_{\mathtt{MC2}}$
$\tilde{A}^{(1)}_{\mathtt{SR2}}$
$\tilde{A}^{(2)}$
$\tilde{B}^{(2)}$
$\mathrm{M}_{\mathcal{R}} = 4$
$\tilde{A}^{(2)}_{\mathtt{MC1}}$
$\tilde{A}^{(2)}_{\mathtt{SR1}}$
$\tilde{A}^{(2)}_{\mathtt{MC2}}$
$\tilde{A}^{(2)}_{\mathtt{SR2}}$
$\mathtt{MC}^{-1}(H_A)$
$\mathtt{MC}^{-1}(H_B)$

Fig. 20. MitM attack on 7-round `Simpira-2`

6.2 Meet-in-the-Middle Attack on 11-Round `Simpira-4`

Figure 23 is an 11-round MitM characteristic of `Simpira`-4. Figure 28 given in Supplementary Material B in our full version paper [32] is an alternative representation of the MitM characteristic with `MC-then-XOR-Rule` in superposition states. The starting states are $\tilde{A}^{(7)}_{\mathtt{MC1}}$, $\tilde{C}^{(6)}_{\mathtt{MC1}}$, $\tilde{A}^{(6)}_{\mathtt{MC1}}$, and $\tilde{C}^{(7)}_{\mathtt{MC1}}$. The initial DoFs for 🟥 and 🟦 are $\lambda_{\mathcal{R}} = 24$ and $\lambda_{\mathcal{B}} = 4$, respectively. Along the forward and backward

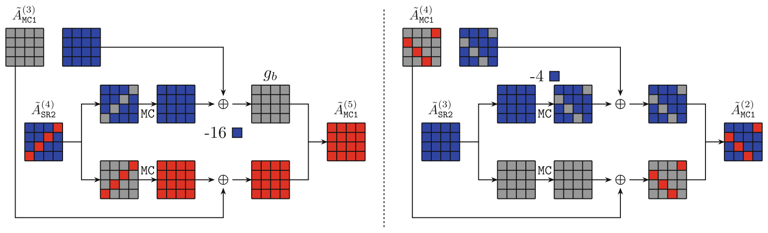

$\tilde{A}^{(3)}_{\texttt{MC1}}$
$\tilde{A}^{(4)}_{\texttt{SR2}}$
MC
MC
g_b
-16 🟦
$\tilde{A}^{(5)}_{\texttt{MC1}}$
$\tilde{A}^{(4)}_{\texttt{MC1}}$
$\tilde{A}^{(3)}_{\texttt{SR2}}$
-4 🟦
MC
MC
$\tilde{A}^{(2)}_{\texttt{MC1}}$

Fig. 21. The `MC-then-XOR-Rule` of `Simpira-2` in superposition framework

Algorithm 2: Preimage Attack on 7-round `Simpira-2`

```
1  for $g_b \in \mathbb{F}_2^{128}$ do
2      $U \leftarrow [\,]$
3      for $v_\mathcal{B} \in \mathbb{F}_2^{12\times 8}$ in $\tilde{A}^{(4)}_{\texttt{MC1}}[0, 2\text{-}5, 7\text{-}10, 13\text{-}15]$ do
4          Compute the 🟦 cells in $\tilde{A}^{(4)}_{\texttt{SR2}}$ from $\tilde{A}^{(4)}_{\texttt{MC1}}$
5          Let $\tilde{A}^{(4)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \in [3, 6, 9, 12]$
6          Compute $\tilde{A}^{(3)}_{\texttt{MC1}}$ by $\texttt{MC}(\tilde{A}^{(4)}_{\texttt{SR2}}) \oplus g_b$    /* Left part of Fig. 21 */
7          Compute $\tilde{A}^{(3)}_{\texttt{SR2}}$ from $\tilde{A}^{(3)}_{\texttt{MC1}}$
8          $c_0 \| c_1 \| c_2 \| c_3 \leftarrow \texttt{MC}(\tilde{A}^{(3)}_{\texttt{SR2}})[1, 6, 11, 12]$    /* Right part of Fig. 21 */
9          $\mathfrak{c}_\mathcal{B} \leftarrow c_0 \| c_1 \| c_2 \| c_3$
10         $U[\mathfrak{c}_\mathcal{B}] \leftarrow v_\mathcal{B}$    /* There are $2^{8\times 8}$ elements $U[\mathfrak{c}_\mathcal{B}]$ given $\mathfrak{c}_\mathcal{B}$ */
11     end
12     for $\mathfrak{c}_\mathcal{B} \in \mathbb{F}_2^{4\times 8}$ do
13         Set $\mathcal{S}$ to be an empty set to store the compatible values of 🟥
14         for $2^{8\lambda_\mathcal{R}}$ values $v_\mathcal{R}$ of the 🟥 bytes in $\tilde{A}^{(4)}_{\texttt{MC1}}$, $\lambda_\mathcal{R} = 4$ do
15             Compute to the 🟥 cells in $\tilde{A}^{(0)}_{\texttt{SR2}}$, $\tilde{A}^{(2)}_{\texttt{SR2}}$, $\tilde{A}^{(4)}_{\texttt{SR2}}$ and $\tilde{A}^{(6)}_{\texttt{SR2}}$
16             As shown in Fig. 22, $\mathrm{M}_\mathcal{R}$=4 bytes equations are derived by
                   $$\left(\tilde{A}^{(2)}_{\texttt{SR2}} \oplus \tilde{A}^{(4)}_{\texttt{SR2}} \oplus \tilde{A}^{(6)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_\mathcal{B})\right)[3, 6, 9, 12] = \tilde{A}^{(0)}_{\texttt{SR2}}[3, 6, 9, 12]$$
               Put the solution into $\mathcal{S}$
17         end
18         for $v_\mathcal{B} \in U[\mathfrak{c}_\mathcal{B}]$ do
19             Compute the 🟦 cells in $\tilde{A}^{(3)}_{\texttt{MC1}}$ as Line 6 for $v_\mathcal{R} \in \mathcal{S}$ do
20                 Reconstruct the (candidate) message $X$ if $X$ is a preimage then
21                     Output $X$ and stop
22                 end
23             end
24         end
25     end
26 end
```

