Meet-in-the-Middle Attacks Revisited: Focusing on Key-recovery and Collision Attacks

No Author Given

No Institute Given

Abstract. At EUROCRYPT 2021, Bao et al. proposed an automatic method for systematically exploring the configuration space of meet-in-the-middle (MITM) preimage attacks. We further extend it into a constraint-based framework for finding exploitable MITM characteristics in the context of key-recovery and collision attacks by taking the subtle peculiarities of both scenarios into account. Moreover, to perform attacks based on MITM characteristics with nonlinear constrained neutral words, which have not been seen before, we present a procedure for deriving the solution spaces of neutral words without solving the corresponding nonlinear equations or increasing the overall time complexities of the attack. We apply our method to concrete symmetric-key primitives, including SKINNY, ForkSkinny, Romulus-H, Saturnin, Grøstl, WHIRLPOOL, and hashing modes with AES-256. As a result, we identify the first 23-round key-recovery attack on SKINNY-n-$3n$ and the first 24-round key-recovery attack on ForkSkinny-n-$3n$ in the single-key model with extremely low memories. Moreover, improved (pseudo) preimage or collision attacks on round-reduced WHIRLPOOL, Grøstl, and hashing modes with AES-256 are obtained. In particular, employing the new representation of the AES key schedule due to Leurent and Pernot (EUROCRYPT 2021), we identify the first preimage attack on 10-round AES-256.

Keywords: Meet-in-the-Middle · Three-subset MITM · Preimage attack · Collision Attack · AES-256 · MILP

1 Introduction

The meet-in-the-middle (MITM) approach is a generic technique for cryptanalysis of symmetric-key primitives, which was first introduced by Diffie and Hellman in 1977 for attacking block ciphers [20]. Many variants of this technique can be found in the literature [23,22,21,26,11]. Its basic idea is best illustrated by performing an MITM attack on a block cipher deliberately made susceptible to this type of attacks. Let $E_K(\cdot)$ be a block cipher whose block size is n-bit such that $C = E_K(P) = F_{K_2}(F_{K_1}(P))$, where $K = K_1||K_2$, and K_1 and K_2 are independent key materials. Therefore, for a given pair of plaintext-ciphertext pair (P, C), the intermediate value V can be computed independently as $F_{K_1}(P)$ and $F^{-1}_{K_2}(C)$ with independent guesses of K_1 and K_2. The correct key guess necessarily satisfies $F_{K_1}(P) = F^{-1}_{K_2}(C)$. Therefore, by searching collisions on the intermediate values computed from P and C, one can reduce the search space from $2^{|K|} = 2^{|K_1|+|K_2|}$ to $2^{|K_1|+|K_2|-n}$ with time complexity $2^{|K_1|} + 2^{|K_2|}$. The remaining key space with $2^{|K_1|+|K_2|-n}$ candidates can be tested against several known plaintext-ciphertext pairs to identify the unique secret key.

However, in practice, it is rare that a target cipher can be clearly separated into two independent halves as the above doubly cascaded F with independent key materials. When a clear separation into two independent chunks is not possible, a variant of the basic MITM strategy (known as three-subset MITM attack) is available. This method was originally proposed by Bogdanov and Rechberger [13], applied to many ciphers [13,55,37,51], and was well summarized by Isobe [35]. Again, let us briefly demonstrate this technique on an ill-designed example with respect the three-subset MITM attack. Let $E_K(\cdot)$ be a block cipher whose block size is n-bit such that it can be divided into three chunks as $C = E_K(P) = H_{K_3||K_2}(G_{K_1||K_2||K_3}(F_{K_1||K_2}(P)))$, where $K = K_1||K_2||K_3$ and K_1, K_2, K_3 are independent. Moreover, some m-bit $(m < n)$ information of a state value inside G can be partially computed along the forward direction from $F_{K_1||K_2}(P)$ without the knowledge of K_3, or computed along the backward direction from $H^{-1}_{K_3||K_2}(C)$ without the knowledge of K_1. The three-subset MITM attack partitions the search space with $2^{|K|} = 2^{|K_1|+|K_2|+|K_3|}$ elements into $2^{|K_2|}$ subspaces of equal size according to the value of $|K_2|$. For each subspace, where the value of $|K_2|$ is fixed, one can perform the basic MITM attack with partial match to reduce the size of the search space from $2^{|K_1|+|K_3|}$ to $2^{|K_1|+|K_3|-m}$ with time complexity $2^{|K_1|} + 2^{|K_3|}$. Under our terminology, which will be introduced in Section 2, one run of the basic version of the MITM attack with a fixed K_2 is called one MITM *episode*. To identify the correct key, $2^{|K_2|}$ episodes has to be performed. Therefore, the overall time complexity can be estimated as $2^{|K_2|}(2^{|K_1|} + 2^{|K_3|} + 2^{|K_1|+|K_3|-m})$. This technique has been applied to many block ciphers [55,51,13,11,35,36].

Although the MITM technique was originally introduced for attacking block ciphers, its development seems to be largely cultivated and promoted in the cryptanalysis of hash functions. In 2008, Sasaki and Aoki successfully achieved preimage attacks on several full versions of HAVAL by combining the MITM approach with the local collision technique [52]. From then on, many MITM preimage attacks together with their enhancements and improvements targeting various hash functions emerged in the literature [4,53,31,60,5,42,3,33,61,1,50,7]. Along the way, several important techniques arise which significantly enhance and enrich the MITM methodology, including the *splice-and-cut* technique [4], the concept of *initial structure* [53], *(indirect-)partial matching* [53,4], *sieve-in-the-middle* [16] and *match-box* technique [28]. Some techniques are formalized as *bicliques* [41,12] and further perceived from differential views [42,27]. These developments in the context of cryptanalysis of hash functions were finally found to be applicable in the MITM attacks on block ciphers. In [62], Wei et al. first applied the splice-and-cut technique to the MITM attacks on block ciphers by connecting the plaintext and ciphertext states with encryption or decryption oracles.

Despite that the principle of how to combine all these techniques in MITM attacks is quite clear, to actually apply them in practice effectively and efficiently is complicated, tedious, and error-prone. Recently, (semi) automatic tools are developed to explore the configuration space of MITM attacks in a more systematic approach. In [51], Sasaki proposed an MILP-based method to search for optimal independent key bits used in the three-subset MITM key-recovery attacks on `GIFT` [6]. However, Sasaki's model is not general enough and the possible positions of neutral words are prefixed. At EUROCRYPT 2021, the MITM preimage attacks on AES-like hashing was throughly modeled as constrained optimization problems which were solved with MILP techniques [7]. This approach outperforms previous work done manually, and many attacks on AES-like hashing [50,63,43] are shown to have room to be further improved. However, this method is described in a way specific to preimage attacks and do not translate directly to MITM-based key-recovery or collision attacks.

Our contribution. We describe the MITM attacks [1] in a unified way as MITM attacks on the so-called *closed computation path*. This view has been long known to our community. Nevertheless, we believe that our treatment is more formal and general. In particular, by introducing some new concepts, we make the description of MITM attacks more expressive and accurate.

Then, we focus our attention on MITM key-recovery and collision attacks on block ciphers and hash functions. We identify the peculiarities specific to these scenarios and show how to deal with them automatically. For the MITM characteristics employed in key-recovery attacks, the degrees of freedom originated from the states in the key schedule data path must not be depleted, while the degrees of freedom originated from the encryption data path must be used up. Also, when searching for candidate configurations for the MITM key-recovery attacks, we should avoid those configurations that lead to attacks requiring the full codebook. We apply our methods to concrete block ciphers `SKINNY` and `ForkSkinny`. and we identify the first 23-round attack on `SKINNY`-n-$3n$ in the single-key model with extremely low memory, penetrating one more round than the designers have expected: *We conclude that meet-in-the-middle attack may work up to at most 22 rounds* [10, Sect. 4.2, page 22]. Interestingly, the characteristics we employed in these attacks impose nontrivial constraints on the neutral words from the key states, which has not been seen before. For collision attacks, they are based on a generalized version of the t-cell partial target preimage attacks, where the words of the target value fulfill t (word-oriented) equations.

Finally, we perform MITM preimage and collision attacks on concrete hash functions (e.g., `Romulus-H` [38], `Saturnin` [15], `WHIRLPOOL` [9], and `Grøstl` [29]). In the attacks on certain hash functions, we encounter some special MITM characteristics where the neutral words are *nonlinearly* constrained. In previous work, the neutral words are linearly constrained and thus the solution space of the neutral words can be obtained efficiently by solving the corresponding system of linear equations. For nonlinear equations, this approach would significantly

[1] We do not consider the Demirci-Selçuk MITM attacks [17,25,19,18] in this paper, which is a quite different technique.

increase the complexities. We propose a technique that is applicable to both the non-linearly and linearly constrained neutral words, overcoming this difficulty without increasing the time complexity of the attacks. Based on this technique, we improve the (pseudo) preimage attacks on round-reduced `Grøstl`-256 and its output transformation by one round. For collision attacks, the first 6-round classical collision attack on `WHIRLPOOL` is provided, breaking a 10-year record for collision attacks on `WHIRLPOOL` in the classical setting. Also, we give the first 6-round collision attack and 8-round collision attack on the output transformations of `Grøstl-256` and `Grøstl-512`, respectively. Interestingly, we notice that all competitive collision attacks on these AES-like hashings are based on the rebound technique [47]. In addition, we offer the first third-party cryptanalysis of `Saturnin-Hash` [15], a second round candidate of the NIST LWC project. A summary of our results on concrete primitives is given in Table 1 and Table 2.

Table 1: Single-key attacks (SK) on `SKINNY`-n-$3n$ and `ForkSkinny`-n-$3n$, where ID and DS-MITM denote impossible differential and Demirci-Selçuk MITM attacks, respectively.

`SKINNY`							
Version	Rounds	Data	Time	Memory	Attack	Setting	Ref.
64-192	22	$2^{47.84}$	$2^{183.97}$	$2^{74.84}$	ID	SK	[59]
	23	2^{52}	2^{188}	2^{8}	MITM	SK	Sect. 4
128-384	22	2^{96}	$2^{382.46}$	$2^{330.99}$	DS-MITM	SK	[58]
	22	$2^{92.22}$	$2^{373.48}$	$2^{147.22}$	ID	SK	[59]
	23	2^{104}	2^{376}	2^{8}	MITM	SK	Sect. 4
`ForkSkinny`							
64-192	24	2^{52}	2^{188}	2^{8}	MITM	SK	Sect. B
128-384	24	2^{104}	2^{376}	2^{8}	MITM	SK	Sect. B
128-256	24	$2^{122.5}$	$2^{124.5}$	$2^{97.5}$	ID	RK	[8]
	26	2^{127}	$2^{250.3}$	2^{160}	ID	RK	[8]

2 A Formal Description of the MITM Technique

We now formally describe the MITM attacks with the notations introduced by Bao et al.'s work [7] in a more unified way. We encourage the readers to carefully go through this section since it not only serves as a recall of Bao et al.'s work, but also introduces some new terminologies that enhance the expressiveness and accuracy of the descriptions of MITM attacks.

Given a computation path that forms a "closed loop", the ultimate goal of the meet-in-the-middle attack is to find a particular value for some intermediate states with which the values for all the states involved in the computation path can be determined, such that the values are compatible with the whole computation path (there are no conflicts between the values due to the involved com-

Table 2: A Summary of the results. Note that we only consider preimage and collision attacks. Distinguishing attacks [39,44,54,14] are not included. Also, note that the complexity of the preimage attack on `Romulus-H` is 2^{248}. This attack does not break 23-round `Romulus-H` since the designers only claim 128-bit security. However, this complexity is better than an exhaustive search, whose complexity is 2^{256}. Similarly, `Saturnin` claims only 224-bit security.

`WHIRLPOOL`						
Target	Attack	Rounds	Time	Memory	Setting	Ref.
Hash function	Collision	4	2^{120}	2^{16}	Classic	[47]
		5	2^{120}	2^{64}	Classic	[30,44]
		6	2^{228}	-	Quantum	[34]
		6	2^{248}	2^{248}	Classic	Sect. 6.2
	Preimage	5	2^{504}	2^{8}	Classic	[50]
		5	$2^{481.5}$	2^{64}		[63]
		6	2^{481}	2^{256}		[56]
Compression function	(Semi-) free-start	5	2^{120}	2^{16}	Classic	[47]
		7	2^{184}	2^{8}		[44]
		8	2^{120}	2^{8}		[56]
`Grøstl-256`						
Hash function	Collision	3	2^{64}	-	Classic	[57]
		5	2^{120}	2^{64}		[48]
	Pseudo preimage	5	$2^{244.8}$	2^{230}	Classic	[63]
		6	2^{252}	2^{251}		Sect. D
Compression function	Semi-free-start	6	2^{112}	2^{64}	Classic	[57]
Output Transformation	Preimage	5	2^{206}	2^{48}	Classic	[63]
		6	2^{240}	2^{152}		Sect. D
	Collision	6	2^{124}	2^{124}	Classic	Sect. F.1
`Grøstl-512`						
Hash function	Collision	5	2^{240}	2^{64}	Quantum	[24]
Compression function		7	2^{152}	2^{56}	Classic	[54]
Output Transformation		8	2^{248}	2^{248}	Classic	Sect. F.2
Hash function	Pseudo preimage	8	$2^{507.3}$	2^{507}	Classic	[63]
`Saturnin-Hash`						
Compression function	Preimage	7	2^{208}	2^{48}	Classic	Sect. E
Hash function		7	2^{232}	2^{48}		
`SKINNY-128-384`, `Romulus-H`, and `AES` hashing mode						
`SKINNY-128-384-DM/MMO`	Preimage	23	2^{120}	2^{8}	Classic	Sect. C
`Romulus-H`		23	2^{248}	2^{8}		Sect. C
`AES-256`		9	2^{120}	2^{8}		[7]
`AES-256`		10	2^{120}	2^{56}		Sect. G
`Romulus-H` compression function	Free-start	23	2^{124}	2^{124}		Sect. C

putation). Let us descend from the abstract highland and consider the closed computation path shown in Figure 1. The upper segment of the computation path constitutes an iterative block cipher with an iterative key schedule, and we

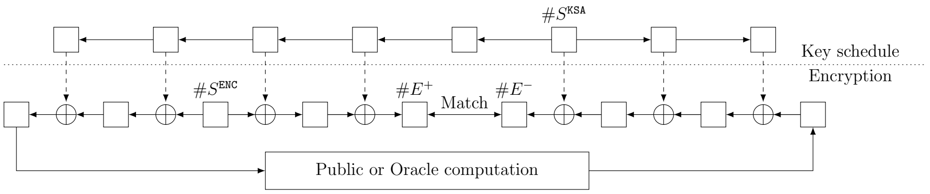


Fig. 1: A high-level overview of the MITM attacks

assume that the states involved in the encryption data path and key schedule data path contains n and $\bar{n}$ w-bit words respectively, which are typically visualized as rectangles with n and $\bar{n}$ cells, respectively. The lower segment of the computation path can be arbitrary. In our context, it can be an oracle of the block cipher appearing in the upper segment of the computation path when we consider an MITM key-recovery attack, or a simple exclusive-or of a given target value when we consider preimage attacks. Before we can perform an MITM attack on the computation path, a configuration or an MITM *characteristic* has to be identified.

MITM Characteristics and Their Visualization. The MITM attack entails the identification of several special states: the starting state $\#S^{\texttt{ENC}}$ (see Figure 1) in the encryption data path, the starting state $\#S^{\texttt{KSA}}$ in the key schedule data path, the ending state $\#E^{+}$ for the forward computation (the computation path starting from $(\#S^{\texttt{ENC}}, \#S^{\texttt{KSA}})$ leading to $\#E^{+}$), and the ending state $\#E^{-}$ for the backward computation (the computation path starting from $(\#S^{\texttt{ENC}}, \#S^{\texttt{KSA}})$ leading to $\#E^{-}$). Moreover, the cells of $(\#S^{\texttt{ENC}}, \#S^{\texttt{KSA}})$ are partitioned into different subsets with different meanings. Let $\mathcal{B}^{\texttt{ENC}}$, $\mathcal{B}^{\texttt{KSA}}$, $\mathcal{R}^{\texttt{ENC}}$, $\mathcal{R}^{\texttt{KSA}}$, $\mathcal{M}^{+}$, and $\mathcal{M}^{-}$ be some ordered subsets of $\mathcal{N} = \{0, 1, \cdots, n-1\}$ or $\overline{\mathcal{N}} = \{0, 1, \cdots, \bar{n}-1\}$ such that $\mathcal{B}^{\texttt{ENC}} \cap \mathcal{R}^{\texttt{ENC}} = \emptyset$, $\mathcal{B}^{\texttt{KSA}} \cap \mathcal{R}^{\texttt{KSA}} = \emptyset$, $\mathcal{G}^{\texttt{ENC}} = \mathcal{N} - \mathcal{B}^{\texttt{ENC}} \cup \mathcal{R}^{\texttt{ENC}}$ and $\mathcal{G}^{\texttt{KSA}} = \overline{\mathcal{N}} - \mathcal{B}^{\texttt{KSA}} \cup \mathcal{R}^{\texttt{KSA}}$. We will use these index sets to reference the cells of the states. For example, for a 16-cell state $\#S$ and $\mathcal{M}^{+} = [0, 1, 3]$, we have $\#S[\mathcal{M}^{+}] = \#S[0, 1, 3] = (\#S[0], \#S[1], \#S[3])$.

The cells $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$, visualized as 🟦 cells, are called neutral words of the forward computation, and the cells $(\#S^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$, visualized as 🟥 cells, are called neutral words of the backward computation. The initial degrees of freedom for the forward and backward computation are defined as $\lambda^{+} = |\mathcal{B}^{\texttt{ENC}}| + |\mathcal{B}^{\texttt{KSA}}|$ and $\lambda^{-} = |\mathcal{R}^{\texttt{ENC}}| + |\mathcal{R}^{\texttt{KSA}}|$ respectively, that is, the numbers of 🟦 cells and 🟥 cells in the starting states. In addition, $E^{+}[\mathcal{M}^{+}]$ are visualized as 🟦 cells, and $E^{-}[\mathcal{M}^{-}]$ are visualized as 🟥 cells. Finally, $\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$ and $\#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]$ are visualized as ■ cells.

We then define a sequence of l^{+} functions $\boldsymbol{\pi}^{+} = (\pi_1^{+}, \cdots, \pi_{l^{+}}^{+})$ whose values can be computed with the knowledge of the ■ cells $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ and 🟦 cells $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ in the starting states, where

$$\pi_i^{+} : \mathbb{F}_2^{w \cdot (|\mathcal{G}^{\texttt{ENC}}| + |\mathcal{G}^{\texttt{KSA}}| + |\mathcal{B}^{\texttt{ENC}}| + |\mathcal{B}^{\texttt{KSA}}|)} \to \mathbb{F}_2^{w}$$

is a function mapping $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ to a w-bit word $\pi_i^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$. Similarly, we define a sequence of l^- functions $\boldsymbol{\pi}^- = (\pi_1^-, \cdots, \pi_{l^-}^-)$ whose values can be computed with the knowledge of the ■ cells $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ and 🟥 cells $(\#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$. $\boldsymbol{\pi}^+$ and $\boldsymbol{\pi}^-$ will be used to represent certain constraints on the neutral words of the forward and backward computations, respectively. A valid MITM characteristic satisfies the following property.

Property 1. For any fixed $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w\cdot l^+}$ and $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w\cdot l^-}$, when the cells $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ are fixed to an arbitrary constant, and the neutral words for the forward computation and backward computation paths fulfill the following systems of equations:

$$\begin{cases} \pi_1^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) = a_1 \\ \pi_2^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) = a_2 \\ \qquad\qquad\cdots\quad\cdots \\ \pi_{l^+}^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) = a_{l^+} \end{cases} \tag{1}$$

and

$$\begin{cases} \pi_1^-(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) = b_1 \\ \pi_2^-(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) = b_2 \\ \qquad\qquad\cdots\quad\cdots \\ \pi_{l^-}^-(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) = b_{l^-} \end{cases} \tag{2}$$

respectively, then the values of the cells $\#E^+[\mathcal{M}^+]$ can be derived from the starting states $(\#S^{\mathtt{ENC}}, \#S^{\mathtt{KSA}})$ along the forward computation path without the knowledge of the neutral words for the backward computation, and the values of the cells $\#E^-[\mathcal{M}^-]$ can be derived from the starting states $(\#S^{\mathtt{ENC}}, \#S^{\mathtt{KSA}})$ along the backward computation path without the knowledge of the neutral words for the forward computation. In short, computations for deriving $\#E^-[\mathcal{M}^+]$ and $\#E^-[\mathcal{M}^-]$ can be carried out independently.

Let us talk more about Property 1. For any given $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ and $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+})$, the solution space of $(\#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ induced by Equation (1) is denoted by

$$\mathbb{B}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^+).$$

Since there are $\lambda^+ = |\mathcal{B}^{\mathtt{ENC}}| + |\mathcal{B}^{\mathtt{KSA}}|$ w-bit variables and l^+ equations, we expect $2^{w\cdot(\lambda^+ - l^+)}$ solutions, and we call $\mathrm{DoF}^+ = \lambda^+ - l^+$ the *degrees of freedom for the forward computation*. Similarly, the solution space of $(\#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$ induced by Equation (2) is denoted by $\mathbb{R}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^-)$. Since there are $\lambda^- = |\mathcal{R}^{\mathtt{ENC}}| + |\mathcal{R}^{\mathtt{KSA}}|$ w-bit variables and l^- equations, we expect $2^{w\cdot(\lambda^- - l^-)}$ solutions, and we call $\mathrm{DoF}^- = \lambda^- - l^-$ the *degrees of freedom for the backward computation.*

Let F^+ be the function computing $\#E^+[\mathcal{M}^+]$ from $(\#S^{\mathtt{ENC}}, \#S^{\mathtt{KSA}})$, that is, $\#E^+[\mathcal{M}^+]$ can be computed as

$$F^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}],\ \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}],\ \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]),$$

and similarly, $\#E^-[\mathcal{M}^-]$ can be computed as

$$F^-(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]).$$

Property 1 implies that

$$F^-(\alpha, x, \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) = F^-(\alpha, y, \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$$

for any given $x, y \in \mathbb{B}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^+)$ and $\alpha \in \mathbb{F}_2^{|\mathcal{G}^{\mathtt{ENC}}|+|\mathcal{G}^{\mathtt{KSA}}|}$. Similarly, for any $u, v \in \mathbb{R}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^-)$, we have

$$F^+(\alpha, \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], u) = F^+(\alpha, \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], v).$$

Consequently, for any given $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}]) = \alpha$, and $\mathfrak{c}^+$, and $\mathfrak{c}^-$, we can perform a matching process given in Algorithm 1.

Algorithm 1: One MITM episode

```
Fix $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ to a constant $\alpha$
Fix $\mathfrak{c}^+$, and $\mathfrak{c}^-$ to some constants
Fix $x^*$ to be an element in $\mathbb{B}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^+)$
Fix $u^*$ to be an element in $\mathbb{R}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^-)$
$L \leftarrow [\,]$
for all $(\#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) \in \mathbb{B}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^+)$ do
    $E^+[\mathcal{M}^+] \leftarrow F^+(\alpha, \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], u^*)$
    Insert $E^+[\mathcal{M}^+]$ into $L$
for all $(\#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) \in \mathbb{R}(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathfrak{c}^-)$ do
    $E^-[\mathcal{M}^-] \leftarrow F^-(\alpha, x^*, \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$
    for $E^+[\mathcal{M}^+]$ in $L$ matching with $E^-[\mathcal{M}^-]$ do
        Test for full match between $E^+[\mathcal{M}^+]$ and $E^-[\mathcal{M}^-]$
```

In real MITM attacks, Algorithm 1 will be performed multiple times for many different α, $\mathfrak{c}^+$, and $\mathfrak{c}^-$, each time is called one MITM *episode*. Variables that remain constant within each episode are called *episodic constants*, and variables remain constant in the whole life cycle of an attack (remaining constant across different episodes) are called *global constants*. Thus *global constants* are always *episodic constants*. The ■ cells used in [7] and this work capture the episodic constants, whose values can change across different episodes.

Within each episode, $(2^w)^{\mathrm{DoF}^+}$ times of forward computation are carried out, and $(2^w)^{\mathrm{DoF}^-}$ times of backward computations are carried out, which are referred to as *forward threads* and *backward threads*. Each forward thread and backward thread *within the same episode* gives a pair of values for $(\#E^+[\mathcal{M}^+], \#E^-[\mathcal{M}^-])$ which are computed along the forward and backward computation paths from a *common value* of the starting states $(\#S^{\mathtt{ENC}}, \#S^{\mathtt{KSA}})$, and thus can be tested for match according to the computation connecting $\#E^+$ and $\#E^-$ in the closed loop. Note that testing pairs computed from different values of the starting point (e.g., pairs formed from different episodes) is meaningless. In each episode, we have $(2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^-}$ *paired threads*. If the computation connecting $\#E^+[\mathcal{M}^+]$ and $\#E^-[\mathcal{M}^-]$ forms an m-cell filter, then there are about $(2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^- - m}$ paired threads will pass the filter and be tested for a full match. We call $\mathrm{DoM} = m$ the degrees of match or the strength of the filter. Finally, we emphasize again that the MITM procedure given in Algorithm 1 is performed for some fixed $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+, \mathfrak{c}^-)$, and we say $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+, \mathfrak{c}^-)$ defines the *context* of the MITM episode.

Automatic Search for MITM Characteristics. For a given closed computation path shown in Figure 1, a configuration of the states $\#S^{\texttt{ENC}}$, $\#S^{\texttt{KSA}}$, $\#E^+$, $\#E^-$, and the parameters $\mathcal{B}^{\texttt{ENC}}$, $\mathcal{B}^{\texttt{KSA}}$, $\mathcal{R}^{\texttt{ENC}}$, $\mathcal{R}^{\texttt{KSA}}$, $\mathcal{M}^+$, $\mathcal{M}^-$, DoF^+, DoF^-, $\boldsymbol{\pi}^+$, $\boldsymbol{\pi}^-$, and DoM satisfying Property 1 is called an MITM characteristic. At EUROCRYPT 2021, Bao et al. presented an MILP-based method for finding optimal MITM characteristics for preimage attacks, and we refer the reader to [7] for more details. Here, we only mention that an MILP characteristic can be visualized with the following coloring scheme on the states of the closed computation path and the ith cell of a state $\#S$ is encoded with a pair of 0-1 variables $(x_i^{\#S}, y_i^{\#S})$ in the MILP models according to the following rule:

- ■ `Gray` (`G`), $(x_i^{\#S}, y_i^{\#S}) = (1, 1)$: known *episodic* constants.
- 🟥 `Red` (`R`), $(x_i^{\#S}, y_i^{\#S}) = (0, 1)$: neutral words for backward computation or dependent on ■ cells and neutral words for backward computation.
- 🟦 `Blue` (`B`), $(x_i^{\#S}, y_i^{\#S}) = (1, 0)$: neutral words for forward computation or dependent on ■ cells and neutral words for forward computation.
- □ `White` (`W`), $(x_i^{\#S}, y_i^{\#S}) = (0, 0)$: dependent on 🟦 cells in the backward computation or dependent on 🟥 cells in the forward computation.

3 Automatic MITM Key-recovery Attacks

We describe the MITM key-recovery attack on a block cipher based on Figure 1 with the lower segment being an encryption or decryption oracle. Before going any further, we introduce some new notations. The initial degrees of freedom from the encryption and key schedule data paths for the forward computation are defined as $\lambda^+_{\texttt{ENC}} = |\mathcal{B}^{\texttt{ENC}}|$ and $\lambda^+_{\texttt{KSA}} = |\mathcal{B}^{\texttt{KSA}}|$, respectively. Similarly, The initial degrees of freedom from the encryption and key schedule data paths for the backward computation are defined as $\lambda^-_{\texttt{ENC}} = |\mathcal{R}^{\texttt{ENC}}|$ and $\lambda^-_{\texttt{KSA}} = |\mathcal{R}^{\texttt{KSA}}|$, respectively. Under these notations, we have $\lambda^+ = \lambda^+_{\texttt{ENC}} + \lambda^+_{\texttt{KSA}}$ and $\lambda^- = \lambda^-_{\texttt{ENC}} + \lambda^-_{\texttt{KSA}}$.

For an MITM characteristic, we say that the degrees of freedom from the encryption data path for the forward computation is used up if for any given $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$, we partition the solution space

$$\mathbb{B}(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$$

of $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ due to Equation (1) into subspaces according to the value of $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]$, then each space contains exactly one element. That is, the values of the 🟦 cells in $\#S^{\texttt{ENC}}$ can be fully determined by the 🟦 cells in $\#S^{\texttt{KSA}}$ for a given $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$. Similarly, we say that the degrees of freedom from the encryption data path for the backward computation is used up if the values of the 🟥 cells in $\#S^{\texttt{ENC}}$ can be fully determined by the 🟥 cells in $\#S^{\texttt{KSA}}$ for a given $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^-)$.

Now, Let us recall from Section 2 that the goal of the MITM attack is to find a particular value for some intermediate states in the closed computation path shown in Figure 1 with which the values for all the states involved in the computation path can be determined, such that the values derived are compatible with the whole computation path. Specifically, in the context of MITM key-recovery attacks, our goal can be formulated as follows.

Goal 1. *Identify a value K for the key register hosting the master key, and a value for one full state in the encryption data path, with which we can derive the values of all states involved. We require that the values for all states are compatible and K equals to the secret key hiding in the oracle.*

The above goal indicates that in the MITM key-recovery attack, the full key space must be (implicitly) tested, since a compatible assignment of values to the states is not enough (unlike MITM preimage attacks), and we must identify the unique secret key. Secondly, in the key-recovery attack, we prefer not to exhaust the full codebook of the targeted cipher. These particularities result in the following requirements for the MITM characteristic:

I. The degrees of freedom for the forward computation or backward computation from $\#S^{\texttt{KSA}}$ cannot be depleted (i.e., $\text{DoF}^+ > 0$ and $\text{DoF}^- > 0$), while the degrees of freedom for *both* the forward computation and backward computation from $\#S^{\texttt{ENC}}$ should be used up.

II. In the MITM characteristic, we require that there is at least one ■ cell (*episodic constant*) in the plaintext state, which will be set to *global constant* in the actual attack to avoid using the full codebook.

To ensure (I), we require the corresponding systems of equations of the MITM characteristic given in Equation (1) and (2) to satisfy the following conditions. For Equation (1), there are $l^+_{\texttt{KSA}}$ equations (without loss of generality, we assume these are the first $l^+_{\texttt{KSA}}$ equations) do not involve $\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$ and $S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}]$. The remaining $l^+ - l^+_{\texttt{KSA}}$ equations are used to exhaust the degrees of freedom from the encryption data path, and thus $|\lambda^+_{\texttt{ENC}}| = |\mathcal{B}^{\texttt{ENC}}| = l^+ - l^+_{\texttt{KSA}}$. Under this, we have $\text{DoF}^+ = \lambda^+_{\texttt{KSA}} - l^+_{\texttt{KSA}}$. In addition, for each constant $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$, and each solution for $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]$ of the first $l^+_{\texttt{KSA}}$ equations, we can derive one and only one solution for $\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}]$ by solving the remaining equations. For Equation (2), there are $l^-_{\texttt{KSA}}$ equations (without loss of generality, we assume these are the first $l^-_{\texttt{KSA}}$ equations) do not involve $\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$ and $S^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}]$. The remaining $l^- - l^-_{\texttt{KSA}}$ equations are used to exhaust the degrees of freedom from the encryption data path, and thus $|\lambda^-_{\texttt{ENC}}| = |\mathcal{R}^{\texttt{ENC}}| = l^- - l^-_{\texttt{KSA}}$. Under this, we have $\text{DoF}^- = \lambda^-_{\texttt{KSA}} - l^-_{\texttt{KSA}}$. In addition, for each constant $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^-)$, and each solution for $\#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]$ of the first $l^-_{\texttt{KSA}}$ equations, we can derive one and only one solution for $\#S^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}]$ by solving the remaining equations.

Requirement (I) may be less obvious than (II), and we will explain it by looking into the algorithmic framework given in Algorithm 2. But before we go into the details, we emphasize that due to these peculiarities, almost all MITM characteristics found by the the method presented in [7] are useless in the context of key-recovery attacks.

Algorithm 2: The MITM key-recovery attack on block ciphers

```
1  Set $|\#S^{\texttt{ENC}}|$ *independent* gray cells to constants, which should contain all the
     gray cells in the plaintext state
2  Collecting a structure of plaintext-ciphertext pairs and store them in a table
     $H$, which traverses the non-constant cells in the plaintext
3  for $\#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}] \in \mathbb{F}_2^{w\cdot|\mathcal{G}^{\texttt{KSA}}|}$ do
4      for $\mathfrak{c}^{+}_{\texttt{KSA}} = (a_1, \cdots, a_{l^{+}_{\texttt{KSA}}}) \in \mathbb{F}_2^{w\cdot l^{+}_{\texttt{KSA}}}$ do
5          for $\mathfrak{c}^{-}_{\texttt{KSA}} = (b_1, \cdots, b_{l^{-}_{\texttt{KSA}}}) \in \mathbb{F}_2^{w\cdot l^{-}_{\texttt{KSA}}}$ do
6              Derive the the value of $\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$
7              $L \leftarrow [\,]$
8              for $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}] \in \mathbb{B}^{\texttt{KSA}}(\#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^{+}_{\texttt{KSA}})$ do
9                  Derive the the value of $\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}]$ and compute $E^{+}[\mathcal{M}^{+}]$
                     along the forward computation path
10                 Insert $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]$ into $L$ indexed by $E^{+}[\mathcal{M}^{+}]$
11             for $\#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}] \in \mathbb{R}^{\texttt{KSA}}(\#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^{-}_{\texttt{KSA}})$ do
12                 Derive the the value of $\#S^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}]$ and Compute $E^{-}[\mathcal{M}^{-}]$
                     along the backward computation path by accessing $H$
13                 for $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}] \in L[E^{-}[\mathcal{M}^{-}]]$ do
14                     Reconstruct the (guessed) key value $K'$ from $\#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]$,
                         $\#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]$, and $\#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]$
15                     Test $K'$ against several plaintext-ciphertext pairs
```

From now on, we use $|\#S|$ denote the number of cells in a state $\#S$. In Line 1 of Algorithm 2, we set $|\#S^{\texttt{ENC}}|$ gray cells, including all the gray cells in the plaintext state to global constants, where $|\#S^{\texttt{ENC}}|$ denotes the number of cells in $\#S^{\texttt{ENC}}$. Since the gray cells in the plaintext states are set to global constant, the attack will not use the full codebook. These $|\#S^{\texttt{ENC}}|$ gray cells are not necessarily within one single state along the computation path. Instead, they can be distributed over multiple states. Moreover, we require that the values of these cells can be set independently to *arbitrary* values without leading to a conflict along the computation path (excluding the computations connecting the ending states). When these constants are set, for any given key, we can derive the values of all the states (including $\#S^{\texttt{ENC}}$), along the computation path (excluding the computation connecting the ending states), which indicates that if the degrees of freedom of $\#S^{\mathtt{ENC}}$ are not exhausted, this constant setting process may lead to conflicts, which is equivalent to setting more than $|\#S^{\mathtt{ENC}}|$ cells of $\#S^{\mathtt{ENC}}$ to constants. Then, each MITM episode is performed within the context defined by the outer loops surrounding the code segment from Line 8 to Line 15.

Complexity Analysis. In Line 2 of Algorithm 2, suppose there are ε gray cells in the plaintext state, then the data complexity $(2^w)^{n-\varepsilon}$. Suppose the states in the encryption data and key schedule data paths contains n and $\bar{n}$ cells, respectively, and the matching part forms an m-cell filter. According Algorithm 2, there are $(2^w)^{\bar{n}-\lambda^+_{\mathtt{KSA}}-\lambda^-_{\mathtt{KSA}}}\cdot(2^w)^{l^+_{\mathtt{KSA}}}\cdot(2^w)^{l^-_{\mathtt{KSA}}} = (2^w)^{\bar{n}-(\mathrm{DoF}^++\mathrm{DoF}^-)}$ MITM episodes, and in each episode $(2^w)^{\mathrm{DoF}^++\mathrm{DoF}^-}$ different keys are tested, where $(2^w)^{\mathrm{DoF}^++\mathrm{DoF}^--m}$ of them will pass the m-cell filter. Therefore, the overall time complexity can be estimated as $(2^w)^{\bar{n}-\mathrm{DoF}^+-\mathrm{DoF}^+}((2^w)^{\mathrm{DoF}^+} + (2^w)^{\mathrm{DoF}^-} + (2^w)^{\mathrm{DoF}^++\mathrm{DoF}^--m})$, which is approximately

$$(2^w)^{\bar{n}-\min\{\mathrm{DoF}^+,\mathrm{DoF}^-,m\}}. \tag{3}$$

4 MITM Attacks on SKINNY and ForkSkinny

SKINNY is a family of lightweight block ciphers designed by Beierle et al. [10] based on the TWEAKEY framework [40]. In this section, we apply our method to SKINNY-n-$3n$ (The version with an n-bit block size, a $3n$-bit key, and a 0-bit tweak) with $n \in \{64, 128\}$. The overall structure of SKINNY-n-$3n$ and its round function are given in Figure 2.

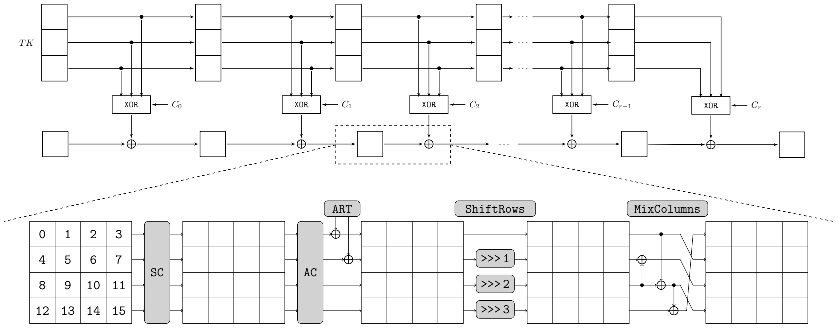


Fig. 2: The hight-level structure of SKINNY-n-$3n$ and its round function (Thanks to https://www.iacr.org/authors/tikz/).

The internal state is viewed as a 4 × 4 square with 16 cells. In each round, the state is updated with five operations: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC). The key register is arranged into three 4 × 4 squares denoted as TK_1, TK_2, and TK_3 respectively. Note that the in each round only the first two rows of the internal state are affected by ART, and the MC operation is non-MDS and thus quite different from the AES-like structures analyzed in [7]. Specifically, we have

$$\texttt{MC}\begin{pmatrix}a\\b\\c\\d\end{pmatrix}=\begin{pmatrix}a\oplus c\oplus d\\a\\b\oplus c\\a\oplus c\end{pmatrix}\quad\text{and}\quad\texttt{MC}^{-1}\begin{pmatrix}\alpha\\\beta\\\gamma\\\delta\end{pmatrix}=\begin{pmatrix}\beta\\\beta\oplus\gamma\oplus\delta\\\beta\oplus\delta\\\alpha\oplus\delta\end{pmatrix}. \tag{4}$$

4.1 Programming the MITM Attacks on SKINNY-n-3n with MILP

Based on the analysis of Section 3, we show how to build the MILP model for finding MITM characteristics of SKINNY-n-3n in the context of key-recovery attacks. We employ the same encoding scheme from [7], where the ith cell of a state $\#S$ is encoded with a pair of 0-1 variables $(x_i^{\#S}, y_i^{\#S})$ according to the rule given in Section 2. Firstly, due to the complexity estimation given by Equation (3), $\min\{\mathrm{DoF}^+, \mathrm{DoF}^-, \mathrm{DoM}\}$ should be maximized in our model. To this end, we introduce an auxiliary variable $v_{\texttt{Obj}}$, impose the constraints

$$\{v_{\texttt{Obj}} \leq \mathrm{DoF}^+, v_{\texttt{Obj}} \leq \mathrm{DoF}^-, v_{\texttt{Obj}} \leq \mathrm{DoM}\}$$

and set the objective function to maximize $v_{\texttt{Obj}}$. In what follows, we describe the constraints for the starting states, ending states, and the states in the computation paths with a special focus on what is different from Bao et al.'s work [7]. First of all, the tweakey schedule algorithm of SKINNY-n-3n only involves in-cell operations and permutations changing the positions of the cells in the tweakey register, which will not alter the color of a cell in our model (only their positions are changed). Therefore, we will not discuss the constraints imposed solely by the tweakey schedule algorithm in the following.

Constraints for the Starting States. As discussed in Section 3, we distinguish the sources of degrees of freedom from the encryption data path (denoted by $\lambda^+_{\texttt{ENC}}$ and $\lambda^-_{\texttt{ENC}}$) and the key schedule data path (denoted by $\lambda^+_{\texttt{KSA}}$ and $\lambda^-_{\texttt{KSA}}$), and the initial degrees of freedom satisfies $\lambda^+ = \lambda^+_{\texttt{ENC}} + \lambda^+_{\texttt{KSA}}$ and $\lambda^- = \lambda^-_{\texttt{ENC}} + \lambda^-_{\texttt{KSA}}$, where $\lambda^+_{\texttt{ENC}} = |\mathcal{B}^{\texttt{ENC}}|$, $\lambda^+_{\texttt{KSA}} = |\mathcal{B}^{\texttt{KSA}}|$, $\lambda^-_{\texttt{ENC}} = |\mathcal{R}^{\texttt{ENC}}|$, and $\lambda^-_{\texttt{KSA}} = |\mathcal{R}^{\texttt{KSA}}|$. We introduce two variables α_i and β_i for each cell in $(\#S^{\texttt{ENC}}, \#S^{\texttt{KSA}})$, where $\alpha_i = 1$ if and only if $(x_i^{\#S}, y_i^{\#S}) = (1, 0)$ and $\beta_i = 1$ if and only if $(x_i^{\#S}, y_i^{\#S}) = (0, 1)$. Then we have the following constraints:

$$\lambda^+_{\texttt{ENC}} = \sum\nolimits_i \alpha_i^{\texttt{ENC}},\ \lambda^+_{\texttt{KSA}} = \sum\nolimits_i \alpha_i^{\texttt{KSA}},\ \lambda^-_{\texttt{ENC}} = \sum\nolimits_i \beta_i^{\texttt{ENC}},\ \lambda^-_{\texttt{KSA}} = \sum\nolimits_i \beta_i^{\texttt{KSA}},$$

and

$$\begin{cases} x_i^{\#S^{\texttt{ENC}}} - \alpha_i^{\texttt{ENC}} \geq 0 \\ y_i^{\#S^{\texttt{ENC}}} - x_i^{\#S^{\texttt{ENC}}} + \alpha_i^{\texttt{ENC}} \geq 0 \\ y_i^{\#S^{\texttt{ENC}}} + \alpha_i^{\texttt{ENC}} \leq 1 \end{cases}, \qquad \begin{cases} y_i^{\#S^{\texttt{ENC}}} - \beta_i^{\texttt{ENC}} \geq 0 \\ x_i^{\#S^{\texttt{ENC}}} - y_i^{\#S^{\texttt{ENC}}} + \beta_i^{\texttt{ENC}} \geq 0 \\ x_i^{\#S^{\texttt{ENC}}} + \beta_i^{\texttt{ENC}} \leq 1 \end{cases},$$

$$\begin{cases} x_i^{\#S^{\texttt{KSA}}} - \alpha_i^{\texttt{KSA}} \geq 0 \\ y_i^{\#S^{\texttt{KSA}}} - x_i^{\#S^{\texttt{KSA}}} + \alpha_i^{\texttt{KSA}} \geq 0 \\ y_i^{\#S^{\texttt{KSA}}} + \alpha_i^{\texttt{KSA}} \leq 1 \end{cases}, \qquad \begin{cases} y_i^{\#S^{\texttt{KSA}}} - \beta_i^{\texttt{KSA}} \geq 0 \\ x_i^{\#S^{\texttt{KSA}}} - y_i^{\#S^{\texttt{KSA}}} + \beta_i^{\texttt{KSA}} \geq 0 \\ x_i^{\#S^{\texttt{KSA}}} + \beta_i^{\texttt{KSA}} \leq 1 \end{cases}.$$

Constraints for the Ending States. We assume that the matching only happens at the `MixColumns`. Let $(\#E^+[4j], \#E^+[4j+1], \#E^+[4j+2], \#E^+[4j+3])^T$ and $(\#E^-[4j], \#E^-[4j+1], \#E^-[4j+2], \#E^-[4j+3])^T$ be the jth column of the ending states $\#E^+$ and $\#E^-$ linked by the `MC` operation. Since `MC` is non-MDS, its constraints are quite different from Bao et al.'s model for MDS matrix, where there is a $(\Sigma - 4)$-cell filter if and only if $\Sigma \geq 5$ out of 8 cells of the two columns are 🟦 or 🟥 cells (see [7, Property 1, page 14]).

For the `MC` operation of `SKINNY`, there may exist an m-cell $(m > 0)$ filter even if $\Sigma < 5$. For example, according to Equation (4), if $\#E^+[4j] =$ 🟦, $\#E^-[4j+1] =$ 🟥 and all other cells are □ , we still get a 1-cell filter due to $\#E^+[4j] = \#E^-[4j+1]$. We can enumerate all possible patterns and convert these local constraints into linear inequalities using the convex hull computation method. In Figure 3, we list some of the possible matching patterns with their filtering strength measured in cells. We introduce a variable $\gamma_j \geq 0$ for the j-th columns of $\#E^+$ and $\#E^-$ such that there is a γ_j-cell filter due to the coloring patterns of $\#E^+$ and $\#E^-$, then we get a DoM-cell filter at the matching point, where $\text{DoM} = \sum_j \gamma_j$ and should be positive according to the complexity analysis given by Equation (3).

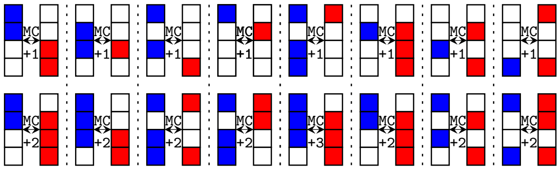


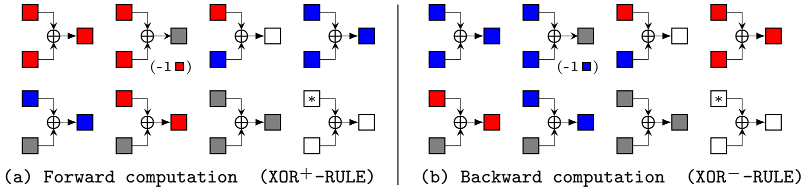
Fig. 3: Some possible coloring patterns at the matching point

Constraints Imposed by the Computation Paths. Along the computation paths leading to the ending states, the initial degrees of freedom are consumed according to the MITM characteristic. Forward computation consumes the degrees of freedom of the neutral words for backward computation while backward computation consumes the degrees of freedom of the neutral words for the forward computation. The consumption of degrees of freedom is counted in cells. Let $\sigma^+_{\texttt{ENC}}$, $\sigma^+_{\texttt{KSA}}$ and $\sigma^-_{\texttt{ENC}}$, $\sigma^-_{\texttt{KSA}}$ be the accumulated degrees of freedom that have been consumed in the backward and forward computation in the encryption and key schedule data paths. Since the degrees of freedom from the encryption data paths for both directions should be used up and the degrees of freedom originated from the key schedule data path should not be exhausted, we require

$$\begin{cases} \lambda^+_{\texttt{ENC}} - \sigma^+_{\texttt{ENC}} = 0, \quad \lambda^-_{\texttt{ENC}} - \sigma^-_{\texttt{ENC}} = 0 \\ \text{DoF}^+ = \lambda^+_{\texttt{KSA}} - \sigma^+_{\texttt{KSA}} \geq 1, \quad \text{DoF}^- = \lambda^-_{\texttt{KSA}} - \sigma^-_{\texttt{KSA}} \geq 1 \end{cases} .$$

According to the semantics of the colors, how a coloring pattern of the input and output states of an operation consumes the degrees of freedom should be be different for the forward and the backward computation paths. Therefore, we will give two sets of rules for different directions of the computation.

`XOR`. The XOR operations exist in the `ART` and `MC`, and we can reuse the `XOR-RULE`$^+$ (for forward computation) and `XOR-RULE`$^-$ (for backward computation) rules gvien in [7]. The coloring patterns and how the degrees of freedom are consumed are visualized in Figure 4.

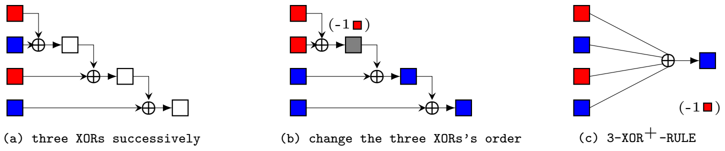


(a) Forward computation (XOR$^+$-RULE)

(b) Backward computation (XOR$^-$-RULE)

Fig. 4: Rules for XOR, where a "*" means that the cell can be any color

`AddRoundTweakey`. `ART` is the operation that the first two rows of the three tweakey states are XORed into the encryption data path. There are three XOR operations and four input cells (three from the tweakey state and one from the encryption data path) involved to produce an output cell. Certainly, we can use the `XOR-RULE` three times to get the constraints. However, this approach misses some important coloring patterns that may lead to better attacks. We take the forward computation for example as shown in Figure 5. If we use `XOR`$^+$`-RULE` three times successively as shown in Figure 5(a), when the 🟥 and 🟦 are the input cells of the XOR, the output cell will be □ , eventually leading to a □ output cell. However, if we change the order of the XOR operations as shown in Figure 5(b), then 🟥 $\oplus$ 🟥 may produce a ▨ cell by consuming one degree of freedom, leading to a 🟦 output cell. To take this into account, we model the rule for three XORs as a whole, named as `3-XOR`$^+$`-RULE`, with Figure 5(c) as an example.



(a) three XORs successively

(b) change the three XORs's order

(c) 3-XOR$^+$-RULE

Fig. 5: The inaccuracy of modeling `3-XOR`$^+$ by applying `XOR`$^+$ successively

For the `3-XOR` operation in the forward computation, we have the following set of rules (denoted by `3-XOR`$^+$`-RULE`):

- ▶ `3-XOR`$^+$`-RULE-1`. If there are 🟥 cells but no □ and 🟦 cells in the input, the output cell is ◼ or 🟥 (partially cancel the impacts of the input 🟥 cells by consuming $\lambda^-_{\texttt{ENC}}$ or $\lambda^-_{\texttt{KSA}}$).
- ▶ `3-XOR`$^+$`-RULE-2`. If there are 🟥 and 🟦 cells but no □ cells in the input, the output cell is □ or 🟦 (partially cancel the impacts from 🟥 on 🟦 by consuming $\lambda^-_{\texttt{ENC}}$ or $\lambda^-_{\texttt{KSA}}$).
- ▶ `3-XOR`$^+$`-RULE-3`. If there are 🟦 cells but no □ and 🟥 cells in the input, the output cell is 🟦.
- ▶ `3-XOR`$^+$`-RULE-4`. If all the input cells are ◼, then the output cell is ◼.
- ▶ `3-XOR`$^+$`-RULE-5`. If there is at least one □ cell in the input, the output is □.

We introduce variables $\delta^-_{\texttt{ENC}}$ and $\delta^-_{\texttt{KSA}}$ to denote the consumed degrees of freedom due to `3-XOR`$^+$`-RULE`. For example, $\delta^-_{\texttt{ENC}} = 1$ means that we consume one degree of freedom from $\lambda^-_{\texttt{ENC}}$ by applying the rule. In order to use up all the degrees of freedom from $\#S^{\texttt{ENC}}$, we should consume $\lambda^-_{\texttt{ENC}}$ first whenever possible. As shown in Figure 6, when there are degrees of freedom in the encryption path, i.e., 🟥 cells, the consumption of degree of freedom is always from $\lambda^-_{\texttt{ENC}}$, i.e., $\delta^-_{\texttt{ENC}} = 1$ and $\delta^-_{\texttt{KSA}} = 0$.

Let $\#a$, $\#b$, $\#c$, $\#d$ be the input cells and $\#e$ be the output cell. Then, the set of rules `3-XOR`$^+$`-RULE` restricts $(x^{\#a}, y^{\#a}, x^{\#b}, y^{\#b}, x^{\#c}, y^{\#c}, x^{\#d}, y^{\#d}, x^{\#e}, y^{\#e}, \delta^-_{\texttt{ENC}})$ and $(x^{\#a}, y^{\#a}, x^{\#b}, y^{\#b}, x^{\#c}, y^{\#c}, x^{\#d}, y^{\#d}, x^{\#e}, y^{\#e}, \delta^-_{\texttt{KSA}})$ to subsets of $\mathbb{F}_2^{11}$, which can be described by a system of linear inequalities by using the convex hull computation method. Some valid coloring patterns due to `3-XOR`$^+$`-RULE` are given in Figure 6. Note that `3-XOR`$^-$`-RULE` can be obtained from `3-XOR`$^+$`-RULE` by exchanging the 🟥 cells and 🟦 cells, since the meanings of 🟥 and 🟦 are dual for the forward and backward computations.

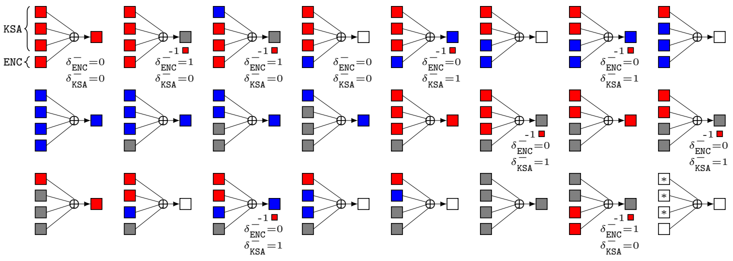
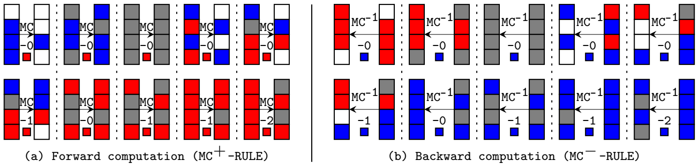

KSA
ENC
$\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=0
-1 🟥 $\delta^-_{\texttt{ENC}}$=1 $\delta^-_{\texttt{KSA}}$=0
-1 🟥 $\delta^-_{\texttt{ENC}}$=1 $\delta^-_{\texttt{KSA}}$=0
$\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=0
-1 🟥 $\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=1
-1 🟥 $\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=1
-1 🟥 $\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=1
-1 🟥 $\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=1
-1 🟥 $\delta^-_{\texttt{ENC}}$=0 $\delta^-_{\texttt{KSA}}$=1
-1 🟥 $\delta^-_{\texttt{ENC}}$=1 $\delta^-_{\texttt{KSA}}$=0
*
*
*

Fig. 6: `3-XOR`$^+$`-RULE`, where a "*" means that the cell can be any color

`MixColumn`. Since `MC` contains only XOR operations, we can use `XOR-RULE` to generate the set of rules `MC-RULE` for `MC`. According to Equation (4), there exists one equation that XORs three cells together to get one cell. We use a similar approach we employed for `3-XOR`$^+$`-RULE` and `3-XOR`$^-$`-RULE` to handle this special equation. Finally, we get the valid propogations of the coloring patterns and list some of them in Figure 7. Note that there are no key additions involved in `MC`, and thus all the consumed degrees of freedom are from $\lambda^+_{\texttt{ENC}}$ and $\lambda^-_{\texttt{ENC}}$.



(a) Forward computation (MC^{+}-RULE)

(b) Backward computation (MC^{-}-RULE)

Fig. 7: MC-RULE

4.2 The MITM Key-recovery Attack on SKINNY-n-$3n$

Solving the model built in Section 4.1, we identify a 23-round MITM characteristic as shown in Figure 8. The starting states are $\#S^{\texttt{ENC}} = Y_1$ and the three tweakey words $\#S^{\texttt{KSA}} = (TK_1^{(1)}, TK_2^{(1)}, TK_3^{(1)})$. The matching process happens at the MC operation between the ending states $\#E^{+} = Z_{12}$ and $\#E^{-} = X_{13}$. There are 3 🟦 cells and 3 🟥 cells in $\#S^{\texttt{KSA}}$, providing $\lambda^{-}_{\texttt{KSA}} = \lambda^{+}_{\texttt{KSA}} = 3$ cells of initial degrees of freedom originated from the key schedule data path. For $\#S^{\texttt{ENC}}$, Y_1 provides $\lambda^{-}_{\texttt{ENC}} = 8$ and $\lambda^{+}_{\texttt{ENC}} = 1$ cells of initial degrees of freedom from the encryption data path. The $\lambda^{+}_{\texttt{ENC}} = 1$ cells of degrees of freedom is used up when computing X_1 from Y_1 by XORing the subtweakey. In the forward computation, the $\lambda^{-}_{\texttt{ENC}} = 8$ cells of degrees of freedom are used up when computing Y_4 from Y_1. For the forward computation, we require $TK_1^{(6)}[7] \oplus TK_2^{(6)}[7] \oplus TK_3^{(6)}[7]$ and $TK_1^{(8)}[1] \oplus TK_2^{(8)}[1] \oplus TK_3^{(8)}[1]$ to be constants, consuming $\sigma^{-}_{\texttt{KSA}} = 2$ cells of degrees of freedom originated from the key schedule data path. Hence, we get $\text{DoF}^{-} = \lambda^{-}_{\texttt{KSA}} - \sigma^{-}_{\texttt{KSA}} = 1$. Similarly, we get $\text{DoF}^{+} = \lambda^{+}_{\texttt{KSA}} - \sigma^{+}_{\texttt{KSA}} = 1$. At the matching point, we have $\text{DoM} = 2$ from the first two column of $\#E^{+}$ and $\#E^{-}$ with Equation (4). The 23-round key-recovery attack is given in Algorithm 3. The data and memory complexity is bounded by Line 2, which is 2^{104} for SKINNY-128-384 and 2^{52} for SKINNY-64-192. According to Equation (3), the time complexity is about 2^{376} for SKINNY-128-384 and 2^{188} for SKINNY-64-192.

Remark. The designers of SKINNY claimed that: *"We conclude that meet-in-the-middle attack may work up to at most 22 rounds (see [10], Sect. 4.2, page 22)"*. Our attack penetrates one more round than expected and is the first 23-round single-key attack on SKINNY-128-384 and SKINNY-64-192. Using the same method, we also analyze ForkSkinny (see Supplement Material B). In addition, we report on some results on Romulus-H as a by-product of the analysis of SKINNY (see Supplement Material C).

5 Exploiting Nonlinearly Constrained Neutral Words in MITM Attacks and Its Applications

According to Property 1 in Section 2, in order to compute the allowable values for the neutral words, one has to solve two systems of equations, i.e., Equation (1)

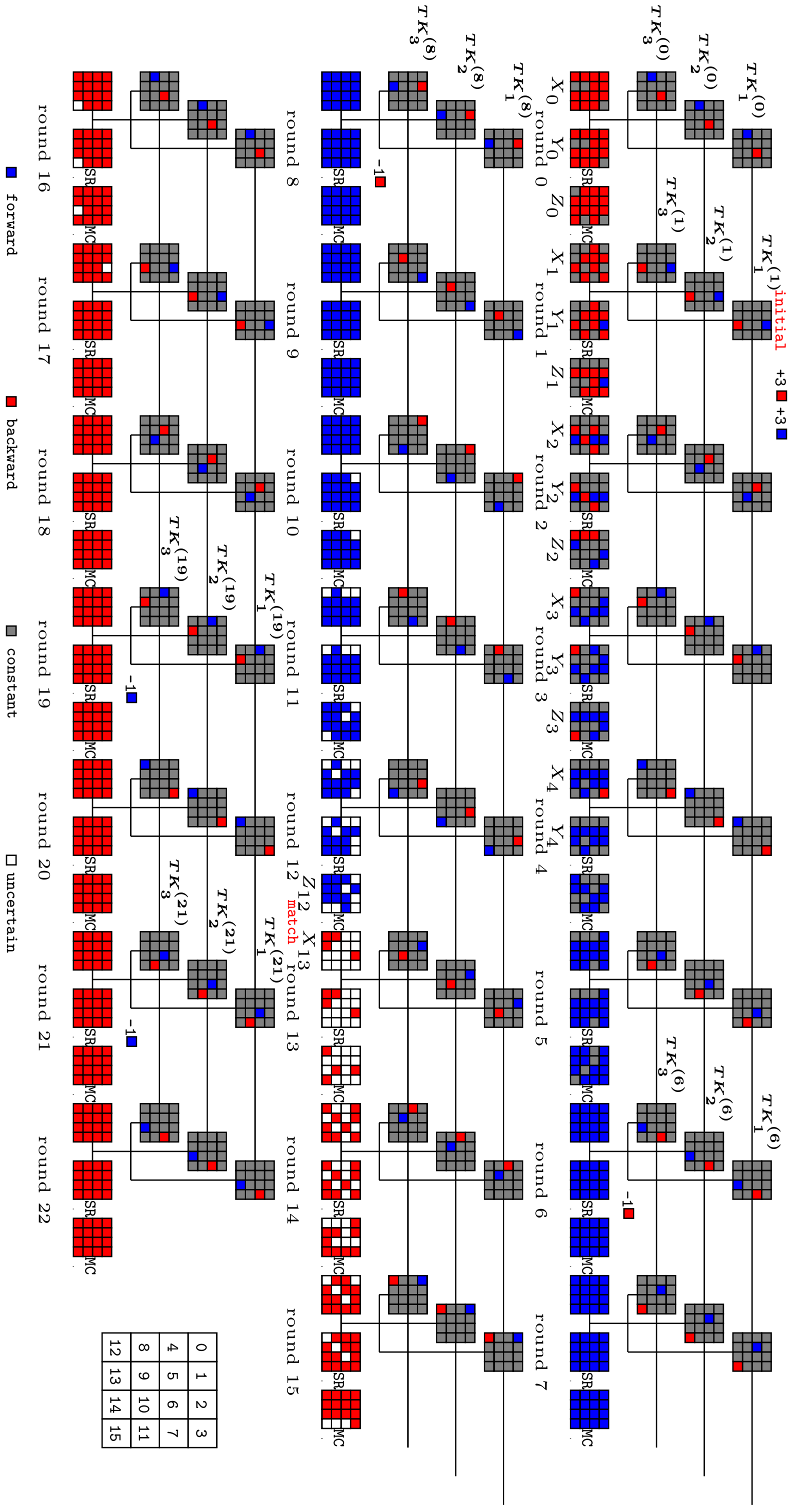


Fig. 8: An MITM key-recovery attack on 23-round SKINNY-n-$3n$

Algorithm 3: The MITM key-recovery attack on $\texttt{SKINNY}$-n-$3n$

```
$X_0[3,9,13] \leftarrow 0,\ X_1[0,2,8,10,13] \leftarrow 0,\ X_2[1,3,9,11] \leftarrow 0,\ Y_2[5] \leftarrow 0,$
  $X_3[0,8] \leftarrow 0,\ Y_4[3] \leftarrow 0$
Collecting structure of plaintext-ciphertext pairs and store them in table $H$,
  which traverses the non-constant 16-3=13 cells in the plaintext
for All possilbe values of the ■ cells in $(TK_1^{(0)}, TK_2^{(0)}, TK_3^{(0)})$ do
    for $(a_1, a_2, b_1, b_2) \in \mathbb{F}_2^{4w}$ do
        $Y_0[3] \leftarrow TK_1^{(0)}[3] \oplus TK_2^{(0)}[3] \oplus TK_3^{(0)}[3],\ Y_0[9,13] \leftarrow X_0[9,13],$
          $Z_0[3,11,12] \leftarrow Y_0[3,9,13],\ X_1[12] \leftarrow X_1[0] \oplus Z_0[12],\ X_1[7] \leftarrow Z_0[3],$
          $X_1[15] \leftarrow Z_0[3] \oplus Z_0[11],\ X_2[15] \leftarrow X_2[3] \oplus Z_1[15],\ X_3[4] \leftarrow Z_2[0]$
        Derive the solution space of the 🟥 cells in the $TK$ by
          $\begin{cases} TK_1^{(6)}[7] \oplus TK_2^{(6)}[7] \oplus TK_3^{(6)}[7] = a_1 \\ TK_1^{(8)}[1] \oplus TK_2^{(8)}[1] \oplus TK_3^{(8)}[1] = a_2 \end{cases}.$
        Derive the solution space of the 🟦 cells in the $TK$ by
          $\begin{cases} TK_1^{(19)}[4] \oplus TK_2^{(19)}[4] \oplus TK_3^{(19)}[4] = b_1 \\ TK_1^{(21)}[6] \oplus TK_2^{(21)}[6] \oplus TK_3^{(21)}[6] = b_2 \end{cases}.$
        Initialize $L$ to be an empty hash table
        for the value in the solution space of 🟥 cells in $TK$ do
            Compute $X_{13}[8]$ along the backward computation path:
              $X_4 \rightarrow X_0 \rightarrow E_K(X_0) \rightarrow X_{13}$ by accessing $H$
            Insert relative information into $L$ indexed by $X_{13}[8]$
        for the value in the solution space of 🟦 cells in $TK$ do
            Compute $Z_{12}[4]$ and $Z_{12}[8]$ along the forward computation path:
              $X_1 \rightarrow Z_{12}$
            for Candidate keys in $L[Z_{12}[4] \oplus Z_{12}[8]]$ do
                Test the guessed key with several plaintext-ciphertext pairs
```

and (2). In previous MITM preimage attacks [50,7], the two systems of equations are linear (or can be reduced to linear equations involving certain cells not from the starting states that implicitly define the spaces of the neutral words). Hence, it is easy to derive the solution spaces $\mathbb{B}(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^{+})$ and $\mathbb{R}(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^{-})$ by solving the systems of equations, whose cost can be ignored compared with the overall complexity. However, in practice, we encounter many interesting MITM characteristics with nonlinear constrained neutral words, and there is no efficient method for solving them. We present a table based technique in Algorithm 4 which can be applied in attacks relying on such MITM characteristics without solving the equations or increasing the overall time complexities.

Algorithm 4: Computing the solution spaces of the neutral words

```
Input: $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}]) \in \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\mathtt{ENC}}|+|\mathcal{G}^{\mathtt{KSA}}|)}$
Output: $V$, $U$
$V \leftarrow [\,]$, $U \leftarrow [\,]$
for $(\#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) \in \mathbb{F}_2^{w\cdot(|\mathcal{B}^{\mathtt{ENC}}|+|\mathcal{B}^{\mathtt{KSA}}|)}$ do
    $\boldsymbol{v} \leftarrow \boldsymbol{\pi}^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ by Equation 1
    Insert $(\#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ into $V$ at index $\boldsymbol{v}$
for $(\#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}]) \in \mathbb{F}_2^{w\cdot(|\mathcal{R}^{\mathtt{ENC}}|+|\mathcal{R}^{\mathtt{KSA}}|)}$ do
    $\boldsymbol{u} \leftarrow \boldsymbol{\pi}^-(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$ by Equation 2
    Insert $(\#S^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$ into $U$ at index $\boldsymbol{u}$
```

Algorithm 4 obtains the solution spaces of the neutral words for all possible $\mathfrak{c}^+$ and $\mathfrak{c}^-$ under a given value of $(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ with time complexity $(2^w)^{\lambda^+} + (2^w)^{\lambda^-}$ and memory complexity $(2^w)^{\lambda^+} + (2^w)^{\lambda^-}$. After running Algorithm 4, $V[\boldsymbol{v}]$ stores the solution space of

$$\boldsymbol{\pi}^+(\#S^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \#S^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \#S^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}]) = \boldsymbol{v},$$

which consists about $2^{w\cdot(\lambda^+-l^+)} = 2^{w\cdot\mathrm{DoF}^+}$ values for the neutral words for the forward computation. Similarly, under each index $\boldsymbol{u}$ of U, there are about $2^{w\cdot(\lambda^--l^-)} = 2^{w\cdot\mathrm{DoF}^-}$ values for the neutral words for the backward computation. Algorithm 4 can be plugged into the procedure for MITM attacks to deal with MITM characteristics with nonlinearly constrained neutral words. For example, applying the technique to the MITM preimage attack gives Algorithm 5. Next, we show the time complexity is not increased.

Complexity Analysis. In each MITM episode within the context defined by the “For” loops surrounding the code segment from Line 6 to Line 14 of Algorithm 5, we test $2^{w\cdot(\mathrm{DoF}^++\mathrm{DoF}^-)}$ messages and we expect $2^{w\cdot(\mathrm{DoF}^++\mathrm{DoF}^--m)}$ of them to pass the m-cell filter, and averagely, there are about $2^{w\cdot(\mathrm{DoF}^++\mathrm{DoF}^--h)}$ preimages passing the check at Line 13 for each episode. The time complexity to perform one MITM episode is

$$(2^w)^{\mathrm{DoF}^+} + (2^w)^{\mathrm{DoF}^-} + (2^w)^{\mathrm{DoF}^++\mathrm{DoF}^--m}. \tag{5}$$

Then, we estimate the size of $\mathbb{G}$ in Line 1 of Algorithm 5, which determines the number of MITM episodes performed. Suppose $|\mathbb{G}| = (2^w)^x$, to produce one preimage, we require that $(2^w)^x \cdot (2^w)^{l^++l^-} \cdot (2^w)^{\mathrm{DoF}^++\mathrm{DoF}^-} = (2^w)^h$ or $x = h - (\lambda^+ + \lambda^-)$. Hence, we consider two situations depending on $\lambda^+ + \lambda^-$.

- $\lambda^+ + \lambda^- \geq h$: In this case, we set $x = 0$, then $|\mathbb{G}| = 1$. At Line 3 and Line 4 of Algorithm 5, we only need to traverse $(2^w)^{h-(\mathrm{DoF}^++\mathrm{DoF}^-)}$ values of $(\mathfrak{c}^+, \mathfrak{c}^-) \in \mathbb{F}_2^{w\cdot l^++w\cdot l^-}$, where $h-(\mathrm{DoF}^++\mathrm{DoF}^-) \leq l^++l^-$ due to $\lambda^++\lambda^- \geq h$,

Algorithm 5: The framework of the MITM preimage attack on AES-like hashing with non-linearly constrained neutral words

```
1  for $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]) \in \mathbb{G} \subseteq \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\texttt{ENC}}|+|\mathcal{G}^{\texttt{KSA}}|)}$ do
2      Call Algorithm 4 to build $V$, $U$
3      for $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w\cdot l^+}$ do
4          for $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w\cdot l^-}$ do
5              $L \leftarrow [\,]$
6              for $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in V[\mathfrak{c}^+]$ do
7                  Compute $E^+[\mathcal{M}^+]$ along the forward computation path
8                  Insert $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ into $L$ indexed by $E^+[\mathcal{M}^+]$

9              for $(\#S^{\texttt{ENC}}[\mathcal{R}^{\texttt{KSA}}], \#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) \in U[\mathfrak{c}^-]$, do
10                 Compute $E^-[\mathcal{M}^-]$ along the backward computation path

11                 for $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in L[E^-[\mathcal{M}^-]]$ do
12                     Reconstruct the (candidate) message $X$
13                     if $X$ is a preimage then
14                         Output $X$ and Stop.
```

to find the preimage. Then, together with Equation (5), we have the overall time complexity:

$$(2^w)^{\lambda^+} + (2^w)^{\lambda^-} + (2^w)^{h-\min(\mathrm{DoF}^+,\ \mathrm{DoF}^-,\ m)}.$$

- $\lambda^+ + \lambda^- < h$: Set $x = h - (\lambda^+ + \lambda^-)$, and we need to build 2^x V and U in Line 2 of Algorithm 5. Hence, we get the overall complexity:

$$(2^w)^{h-\lambda^+} + (2^w)^{h-\lambda^-} + (2^w)^{h-\min(\mathrm{DoF}^+,\ \mathrm{DoF}^-,\ m)}. \tag{6}$$

Moreover, the memory complexity for both situations is

$$(2^w)^{\lambda^+} + (2^w)^{\lambda^-} + (2^w)^{\min(\mathrm{DoF}^+,\ \mathrm{DoF}^-)}. \tag{7}$$

We apply Algorithm 5 to `Grøstl`-256, and `Saturnin-Hash`, and we obtain some improved cryptanalytic results, which are given in the Supplementary Materials D and E.

6 MITM-based Collision Attacks and Its Applications

Suppose that there is an algorithm that can produce a different t-cell partial target preimage. Then we expect to find a collision by running the algorithm $2^{w\cdot(h-t)/2}$ times to identify a collision on the h-cell hash value. At FSE 2012 [46], Li, Isobe, and Shibutani employed this strategy to convert the MITM-based partial target preimage attacks into pseudo collision attacks. First, we consider a generalization of partial target preimage attacks.

Let $\mathbb{T}$ be the space of all possible values of the output of the hash function. For a predefined partition of $\mathbb{T}$ into $(2^w)^t$ subspaces with an equal size. We call an algorithm a t-cell partial target preimage attack if it can produce a message whose hash value is a random element in a given subspace. For example, an algorithm generating a message such that the first word of its hash value is always 0 is a 1-cell partial target preimage attack. An algorithm generating a message such that the XOR of the first and second words of its hash value is always 0 is also a 1-cell partial target preimage attack. Given an MITM characteristic, the framework for a collision attack is described in Algorithm 6. Note that the call to Algorithm 6 can be replaced by an ordinary equation solving procedure to save the memory if the involved equations are linear or easy to solve. To be clear on how to set the objective functions in our MILP models, we need to understand how the complexity of the attack is related to the parameters specified in the MITM characteristic.

Algorithm 6: The framework of the MITM collision attack on AES-like hashing with non-linearly constrained starting states

```
Setting the selected $t$ cells of $\#T$ to constants
$H \leftarrow [\,]$
for $(\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]) \in \mathbb{G} \subseteq \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\texttt{ENC}}|+|\mathcal{G}^{\texttt{KSA}}|)}$ do
    $V \leftarrow [\,]$, $U \leftarrow [\,]$
    Call Algorithm 4 to populate $V$ and $U$
    for $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w\cdot l^+}$ do
        for $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w\cdot l^-}$ do
            $L \leftarrow [\,]$
            for $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in V[\mathfrak{c}^+]$ do
                Compute $E^+[\mathcal{M}^+]$ along the forward computation path
                Insert $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ into $L$ indexed by $E^+[\mathcal{M}^+]$
            for $(\#S^{\texttt{ENC}}[\mathcal{R}^{\texttt{KSA}}], \#S^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) \in U[\mathfrak{c}^-]$, do
                Compute $E^-[\mathcal{M}^-]$ along the backward computation path
                for $(\#S^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \#S^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in L[E^-[\mathcal{M}^-]]$ do
                    Reconstruct the (candidate) message $X$
                    if $X$ is a $t$-cell partial target preimage then
                        Insert $X$ into $H$ indexed by the hash value of $X$
                        Stop when there is a collision
```

Complexity Analysis. In the MITM t-cell partial target preimage attack, if the matching process results in an m-cell filter, then we have $m \leq t$, because the matching information is derived from the known cells of the target T. To determine the overall complexity of the algorithm, we need to determine how many MITM episodes (Line 9 to 18 of Algorithm 6) are required. According to the analysis of Algorithm 4 in Section 5, the time complexity for building U and V is $(2^w)^{\lambda^+} + (2^w)^{\lambda^-}$. In each MITM episode within the context defined by the "For" loops surrounding the code segment from Line 9 to Line 18, we test $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^-)}$ messages and we expect $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^- - m)}$ of them to pass the m-cell filter, and averagely, there are about $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^- - t)}$ messages are inserted into the hash table H. Therefore, we need about $(2^w)^{\frac{h-t}{2} - (\mathrm{DoF}^+ + \mathrm{DoF}^- - t)}$ episodes to produce one collision. The time to perform one MITM episode is

$$(2^w)^{\mathrm{DoF}^+} + (2^w)^{\mathrm{DoF}^-} + (2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^- - m} + (2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^- - t}. \tag{8}$$

Suppose in Line 3 of Algorithm 6 we have $\mathbb{G} = 2^{w\cdot x}$. Then, $(2^w)^x \cdot (2^w)^{l^+} \cdot (2^w)^{l^-}$ matching episodes are performed. Hence, we have

$$(2^w)^x \cdot (2^w)^{l^+} \cdot (2^w)^{l^-} = (2^w)^{\frac{h-t}{2} - (\mathrm{DoF}^+ + \mathrm{DoF}^- - t)}.$$

We get $x = \frac{h}{2} - (\lambda^+ + \lambda^- - \frac{t}{2})$. Hence, we consider two situations:

- $\lambda^+ + \lambda^- \geq \frac{h+t}{2}$: In this case, we set $x = 0$. At Line 6 and Line 7 of Algorithm 6, we only need to traverse $(2^w)^{\frac{h-t}{2} - (\mathrm{DoF}^+ + \mathrm{DoF}^- - t)}$ values of $(\mathfrak{c}^+, \mathfrak{c}^-) \in \mathbb{F}_2^{w\cdot l^+ + w\cdot l^-}$, where $\frac{h-t}{2} - (\mathrm{DoF}^+ + \mathrm{DoF}^- - t) \leq l^+ + l^-$ due to $\lambda^+ + \lambda^- \geq \frac{h+t}{2}$, to find the collision. Then, together with Equation 8, we have the overall time complexity:

$$(2^w)^{\lambda^+} + (2^w)^{\lambda^-} + (2^w)^{\frac{h}{2} - \min\{\mathrm{DoF}^+ - \frac{t}{2},\ \mathrm{DoF}^- - \frac{t}{2},\ m - \frac{t}{2},\ \frac{t}{2}\}}. \tag{9}$$

- $\lambda^+ + \lambda^- < \frac{h+t}{2}$: Set $x = \frac{h}{2} - (\lambda^+ + \lambda^- - \frac{t}{2})$, and we need to build 2^x V and U in Line 5 of Algorithm 6. Hence, we get the overall complexity:

$$(2^w)^{\frac{h}{2} - (\lambda^+ - \frac{t}{2})} + (2^w)^{\frac{h}{2} - (\lambda^- - \frac{t}{2})} + (2^w)^{\frac{h}{2} - \min\{\mathrm{DoF}^+ - \frac{t}{2},\ \mathrm{DoF}^- - \frac{t}{2},\ m - \frac{t}{2},\ \frac{t}{2}\}}, \tag{10}$$

  which is approximately $(2^w)^{\frac{h}{2} - \min\{\mathrm{DoF}^+ - \frac{t}{2},\ \mathrm{DoF}^- - \frac{t}{2},\ m - \frac{t}{2},\ \frac{t}{2}\}}$, since we always have $\mathrm{DoF}^+ \leq \lambda^+$ and $\mathrm{DoF}^- \leq \lambda^-$.

The memory complexity in both situations is

$$(2^w)^{\lambda^+} + (2^w)^{\lambda^-} + (2^w)^{\min\{\mathrm{DoF}^+, \mathrm{DoF}^-\}} + (2^w)^{\frac{h-t}{2}}. \tag{11}$$

where the $(2^w)^{\frac{h-t}{2}}$ is to store the t-cell partial target preimages in H. Consequently, for an attack efficient than the trivial birthday attack, we have $\min\{\mathrm{DoF}^+ - \frac{t}{2},\ \mathrm{DoF}^- - \frac{t}{2},\ m - \frac{t}{2},\ \frac{t}{2}\} > 0$, $\lambda^+ < \frac{h}{2}$ and $\lambda^- < \frac{h}{2}$, or

$$\begin{cases} \mathrm{DoF}^+ > \frac{t}{2},\ \mathrm{DoF}^- > \frac{t}{2} \\ \frac{t}{2} < m \leq t \\ \lambda^+ < \frac{h}{2},\ \lambda^- < \frac{h}{2} \end{cases}.$$

6.1 Automatic Search for MITM-based Collision Attacks

First of all, The objective function of the model is to maximize

$$\min(\mathrm{DoF}^{+}-\frac{t}{2},\mathrm{DoF}^{+}-\frac{t}{2},m-\frac{t}{2},\frac{t}{2})$$

according to Equation (10). In what follows, we only discuss the main particularity of MITM-based collision attacks, which lies in the matching part. To be more specific, the degree of match (DoM) is derived differently from other attacks discussed in the work. To be concrete, we consider AES-like hashings like `WHIRLPOOL` and `Grøstl`, which includes the `MixColumn`(`MC`) or `MixRows`(`MR`) operation in their last rounds. To determine the degree of match, we consider two situations according to the position where the match happens.

The matching point is placed at the last round. Suppose that the MDS matrix of the `MC` operation at the matching point operates on k cells, which links the state Z in the last round to the XOR sum of the input state X of the first round and the target T, i.e., $\mathtt{MC}(Z) = X \oplus T$. Suppose that from the forward and backward computation α 🟦 cells and β 🟥 cells are known. Without loss of generality, we assume $(Z[0],\cdots,Z[\alpha-1])^T$ of Z is known as 🟦, and $(X[0],\cdots,X[\beta-1])^T$ of X is known as 🟥. From

$$\mathtt{MC}\cdot\begin{pmatrix} Z[0] \\ \vdots \\ Z[\alpha-1] \\ \vdots \\ Z[k-1] \end{pmatrix} = \begin{pmatrix} X[0]\oplus T[0] \\ X[1]\oplus T[1] \\ \vdots \\ X[\beta-1]\oplus T[\beta-1] \\ \vdots \end{pmatrix},$$

we get β linear equations with k variables $Z[0], Z[1],\cdots,Z[k-1]$ on the left, and 2β variables $X[0],\cdots,X[\beta-1]$, $T[0],\cdots,T[\beta-1]$ on the right. There are $k-\alpha$ unknowns $Z[\alpha],\cdots,Z[k-1]$ on the left. Hence, if $\beta > k-\alpha$, we can represent the $k-\alpha$ unknowns by other variables by consuming $k-\alpha$ linear equations. At last, we have $\Sigma = \beta-(k-\alpha)$ linear equations left:

$$\begin{cases} \zeta_1(Z[0],\cdots,Z[\alpha-1]) = \phi_1(X[0],\cdots,X[\beta-1]) \oplus \varphi_1(T[0],\cdots,T[\beta-1]), \\ \zeta_2(Z[0],\cdots,Z[\alpha-1]) = \phi_2(X[0],\cdots,X[\beta-1]) \oplus \varphi_2(T[0],\cdots,T[\beta-1]), \\ \qquad\qquad\vdots \\ \zeta_\Sigma(Z[0],\cdots,Z[\alpha-1]) = \phi_\Sigma(X[0],\cdots,X[\beta-1]) \oplus \varphi_\Sigma(T[0],\cdots,T[\beta-1]), \end{cases} \tag{12}$$

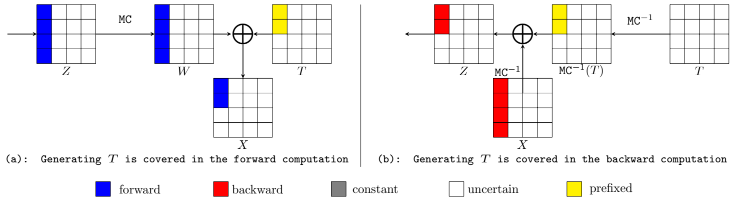
where $\zeta_i(\cdot)$, $\phi_i(\cdot)$, $\varphi_i(\cdot)$ are linear equations. By assigning $t \le \Sigma = \beta+\alpha-k$ conditions on the target T in the Equation (12):

$$\begin{cases} \varphi_1(T[0],\cdots,T[\beta-1]) = \tau_1, \\ \varphi_2(T[0],\cdots,T[\beta-1]) = \tau_2, \\ \qquad\qquad\vdots \\ \varphi_t(T[0],\cdots,T[\beta-1]) = \tau_t, \end{cases} \tag{13}$$

where $\boldsymbol{\tau} = (\tau_1,\cdots,\tau_t) \in \mathbb{F}_2^{w\cdot t}$, we get a t-cell filter:

$$\begin{cases} \zeta_1(Z[0],\cdots,Z[\alpha-1]) = \phi_1(X[0],\cdots,X[\beta-1]) \oplus \tau_1, \\ \zeta_2(Z[0],\cdots,Z[\alpha-1]) = \phi_2(X[0],\cdots,X[\beta-1]) \oplus \tau_2, \\ \qquad\qquad\vdots \\ \zeta_{\tau_t}(Z[0],\cdots,Z[\alpha-1]) = \phi_\Sigma(X[0],\cdots,X[\beta-1]) \oplus \tau_t. \end{cases}$$

In summary, we have the constraints $\text{DoF} = t \leq \Sigma = \beta + \alpha - k$ and $\beta + \alpha \geq k$. Therefore, in the MILP model for this case, we can ignore the coloring information of T. After identifying an MITM characteristic with configurations for (α, β, m, t), the t conditions on T can be derived accordingly with Equation (13).



(a): Generating T is covered in the forward computation

(b): Generating T is covered in the backward computation

Fig. 9: The matching point is not placed at the last round.

The matching point is not at the last round. In this case, the XOR of the target T can happen in the forward computation (see Figure 9(a)) or in the backward computation (see Figure 9(b)). The yellow cells are prefixed constants, which can be represented as 0-1 variables in the same way as the `Gray (G)` cells: If the ith cell of T is yellow, then $(x_i^T, y_i^T) = (1, 1)$. Other cells of T are `White (W)`, encoded as $(x_j^T, y_j^T) = (0, 0)$.

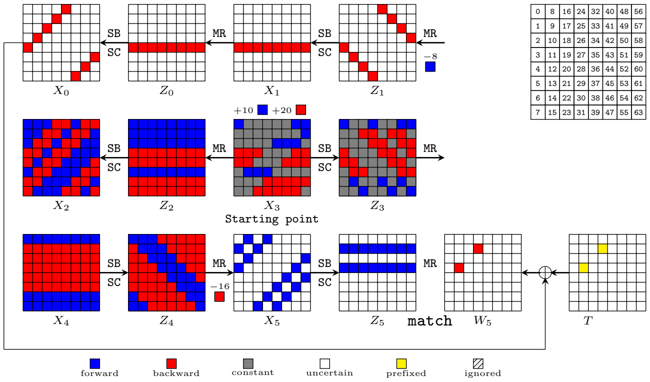
In the case shown in Figure 9(a), the rules of xoring the tag T is the same to the $\texttt{XOR}^{+}\texttt{-RULE}$ by regarding the 🟨 cells as ⬛ cells. Moreover, we require that the 🟨 cells in T align with the 🟦 cells in X as shown in Figure 9(a). Hence, the constraint $x_i^T \leq x_i^X$ is added to avoid the transition □ ⊕ 🟨 → □. Therefore, for the number t of conditions imposed on T, we have $t = \sum_i x_i^T$.

In the case of Figure 9(b), we consider the positions of 🟨 cells in $\texttt{MC}^{-1}(T)$. The rules of xoring the tag T is the same to the $\texttt{XOR}^{+}\texttt{-RULE}$ by regarding the 🟨 cells as ⬛ cells. In addition, we require that the 🟨 cells in $\texttt{MC}^{-1}(T)$ align with the 🟥 cells in Z. Hence, the constraint $y_i^{\texttt{MC}^{-1}(T)} \leq y_i^Z$ is added to avoid the transition □ ⊕ 🟨 → □. Therefore, for the number t of conditions imposed on T, we have $t = \sum_i y_i^{\texttt{MC}^{-1}(T)}$.

6.2 Collision Attacks on `WHIRLPOOL` and `Grøstl`

The `WHIRLPOOL` hash function [9], designed by Barreto and Rijmen, is an ISO/IEC standard. Its compression function is built by plug an AES-like cipher into the Miyaguchi-Preneel construction. During the last 20 years, `WHIRLPOOL` has withstood extensive cryptanalysis [47,44,34,56] and the best collision attack in the classical setting reaches 5 rounds [30,44]. Recently, Hosoyamada and Sasaki introduced a quantum collision attack on 6-round `WHIRLPOOL` [34].

In this section, we give the first 6-round collision attack on `WHIRLPOOL` in the classical setting, breaking the 10-year record for collision attacks on `WHIRLPOOL`.



0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

Fig. 10: An MITM attack on 6-round WHIRLPOOL

Applying the automatic model of MITM collision attack to WHIRLPOOL, we find a new 6-round MITM characteristic shown in Figure 10. We apply Algorithm 6 to WHIRLPOOL based on this MITM characteristic. The starting state is X_3. Then, we have $\lambda^+ = 10$ and $\lambda^- = 20$, $w = 8$. According to Property 1, we have $l^+ = 8$ and $\mathfrak{c}^+ = (a_1, \cdots, a_8) \in \mathbb{F}_2^{8\times 8}$; $l^- = 16$ and $\mathfrak{c}^- = (b_1, \cdots, b_{16}) \in \mathbb{F}_2^{8\times 16}$. Then we build similar equations to Equation (14), (15), (16) in the attack on Grøstl in Section D. Therefore, we call Algorithm 4 to build V and U. $\text{DoF}^+ = \lambda^+ - l^+ = 2$, $\text{DoF}^- = \lambda^- - l^- = 4$, $t = m = 2$ and $h = 64$. The time complexity is $(2^8)^{\frac{64}{2}-(10-\frac{2}{2})} + (2^8)^{\frac{64}{2}-(20-\frac{2}{2})} + (2^8)^{\frac{64}{2}-\min\{2-\frac{2}{2},\ 4-\frac{2}{2},\ 2-\frac{2}{2},\ \frac{2}{2}\}} \approx 2^{248}$ according to Equation (10), and the memory complexity is about 2^{248}. We also apply the method to Grøstl, and the results are given in Supplementary Material F.

7 Conclusion and Open Problems

Taking Bao et al's work (EUROCRYPT 2021) on automatic MITM preimage attacks as a starting point, we formulate the MITM attacks in a more formal, expressive, and accurate way. Based on this formulation, we investigate the peculiarities of MITM-based key-recovery attacks on block ciphers and collision attacks on AES-like hash functions and model them in the constraint programming paradigm. Now, we have a fairly powerful tool for finding exploitable MITM characteristics in key-recovery, (pseudo) preimage, and collision attacks on word oriented designs. Moreover, we present a generic procedure for dealing with nonlinearly constrained neutral words without increasing the overall time complexities of the attacks relying on them. We apply our method to concrete keyed and unkeyed primitives, leading to attacks improving the state-of-the-art. At this point, we would like propose an open problem: Is it possible to search for bit-level MITM characteristics automatically, and to what extent it can improve the current cryptanalytic results? In particular, we face the following difficulties in building a bit-oriented model: 1) Programming the bit-level propagation of neutral words inside the S-box. 2) Matching may happen in both the S-box layer and linear layer. 3) Even if the model can be constructed, it will be extremely difficult to solve due to a large number of variables and constraints. For bit-oriented models, we think the work from Fuhr, Minaud, and Yu [51,28] is good starting point.

References

1. Riham AlTawy and Amr M. Youssef. Preimage attacks on reduced-round Stribog. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 2014, Proceedings*, volume 8469, pages 109–125. Springer, 2014.
2. Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A new primitive for authenticated encryption of very short messages. In *ASIACRYPT 2019, Proceedings, Part II*, pages 153–182, 2019.
3. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In Mitsuru Matsui, editor, *ASIACRYPT 2009, Proceedings*, volume 5912, pages 578–597. Springer, 2009.
4. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008, Revised Selected Papers*, volume 5381, pages 103–119. Springer, 2008.
5. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO 2009, Proceedings*, volume 5677, pages 70–89. Springer, 2009.
6. Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small Present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017, Proceedings*, volume 10529, pages 321–345. Springer, 2017.
7. Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. Cryptology ePrint Archive, Report 2020/467, 2020. https://eprint.iacr.org/2020/467.
8. Augustin Bariant, Nicolas David, and Gaëtan Leurent. Cryptanalysis of Forkciphers. *IACR Trans. Symmetric Cryptol.*, 2020(1):233–265, 2020.
9. Paulo S. L. M. Barreto and Vincent Rijmen. The WHIRLPOOL Hashing Function, 2000. Revised in 2003.
10. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016, Proceedings, Part II*, pages 123–153. Springer, 2016.
11. Eli Biham, Orr Dunkelman, Nathan Keller, and Adi Shamir. New attacks on IDEA with at least 6 rounds. *J. Cryptol.*, 28(2):209–239, 2015.
12. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *ASIACRYPT 2011, Proceedings*, pages 344–371, 2011.
13. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010, Revised Selected Papers*, volume 6544, pages 229–240. Springer, 2010.

14. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In Antoine Joux, editor, *FSE 2011, Revised Selected Papers*, volume 6733, pages 252–269. Springer, 2011.
15. Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. *IACR Trans. Symmetric Cryptol.*, 2020(S1):160–207, 2020.
16. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 222–240, 2013.
17. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *FSE 2008, Revised Selected Papers*, volume 5086, pages 116–126. Springer, 2008.
18. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Proceedings, Part II*, volume 9815, pages 157–184. Springer, 2016.
19. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013, Proceedings*, volume 7881, pages 371–387. Springer, 2013.
20. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.
21. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Key recovery attacks on 3-round Even-Mansour, 8-step LED-128, and full AES. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Proceedings, Part I*, volume 8269, pages 337–356. Springer, 2013.
22. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Cryptanalysis of iterated Even-Mansour schemes with two keys. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Proceedings, Part I*, volume 8873, pages 439–457. Springer, 2014.
23. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. New attacks on Feistel structures with improved memory complexities. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 433–454. Springer, 2015.
24. Xiaoyang Dong, Siwei Sun, Danping Shi, Fei Gao, Xiaoyun Wang, and Lei Hu. Quantum collision attacks on AES-like hashing with low quantum random access memories. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Proceedings, Part II*, volume 12492, pages 727–757. Springer, 2020.
25. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In Masayuki Abe, editor, *ASIACRYPT 2010, Proceedings*, volume 6477, pages 158–176. Springer, 2010.
26. Orr Dunkelman, Gautham Sekar, and Bart Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In *INDOCRYPT 2007, Proceedings*, pages 86–100, 2007.
27. Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman. Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 683–701. Springer, 2015.

28. Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 61–81, 2014.
29. 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.
30. Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010, Revised Selected Papers*, pages 365–383, 2010.
31. 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*, pages 56–75, 2010.
32. Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In Matthew J. B. Robshaw, editor, *FSE 2006, Revised Selected Papers*, volume 4047, pages 210–225. Springer, 2006.
33. Deukjo Hong, Bonwook Koo, and Yu Sasaki. Improved preimage attack for 68-step HAS-160. In Dong Hoon Lee and Seokhie Hong, editors, *ICISC 2009, Revised Selected Papers*, volume 5984, pages 332–348. Springer, 2009.
34. Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Proceedings, Part II*, volume 12106, pages 249–279. Springer, 2020.
35. Takanori Isobe. A single-key attack on the full GOST block cipher. *J. Cryptol.*, 26(1):172–189, 2013.
36. Takanori Isobe and Kyoji Shibutani. Security analysis of the lightweight block ciphers XTEA, LED and Piccolo. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 2012, Proceedings*, volume 7372, pages 71–86. Springer, 2012.
37. Takanori Isobe and Kyoji Shibutani. Generic Key Recovery Attack on Feistel Scheme. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Proceedings, Part I*, volume 8269, pages 464–485. Springer, 2013.
38. Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Romulus for Round 3. NIST Lightweight Crypto Standardization process (Round 2), 2020.
39. Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved rebound attack on the finalist Grøstl. In Anne Canteaut, editor, *FSE 2012, Revised Selected Papers*, volume 7549, pages 110–126. Springer, 2012.
40. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Proceedings, Part II*, volume 8874, pages 274–288. Springer, 2014.
41. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family. *IACR Cryptology ePrint Archive*, 2011:286, 2011.
42. Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced SHA-1. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012, Proceedings*, volume 7417, pages 367–383. Springer, 2012.
43. 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.

44. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full WHIRLPOOL compression function. In *ASIACRYPT 2009, Proceedings*, pages 126–143, 2009.
45. Gaëtan Leurent and Clara Pernot. New Representations of the AES Key Schedule. Cryptology ePrint Archive, Report 2020/1253, 2020. https://eprint.iacr.org/2020/1253.
46. Ji Li, Takanori Isobe, and Kyoji Shibutani. Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to SHA-2. In Anne Canteaut, editor, *FSE 2012, Revised Selected Papers*, volume 7549, pages 264–286. Springer, 2012.
47. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced WHIRLPOOL and Grøstl. In *FSE 2009, Revised Selected Papers*, pages 260–276, 2009.
48. Florian Mendel, Vincent Rijmen, and Martin Schläffer. Collision attack on 5 rounds of Grøstl. In *FSE 2014, Revised Selected Papers*, pages 509–521, 2014.
49. National Institute of Standards and Technology (NIST). Lightweight cryptography (LWC) standardization process, 2020. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-2-Candidates.
50. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to WHIRLPOOL. In Antoine Joux, editor, *FSE 2011, Revised Selected Papers*, volume 6733, pages 378–396. Springer, 2011.
51. Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In Atsuo Inomata and Kan Yasuda, editors, *IWSEC 2018, Proceedings*, volume 11049, pages 227–243. Springer, 2018.
52. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT 2008, Proceedings*, volume 5350, pages 253–271. Springer, 2008.
53. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *EUROCRYPT 2009, Proceedings*, volume 5479, pages 134–152. Springer, 2009.
54. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active super-sbox analysis: Applications to ECHO and Grøstl. In *ASIACRYPT 2010, Proceedings*, pages 38–55, 2010.
55. Yu Sasaki, Lei Wang, Yasuhide Sakai, Kazuo Sakiyama, and Kazuo Ohta. Three-subset meet-in-the-middle attack on reduced XTEA. In *AFRICACRYPT 2012*, pages 138–154. Springer, 2012.
56. Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on WHIRLPOOL: Improved preimage and collision attacks. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012, Proceedings*, volume 7658, pages 562–579. Springer, 2012.
57. Martin Schläffer. Updated differential analysis of Grøstl. In *Grøstl website*, 2011.
58. Danping Shi, Siwei Sun, Patrick Derbez, Yosuke Todo, Bing Sun, and Lei Hu. Programming the Demirci-Selçuk meet-in-the-middle attack with constraints. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018, Proceedings, Part II*, volume 11273, pages 3–34. Springer, 2018.
59. Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. Impossible differential cryptanalysis of reduced-round SKINNY. In Marc Joye and Abderrahmane Nitaj, editors, *AFRICACRYPT 2017, Proceedings*, volume 10239, pages 117–134, 2017.
60. Lei Wang and Yu Sasaki. Finding preimages of Tiger up to 23 steps. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010, Revised Selected Papers*, volume 6147, pages 116–133. Springer, 2010.

61. Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. Second preimage attacks on step-reduced RIPEMD/RIPEMD-128 with a new local-collision approach. In Aggelos Kiayias, editor, *CT-RSA 2011, Proceedings*, volume 6558, pages 197–212. Springer, 2011.
62. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved meet-in-the-middle cryptanalysis of KTANTAN (poster). In Udaya Parampalli and Philip Hawkes, editors, *ACISP 2011, Proceedings*, volume 6812, pages 433–438. Springer, 2011.
63. 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 *FSE 2012, Revised Selected Papers*, pages 127–145, 2012.

Supplementary Material

A Detailed Explaination of the MITM Attack on 23-round Skinny-n-$3n$

According to Algorithm 2, we first set $|\#S^{\texttt{ENC}}|$ *independent* gray cells to constants. Therefore, we set $|\#S^{\texttt{ENC}}| = 16$ cells as 0 in the encryption data path as shown in Line 1 of Algorithm 3. Among them, the three cells of $X_0[3, 9, 13]$ are set as 0 to meet the requirement (II) to avoid using full codebook in Section 3. Hence, we only traverse the 13 non-constant cells of X_0 to collect data as shown in Line 2 of Algorithm 3. In Line 5 of Algorithm 3, after fixing one value for the ■ cells in $(TK_1^{(0)}, TK_2^{(0)}, TK_3^{(0)})$, we determine the other unfixed ■ cells in the encryption data path, i.e., $\#S^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$ in Line 6 of Algorithm 2. In detail, $Y_0[3] = X_0[3] \oplus TK_1^{(0)}[3] \oplus TK_2^{(0)}[3] \oplus TK_3^{(0)}[3]$ by adding $X_0[3] = 0$ with subtweakey cell; $Y_0[9, 13] = X_0[9, 13]$ since the subtweakey is only XORed into the first two rows. Now, the ■ cells in Y_0 are known. According to SR, we get the ■ cells $Z_0[3, 11, 12] = Y_0[3, 9, 13]$. Due to MC in Equation (4), we compute $X_1[12] = X_1[0] \oplus Z_0[12]$, $X_1[7] = Z_0[3]$ and $X_1[15] = Z_0[3] \oplus Z_0[11]$, where $X_1[0] = 0$ according to Line 1 of Algorithm 3. Totally, 8 ■ cells in X_1 are fixed (together with $X_1[0, 2, 8, 10, 13] = 0$ in Line 1 of Algorithm 3). After adding subtweakey into X_1, the ■ cells in Y_1 and then Z_1 are fixed, where Y_1 is acting as a starting point with $\lambda^-_{\texttt{ENC}} = 8$. From Z_1 to X_2, we get $X_2[15] = X_2[3] \oplus Z_1[15]$ due to Equation (4), where $X_2[3] = 0$ according to Line 1 of Algorithm 3. Here $X_2[1, 3, 9, 11] = 0$ and $Y_2[5] = 0$ act as a reduction of degrees of freedom for 🟥 cells and denote as $\sigma^-_{\texttt{ENC},2} = 5$ in round 2. The ■ cells in Y_2 and Z_2 are fixed by adding subtweakey to X_2. From Z_2 to X_3, we have $X_3[4] = Z_2[0]$ with Equation (4). Here $X_3[0, 8] = 0$ acts as a reduction of degrees of freedom for 🟥 cells and denote as $\sigma^-_{\texttt{ENC},3} = 2$ in round 3. At last, in round 4, from X_4 to Y_4, the red cell is cancelled by adding subtweakey to get $Y_4[3] = 0$, which acts as a degree of freedom reduction for 🟥 cells and denote as $\sigma^-_{\texttt{ENC},4} = 1$. Now, all the ■ cells in the encryption data path are fixed. Totally, $\sigma^-_{\texttt{ENC}} = \sigma^-_{\texttt{ENC},2} + \sigma^-_{\texttt{ENC},3} + \sigma^-_{\texttt{ENC},4} = 5 + 2 + 1 = 8$ cells of degrees of freedom are used. