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 [37,42]. Typical examples are PGV-modes [42], 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` [36]. 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` [34], 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* [48] 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 [40,25,14,33] and hash functions [48,3,30]. 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 [48], bicliques [13], 3-subset MitM [14], indirect-partial matching [3,48], sieve-in-the-middle [17], match-box [27], dissection [23], MitM with guess-and-determine [49], differential-aided MitM [35,26,16], algebraic MitM [39], two-stage MitM [5], quantum MitM [50], etc. Till now, the MitM attack and its variants have broken MD4 [38,30], MD5 [48], KeeLoq [32], HAVAL [4,47], GOST [33], GEA-1/2 [10,1], 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.* [52] introduced a search algorithm for MitM attacks on `Grøstl`. In [44], 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 [50] 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.* [43] 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 [3,2,48,30]. At ACNS 2013, Sasaki *et al.* [46] 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 [50] 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 [3,48], 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* [2,48]. 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 [6,24,7,43] 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} [34] to 2^{193}, and improve the preimage attack on `Simpira-2` from previous 5 rounds [50] to ours 7 rounds.

For the second contribution, We first generalize Sasaki's multi-round matching strategy for Feistel [46] into full-round matching. At ACNS 2013, Sasaki [46] 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 [46] 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 [50] on `Simpira v2` by improving the attack on `Simpira-2` from 5 rounds [50] to ours 7 rounds, and improving the attack on `Simpira-4` from 9 rounds [50] to ours 11 rounds. As stated by Schrottenloher and Stevens [51, 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 as shown in Section D 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` [34] 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 [50] 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 [50], 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 [51, 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}	[46]
		Classical	12	2^{113}	2^{48}	2^{128}	Sect. 5
`Simpira-2`	Preimage	Classical	5	2^{128}	-	2^{256}	[50]
		Quantum	5	2^{64}	-	2^{128}	[50]
		Classical	7	2^{225}	2^{96}	2^{256}	Sect. 6.1
`Simpira-4`	Preimage	Classical	9	2^{128}	-	2^{256}	[50]
		Quantum	9	2^{64}	-	2^{128}	[50]
		Classical	11	2^{225}	2^{160}	2^{256}	Sect. 6.2
`Simpira-6`	Preimage	Classical	11	$2^{193.6}$	2^{193}	2^{256}	Sect. C
`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}	Sect. D
`Areion256-DM`	Preimage	Classical	5	2^{248}	2^{8}	2^{256}	[34]
		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}	[34]
		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

$A^{(r)}_{\texttt{SB}}$: the internal state after operation `SB` in round r, $r \geq 0$
$A^{(r)}_{\texttt{SB}}[i]$: the i-th byte of the internal state $A^{(r)}_{\texttt{SB}}$
■, $\mathcal{R}$: 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
$\mathrm{DoF}_{\mathcal{R}}$: the byte number of DoF of the ■ neutral words
$\mathrm{DoF}_{\mathcal{B}}$: 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

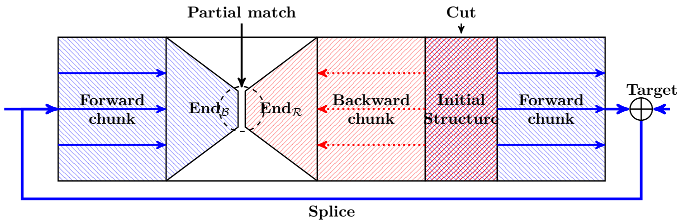


Fig. 1: The closed computation path of the MitM attack

Since the pioneering works on preimage attacks on Merkle–Damgård hashing, e.g. `MD4`, `MD5`, and `HAVAL` [38,48,3,30], techniques such as *splice-and-cut* [3], *initial structure* [48] and *(indirect-) partial matching* [2,48] have been invented to significantly improve the MitM approach. In Figure 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 [45,46] 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 $\mathrm{DoF}_{\mathcal{R}}$ or $\mathrm{DoF}_{\mathcal{B}}$.
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 $2^{8\cdot\mathrm{DoF}_{\mathcal{R}}}$ 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:

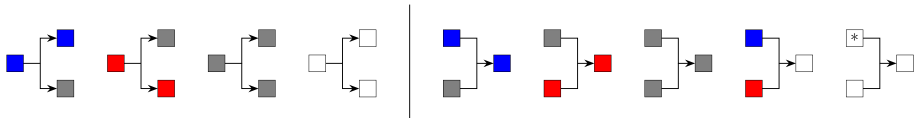
$$2^{h-8\cdot\min(\mathrm{DoF}_{\mathcal{R}},\mathrm{DoF}_{\mathcal{B}},\mathrm{DoM})}. \tag{1}$$

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 [45,49], 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, 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 named as 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 Figure 2.



(a) Rules for separation (b) Rules for combination

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.

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 Figure 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 [42], such as Davies-Meyer (DM, Figure 4), Matyas-Meyer-Oseas (MMO, Figure 5) and Miyaguchi-Preneel (MP), is a common way to build the compression functions for hashing.

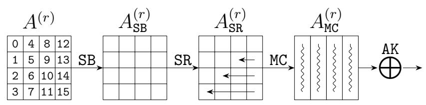


Fig. 3: One round AES

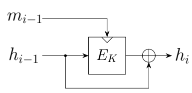


Fig. 4: DM

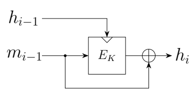


Fig. 5: MMO

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) [53]. 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 Figure 6. Figure 7 is an equivalent transformation of Figure 6, where $\tilde{A}^{(r)} = \mathtt{MC}^{-1}(A^{(r)})$, $\tilde{B}^{(r)} = \mathtt{MC}^{-1}(B^{(r)})$, $\tilde{A}^{(r+1)} = \mathtt{MC}^{-1}(A^{(r+1)})$, and $\tilde{B}^{(r+1)} = \mathtt{MC}^{-1}(B^{(r+1)})$. The round function of GFN adopts multiple branches, e.g., the round function of 4-branch Simpira v2 in Figure 8.

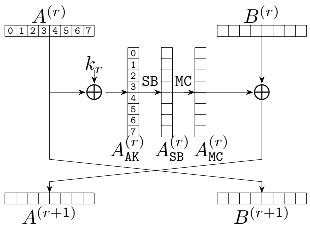


Fig. 6: One round Feistel-SP

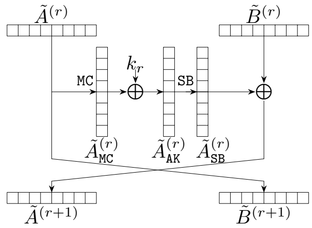


Fig. 7: Equivalent transform of Feistel-SP

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.

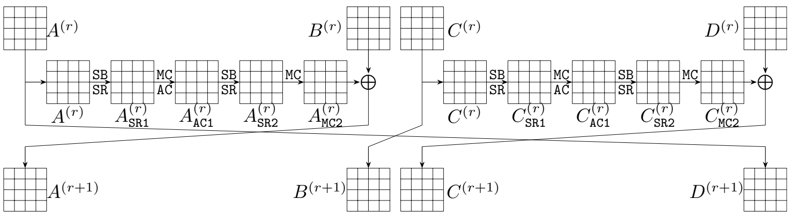


Fig. 8: The round function of `Simpira`-4

2.6 Areion

`Areion` [34] 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 Figure 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

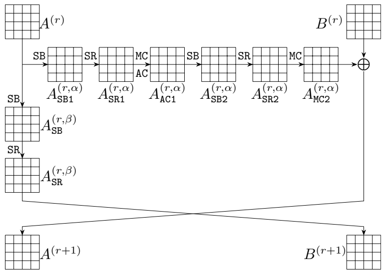


Fig. 9: One round `Areion-256`

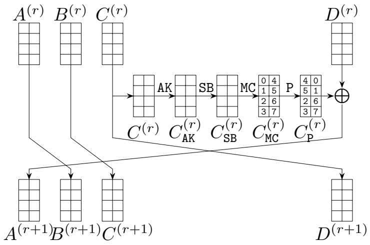


Fig. 10: One round `Lesamnta-LW`

a Merkle-Damgård iterated hash function [41,20]. Figure 11 shows a hash with two message blocks, where the i-th compression function (CF) is $\texttt{CF}(h_{i-1}, m_i) = E(h_{i-1}^0, m_i \| h_{i-1}^1) = h_i$, with h_{i-1}^0, h_{i-1}^1, $m_i \in \mathbb{F}_2^{128}$, h_{i-1}, $h_i \in \mathbb{F}_2^{256}$, and $h_{i-1} = h_{i-1}^0 \| h_{i-1}^1$. 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 $m_i = A^{(r)} \| B^{(r)}$, $h_{i-1}^1 = C^{(r)} \| D^{(r)}$. `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`.

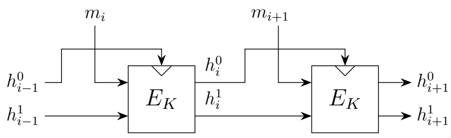
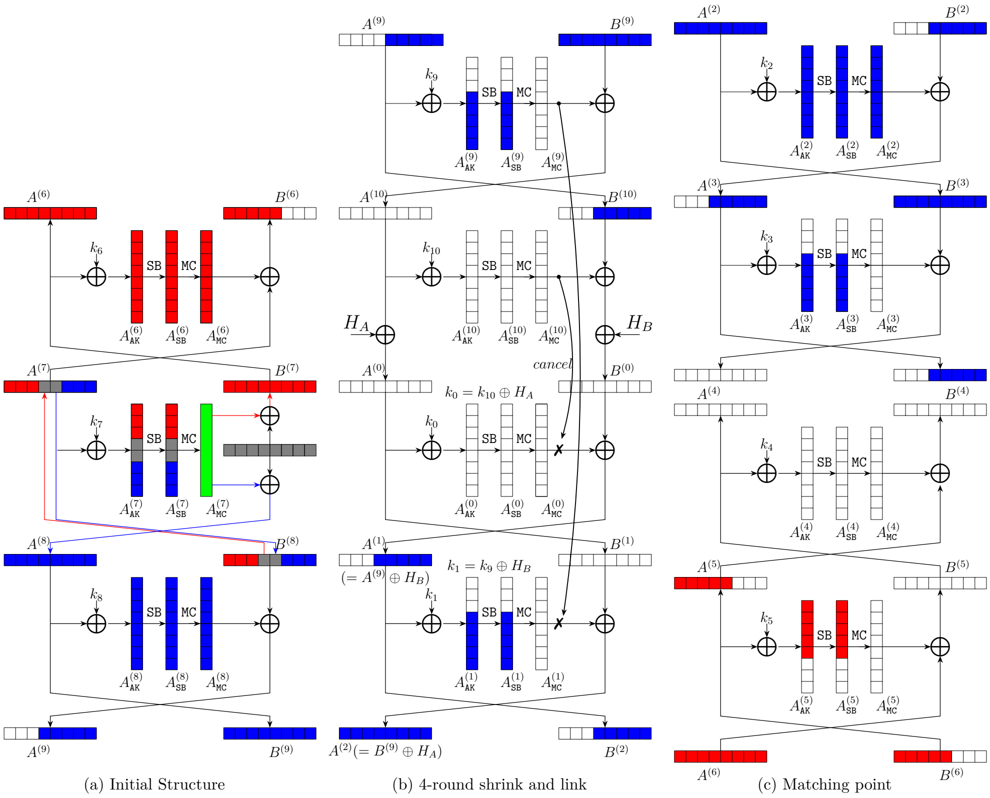


Fig. 11: `Lesamnta-LW` hash with two message blocks

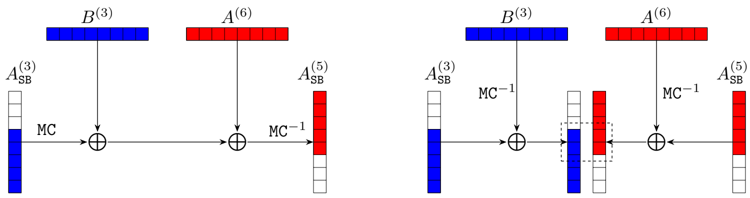
2.8 Sasaki's preimage attack on Feistel-SP

At ACNS 2013, Sasaki [46] introduced the MitM preimage attacks on MMO hashing mode with Feistel-SP block ciphers by omitting the last network twist. In Figure 12(a), $A_{\texttt{AK}}^{(6)}$ and $A_{\texttt{AK}}^{(7)}$ 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 $\text{DoF}_{\mathcal{R}} = 11 - 8 = 3$ and $\text{DoF}_{\mathcal{B}} = 3$, respectively. In Figure 12(b), by assigning conditions $k_0 = k_{10} \oplus H_A$ and $k_1 = k_9 \oplus H_B$, we have $A_{\texttt{MC}}^{(10)} = A_{\texttt{MC}}^{(0)}$ and $A_{\texttt{MC}}^{(9)} = A_{\texttt{MC}}^{(1)}$. Therefore, $A^{(2)} = B^{(9)} \oplus H_A$ and $B^{(2)} = A^{(9)} \oplus H_B$. In Figure 12(c), Sasaki applied a linear transformation in the computation from $A_{\texttt{SB}}^{(3)}$ to $A_{\texttt{SB}}^{(5)}$ to derive a multi-round matching with DoM = 2 as shown in Figure 13. The time complexity is $2^{8\times(16-\min\{3,3,2\})} = 2^{112}$.



(a) Initial Structure (b) 4-round shrink and link (c) Matching point

Fig. 12: Sasaki's attack



(a) Original (b) Transformed

Fig. 13: Matching in Sasaki's attack

3 Generalization on Matching Strategy in MitM

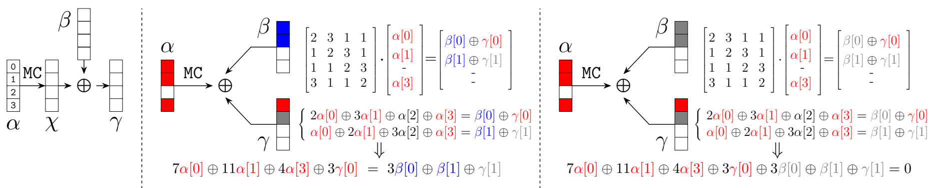
In the matching point of the MitM attack, with forward and backward computations, if two matching states F^+ and F^- are determined only by the ■ and ■, respectively, then, the relation $F^+ = F^-$ acts as a *direct partial matching*. This simple matching strategy is frequently used in previous works [48,45]. 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 Figure 14(a). Suppose from the matching states, there exist $\mathrm{M}_{\mathcal{R}}$ byte-equations $\pi_{\mathcal{R}} = 0$, $\mathrm{M}_{\mathcal{B}}$ 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 [45,46], 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 $\mathrm{M}_{\mathcal{R}} = 1$ 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 Figure 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) $\mathrm{M}_{\mathcal{R}} = 1$ bytes matching

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 Figure 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 $\mathtt{OR}(\omega_i^{\beta},\omega_i^{\gamma})$, 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 Figure 14, where $\chi = \beta \oplus \gamma$. The rule to generate χ follows the `XOR-Rule` in [6], (i.e. 🟦 ⊕ 🟥 = □, 🟦 ⊕ 🟦 = 🟦, □ ⊕ 🟦 = □, 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

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

where $\mathtt{AND}(x_i, y_i) = 1$ 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

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

For the `MC` then `XOR` operations in Figure 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

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

4 Automatic Model for Transformed Feistel Struture

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 [46], Sasaki proposed a matching strategy for Feistel with a linear transformation. As shown in Figure 13, it is hard to see any matching in the original Figure 13(a). However, after a linear transformation in Figure 13(b), the two-byte matching is obviously obtained. Besides the attack on balanced Feistel-SP, Sasaki [46] also built MitM attacks on GFN with SP round function, where the matching point covers 7 consecutive rounds. A similar linear transformation as in Figure 13(b) is also applied, but involves more internal states.

Inspired by Sasaki's matching strategy [46], 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 Figure 13(a) by replacing $B^{(3)}$ by $\mathtt{MC}(A_{\mathtt{SB}}^{(7)}) \oplus B^{(7)} \oplus H_A$ and replacing $A^{(6)}$ by $B^{(7)}$, where the internal states $A_{\mathtt{SB}}^{(7)}$ and $B^{(7)}$ come from the initial structure. Therefore, Figure 13 becomes Figure 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 $\square$ 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 Figure 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 Figure 15(a) to obtain Figure 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 Figure 15(a), $A_{\mathtt{SB}}^{(3)}$ will be separated into two states following the rules in Figure 2, then one of the two states will be all $\square$ 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 Figure 15(b), each byte of $A_{\mathtt{SB}}^{(3)}$ will be involved in the matching path individually. For example, considering the 4-th byte, there is a one-byte equation

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

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

III The transformed structure in Figure 15(b) is easy to program in the automatic tool. As shown in Equation (5), each byte can be individually considered, which is very friendly than the untransformed case in Figure 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 (Figure 23), there are more states involved in matching than that in Figure 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.

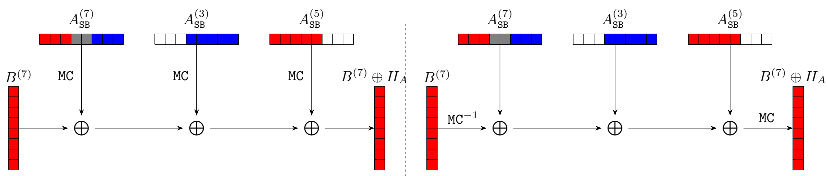


Fig. 15: Full-match in Feistel-SP

We find that the transformation in Figure 15(b) can be directly obtained if we consider MitM attacks on an equivalent transformation of Feistel-SP, i.e., Figure 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 Figure 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 Figure 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 $\left(\tilde{A}^{(r)}, \tilde{B}^{(r)}, \tilde{C}^{(r)}, \tilde{D}^{(r)}\right) = \left(\mathtt{MC}^{-1}(A^{(r)}), \mathtt{MC}^{-1}(B^{(r)}), \mathtt{MC}^{-1}(C^{(r)}), \mathtt{MC}^{-1}(D^{(r)})\right)$, 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 Figure 16, where $\tilde{A}^{(r+1)} = \tilde{B}^{(r)} \oplus \tilde{A}^{(r)}_{\mathtt{SR2}}$. In this way, $\tilde{A}^{(r)}_{\mathtt{MC1}} = \mathtt{MC}(\tilde{A}^{(r)}) = A^{(r)}$, then $\tilde{A}^{(r)}_{\mathtt{SR2}} = A^{(r)}_{\mathtt{SR2}}$. According to the predefined $\tilde{B}^{(r)} = \mathtt{MC}^{-1}(B^{(r)})$, $\tilde{A}^{(r+1)}$ is equivalent to $\mathtt{MC}^{-1}(B^{(r)}) \oplus A^{(r)}_{\mathtt{SR2}}$. 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 (Figure 8). This is also true for $\tilde{B}^{(r+1)}$, $\tilde{C}^{(r+1)}$ and $\tilde{D}^{(r+1)}$.

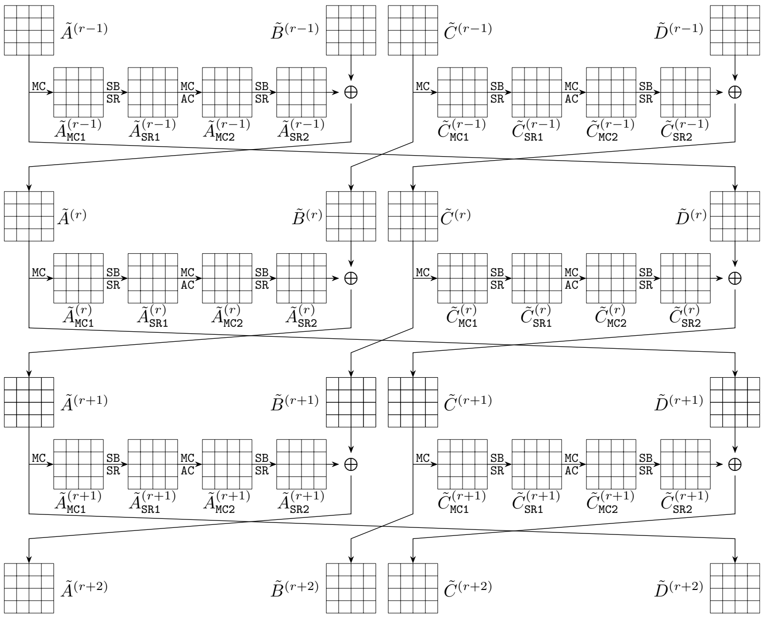
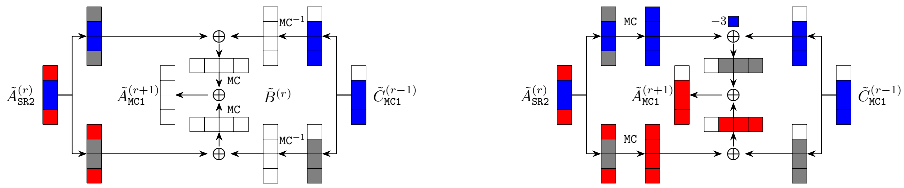


Fig. 16: Equivalent transform of Simpira-4

MILP Constraints for the Computation Paths. As shown in Figure 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 Figure 17(a), when applying `MC-Rule` on the superposition states of $\tilde{C}^{(r-1)}_{\texttt{MC1}}$, it will lead to all □ cells. Subsequently, $\tilde{A}^{(r+1)}_{\texttt{MC1}}$ will end up with a full column of □ cells. However, if we apply the `MC-then-XOR-Rule` with superposition framework as shown in Figure 17(b), three ■ cells will be preserved by consuming three ■ cells. This also fits our intuition, i.e. more linear operations yield higher possibility of generating unknown cells.



(a) Model `MC-Rule` and `XOR-Rule` separately (b) Model the link by `MC-then-XOR-Rule`

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

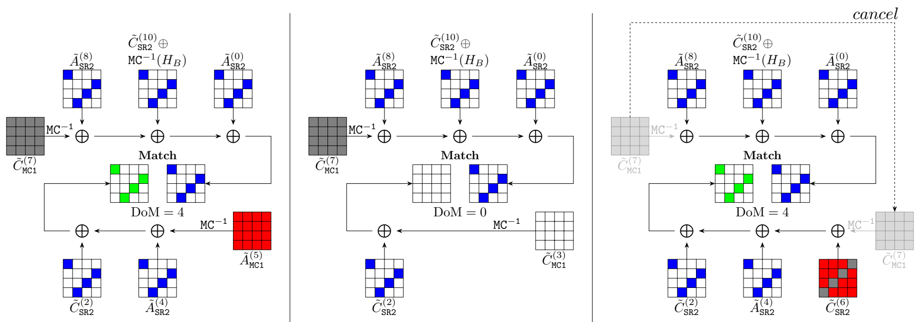
MILP Constraints for the Full-Round Match. In Figure 12(c), the ending states are $(A^{(4)}, B^{(4)})$ computed from two opposite directions. With a linear transformation, two-byte partial matching is deduced as shown in Figure 13. The matching phase involves two rounds of forward and two rounds of backward, 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 Figure 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 Equation (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 (Figure 23) on transformed `Simpira`-4 in Section 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 Figure 18(a). If a shorter round is considered for matching, e.g., $(6, 2)$-round in Figure 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 Figure 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 Equation (7):

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

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.



(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 Figure 18(c). For the general case, assume the matching phase consists of two starting states I_1 and I_2, e.g., in Figure 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 Equation (7), the generic full-round matching equation can be written as

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

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 Section 3, if all these involved bytes are not $\square$ 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 Figure 19, which improves Sasaki's attack [46] 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

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

with $\tilde{A}_{\texttt{SB}}^{(1)} = \tilde{A}_{\texttt{SB}}^{(10)}$ by assigning the same assumption to Sasaki's attack [46], i.e., $k_0 = k_{11} \oplus H_A$ and $k_1 = k_{10} \oplus H_B$. From Equation (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

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

6 Meet-in-the-Middle Attack on Reduced `Simpira v2`

For `Simpira v2` [29] with branch number bigger than 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 [50], we follow the same way of truncation of the output of `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 [50], we introduce the first attack on reduced `Simpira-6` in Supplementary Material C. We also give an experiment based on a new 7-round MitM characteristic of `Simpira-2` in Supplementary Material F.

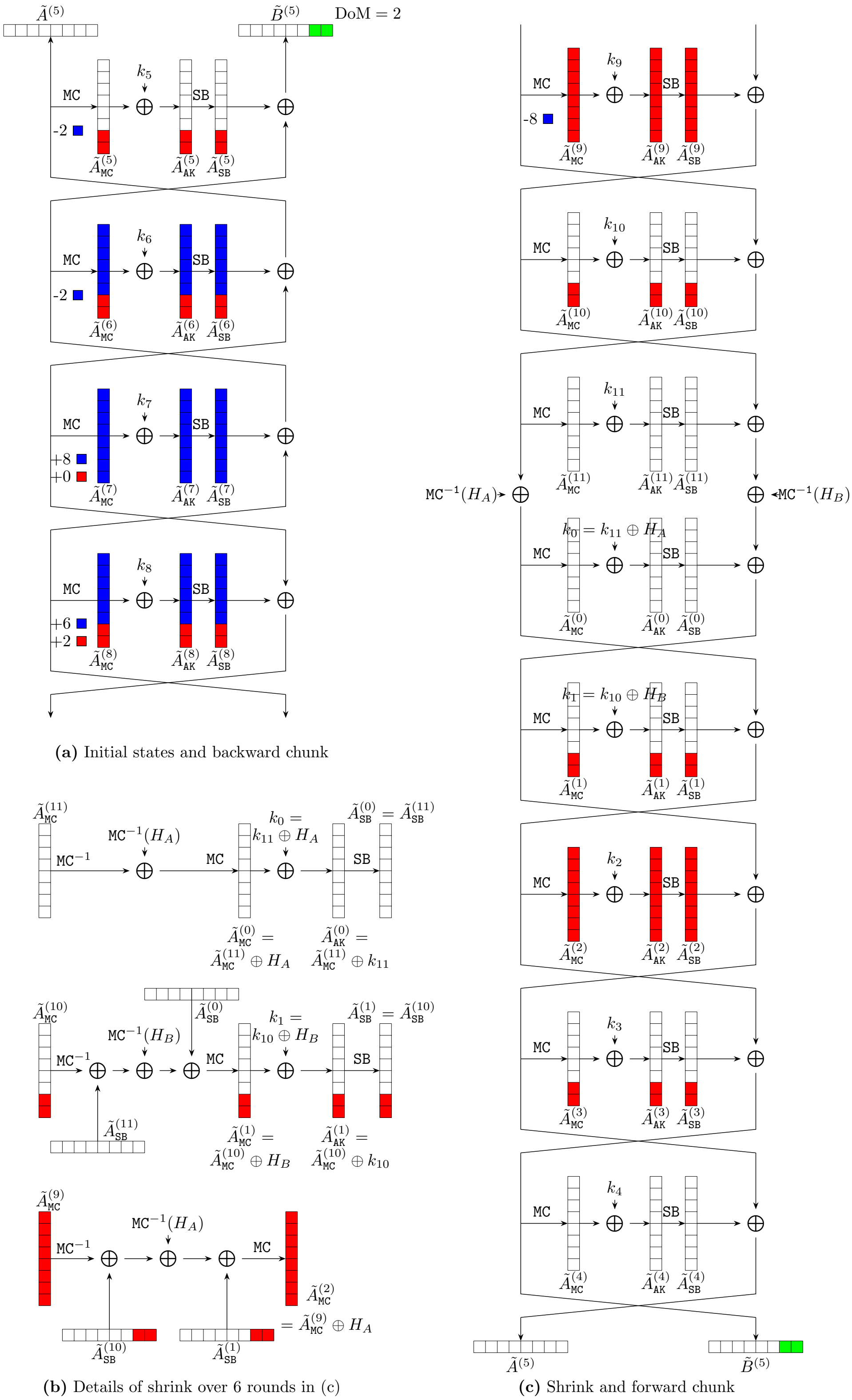


Fig. 19: MitM attack on 12-round `Feistel-SP`

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

Set constraints on key schedule $k_0 = k_{11} \oplus H_A$ and $k_1 = k_{10} \oplus H_B$
for $g_b \in \mathbb{F}_2^{64}$ /* $\texttt{MC}(\tilde{A}^{(8)}_{\texttt{SB}}[0\text{-}5]||0||0) \oplus \tilde{A}^{(7)}_{\texttt{MC}} = g_b$ */
do
 $U \leftarrow [\,]$
 for $v_{\mathcal{B}} \in \mathbb{F}_2^{6\times 8}$ *in* $\tilde{A}^{(8)}_{\texttt{SB}}[0\text{-}5]$ **do**
  $\tilde{A}^{(7)}_{\texttt{MC}} \leftarrow \texttt{MC}(v_{\mathcal{B}}\|0\|0) \oplus g_b$
  Compute throungh `AK` and `SB` to get the values of ■ cells in $\tilde{A}^{(7)}_{\texttt{SB}}$
  $c_0\|c_1 \leftarrow \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SB}})[6,7]$ /* $\tilde{A}^{(6)}_{\texttt{MC}} = \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SB}}) \oplus \tilde{A}^{(8)}_{\texttt{MC}}$ */
  Compute ■ cells in $\tilde{A}^{(6)}_{\texttt{SB}}$
  $c_2\|c_3 \leftarrow \texttt{MC}(\tilde{A}^{(6)}_{\texttt{SB}}[0\text{-}5]\|0\|0)[6,7] \oplus \tilde{A}^{(7)}_{\texttt{MC}}[6,7]$
  $\mathfrak{c}_{\mathcal{B}} \leftarrow c_0\|c_1\|c_2\|c_3$
  $U[\mathfrak{c}_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$ /* There are 2^{16} elements in $U[\mathfrak{c}_{\mathcal{B}}]$ given $\mathfrak{c}_{\mathcal{B}}$ */
 end
 for $\mathfrak{c}_{\mathcal{B}} \in \mathbb{F}_2^{4\times 8}$ **do**
  $L \leftarrow [\,]$
  for $v_{\mathcal{B}} \in U[\mathfrak{c}_{\mathcal{B}}]$ **do**
   Compute backward to the ■ cells in $\tilde{A}^{(6)}_{\texttt{MC}}$. According to Figure 19, derive 2 bytes $End_{\mathcal{B}}$ for matching by

$$End_{\mathcal{B}} \leftarrow \texttt{MC}^{-1}\left(\tilde{A}^{(6)}_{\texttt{MC}}[0-5]\|0\|0\right)[6,7]$$

   $L[End_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$
  end
  for $2^{8\lambda_{\mathcal{R}}}$ *values* $v_{\mathcal{R}}$ *of the* ■ *bytes in* $\tilde{A}^{(8)}_{\texttt{MC}}$, $\lambda_{\mathcal{R}} = 2$ **do**
   Compute backward to the ■ cells in $\tilde{A}^{(5)}_{\texttt{SB}}$
   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 $\tilde{A}^{(2)}_{\texttt{MC}} = \tilde{A}^{(9)}_{\texttt{MC}} \oplus H_A$
   With $\tilde{A}^{(1)}_{\texttt{MC}}$ and $\tilde{A}^{(2)}_{\texttt{MC}}$, compute forward to the ■ cells in $\tilde{A}^{(3)}_{\texttt{SB}}$
   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

$$End_{\mathcal{R}} \leftarrow \texttt{MC}^{-1}\left(\tilde{A}^{(2)}_{\texttt{MC}}\right)[6,7] \oplus \tilde{A}^{(3)}_{\texttt{SB}}[6,7] \oplus \tilde{A}^{(5)}_{\texttt{SB}}[6,7] \oplus \texttt{MC}^{-1}\left(0\|0\|0\|0\|0\|0\|\tilde{A}^{(6)}_{\texttt{MC}}[6,7]\right)[6,7]$$

   for $v_{\mathcal{B}} \in L[End_{\mathcal{R}}]$ **do**
    Reconstruct the (candidate) message X
    if X *is a preimage* **then**
     Output X and stop
    end
   end
  end
 end
end

6.1 Meet-in-the-Middle Attack on 7-round Simpira-2

As shown in Figure 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 Figure 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, $\texttt{MC}^{-1}(\tilde{A}^{(3)}_{\texttt{MC1}})$ makes no contribution to the match and can be canceled without influence as shown in Figure 22.

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

Then, 4 bytes for matching in the Equation (10) indexed by $[3, 6, 9, 12]$ are only determined by the ■ bytes, i.e. $\text{M}_{\mathcal{R}} = 4$. 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 Equation (2), the overall time complexity to mount a MitM attack is

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

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

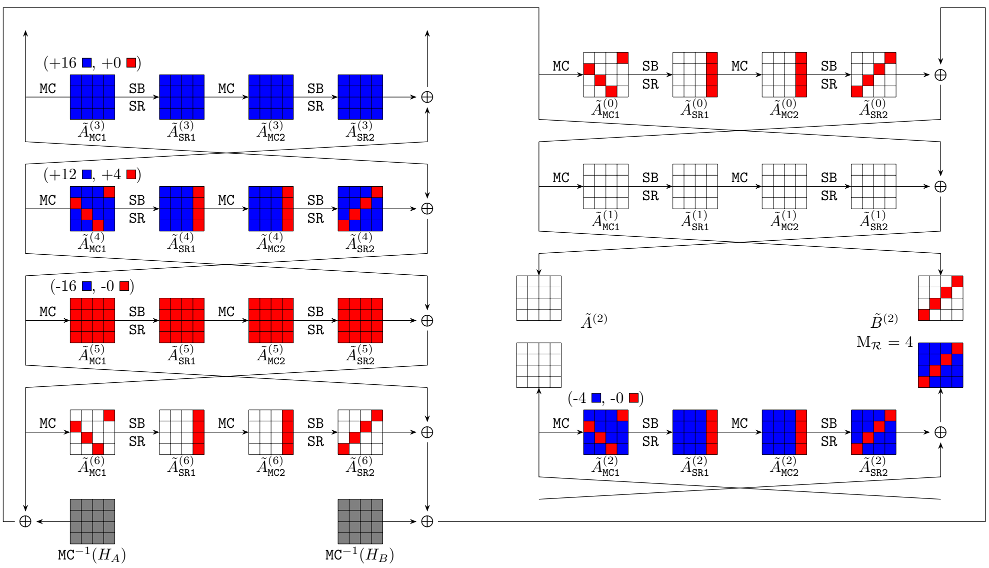


Fig. 20: MitM attack on 7-round Simpira-2

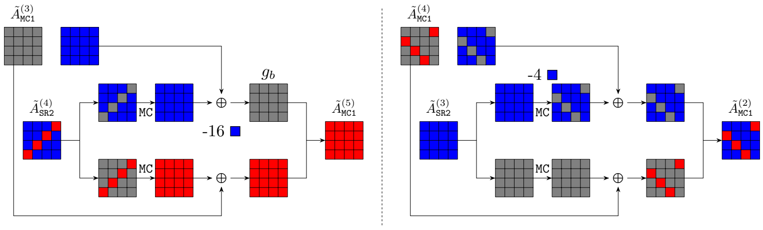


Fig. 21: The `MC-then-XOR-Rule` of `Simpira-2` in superposition framework

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 Figure 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 Figure 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 Figure 22, $\text{M}_{\mathcal{R}}$=4 bytes equations are derived by

$$\left(\tilde{A}^{(2)}_{\texttt{SR2}} \oplus \tilde{A}^{(4)}_{\texttt{SR2}} \oplus \tilde{A}^{(6)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B)\right)[3, 6, 9, 12] = \tilde{A}^{(0)}_{\texttt{SR2}}[3, 6, 9, 12]$$

17   Put the solution into $\mathcal{S}$
18   **end**
19   **for** $v_{\mathcal{B}} \in U[\mathfrak{c}_{\mathcal{B}}]$ **do**
20   Compute the ■ cells in $\tilde{A}^{(3)}_{\texttt{MC1}}$ as Line 6
21   **for** $v_{\mathcal{R}} \in \mathcal{S}$ **do**
22   Reconstruct the (candidate) message X
23   **if** X *is a preimage* **then**
24   Output X and stop
25   **end**
26   **end**
27   **end**
28   **end**
29 **end**

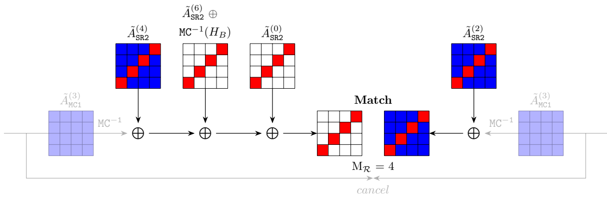


Fig. 22: Full-round matching in 7-round Simpira-2

6.2 Meet-in-the-Middle Attack on 11-round Simpira-4

Figure 23 is an 11-round MitM characteristic on Simpira-4. Figure 28 given in Supplementary Material B is an alternative representation of the MitM characteristic with MC-then-XOR-Rule in superposition framework. The starting states are $\tilde{A}^{(7)}_{\texttt{MC1}}$, $\tilde{C}^{(6)}_{\texttt{MC1}}$, $\tilde{A}^{(6)}_{\texttt{MC1}}$, and $\tilde{C}^{(7)}_{\texttt{MC1}}$. The initial DoFs for ■ and ■ is $\lambda_{\mathcal{R}} = 24$ and $\lambda_{\mathcal{B}} = 4$, respectively. Along the forward and backward computation paths, there are total 20 constrains on ■ and 0 constant constrain on ■, i.e. $l_{\mathcal{R}} = 20$ and $l_{\mathcal{B}} = 0$. Hence, we get $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 4$ and $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 4$. 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 cancelled.

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

Then, 4 bytes in Equation (11) indexed by $[0, 7, 10, 13]$ are derived as the degree of match, i.e. 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 $2^{32} \times 2^{192} + 2^{8 \times \left(32 - \min\{24-20, 4, 4\}\right)} \approx 2^{225}$. 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 Figure 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 $\text{DoF}_{\mathcal{R}} = \text{DoF}_{\mathcal{B}} = 4$. The matching happens between $D^{(17)}$ and the targeted hash value, where DoM = 8. The procedure of the MitM collision attack is given in Algorithm 4, where two message blocks (m_1, m_2) are needed as shown in Figure 11. In our collision attack, we only use the first column of $D^{(17)}$ for matching. At first, we randomly fix the 32-bit partial target as constant, i.e., the first 32-bit $D^{(17)}$. 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

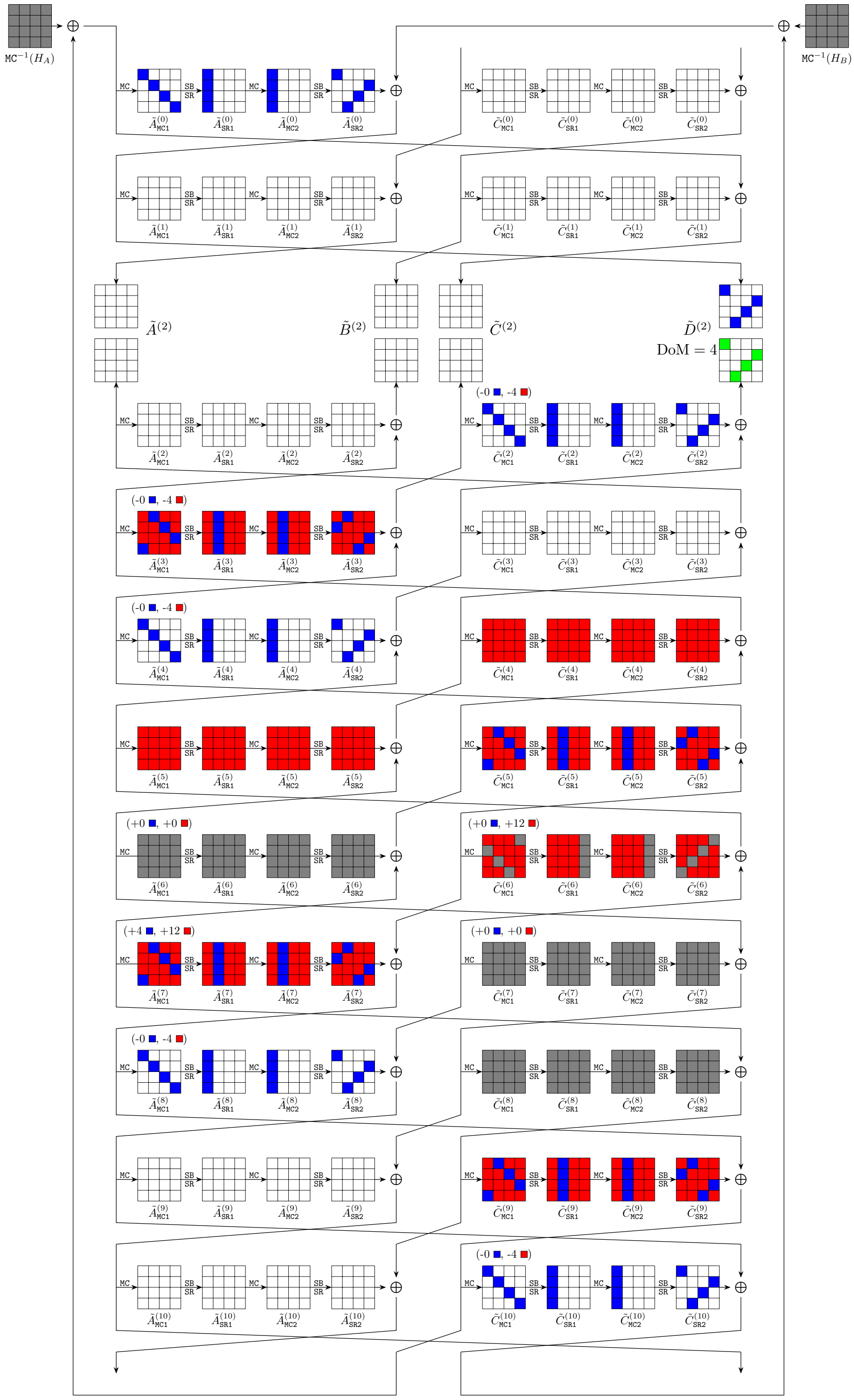


Fig. 23: MitM attack on 11-round Simpira-4

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

for $\tilde{A}^{(6)}_{\texttt{MC1}} \| \tilde{C}^{(6)}_{\texttt{MC1}}[1,6,11,12] \| \tilde{C}^{(7)}_{\texttt{MC1}} \in \mathbb{G}$ /* $|\mathbb{G}| = 2^{32}$ */
do
  for $g_r \in \mathbb{F}_2^{32}$ **do**
    $V \leftarrow [\,]$
    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**
      Compute the $\textcolor{red}{\blacksquare}$ cells in $\tilde{A}^{(7)}_{\texttt{SR2}}$ from $\tilde{A}^{(7)}_{\texttt{MC1}}$
      Let $\tilde{A}^{(7)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \in [1,4,11,14]$
      $\tilde{C}^{(6)}_{\texttt{MC1}}[0,5,10,15] \leftarrow \texttt{MC}(\tilde{A}^{(7)}_{\texttt{SR2}})[0,5,10,15] \oplus g_r$
      From $\tilde{A}^{(7)}_{\texttt{MC1}}$ and $\tilde{A}^{(6)}_{\texttt{MC1}}$, compute the $\textcolor{red}{\blacksquare}$ cells in $\tilde{C}^{(5)}_{\texttt{SR2}}$
      Let $\tilde{C}^{(5)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \in [1,4,11,14]$
      $c_0\|c_1\|c_2\|c_3 \leftarrow \left(\texttt{MC}(\tilde{C}^{(5)}_{\texttt{SR2}}) \oplus \tilde{C}^{(6)}_{\texttt{MC1}}\right)[0,5,10,15]$
      From the known values, compute the $\textcolor{red}{\blacksquare}$ cells in $\tilde{C}^{(9)}_{\texttt{SR2}}$, $\tilde{C}^{(4)}_{\texttt{SR2}}$, $\tilde{A}^{(3)}_{\texttt{SR2}}$, and let the remaining $\textcolor{blue}{\blacksquare}$ cells be 0
      $c_4\|c_5\|c_6\|c_7 \leftarrow \texttt{MC}\left(\tilde{C}^{(9)}_{\texttt{SR2}}\right)[0,5,10,15]$,
      $c_8\|c_9\|c_{10}\|c_{11} \leftarrow \texttt{MC}\left(\tilde{C}^{(4)}_{\texttt{SR2}}\right)[3,4,9,14]$
      $c_{12}\|c_{13}\|c_{14}\|c_{15} \leftarrow \texttt{MC}\left(\tilde{A}^{(3)}_{\texttt{SR2}}\right)[0,5,10,15]$,
      $\mathfrak{c}_{\mathcal{R}} \leftarrow c_0\|c_1\|\cdots\|c_{14}\|c_{15}$
      $V[\mathfrak{c}_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
    end
    for $\mathfrak{c}_{\mathcal{R}} \leftarrow \mathbb{F}_2^{16\times 8}$ **do**
      $L \leftarrow [\,]$
      for $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ **do**
        Compute the $\textcolor{red}{\blacksquare}$ cells in $\tilde{C}^{(6)}_{\texttt{SR2}}$. According to Figure 18(c), derive 4 bytes $End_{\mathcal{R}}$ for matching by

$$End_{\mathcal{R}} \leftarrow \left(\tilde{C}^{(6)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B)\right)[0,7,10,13]$$

        $L[End_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
      end
      for $2^{8\lambda_{\mathcal{B}}}$ *values* $v_{\mathcal{B}}$ *of the* $\textcolor{blue}{\blacksquare}$ *bytes in* $\tilde{A}^{(7)}_{\texttt{MC1}}$, $\lambda_{\mathcal{B}} = 4$ **do**
        Compute backward to the $\textcolor{blue}{\blacksquare}$ cells in $\tilde{A}^{(4)}_{\texttt{SR2}}$ and $\tilde{C}^{(2)}_{\texttt{SR2}}$
        Compute forward to the $\textcolor{blue}{\blacksquare}$ cells in $\tilde{A}^{(8)}_{\texttt{SR2}}$, $\tilde{C}^{(10)}_{\texttt{SR2}}$ and $\tilde{A}^{(0)}_{\texttt{SR2}}$
        As in Figure 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**
          Reconstruct the (candidate) message X
          if X *is a preimage* **then**
            Output X and stop
          end
        end
      end
    end
  end
end

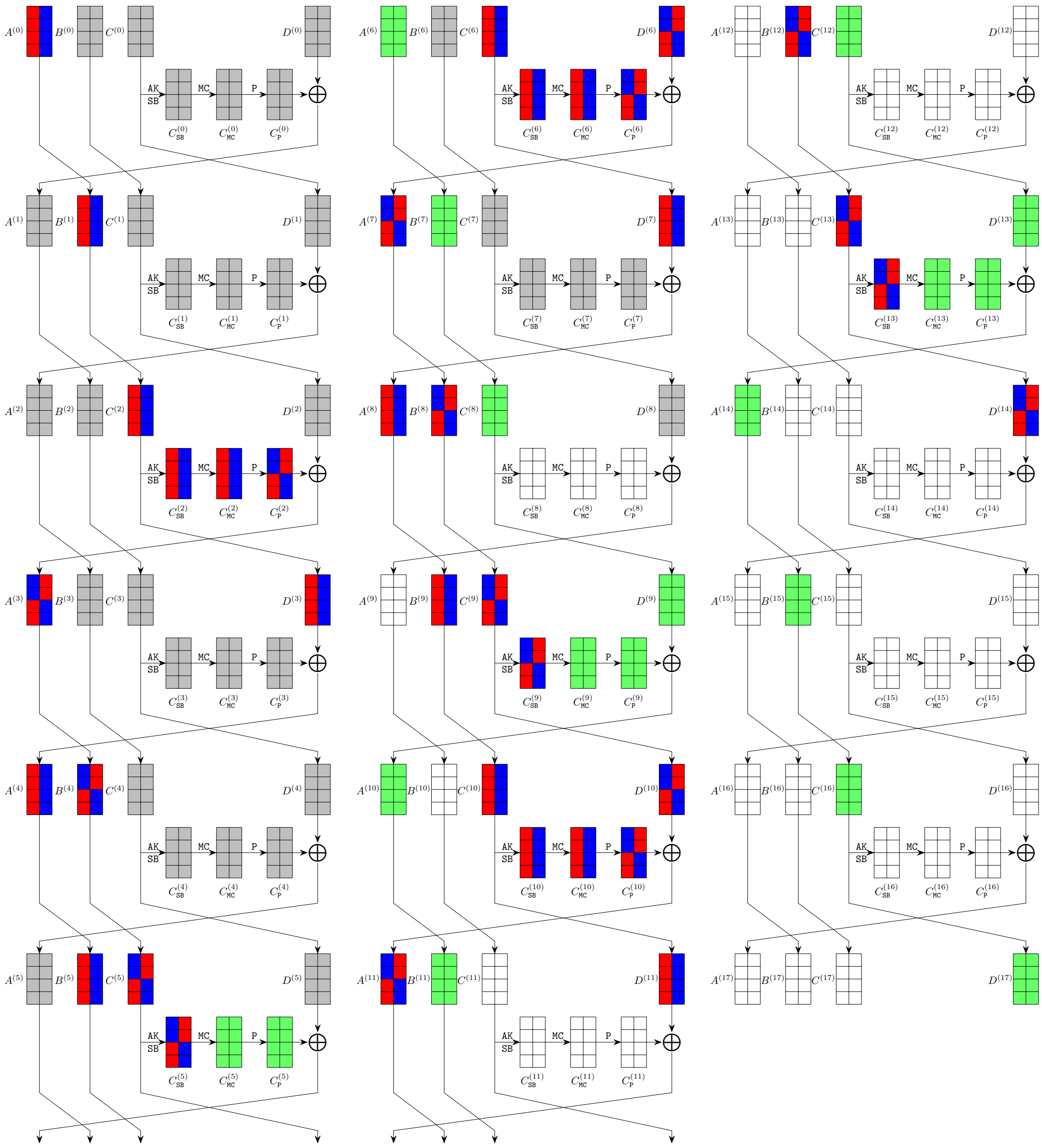


Fig. 24: MitM attack on 17-round Lesamnta-LW

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 linear transformation of collision.

Besides, we also found a 20-round MitM collision attack on `Lesamnta-LW` when targeting on 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 better than the designers' security claim against collision attack, which is 2^{120}. We still put the 20-round MitM characteristic in Section D to clearly specify the superiority of our new model.

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

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

Based on DM hashing mode, Isobe *et al.* [34] 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 $\texttt{SR} \circ \texttt{SB}$ for `Areion-256`. If we just transform it like `Simpira`, the left 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 Section 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 Figure 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 Figure 14, where DoM = 6. According to Section 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. The time to construct table U is $2^{8\cdot\lambda_{\mathcal{B}}} = 2^{184}$. Hence, we have the time complexity $2^8 \cdot 2^{184} + 2^{8\times\left(32-\min\{23-15,8,8\}\right)} \approx 2^{193}$. 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 Figure 34 and Algorithm 8 in Supplementary Material E. 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 $\text{DoF}_{\mathcal{R}} = \text{DoF}_{\mathcal{B}} = 2$. The matching happens between $A^{(1,\alpha)}_{\texttt{SR2}}$ and $B^{(1)} \oplus A^{(2)}$, by combining `MixColumn` and `XOR` operations as Figure 14, where 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 $2^{48} \cdot 2^{176} + 2^{8\times\left(32-\min\{22-20,4-2,2\}\right)} \approx 2^{240}$. 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 Figure 35, 36, and Algorithm 9 in Supplementary Material E. 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 $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 2$ and $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 2$. The matching phase happens between $C^{(9,\beta)}_{\texttt{SR}}$ and $B^{(10)}$ through `MixColumn`, where DoM = 2. The time complexity to precompute V is $2^{8\cdot\lambda_{\mathcal{R}}} = 2^{240}$. The time complexity is $2^{240} + 2^{8\times\left(32-\min\{30-28,\ 2,\ 2\}\right)} \approx 2^{241}$. The overall memory complexity is 2^{48} to store V.

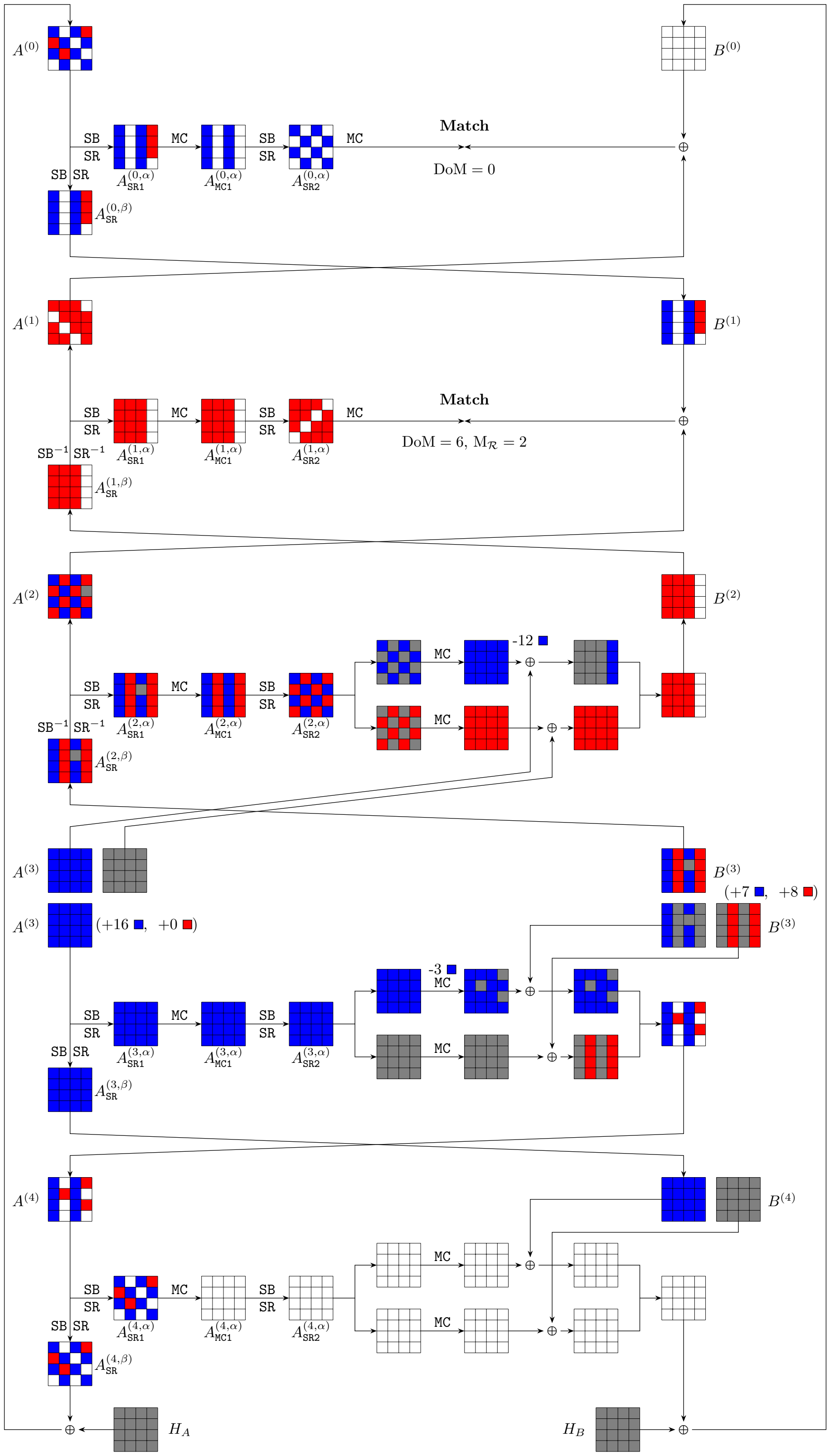


Fig. 25: MitM attack on 5-round `Areion-256`

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 significanlty 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), 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), Key Research Project of Zhejiang Province, China (2023C01025).

References

1. Dor Amzaleg and Itai Dinur. Refined cryptanalysis of the GPRS ciphers GEA-1 and GEA-2. *IACR Cryptol. ePrint Arch.*, page 424, 2022.
2. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In *ASIACRYPT 2009, Proceedings*, volume 5912, pages 578–597. Springer, 2009.
3. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *SAC 2008*, volume 5381, pages 103–119. Springer, 2008.
4. Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage attacks on 3-pass HAVAL and step-reduced MD5. In *SAC 2008*, volume 5381, pages 120–135.
5. Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New attacks on lowmc instances with a single plaintext/ciphertext pair. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 303–331. Springer, 2021.
6. Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In *EUROCRYPT 2021, Part I*, volume 12696, pages 771–804.
7. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. Superposition meet-in-the-middle attacks: Updates on fundamental security of aes-like hashing. In *CRYPTO 2022, Proceedings, Part I*, volume 13507, pages 64–93. Springer, 2022.
8. Paulo S.L.M. Barreto and Vincent Rijmen. The WHIRLPOOL hashing function. *Submitted to NESSIE*, 2000.

9. Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Lightweight AEAD and hashing using the sparkle permutation family. *IACR Trans. Symmetric Cryptol.*, 2020(S1):208–261, 2020.
10. Christof Beierle, Patrick Derbez, Gregor Leander, Gaëtan Leurent, Håvard Raddum, Yann Rotella, David Rupprecht, and Lukas Stennes. Cryptanalysis of the GPRS encryption algorithms GEA-1 and GEA-2. In *EUROCRYPT 2021, Proceedings, Part II*, volume 12697, pages 155–183. Springer, 2021.
11. Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. SHA-3 proposal: ECHO. *Submission to NIST (updated)*, page 113, 2009.
12. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30):320–337, 2009.
13. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *ASIACRYPT 2011, Proceedings*, pages 344–371.
14. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *SAC 2010*, volume 6544, pages 229–240. Springer, 2010.
15. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In *CRYPTO 2011, Proceedings*, volume 6841, pages 169–187. Springer, 2011.
16. Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 1640, 2022.
17. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In *CRYPTO 2013, Proceedings, Part I*, pages 222–240.
18. Don Coppersmith. The data encryption standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, 1994.
19. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
20. Ivan Damgård. A design principle for hash functions. In *CRYPTO '89*, pages 416–427.
21. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In *CRYPTO 2016, Proceedings, Part II*, volume 9815, pages 157–184. Springer, 2016.
22. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.
23. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *CRYPTO 2012*, volume 7417, pages 719–740.
24. Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO 2021, Proceedings, Part III*, volume 12827, pages 278–308. Springer.
25. Orr Dunkelman, Gautham Sekar, and Bart Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In *INDOCRYPT 2007, Proceedings*, volume 4859, pages 86–100. Springer, 2007.
26. Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman. Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 683–701. Springer, 2015.

27. Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In *FSE 2014*, pages 61–81, 2014.
28. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. In *Symmetric Cryptography*, 2009.
29. Shay Gueron and Nicky Mouha. Simpira v2: A family of efficient permutations using the AES round function. In *ASIACRYPT 2016, Proceedings, Part I*, volume 10031, pages 95–125, 2016.
30. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In *ASIACRYPT 2010, Proceedings*, volume 6477, pages 56–75.
31. Shoichi Hirose, Kota Ideguchi, Hidenori Kuwakado, Toru Owada, Bart Preneel, and Hirotaka Yoshida. A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-lw. In *ICISC 2010*, volume 6829, pages 151–168. Springer, 2010.
32. Sebastiaan Indesteege, Nathan Keller, Orr Dunkelman, Eli Biham, and Bart Preneel. A practical attack on KeeLoq. In *EUROCRYPT 2008, Proceedings*, volume 4965, pages 1–18. Springer, 2008.
33. Takanori Isobe. A single-key attack on the full GOST block cipher. *J. Cryptol.*, 26(1):172–189, 2013.
34. Takanori Isobe, Ryoma Ito, Fukang Liu, Kazuhiko Minematsu, Motoki Nakahashi, Kosei Sakamoto, and Rentaro Shiba. Areion: Highly-efficient permutations and its applications to hash functions for short input. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):115–154, 2023.
35. Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced SHA-1. In *CRYPTO 2012, Proceedings*, volume 7417, pages 367–383.
36. Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - efficient short-input hashing for post-quantum applications. *IACR Trans. Symmetric Cryptol.*, 2016(2):1–29, 2016.
37. Xuejia Lai and James L. Massey. Hash function based on block ciphers. In *EUROCRYPT 1992, Proceedings*, volume 658, pages 55–70. Springer, 1992.
38. Gaëtan Leurent. MD4 is not one-way. In *FSE 2008*, volume 5086, pages 412–428.
39. Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on LowMC. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2022.
40. Stefan Lucks. Attacking triple encryption. In *FSE '98, Proceedings*, volume 1372, pages 239–253. Springer, 1998.
41. Ralph C. Merkle. A certified digital signature. In *CRYPTO 1989, Proceedings*, pages 218–238, 1989.
42. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93*, volume 773, pages 368–378.
43. Lingyue Qin, Jialiang Hua, Xiaoyang Dong, Hailun Yan, and Xiaoyun Wang. Meet-in-the-middle preimage attacks on sponge-based hashing. In *EUROCRYPT 2023, Proceedings, Part IV*, volume 14007, pages 158–188. Springer, 2023.
44. Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In *IWSEC 2018*, volume 11049, pages 227–243.
45. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In *FSE 2011*, pages 378–396. Springer, 2011.

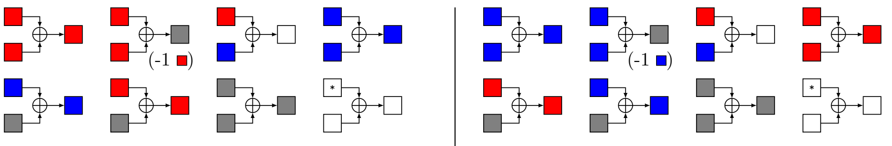
46. Yu Sasaki. Preimage attacks on feistel-sp functions: Impact of omitting the last network twist. In *ACNS 2013, Proceedings*, volume 7954, pages 170–185. Springer, 2013.
47. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In *ASIACRYPT 2008, Proceedings*, volume 5350, pages 253–271. Springer, 2008.
48. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In *EUROCRYPT 2009, Proceedings*, volume 5479, pages 134–152. Springer, 2009.
49. Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on whirlpool: Improved preimage and collision attacks. In *ASIACRYPT 2012, Proceedings*, volume 7658, pages 562–579. Springer, 2012.
50. André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. In *CRYPTO 2022, Proceedings, Part III*, volume 13509, pages 717–747. Springer, 2022.
51. André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. *IACR Cryptol. ePrint Arch.*, page 189, 2022.
52. Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (pseudo) preimage attack on round-reduced grøstl hash function and others. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 127–145. Springer, 2012.
53. Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In *CRYPTO 1989, Proceedings*, volume 435, pages 461–480. Springer, 1989.

Supplementary Material

A MILP models for MitM Attack

We briefly introduce the `MC-Rule` and `XOR-Rule` in [6].

The `XOR-Rule`. For the `XOR` operation in two different directions, the coloring rules of the input and output cells are shown in Figure 26.



(a) For the forward computation (b) For the backward computation

Fig. 26: The `XOR-Rule` in [6], where a “*” means that the cell can be any color

Let $\alpha[i]$, $\beta[i]$ denote the input bytes and $\gamma[i]$ denote the output byte, where $0 \leq i \leq 15$. Let d_i denote the consumed degree of freedom, where $d_i = 1$ describes consuming one DoF to let the output be ◼. The set of rules restrict $(x_i^\alpha, y_i^\alpha, x_i^\beta, y_i^\beta, x_i^\gamma, y_i^\gamma, d_i)$ to a subset of $\mathbb{F}_2^7$, which can be described by a system of linear inequalities by using the convex hull computation.

The `MC-Rule`. The rules of the `MC` operation are also formalized in two different directions. Taking the forward computation as an example, the set of rules is given as following:

1. If there is at least one □ in the input column, all the outputs are □;
2. If there are 🟦 but no □ and no 🟥 in the input column, then all the outputs are 🟦;
3. If all the inputs are ◼, then all the outputs are ◼;
4. If there are 🟥 and 🟦 but no □ in the input column, each output must be 🟦 or □. Moreover, the sum of the numbers of 🟦 and ◼ in the input and output columns must be no more than 3;
5. If there are 🟥 but no □ and no 🟦 in the input column, then each output must be 🟥 or ◼. Moreover, the number of ◼ in the input and output columns must be no more than 3.

Some examples of valid coloring schemes of the `MC-Rule` in the forward computation are shown in Figure 27.

The above rules can also be described by linear inequalities. In [6], they use three 0-1 indicator variables μ, υ, ω for the input column to describe each case:

1. $\mu = 1, \upsilon = 0, \omega = 0$ if and only if case 1 is fulfilled;

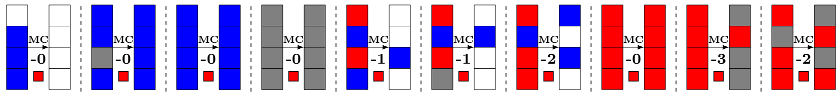


Fig. 27: Some valid coloring schemes for MC-Rule in forward computation in [6]

2. $\mu = 0, \upsilon = 1, \omega = 0$ if and only if case 2 is fulfilled;
3. $\mu = 0, \upsilon = 1, \omega = 1$ if and only if case 3 is fulfilled;
4. $\mu = 0, \upsilon = 0, \omega = 0$ if and only if case 4 is fulfilled;
5. $\mu = 0, \upsilon = 0, \omega = 1$ if and only if case 5 is fulfilled.

Let $(\alpha[0], \alpha[1], \alpha[2], \alpha[3])^T$ and $(\beta[0], \beta[1], \beta[2], \beta[3])^T$ be the input and output columns. Let $\mu = 1$ if and only if there exists $i \in \{0, 1, 2, 3\}$ such that $(x_i^\alpha, y_i^\alpha) = (0, 0)$. Let $\upsilon = 1$ if and only if $x_i^\alpha = 1$ for each $i \in \{0, 1, 2, 3\}$. Let $\omega = 1$ if and only if $y_i^\alpha = 1$ for each $i \in \{0, 1, 2, 3\}$. Then, with the help of μ, υ, ω, the MC-Rule in the forward computation can be a system of inequalities:

$$\begin{cases} \sum_{i=0}^{3} x_i^\alpha - 4\upsilon \geq 0; \\ \sum_{i=0}^{3} x_i^\alpha - \upsilon \leq 3. \end{cases} \quad \Bigg| \quad \begin{cases} \sum_{i=0}^{3} x_i^\beta + 4\mu \leq 4; \\ \sum_{i=0}^{3} y_i^\beta + 4\mu \leq 4; \\ \sum_{i=0}^{3} y_i^\beta - 4\omega = 0; \end{cases} \quad \begin{cases} \sum_{i=0}^{3} (x_i^\alpha + x_i^\beta) - 5\upsilon \leq 3; \\ \sum_{i=0}^{3} (x_i^\alpha + x_i^\beta) - 8\upsilon \geq 0. \end{cases}$$

B The MC-then-XOR-Rule in superposition states of the 11-round attack on Simpira-4

The MC-then-XOR-Rule in superposition states of the MitM attack on 11-round Simpira-4 is given in Figure 28.

C Meet-in-the-Middle Attack on 11-round Simpira-6

For Simpira-6, we find a 11-round preimage attack as shown in Figure 29. The starting states are $\tilde{A}^{(5)}_{\texttt{MC1}}, \tilde{C}^{(5)}_{\texttt{MC1}}, \tilde{E}^{(5)}_{\texttt{MC1}}, \tilde{A}^{(6)}_{\texttt{MC1}}, \tilde{C}^{(6)}_{\texttt{MC1}}$ and $\tilde{E}^{(6)}_{\texttt{MC1}}$ with $\lambda_{\mathcal{B}} = \lambda_{\mathcal{R}} = 24$ initial DoFs of ■ and ■. For the consuming DoFs of backward neutral words, there are 6 linear constraints and 10 nonlinear constraints as shown in Figure 30. And there are 16 nonlinear constraints on forward neutral words as shown in Figure 30. The consuming DoFs of forward and backward neutral words are all 16, i.e. $l_{\mathcal{B}} = l_{\mathcal{R}} = 16$. Then, we have $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 8$ and $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 8$. We find a full-round match through $\tilde{D}^{(2)}$. Similar to Simpira-2, $\texttt{MC}^{-1}(\tilde{C}^{(5)}_{\texttt{MC1}})$ is canceled in both directions. The matching phase is

$$\tilde{A}^{(6)}_{\texttt{SR2}} \oplus \tilde{C}^{(8)}_{\texttt{SR2}} \oplus \tilde{E}^{(10)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B) \oplus \tilde{A}^{(0)}_{\texttt{SR2}} = \tilde{E}^{(4)}_{\texttt{SR2}} \oplus \tilde{C}^{(2)}_{\texttt{SR2}} \tag{12}$$

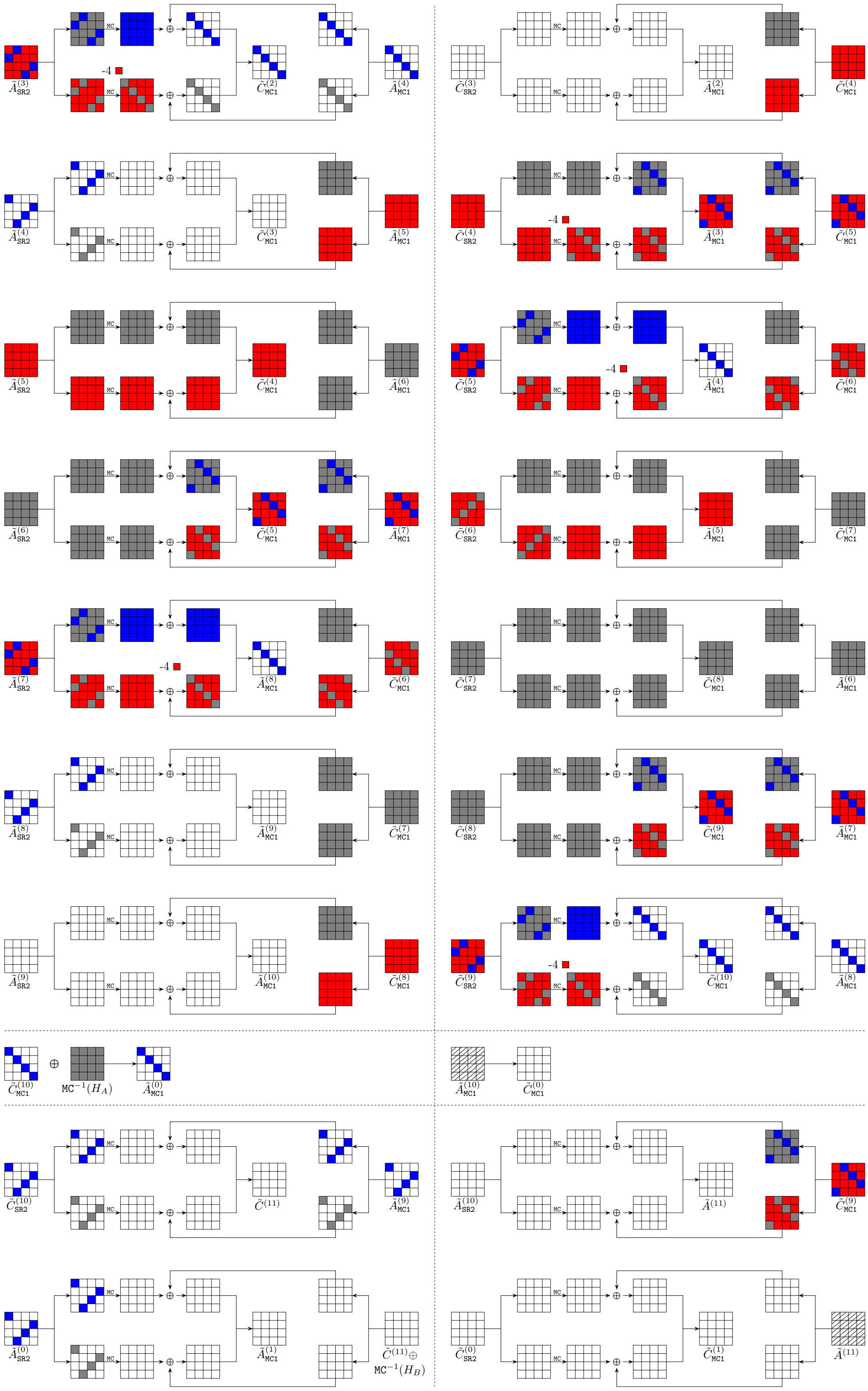


Fig. 28: MitM attack on 11-round `Simpira`-4 in `MC-then-XOR-Rule` representation

as shown in Figure 31, providing 8 bytes degree of match, i.e. DoM = 8.

For the nonlinear constraints, we use the table-based method to build two hash table U and V. Each table needs about 2^{192} time and 2^{192} memory to construct. The detailed attack is proposed in Algorithm 5. According to [24], we only need to traverse about 1 value of the ■ cells in starting states. Hence, the total time to apply Algorithm 5 is about

$$2^{192} + 2^{192} + 2^{8\times\left(\min\{24-16, 24-16, 8\}\right)} \approx 2^{193.6}.$$

The overall memory needed is about 2^{193} to store U and V.

Fig. 29: MitM attack on 11-round Simpira-6

Algorithm 5: Preimage Attack on 11-round `Simpira-6`

Fix the ■ cells in $\tilde{A}^{(5)}_{\texttt{MC1}}$, $\tilde{C}^{(5)}_{\texttt{MC1}}$, $\tilde{E}^{(5)}_{\texttt{MC1}}$, $\tilde{A}^{(6)}_{\texttt{MC1}}$, $\tilde{C}^{(6)}_{\texttt{MC1}}$, $\tilde{E}^{(6)}_{\texttt{MC1}}$

$U \leftarrow [\,]$, $V \leftarrow [\,]$

Traversing $2^{8\cdot\lambda_{\mathcal{R}}} = 2^{192}$ ■ values in $\tilde{C}^{(5)}_{\texttt{MC1}}$, $\tilde{C}^{(6)}_{\texttt{MC1}}$ and $\tilde{E}^{(6)}_{\texttt{MC1}}$, compute the 16 byte constraints (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{128}$), and store the 24 ■ bytes in table $V[c_{\mathcal{R}}]$

Traversing $2^{8\cdot\lambda_{\mathcal{B}}} = 2^{192}$ ■ values in $\tilde{A}^{(5)}_{\texttt{MC1}}$ and $\tilde{C}^{(6)}_{\texttt{MC1}}$, compute the 16 byte constraints (denoted as $c_{\mathcal{B}} \in \mathbb{F}_2^{128}$), and store the 24 ■ bytes in table $U[c_{\mathcal{B}}]$

for $c_{\mathcal{R}} \in \mathbb{F}_2^{16\times 8}$ **do**
- **for** $c_{\mathcal{B}} \in \mathbb{F}_2^{16\times 8}$ **do**
  - $L \leftarrow [\,]$
  - **for** $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ **do**
    - Compute to the ■ cells in $\tilde{A}^{(0)}_{\texttt{SR2}}$, $\tilde{E}^{(10)}_{\texttt{SR2}}$, $\tilde{C}^{(8)}_{\texttt{SR2}}$ and $\tilde{E}^{(4)}_{\texttt{SR2}}$
    - Let the ■ cells in $\tilde{E}^{(4)}_{\texttt{SR2}}$ and $\tilde{E}^{(10)}_{\texttt{SR2}}$ be 0
    - As shown in Figure 31, 8 bytes $End_{\mathcal{R}}$ are derived by

      $$End_{\mathcal{R}} \leftarrow \left(\tilde{A}^{(0)}_{\texttt{SR2}} \oplus \tilde{E}^{(10)}_{\texttt{SR2}} \oplus \tilde{C}^{(8)}_{\texttt{SR2}} \oplus \tilde{E}^{(4)}_{\texttt{SR2}}\right)[0,1,4,7,10,11,13,14]$$

    - $L[End_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
  - **end**
  - **for** $v_{\mathcal{B}} \in V[\mathfrak{c}_{\mathcal{B}}]$ **do**
    - Compute to the ■ cells in $\tilde{C}^{(2)}_{\texttt{SR2}}$, $\tilde{E}^{(4)}_{\texttt{SR2}}$ and $\tilde{E}^{(10)}_{\texttt{SR2}}$
    - Let the ■ cells in $\tilde{E}^{(4)}_{\texttt{SR2}}$ and $\tilde{E}^{(10)}_{\texttt{SR2}}$ be 0
    - As shown in Figure 31, 8 bytes $End_{\mathcal{B}}$ are derived by

      $$End_{\mathcal{B}} \leftarrow \left(\tilde{C}^{(2)}_{\texttt{SR2}} \oplus \tilde{E}^{(4)}_{\texttt{SR2}} \oplus \tilde{E}^{(10)}_{\texttt{SR2}} \oplus \texttt{MC}^{-1}(H_B)\right)[0,1,4,7,10,11,13,14]$$

    - **for** $v_{\mathcal{R}} \in L[End_{\mathcal{B}}]$ **do**
      - Reconstruct the (candidate) message X
      - **if** X *is a preimage* **then**
        - Output X and stop
      - **end**
    - **end**
  - **end**
- **end**

end

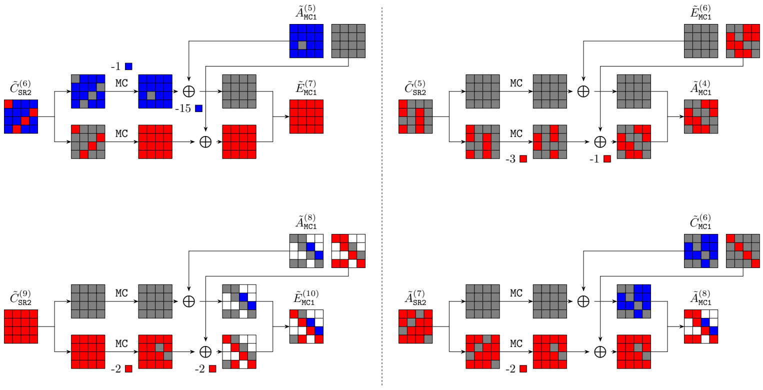


Fig. 30: The consumption of the initial degree in the `MC-Then-XOR` representation of `Simpira-6`

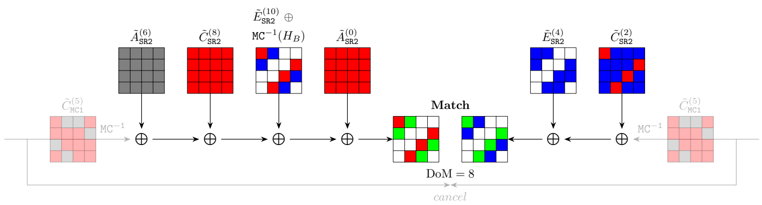


Fig. 31: Full-round match in 11-round `Simpira-6`

D Meet-in-the-Middle Attack on 20-round `lesamnta-LW`

Instead of directly using the output as the filter, we apply a linear transformation on the output and try to find out whether there exist useful filters for its transformation in this attack. We finally find a 20-round MitM characteristic for `Lesamnta-LW` as shown in Figure 32. The initial DoFs for ■ and ■ are $\lambda_{\mathcal{B}} = 4$ and $\lambda_{\mathcal{R}} = 4$, respectively. Along the forward computation path, there are 0 constraints on ■ and 3 constraints on ■, i.e. $l_{\mathcal{B}} = 0$ and $l_{\mathcal{R}} = 3$. Hence, we have $\text{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 4$ and $\text{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 1$. The matching happens on $\texttt{MC}^{-1} \circ \texttt{P}^{-1}(D^{(20)})$ where the 6th byte is fixed as $\alpha \in \mathbb{F}_2^8$, so that a one-byte filter as shown in Figure 33 is obtained. The matching equation is also given in Equation (13).

$$\left(\texttt{MC}^{-1} \circ \texttt{P}^{-1}(D^{(0)}) \oplus C_{\texttt{SB}}^{(0)} \oplus C_{\texttt{SB}}^{(4)} \oplus C_{\texttt{SB}}^{(8)} \oplus C_{\texttt{SB}}^{(12)} \oplus C_{\texttt{SB}}^{(16)}\right)[6] = \texttt{MC}^{-1} \circ \texttt{P}^{-1}(D^{(20)})[6]. \tag{13}$$

The procedure of the MitM collision attack is given in Algorithm 6, where two 128-bit message blocks (m_1, m_2) are needed as shown in Figure 11 to get enough degree of freedom from the message. In Line 19, the size of L_1 is about $2^{32-8} = 2^{24}$. From Line 16 to 24, about $2^8 \times 2^{24} = 2^{32}$ (m_1, m_2, h) are computed, which satisfy the matching Equation (13). To collide in the remaining $256 - 8 = 248$

bits, we need to obtain $2^{(256-8)/2} = 2^{124}$ such 1-byte partial target preimages. To derive 2^{124} such (m_1, m_2, h), 2^4 m_1 and 2^{64} values of $B^{(0)}$ in m_2 are needed. Together with 2^{24} $\mathfrak{c}_{\mathcal{R}}$, $2^{4+64+24+32} = 2^{124}$ (m_1, m_2, h) can be derived. The time to construct table V is 2^{32}. Therefore, the total time complexity is $2^{4+64} \times (2^{32} + 2^{24+32}) = 2^{124}$. The time complexity is better than the generic birthday bound 2^{128}. But it is not better than the designers' security claim against collision attack, which is 2^{120}.

Algorithm 6: Collision Attack on 20-round `Lesamnta-LW`

Fix the third byte in the second column of $\mathtt{MC}^{-1} \circ \mathtt{P}^{-1}(D^{(20)})$ to be $\alpha \in \mathbb{F}_2^8$
for 2^4 *possible values of* m_1 **do**
 for 2^{64} *possible values of* $B^{(0)}$ *in* m_2 **do**
  $V \leftarrow [\,]$
  for $v_{\mathcal{R}} \in \mathbb{F}_2^{8\cdot 4}$ *in* $A^{(0)}[0,1,2,3]$ **do**
   Set the ■ (blue) bytes in $A^{(0)}$ to be 0
   Computer forward to the ■ (red) bytes in $C_{\mathtt{SB}}^{(5)}$, $C_{\mathtt{SB}}^{(9)}$ and $C_{\mathtt{SB}}^{(13)}$
   $c_0 \leftarrow \mathtt{MC}(0\|0\|C_{\mathtt{SB}}^{(5)}[2,3,4,5]\|0\|0)[6]$
   $c_1 \leftarrow \mathtt{MC}(0\|0\|C_{\mathtt{SB}}^{(9)}[2,3,4,5]\|0\|0)[6]$
   $c_2 \leftarrow \mathtt{MC}(0\|0\|C_{\mathtt{SB}}^{(13)}[2,3,4,5]\|0\|0)[6]$
   $\mathfrak{c}_{\mathcal{R}} \leftarrow c_0\|c_1\|c_2$
   $V[\mathfrak{c}_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
  end
  for $\mathfrak{c}_{\mathcal{R}} \in \mathbb{F}_2^{8\cdot 3}$ **do**
   $L_1 \leftarrow [\,]$
   for $2^{8\lambda_{\mathcal{B}}}$ *values* $v_{\mathcal{B}}$ *of* ■ (blue) *bytes in* $A^{(0)}$, $\lambda_{\mathcal{B}} = 4$ **do**
    Compute the ■ (blue) bytes in $C_{\mathtt{SB}}^{(8)}$, $C_{\mathtt{SB}}^{(12)}$ and $C_{\mathtt{SB}}^{(16)}$
    if $(\mathtt{MC}^{-1} \circ \mathtt{P}^{-1}(D^{(0)}) \oplus C_{\mathtt{SB}}^{(0)} \oplus C_{\mathtt{SB}}^{(4)} \oplus C_{\mathtt{SB}}^{(8)} \oplus C_{\mathtt{SB}}^{(12)} \oplus C_{\mathtt{SB}}^{(16)})[6]$ *is equal to* α **then**
     Store $v_{\mathcal{B}}$ in L_1
    end
   end
   for $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ **do**
    Compute the 256-bit target $h = (A^{(20)}, B^{(20)}, C^{(20)}, D^{(20)})$ from the ■ (red) bytes in $v_{\mathcal{R}}$ and the ■ (blue) bytes in L_1 and store the (m_1, m_2, h) in L indexed by h
   end
  end
 end
end
if *the size of* L *is* $2^{(256-8)/2} = 2^{124}$ **then**
 Check L and return (m_1, m_2) and (m_1', m_2') with the same h
end

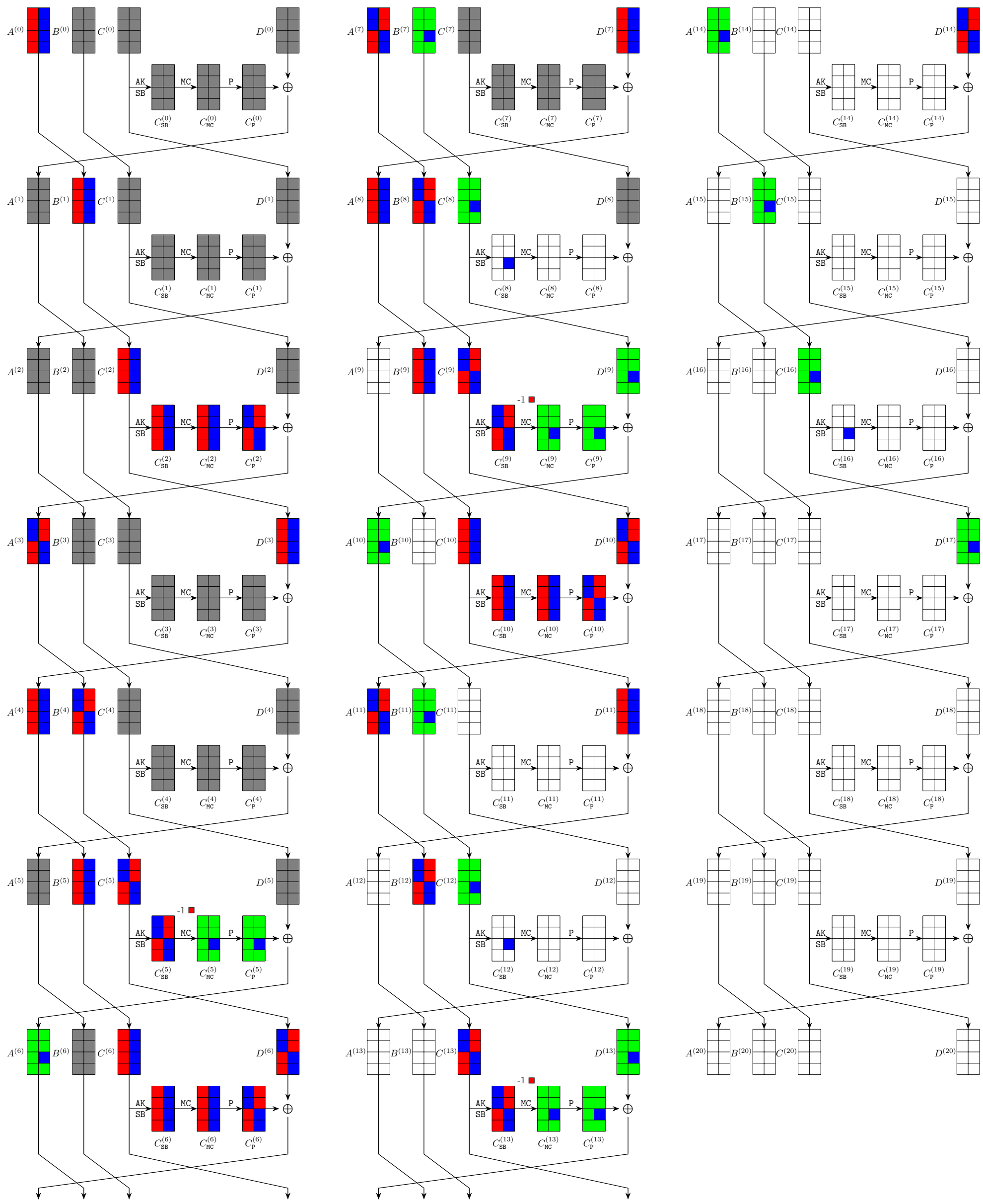


Fig. 32: Collision Attack on 20-round Lesamnta-LW

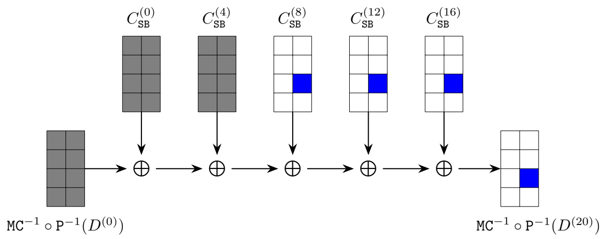


Fig. 33: Matching phase of collision Attack on 20-round `Lesamnta-LW`

E Figures and Algorithms for MitM Attacks on Reduced `Areion`

- The details of the preimage Attack on 5-round `Areion-256` in Figure 25 are described in Algorithm 7.
- A 7-round preimage attack on `Areion-256` is shown in Figure 34. The left part of Figure 34 shows how the attributes are propagated through the forward chunk. The right part provides the details of the backward chunk and how the matching phase is deployed. The detailed attack procedure is shown in Algorithm 8.
- By applying the automatic model on `Areion-512`, we found a 11-round preimage attack as shown in Figure 35 and Figure 36. Figure 35 shows the attributes propagation through the forward chunk. Then, the backward chunk and matching phase are displayed in Figure 36. The detailed attack procedure is shown in Algorithm 9.

F An experiment on 7-round preimage attack on `Simpira-2`

To verify the correctness, we give a 7-round preimage attack on `Simpira`-2 as shown in Figure 37. The starting states are $\tilde{A}_{\mathtt{MC1}}^{(3)}$ and $\tilde{A}_{\mathtt{MC1}}^{(4)}$. The initial DoFs for ■ and ■ are $\lambda_{\mathcal{B}} = 1$, $\lambda_{\mathcal{R}} = 17$, respectively. As shown in Figure 38(a), there are 0 constraints on forward neutral words and 16 constraints on backward neutral words, i.e. $l_{\mathcal{B}} = 0$ and $l_{\mathcal{R}} = 16$. Then, we have $\text{DoF}_{\mathcal{B}} = 1$ and $\text{DoF}_{\mathcal{R}} = 1$. The matching points are $\tilde{A}^{(1)}$ and $\tilde{B}^{(1)}$ and only $\tilde{B}^{(1)}$ can be utilized as shown in Figure 38(b), which can be represented as Equation (14),

$$\tilde{A}_{\mathtt{SR2}}^{(3)} \oplus \tilde{A}_{\mathtt{SR2}}^{(1)} = \tilde{A}_{\mathtt{SR2}}^{(5)} \oplus \mathtt{MC}^{-1}(H_A) \quad (14)$$

where $\mathtt{MC}^{-1}(\tilde{A}_{\mathtt{MC1}}^{(4)})$ can be canceled in both directions. Then 8 bytes for matching are derived in (14) indexed by $[2, 3, 5, 6, 8, 9, 12, 15]$.

Since we only traverse 1 ■ and 1 ■ in each MitM episode, 2 bytes indexed by $[9, 12]$ in Equation (14) are enough to form a filter. Then, there will be about one

Algorithm 7: Preimage Attack on 5-round `Areion-256`

1 **for** $B^{(3)}[9] \in \mathbb{F}_2^8$ **do**
2   **for** $g_b \in \mathbb{F}_2^{96}$ **do**
3     $U \leftarrow [\,]$
4   **for** $v_{\mathcal{B}} \in \mathbb{F}_2^{11\times 8}$ *in* $A^{(3)}[12\text{-}15]$ *and* $B^{(3)}[0\text{-}3, 8, 10, 11]$ **do**
5   Compute $A^{(2,\alpha)}_{\texttt{SR2}}[0, 2, 5, 7, 8, 10, 13, 15]$ from $B^{(3)}$
6   Let $A^{(2,\alpha)}_{\texttt{SR2}}[i] \leftarrow 0$, where $i \notin [0, 2, 5, 7, 8, 10, 13, 15]$
7       $A^{(3)}[0\text{-}11] \leftarrow \texttt{MC}(A^{(2,\alpha)}_{\texttt{SR2}})[0\text{-}11] \oplus g_b$
8   Compute $A^{(3,\alpha)}_{\texttt{SR2}}$ from $A^{(3)}$
9       $c_0 \| c_1 \| c_2 \leftarrow \texttt{MC}(A^{(3,\alpha)}_{\texttt{SR2}})[5, 12, 14]$
10       $\mathfrak{c}_{\mathcal{B}} \leftarrow c_0 \| c_1 \| c_2$
11       $U[\mathfrak{c}_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$
12   **end**
13   **for** $\mathfrak{c}_{\mathcal{B}} \in \mathbb{F}_2^{3\times 8}$ **do**
14       $L \leftarrow [\,]$
15   **for** $v_{\mathcal{B}} \in U[\mathfrak{c}_{\mathcal{B}}]$ **do**
16   Compute $A^{(3)}[0\text{-}11]$ as Line 7, and then compute the 🟦 cells in $A^{(2)}$ and $B^{(1)}$
17   Let the 🟥 cells in $A^{(2)}$ and $B^{(1)}$ be 0, then 6 bytes $End_{\mathcal{B}}$ are derived by

$$End_{\mathcal{B}} \leftarrow \texttt{MC}^{-1}\left(A^{(2)} \oplus B^{(1)}\right)[0, 1, 2, 8, 10, 11]$$

18         $L[End_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$
19   **end**
20   **for** $2^{8\lambda_{\mathcal{R}}}$ *values* $v_{\mathcal{R}}$ *of the* 🟥 *bytes in* $B^{(3)}$, $\lambda_{\mathcal{R}} = 8$ **do**
21   Compute the 🟥 cells in $A^{(2)}$, $B^{(1)}$ and $A^{(1)}_{\texttt{SR2}}$
22         $\mathrm{M}\mathcal{R} = 2$ bytes compatibility of 🟥 cells are tested by

$$\begin{bmatrix} 9\cdot b + d\cdot d, & d\cdot e + b\cdot b \\ e\cdot b + 9\cdot d, & 9\cdot e + d\cdot b \\ b\cdot b + e\cdot d, & e\cdot e + 9\cdot b \end{bmatrix}^T \times \begin{bmatrix} A^{(2)}[12] \oplus B^{(1)}[12] \\ A^{(2)}[13] \oplus B^{(1)}[13] \\ A^{(2)}[14] \oplus B^{(1)}[14] \end{bmatrix} = \begin{bmatrix} b\cdot A^{(1)}_{\texttt{SR2}}[13] \oplus d\cdot A^{(1)}_{\texttt{SR2}}[14] \\ e\cdot A^{(1)}_{\texttt{SR2}}[14] \oplus b\cdot A^{(1)}_{\texttt{SR2}}[15] \end{bmatrix}$$

23   **if** *The compatibility test is passed* **then**
24   Let the 🟦 in $A^{(2)}$ be 0, then 6 bytes $End_{\mathcal{R}}$ are derived by

$$End_{\mathcal{R}} \leftarrow \left(\texttt{MC}^{-1}(A^{(2)}) \oplus A^{(1)}_{\texttt{SR2}}\right)[0, 1, 2, 8, 10, 11]$$

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**
34   **end**
35 **end**

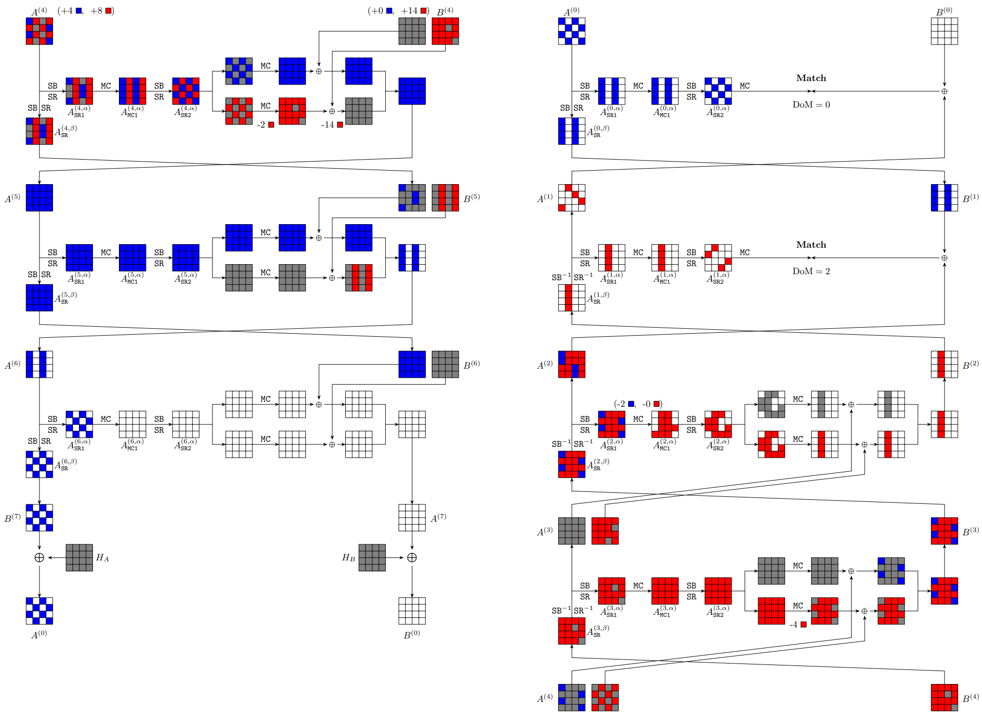


Fig. 34: MitM attack on 7-round `Areion-256`

Algorithm 8: Preimage Attack on 7-round `Areion-256`

1 **for** $A^{(4)}[5,7,8,10]||B^{(4)}[9,15] \in \mathbb{F}_2^{8\times 6}$ **do**
2   **for** $g_r \in \mathbb{F}_2^{112}$ **do**
3     $V \leftarrow [\,]$
4   **for** $v_{\mathcal{R}} \in \mathbb{F}_2^{8\times 8}$ *in* $A^{(4)}[1,3,4,6,9,11,12,14]$ **do**
5   Compute $A^{(4,\alpha)}_{\mathtt{SR2}}[1,3,4,6,9,11,12,14]$
6   Let $A^{(4,\alpha)}_{\mathtt{SR2}}[i] \leftarrow 0$, where $i \notin [1,3,4,6,9,11,12,14]$
7       $c_0\|c_1 \leftarrow \mathtt{MC}(A^{(4,\alpha)}_{\mathtt{SR2}})[9,15]$
8   Let $B^{(4)}[0\text{-}8, 10\text{-}14] \leftarrow \mathtt{MC}(A^{(4,\alpha)}_{\mathtt{SR2}})[0\text{-}8, 10\text{-}14] \oplus g_r$
9   Compute $A^{(3,\alpha)}_{\mathtt{SR2}}$ from $B^{(4)}$
10       $c_2\|c_3\|c_4\|c_5 \leftarrow \mathtt{MC}(A^{(3,\alpha)}_{\mathtt{SR2}})[0,2,13,15]$
11       $\mathfrak{c}_{\mathcal{R}} \leftarrow c_0\|c_1\|c_2\|c_3\|c_4\|c_5$
12       $V[\mathfrak{c}_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
13   **end**
14   **for** $\mathfrak{c}_{\mathcal{R}} \in \mathbb{F}_2^{6\times 8}$ **do**
15   **for** $\mathfrak{c}_{\mathcal{B}} \in \mathbb{F}_2^{2\times 8}$ **do**
16         $L \leftarrow [\,]$
17   Compute $A^{(2,\beta)}_{\mathtt{SR}}[0,2,13,15]$ by $A^{(4)}[0,2,13,15] \oplus \mathfrak{c}_{\mathcal{R}}[2-5]$
18   Through $\mathtt{SR} \circ \mathtt{SB} \circ \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}$, $A^{(2,\beta)}_{\mathtt{SR}} = A^{(2,\alpha)}_{\mathtt{SR2}}$
19   Derive the solution space $\mathcal{S}$ of the ■ cells in $A^{(4)}$ by

$$\begin{cases} 3 \cdot A^{(4)}[0] \oplus A^{(4)}[2] & = \mathfrak{c}_{\mathcal{B}}[0] \\ 3 \cdot A^{(4)}[15] \oplus A^{(4)}[13] & = \mathfrak{c}_{\mathcal{B}}[1] \end{cases}$$

20   **for** $v_{\mathcal{B}} \in \mathcal{S}$ **do**
21   Compute the ■ cells in $A^{(2)}$ and $B^{(1)}$, 2 bytes $End_{\mathcal{B}}$ are derived by
22

$$End_{\mathcal{B}} \leftarrow \begin{bmatrix} 9 \cdot \left(B^{(1)}[0] \oplus A^{(2)}[0]\right) \oplus e \cdot \left(B^{(1)}[1] \oplus A^{(2)}[1]\right) \oplus b \cdot B^{(1)}[2] \oplus d \cdot B^{(1)}[3] \\ b \cdot B^{(1)}[8] \oplus d \cdot B^{(1)}[9] \oplus 9 \cdot \left(B^{(1)}[10] \oplus A^{(2)}[10]\right) \oplus e \cdot \left(B^{(1)}[11] \oplus A^{(2)}[11]\right) \end{bmatrix}$$

23           $L[End_{\mathcal{B}}] \leftarrow v_{\mathcal{B}}$
24   **end**
25   **for** $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ **do**
26   Compute $B^{(4)}$ as Line 8, and then compute the ■ cells in A^2 and $A^{(1,\alpha)}_{\mathtt{SR2}}$ with $\mathfrak{c}_{\mathcal{B}}$, 2 bytes $End_{\mathcal{R}}$ are derived by

$$End_{\mathcal{R}} \leftarrow \begin{bmatrix} b \cdot A^{(2)}[2] \oplus d \cdot A^{(2)}[3] \oplus A^{(1,\alpha)}_{\mathtt{SR2}}[1] \\ b \cdot A^{(2)}[8] \oplus d \cdot A^{(2)}[9] \oplus A^{(1,\alpha)}_{\mathtt{SR2}}[11] \end{bmatrix}$$

27   **for** $v_{\mathcal{B}} \in L[End_{\mathcal{R}}]$ **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**
37 **end**

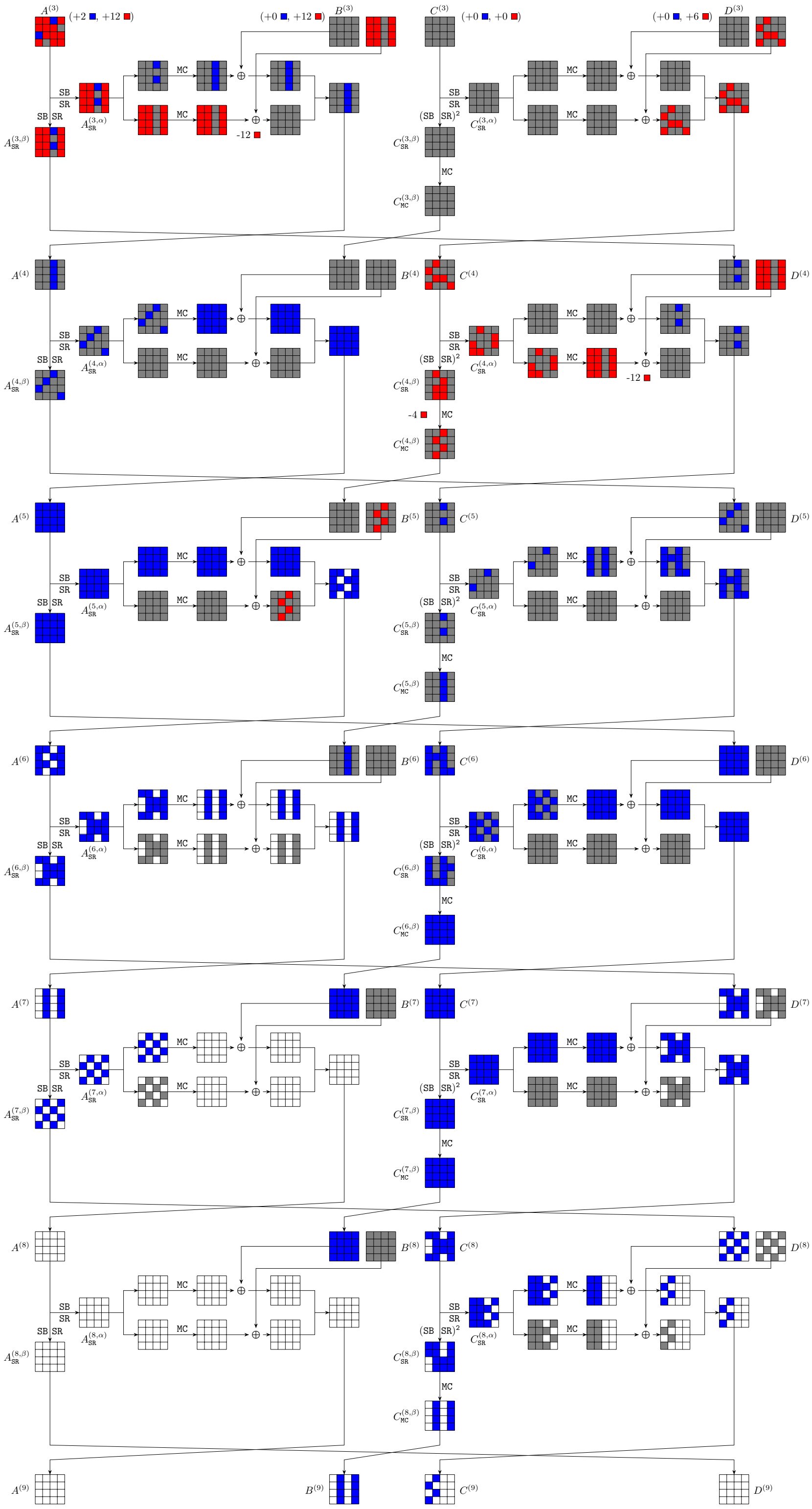


Fig. 35: Forward Chunk of the MitM attack on 11-round `Areion-512`

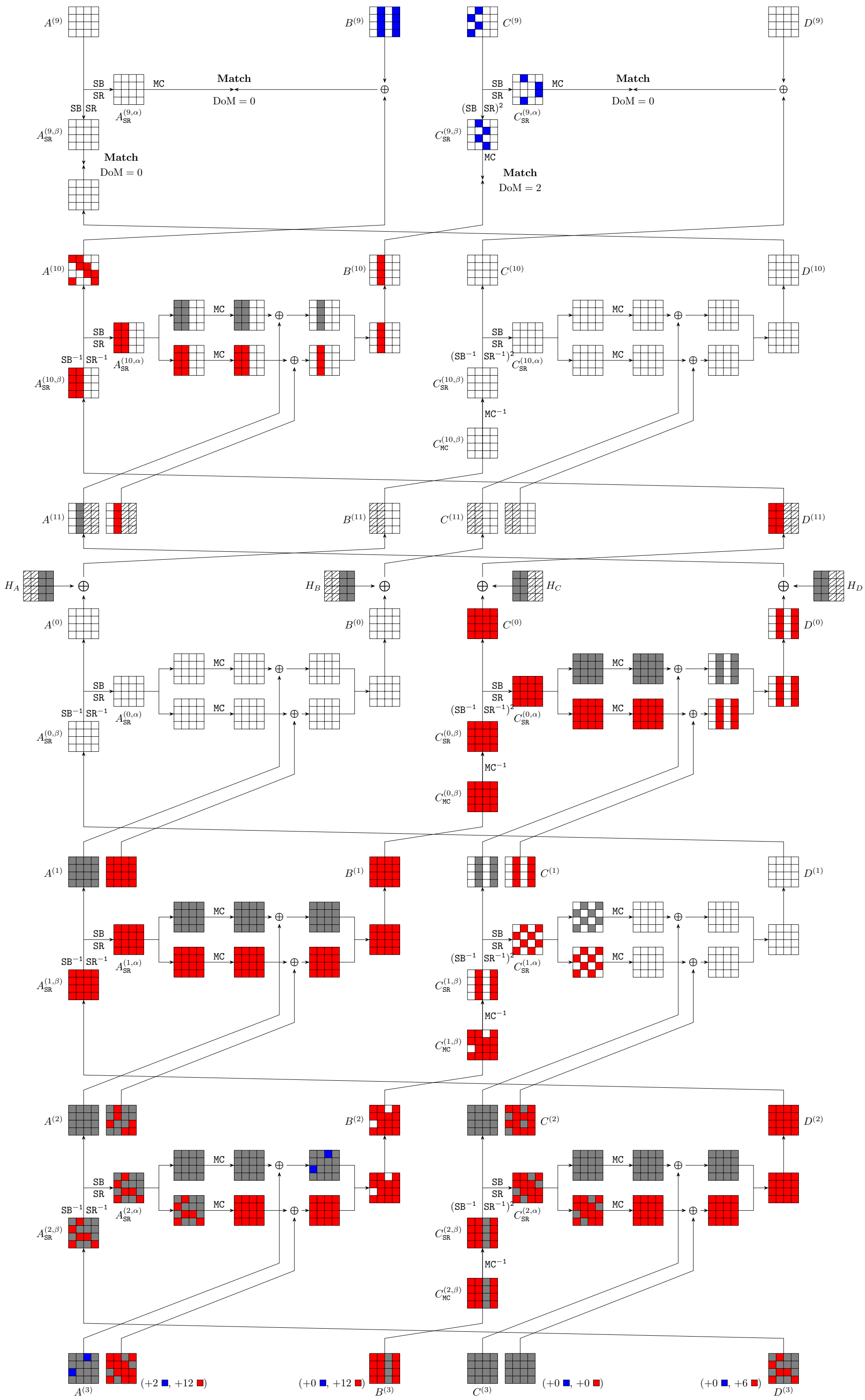


Fig. 36: Backward Chunk and Match Phase of the MitM attack on 11-round `Areion-512`

Algorithm 9: Preimage Attack on 11-round `Areion-512`

for $A^{(3)}[7,13] \| B^{(3)}[8\text{-}11] \| C^{(3)} \| D^{(3)}[0,2,5,7\text{-}9,11\text{-}14] \in \mathbb{G}$ /* $|\mathbb{G}| = 1$ */
do
  for $g_r \in \mathbb{F}_2^{24\times 8}$ **do**
    $V \leftarrow [\ ]$
    for $v_{\mathcal{R}} \in \mathbb{F}_2^{6\times 8}$ *in* $D^{(3)}[1,3,4,6,10,15]$ **do**
      From $C^{(3)}$ and $D^{(3)}$, compute $C_{\texttt{SR}}^{(4,\beta)}$
      $c_0 \| c_1 \| c_2 \| c_3 \leftarrow \texttt{MC}(C_{\texttt{SR}}^{(4,\beta)})[4,6,9,11]$
      $\mathfrak{c}_{\mathcal{R}} \leftarrow c_0 \| c_1 \| c_2 \| c_3$
      $V[\mathfrak{c}_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
    end
    for $\mathfrak{c}_{\mathcal{R}} \in \mathbb{F}_2^{4\times 8}$ **do**
      $L \leftarrow [\ ]$
      for $v_{\mathcal{R}} \in V[\mathfrak{c}_{\mathcal{R}}]$ **do**
        From $C^{(3)}$ and $D^{(3)}$, compute $C_{\texttt{SR}}^{(4,\alpha)}$
        $D^{(4)}[0\text{-}7,12\text{-}15] \leftarrow \texttt{MC}(C_{\texttt{SR}}^{(4,\alpha)})[0\text{-}7,12\text{-}15] \oplus g_r[0\text{-}11]$
        Through $\texttt{SR}^{-1} \circ \texttt{SB}^{-1}$, 12 red ■ cells in $A^{(3)}$ can be derived
        From $A^{(3)}$, compute $A_{\texttt{SR}}^{(3,\alpha)}$, and the 12 red ■ cells in $B^{(3)}$ can be derived by $\texttt{MC}(A_{\texttt{SR}}^{(3,\alpha)})[0\text{-}7,12\text{-}15] \oplus g_r[12\text{-}23]$
        Compute backward to the 4 red ■ cells $B^{(10)}[4\text{-}7]$, 2 bytes $End_{\mathcal{R}}$ are derived by

$$End_{\mathcal{R}} \leftarrow \begin{bmatrix} e \cdot B^{(10)}[4] \oplus b \cdot B^{(10)}[5] \oplus d \cdot B^{(10)}[6] \oplus 9 \cdot B^{(10)}[7] \\ d \cdot B^{(10)}[4] \oplus 9 \cdot B^{(10)}[5] \oplus e \cdot B^{(10)}[6] \oplus b \cdot B^{(10)}[7] \end{bmatrix}$$

        $L[End_{\mathcal{R}}] \leftarrow v_{\mathcal{R}}$
      end
      for $2^{8\lambda_{\mathcal{B}}}$ *values* $v_{\mathcal{B}}$ *of the* blue ■ *bytes in* $A^{(3)}$, $\lambda_{\mathcal{B}} = 2$ **do**
        Compute forward to the 2 blue ■ cells $C_{\texttt{SR}}^{(9,\beta)}[4,6]$ as $End_{\mathcal{B}}$

$$End_{\mathcal{B}} \leftarrow C_{\texttt{SR}}^{(9,\beta)}[4,6]$$

        for $v_{\mathcal{R}} \in L[End_{\mathcal{B}}]$ **do**
          Reconstruct the (candidate) message X
          if X *is a preimage* **then**
            Output X and stop
          end
        end
      end
    end
  end
end

valid (■, ■) pair passing the filter on average. The theoretical time to perform one MitM episode is about $2^8 + 2^8 + 2^{8+8-16} \approx 2^9$. The memory complexity is 2^8 to store $(\tilde{A}^{(3)}_{\texttt{SR2}} \oplus \tilde{A}^{(5)}_{\texttt{SR2}})[9, 12]$. For more visualization, we set the partial target $\texttt{MC}^{-1}(H_A)[9, 12]$ to be 0 in global, then we get 2-byte matching for MitM. After applying $\texttt{MC}^{-1}$ to the input and output of `Simpira-2` derived from the valid starting states, the `XOR` of them should be zero at the 9th and 12th cells in the first branch. To find the 2-byte partial target preimage, exhaustive attack needs 2^{16} to find one partial target preimage.

In our practical experiment, we set the number of MitM episodes to 2^{10}, and we get about 2^{10} partial target preimages, which is very close to our expectation. Some of examples of $\left(\texttt{MC}^{-1}(A_0), \texttt{MC}^{-1}(B_0)\right)$ and $\left(\texttt{MC}^{-1}(A_7), \texttt{MC}^{-1}(B_7)\right)$ are listed in Table 2. The total time to generate the 2^{10} partial target preimage is $2^{10} \times 2^9 = 2^{19}$. We run the experiment on a platform of Interl I9 CPU with 32 GB memory, the running time is about a few minutes. The source codes of the experiment is also given in https://github.com/Hql-code/MitM-Feistel.

Table 2: Preimage examples of 7-round `Simpira-2`

Round	$\left(\texttt{MC}^{-1}(A_0), \texttt{MC}^{-1}(B_0)\right)$	$\left(\texttt{MC}^{-1}(A_7), \texttt{MC}^{-1}(B_7)\right)$
$r = 7$	90d64cee 5dceafc3 c0600c7b 1a4ecd95 cbce2e53 fe452225 e49464ea 31d57501	ce623383 274f3cb0 bf603c92 1a43ae10 ca060030 b89b5a75 4352d9a3 fb5c6f95
	219dd799 af5d9326 87410dcb 5fadba6a 26521560 62354ef3 1cbf6fdb 8db95614	f88c49b3 61cdd8b 741d2f9 5f0f64eb 9703c507 d8aee01e 7e1bc2ae e85d7259
	9cb34f0f ed08af07 8fbb1c33 ca4e3c92 95d3841e 232b28a6 ab1bdb41 8b2bc8db	9dff40f 8195fd0a 2cbb09e0 ca1afff9 ae1b2d86 e7f9c6cb 9076aed d62eb53b
	6f18080e b0935918 6893fef3 93bf08f4 e06c01aa 74d76a05 dafb0f98 4746b05a	8a4a7728 45923513 7593a79d 93f6fd98 2678a9a8 df2d0006 bd8c0429 d5ce8dc5
	79f752e0 1e59a66f 204b02e9 cb95b488 add1cef2 51af4f9a eb5f39f2 7fdc7f6d	bc841e8 e727f743 4d4b5507 cb316893 e762b3b6 467a8df 2b829bef f0bf4704
	bce3ef4b 966eeb2e 30c03e53 e8ac0c7f ccf29d00 25ccbe2 c727d13c 8ead16f0	f8c98bde b306b517 5dc03fb9 e829952c fb17b554 1e6c1dc1 abd27c6d 52864d1c

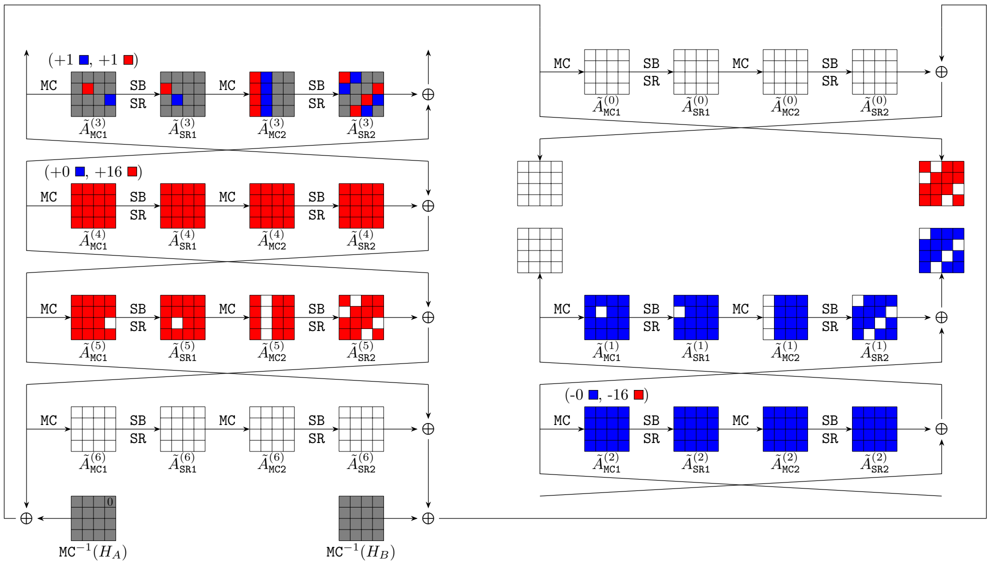
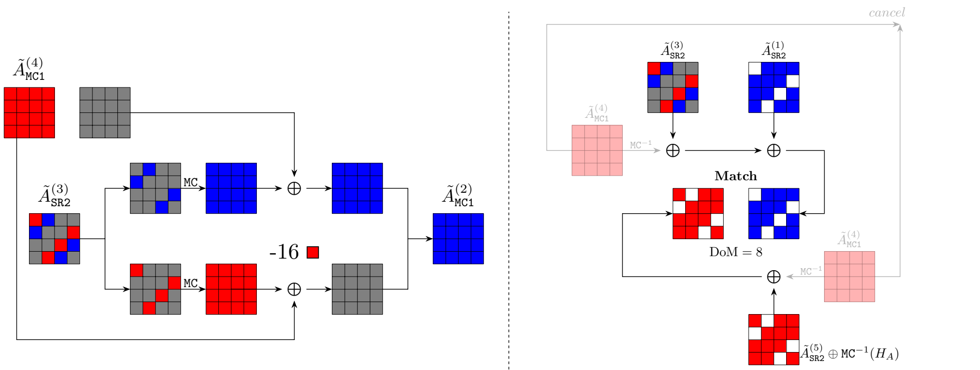


Fig. 37: MitM attack on 7-round `Simpira-2`



(a) The `MC-then-XOR-Rule` of `Simpira-2` in superposition framework

(b) Full-round matching in 7-round `Simpira-2`

Fig. 38: The details of consumption and matching phase in MitM attack on 7-round `Simpira-2`