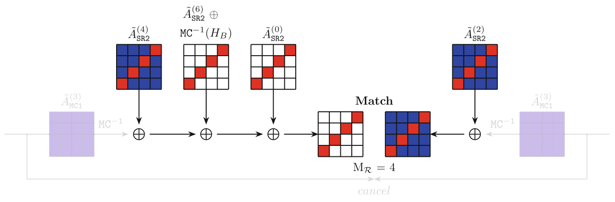

$\tilde{A}^{(6)}_{\texttt{SR2}} \oplus$
$\texttt{MC}^{-1}(H_B)$
$\tilde{A}^{(4)}_{\texttt{SR2}}$
$\tilde{A}^{(0)}_{\texttt{SR2}}$
$\tilde{A}^{(2)}_{\texttt{SR2}}$
$\tilde{A}^{(3)}_{\texttt{MC1}}$
$\texttt{MC}^{-1}$
Match
$\tilde{A}^{(3)}_{\texttt{MC1}}$
$\texttt{MC}^{-1}$
$\mathrm{M}_{\mathcal{R}} = 4$
cancel

Fig. 22. Full-round matching in 7-round `Simpira`-2

computation paths, there are a total of 20 constraints on 🟥 and 0 constant constraints on 🟦, i.e., $l_{\mathcal{R}} = 20$ and $l_{\mathcal{B}} = 0$. Hence, we get $\mathrm{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 4$ and $\mathrm{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 4$. The matching points are $(\tilde{A}^{(2)}, \tilde{B}^{(2)}, \tilde{C}^{(2)}, \tilde{D}^{(2)})$. The full-matching equation is (11), where $\texttt{MC}^{-1}(\tilde{C}^{(7)}_{\texttt{MC1}})$ appears in both directions and can be cancelled.

$$\tilde{A}^{(8)}_{\texttt{SR2}} \oplus \tilde{C}^{(10)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B) \oplus \tilde{A}^{(0)}_{\texttt{SR2}} = \tilde{C}^{(6)}_{\texttt{SR2}} \oplus \tilde{A}^{(4)}_{\texttt{SR2}} \oplus \tilde{C}^{(2)}_{\texttt{SR2}}. \tag{11}$$

Then, 4 bytes in Eq. (11) indexed by $[0, 7, 10, 13]$ are derived as the degree of match, i.e. $\mathrm{DoM} = 4$. The 11-round attack is given in Algorithm 3. The time to construct V is $2^{8 \cdot \lambda_{\mathcal{R}}} = 2^{192}$ and memory is $2^{8 \cdot (\lambda_{\mathcal{R}} - 4)} = 2^{160}$. We need to traverse 2^{32} values of the ■ in $\tilde{A}^{(6)}_{\texttt{MC1}}$, $\tilde{C}^{(6)}_{\texttt{MC1}}$ and $\tilde{C}^{(7)}_{\texttt{MC1}}$. Hence, the total time complexity can be computed by $2^{32} \times 2^{192} + 2^{8 \times (32 - \min\{24-20, 4, 4\})} \approx 2^{225}$. The overall memory is 2^{160} to store V.

7 Meet-in-the-Middle Attack on 17-Round `Lesamnta-LW`

We also apply our automated model to `Lesamnta-LW` [31]. Since the `Lesamnta-LW` does not have the feed-forward mechanism, there are only two forward chunks. We find a 17-round MitM characteristic for `Lesamnta-LW` without linear transformation, which is shown in Fig. 24. The initial DoFs for 🟦 and 🟥 are $\lambda_{\mathcal{B}} = 4$, $\lambda_{\mathcal{R}} = 4$, respectively. Without consuming DoF of 🟥/🟦 in the computation from round 0 to round 17, there is $\mathrm{DoF}_{\mathcal{R}} = \mathrm{DoF}_{\mathcal{B}} = 4$. The matching happens between $D^{(17)}$ and the targeted hash value, where $\mathrm{DoM} = 8$. The attack procedure is given in Algorithm 4, where two message blocks (m_1, m_2) are needed as shown in Fig. 11. In this attack, we only use the first column of $D^{(17)}$ for matching. At first, we randomly fix the first 32-bit in $D^{(17)}$ as constant. Then, in one MitM episode, we can get $2^{32+32-32} = 2^{32}$ (m_1, m_2) satisfying the 32-bit partial target. When we find $2^{(256-32)/2} = 2^{112}$ different (m_1, m_2, h) with the same fixed 32-bit partial target, we can find a collision on the remaining $(256 - 32)$ bits of the full 256-bit target. The time complexity is $2^{16+64} \cdot (2^{32} + 2^{32} + 2^{32}) \approx 2^{113.58}$. The memory complexity is 2^{112}. The same time and memory cost can also be obtained when considering the linear transformation of collision.

Besides, we also found a 20-round MitM collision attack on `Lesamnta-LW` when targeting the linear transformation of collision, the overall time complexity is 2^{124} which is better than the generic birthday bound 2^{128}. However, it's not

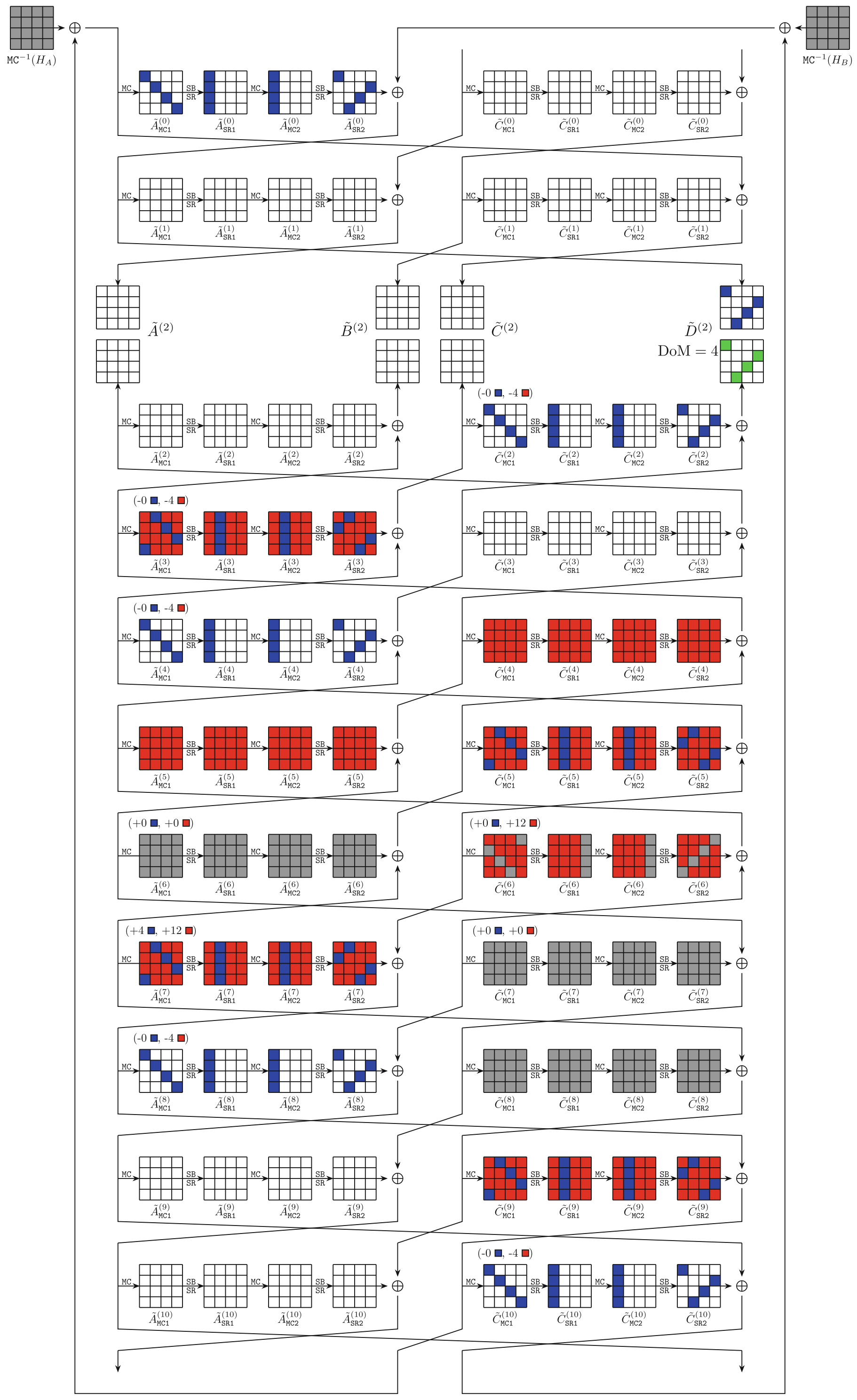


Fig. 23. MitM attack on 11-round Simpira-4

Algorithm 3: Preimage Attack on 11-round Simpira-4

```
1  for $\tilde{A}^{(6)}_{\texttt{MC1}}\|\tilde{C}^{(6)}_{\texttt{MC1}}[1,6,11,12]\|\tilde{C}^{(7)}_{\texttt{MC1}} \in \mathbb{G}$    /* $|\mathbb{G}| = 2^{32}$ */
2  do
3      for $g_r \in \mathbb{F}_2^{32}$ do
4          $V \leftarrow [\,]$
5          for $v_{\mathcal{R}} \in \mathbb{F}_2^{20\times 8}$ in $\tilde{A}^{(7)}_{\texttt{MC1}}[0\text{-}2, 5\text{-}8, 10\text{-}13, 15]$ and $\tilde{C}^{(6)}_{\texttt{MC1}}[2\text{-}4, 7\text{-}9, 13, 14]$ do
6              Compute the 🟥 cells in $\tilde{A}^{(7)}_{\texttt{SR2}}$ from $\tilde{A}^{(7)}_{\texttt{MC1}}$
7              Let $\tilde{A}^{(7)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \in [1,4,11,14]$
8              $\tilde{C}^{(6)}_{\texttt{MC1}}[0,5,10,15] \leftarrow \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SR2}})[0,5,10,15] \oplus g_r$
9              From $\tilde{A}^{(7)}_{\texttt{MC1}}$ and $\tilde{A}^{(6)}_{\texttt{MC1}}$, compute the 🟥 cells in $\tilde{C}^{(5)}_{\texttt{SR2}}$
10             Let $\tilde{C}^{(5)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \in [1,4,11,14]$
11             $c_0\|c_1\|c_2\|c_3 \leftarrow \left(\texttt{MC}(\tilde{C}^{(5)}_{\texttt{SR2}}) \oplus \tilde{C}^{(6)}_{\texttt{MC1}}\right)[0,5,10,15]$
12             From the known values, compute the 🟥 cells in $\tilde{C}^{(9)}_{\texttt{SR2}}$, $\tilde{C}^{(4)}_{\texttt{SR2}}$, $\tilde{A}^{(3)}_{\texttt{SR2}}$, and
               let the remaining 🟦 cells be 0
13             $c_4\|c_5\|c_6\|c_7 \leftarrow \texttt{MC}\left(\tilde{C}^{(9)}_{\texttt{SR2}}\right)[0,5,10,15]$
14             $c_8\|c_9\|c_{10}\|c_{11} \leftarrow \texttt{MC}\left(\tilde{C}^{(4)}_{\texttt{SR2}}\right)[3,4,9,14]$
15             $c_{12}\|c_{13}\|c_{14}\|c_{15} \leftarrow \texttt{MC}\left(\tilde{A}^{(3)}_{\texttt{SR2}}\right)[0,5,10,15]$
16             $\mathfrak{c}_{\mathcal{R}} \leftarrow c_0\|c_1\|\cdots\|c_{14}\|c_{15}$
17             $V[\mathfrak{c}_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
18         end
19         for $\mathfrak{c}_{\mathcal{R}} \leftarrow \mathbb{F}_2^{16\times 8}$ do
20             $L \leftarrow [\,]$
21             for $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ do
22                 Compute the 🟥 cells in $\tilde{C}^{(6)}_{\texttt{SR2}}$. According to Fig. 18(c), derive 4
                   bytes $End_{\mathcal{R}}$ for matching by
                       $End_{\mathcal{R}} \leftarrow \left(\tilde{C}^{(6)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B)\right)[0,7,10,13]$
                   $L[End_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
23             end
24             for $2^{8\lambda_{\mathcal{B}}}$ values $v_{\mathcal{B}}$ of the 🟦 bytes in $\tilde{A}^{(7)}_{\texttt{MC1}}$, $\lambda_{\mathcal{B}} = 4$ do
25                 Compute backward to the 🟦 cells in $\tilde{A}^{(4)}_{\texttt{SR2}}$ and $\tilde{C}^{(2)}_{\texttt{SR2}}$
26                 Compute forward to the 🟦 cells in $\tilde{A}^{(8)}_{\texttt{SR2}}$, $\tilde{C}^{(10)}_{\texttt{SR2}}$ and $\tilde{A}^{(0)}_{\texttt{SR2}}$
27                 As in Fig. 18(c), 4 bytes $End_{\mathcal{B}}$ for matching are derived by
                       $End_{\mathcal{B}} \leftarrow \left(\tilde{A}^{(8)}_{\texttt{SR2}} \oplus \tilde{C}^{(10)}_{\texttt{SR2}} \oplus \tilde{A}^{(0)}_{\texttt{SR2}} \oplus \tilde{C}^{(2)}_{\texttt{SR2}} \oplus \tilde{A}^{(4)}_{\texttt{SR2}}\right)[0,7,10,13]$
                   for $v_{\mathcal{R}} \in L[End_{\mathcal{B}}]$ do
28                     Reconstruct the (candidate) message $X$
29                     if $X$ is a preimage then
30                         Output $X$ and stop
31                     end
32                 end
33             end
34         end
35     end
36 end
```

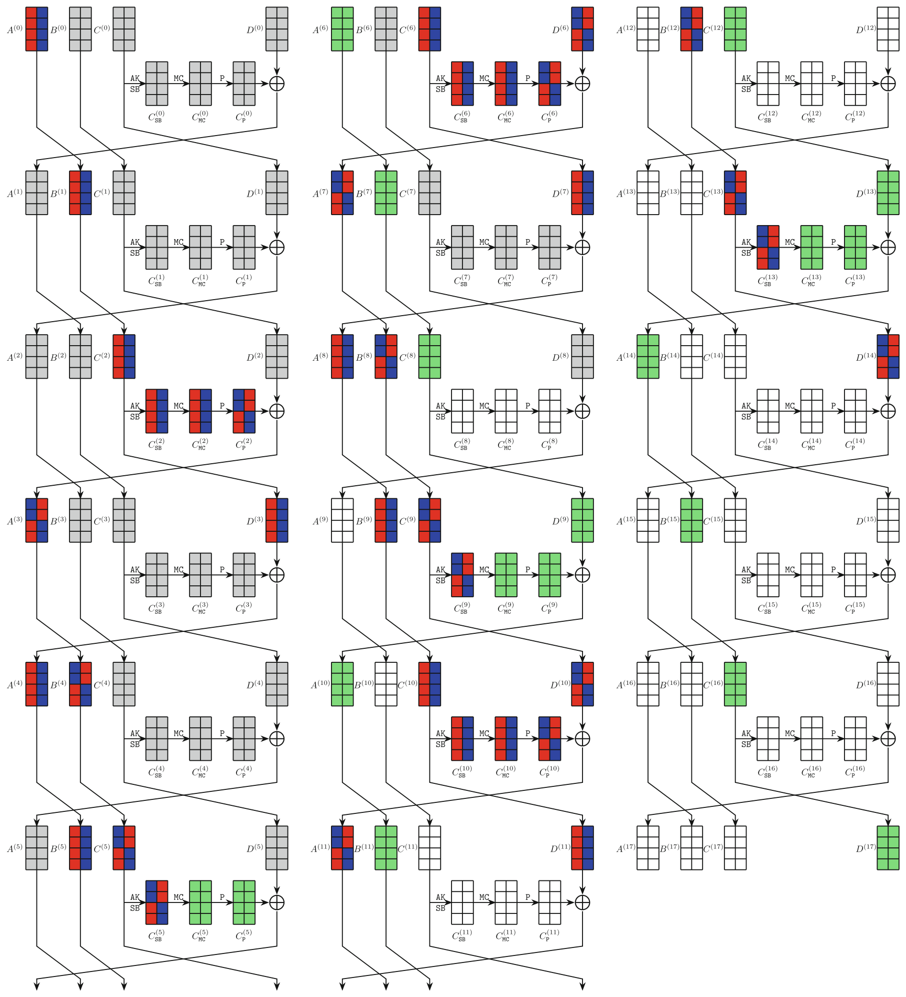


Fig. 24. MitM attack on 17-round Lesamnta-LW

better than the designers' security claim against collision attack, which is 2^{120}. We still put the 20-round MitM characteristic in Supplementary Material D in our full version paper [32] to clearly specify the superiority of our new model.

8 Meet-in-the-Middle Attack on Reduced `Areion`

Based on DM hashing mode, Isobe *et al.* [35] built hash functions `Areion256-DM` and `Areion512-DM`. This section studies the MitM preimage attacks on these two ciphers. However, in the left branch of `Areion`, there exist additional operations, such as $\mathtt{SR} \circ \mathtt{SB}$ for `Areion-256`. If we just transform it like `Simpira`, the left

Algorithm 4: Collision Attack on 17-round `Lesamnta-LW`

```
Fix the first 32 bits of $D^{(17)}$, i.e. 4 bytes of the first column
for $2^{16}$ possible values of $m_1$ do
    for $2^{64}$ possible values of $B^{(0)}$ in $m_2$    /* The 128-bit message block is
        placed in $A^{(0)}$ and $B^{(0)}$                                   */
    do
        for $2^{8\lambda_{\mathcal{R}}}$ possible values of the 🟥 bytes in $A^{(0)}$, $\lambda_{\mathcal{R}} = 4$ do
            Set the 🟦 bytes in $A^{(0)}$ to 0
            Compute forward to the 🟩 bytes in $D^{(17)}$, and store in $L_1$ indexed
            by the first 32 bits of $D^{(17)}$
        end
        for $2^{8\lambda_{\mathcal{B}}}$ possible values of the 🟦 bytes in $A^{(0)}$, $\lambda_{\mathcal{B}} = 4$ do
            Set the 🟥 bytes in $A^{(0)}$ to 0
            Compute forward to the 🟩 bytes in $D^{(17)}$, and store in $L_2$ indexed
            by the first 32 bits of $D^{(17)}$
        end
        for values matched between $L_1$ and $L_2$ do
            Compute the 256-bit target $h = (A^{(17)}, B^{(17)}, C^{(17)}, D^{(17)})$ from the
            matched 🟥 and 🟦 bytes and store the $(m_1, m_2, h)$ in $L$ indexed by $h$
            if the size of $L$ is $2^{(256-32)/2} = 2^{112}$ then
                Check $L$ and return $(m_1, m_2)$ and $(m_1', m_2')$ with the same $h$
            end
        end
    end
end
```

branch still preserved additional operations so that the full-round matching (only `XOR`ed states) cannot be applied. Therefore, we use the generalized matching strategy proposed in Sect. 3 to detect matching equations at two consecutive rounds, together with the superposition MitM technique.

8.1 Meet-in-the-Middle Attack on 5-Round `Areion-256`

By applying the automatic MitM attack, we find a 5-round preimage attack on `Areion-256` as shown in Fig. 25. The starting states are $A^{(3)}$ and $B^{(3)}$. The initial DoFs for 🟥 and 🟦 are $\lambda_{\mathcal{R}} = 8$ and $\lambda_{\mathcal{B}} = 23$, respectively. The consuming degrees for backward and forward are 0 and 15, i.e. $l_{\mathcal{R}} = 0$ and $l_{\mathcal{B}} = 15$. Then we have $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 8$ and $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 8$. The matching happens between $A^{(1,\alpha)}_{\texttt{SR2}}$ and $B^{(1)} \oplus A^{(2)}$, by combining `MixColumn` and `XOR` operations as Fig. 14, where DoM = 6. According to Sect. 3, we get additional $\text{M}_{\mathcal{R}} = 2$ bytes from the last column of $B^{(1)} \oplus A^{(2)}$, which are determined only by 🟥 cells and can also be used in matching phase.

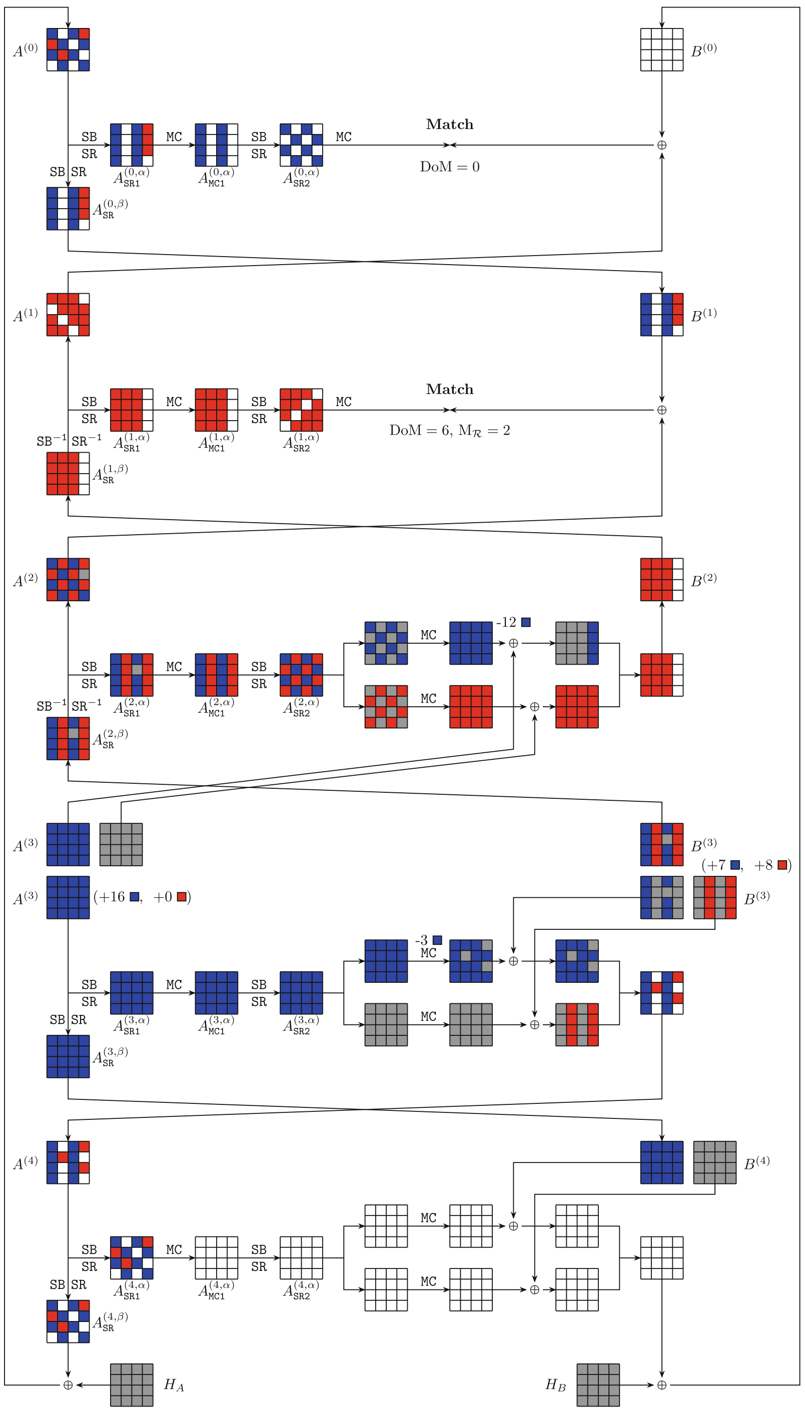
The new 5-round attack on `Areion-256` is given in Algorithm 7 in Supplementary Material E in our full version paper [32]. The time to construct table U is $2^{8\cdot\lambda_{\mathcal{B}}} = 2^{184}$. Hence, we have the time complexity $2^{8} \cdot 2^{184} + 2^{8\times(32-\min\{23-15,8,8\})} \approx 2^{193}$. The overall memory complexity is 2^{88} to store U.

8.2 Meet-in-the-Middle Attack on 7-Round `Areion-256`

The attack figure and algorithm on 7-round `Areion-256` are given in Fig. 34 and Algorithm 8 in Supplementary Material E in our full version paper [32]. The starting states are $A^{(4)}$ and $B^{(4)}$. The initial DoFs for 🟥 and 🟦 are $\lambda_{\mathcal{R}} = 22$ and $\lambda_{\mathcal{B}} = 4$, respectively. The consumed DoFs of 🟥 and 🟦 are $l_{\mathcal{R}} = 20$ and $l_{\mathcal{B}} = 2$, so there is $\mathrm{DoF}_{\mathcal{R}} = \mathrm{DoF}_{\mathcal{B}} = 2$. The matching happens between $A^{(1,\alpha)}_{\mathtt{SR2}}$ and $B^{(1)} \oplus A^{(2)}$, by combining `MixColumn` and `XOR` operations as Fig. 14, where $\mathrm{DoM} = 2$. The time to construct table V is $2^{8\cdot\lambda_{\mathcal{R}}} = 2^{176}$ and memory is $2^{8\cdot(\lambda_{\mathcal{R}}-14)} = 2^{64}$. The overall time complexity is $2^{48} \cdot 2^{176} + 2^{8\times(32-\min\{22-20,4-2,2\})} \approx 2^{240}$. The memory cost is 2^{64} to store V.

8.3 Meet-in-the-Middle Attack on 11-Round `Areion-512`

The attack figure and algorithm on 11-round `Areion-512` are given in Fig. 35, 36, and Algorithm 9 in Supplementary Material E in our full version paper [32]. The starting states are $A^{(3)}$, $B^{(3)}$, $C^{(3)}$ and $D^{(3)}$. The initial DoFs for 🟥 and 🟦 are $\lambda_{\mathcal{R}} = 30$, $\lambda_{\mathcal{B}} = 2$, respectively. The consuming DoF of backward and forward neutral words are $l_{\mathcal{R}} = 28$ and $l_{\mathcal{B}} = 0$. Then, we have $\mathrm{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 2$ and $\mathrm{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 2$. The matching phase happens between $C^{(9,\beta)}_{\mathtt{SR}}$ and $B^{(10)}$ through `MixColumn`, where $\mathrm{DoM} = 2$. The time complexity to precompute V is $2^{8\cdot\lambda_{\mathcal{R}}} = 2^{240}$. The time complexity is $2^{240} + 2^{8\times(32-\min\{30-28,\ 2,\ 2\})} \approx 2^{241}$. The overall memory complexity is 2^{48} to store V.



$A^{(0)}$
$B^{(0)}$
SB
SR
MC
Match
DoM = 0
$A^{(0,\alpha)}_{\mathtt{SR1}}$
$A^{(0,\alpha)}_{\mathtt{MC1}}$
$A^{(0,\alpha)}_{\mathtt{SR2}}$
$A^{(0,\beta)}_{\mathtt{SR}}$
$A^{(1)}$
$B^{(1)}$
$\mathtt{SB}^{-1}$ $\mathtt{SR}^{-1}$
DoM = 6, $\mathrm{M}_{\mathcal{R}} = 2$
$A^{(1,\alpha)}_{\mathtt{SR1}}$
$A^{(1,\alpha)}_{\mathtt{MC1}}$
$A^{(1,\alpha)}_{\mathtt{SR2}}$
$A^{(1,\beta)}_{\mathtt{SR}}$
$A^{(2)}$
$B^{(2)}$
-12
$A^{(2,\alpha)}_{\mathtt{SR1}}$
$A^{(2,\alpha)}_{\mathtt{MC1}}$
$A^{(2,\alpha)}_{\mathtt{SR2}}$
$A^{(2,\beta)}_{\mathtt{SR}}$
$A^{(3)}$
$B^{(3)}$
(+7, +8)
(+16, +0)
-3
$A^{(3,\alpha)}_{\mathtt{SR1}}$
$A^{(3,\alpha)}_{\mathtt{MC1}}$
$A^{(3,\alpha)}_{\mathtt{SR2}}$
$A^{(3,\beta)}_{\mathtt{SR}}$
$A^{(4)}$
$B^{(4)}$
$A^{(4,\alpha)}_{\mathtt{SR1}}$
$A^{(4,\alpha)}_{\mathtt{MC1}}$
$A^{(4,\alpha)}_{\mathtt{SR2}}$
$A^{(4,\beta)}_{\mathtt{SR}}$
H_A
H_B

Fig. 25. MitM attack on 5-round `Areion-256`

9 Conclusion

In this paper, we build a new Meet-in-the-Middle automatic tool for Feistel networks. In our model, we generalize the traditional direct or indirect partial matching strategies and also Sasaki's multi-round matching strategy. We also find some equivalent transformations of Feistel and GFN to significantly simplify the MILP models. Applying our new models, we obtain improved preimage attacks on `Feistel-SP-MMO`, `Simpira-2/-4-DM`,16 `Areion-256/-512-DM` and the first 11-round attack on `Simpira-6`. Besides, we significantly improve the collision attack on the ISO standard hash `Lesamnta-LW` by 6 rounds.

Acknowledgements. We thank the anonymous reviewers from ASIACRYPT 2023 for their insightful comments. This work is supported by the National Key R&D Program of China (2022YFB2702804, 2018YFA0704701), the Natural Science Foundation of China (62272257, 62302250, 62072270, 62072207), Shandong Key Research and Development Program (2020ZLYS09), the Major Scientific and Technological Innovation Project of Shandong, China (2019JZZY010133), the Major Program of Guangdong Basic and Applied Research (2019B030302008, 2022A1515140090), Key Research Project of Zhejiang Province, China (2023C01025).

References

1. Amzaleg, D., Dinur, I.: Refined cryptanalysis of the GPRS ciphers GEA-1 and GEA-2. IACR Cryptology ePrint Archive, Paper 2022/424 (2022)
2. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Yu., Wang, L.: Preimages for step-reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_34
3. Aoki, K., Sasaki, Yu.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_7
4. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 120–135. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_8
5. Banik, S., Barooti, K., Vaudenay, S., Yan, H.: New attacks on LowMC instances with a single plaintext/ciphertext pair. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13090, pp. 303–331. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_11
6. Bao, Z., et al.: Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 771–804. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_27
7. Bao, Z., Guo, J., Shi, D., Tu, Y.: Superposition meet-in-the-middle attacks: updates on fundamental security of AES-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology, CRYPTO 2022. LNCS, vol. 13507, pp. 64–93. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_3
8. Barreto, P.S.L.M., Rijmen, V.: The WHIRLPOOL hashing function. Submitted to NESSIE (2000)

9. Beierle, C., et al.: Lightweight AEAD and hashing using the sparkle permutation family. IACR Trans. Symmetric Cryptol. **2020**(S1), 208–261 (2020)
10. Beierle, C., et al.: Cryptanalysis of the GPRS encryption algorithms GEA-1 and GEA-2. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 155–183. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_6
11. Benadjila, R., et al.: SHA-3 proposal: ECHO. Submission to NIST (updated), p. 113 (2009)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document. Submission to NIST (Round 2), vol. 3, no. 30, pp. 320–337 (2009)
13. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Proceedings of the ASIACRYPT 2011, pp. 344–371 (2011)
14. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_16
15. Bouillaguet, C., Derbez, P., Fouque, P.-A.: Automatic search of attacks on round-reduced AES and applications. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 169–187. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_10
16. Boura, C., David, N., Derbez, P., Leander, G., Naya-Plasencia, M.: Differential meet-in-the-middle cryptanalysis. IACR Cryptology ePrint Archive, Paper 2022/1640 (2022)
17. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_13
18. Coppersmith, D.: The data encryption standard (DES) and its strength against attacks. IBM J. Res. Dev. **38**(3), 243–250 (1994)
19. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. ISC, Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-04722-4
20. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
21. Derbez, P., Fouque, P.-A.: Automatic search of meet-in-the-middle and impossible differential attacks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 157–184. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_6
22. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)
23. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_42
24. Dong, X., Hua, J., Sun, S., Li, Z., Wang, X., Hu, L.: Meet-in-the-middle attacks revisited: key-recovery, collision, and preimage attacks. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12827, pp. 278–308. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_10

25. Dunkelman, O., Sekar, G., Preneel, B.: Improved meet-in-the-middle attacks on reduced-round DES. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 86–100. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77026-8_8
26. Espitau, T., Fouque, P.-A., Karpman, P.: Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 683–701. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_33
27. Fuhr, T., Minaud, B.: Match box meet-in-the-middle attack against KATAN. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 61–81. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_4
28. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl - a SHA-3 candidate. In: Symmetric Cryptography (2009)
29. Gueron, S., Mouha, N.: Simpira v2: a family of efficient permutations using the AES round function. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 95–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_4
30. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced meet-in-the-middle preimage attacks: first results on full tiger, and improved results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_4
31. Hirose, S., Ideguchi, K., Kuwakado, H., Owada, T., Preneel, B., Yoshida, H.: A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-LW. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 151–168. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24209-0_10
32. Hou, Q., Dong, X., Qin, L., Zhang, G., Wang, X.: Automated meet-in-the-middle attack goes to Feistel. Cryptology ePrint Archive, Paper 2023/1359 (2023). https://eprint.iacr.org/2023/1359
33. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A practical attack on KeeLoq. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 1–18. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_1
34. Isobe, T.: A single-key attack on the full GOST block cipher. J. Cryptol. **26**(1), 172–189 (2013)
35. Isobe, T., et al.: Areion: highly-efficient permutations and its applications to hash functions for short input. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(2), 115–154 (2023)
36. Knellwolf, S., Khovratovich, D.: New preimage attacks against reduced SHA-1. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 367–383. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_22
37. Kölbl, S., Lauridsen, M.M., Mendel, F., Rechberger, C.: Haraka v2 - efficient short-input hashing for post-quantum applications. IACR Trans. Symmetric Cryptol. **2016**(2), 1–29 (2016)
38. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-47555-9_5
39. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_26

40. Liu, F., Sarkar, S., Wang, G., Meier, W., Isobe, T.: Algebraic meet-in-the-middle attack on LowMC. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology, ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 225–255. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22963-3_8
41. Lucks, S.: Attacking triple encryption. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 239–253. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69710-1_16
42. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
43. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: a synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_31
44. Qin, L., Hua, J., Dong, X., Yan, H., Wang, X.: Meet-in-the-middle preimage attacks on sponge-based hashing. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology, EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 158–188. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30634-1_6
45. Sasaki, Yu.: Integer linear programming for three-subset meet-in-the-middle attacks: application to GIFT. In: Inomata, A., Yasuda, K. (eds.) IWSEC 2018. LNCS, vol. 11049, pp. 227–243. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97916-8_15
46. Sasaki, Yu.: Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 378–396. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_22
47. Sasaki, Yu.: Preimage attacks on Feistel-SP functions: impact of omitting the last network twist. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 170–185. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_11
48. Sasaki, Yu., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_16
49. Sasaki, Yu., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_8
50. Sasaki, Yu., Wang, L., Wu, S., Wu, W.: Investigating fundamental security requirements on whirlpool: improved preimage and collision attacks. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 562–579. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_34
51. Schrottenloher, A., Stevens, M.: Simplified MITM modeling for permutations: new (quantum) attacks. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology, CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 717–747. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15982-4_24
52. Schrottenloher, A., Stevens, M.: Simplified MITM modeling for permutations: new (quantum) attacks. IACR Cryptology ePrint Archive, Paper 2022/189 (2022)
53. Wu, S., Feng, D., Wu, W., Guo, J., Dong, L., Zou, J.: (Pseudo) preimage attack on round-reduced `Grøstl` hash function and others. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 127–145. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_8
54. Zheng, Y., Matsumoto, T., Imai, H.: On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_42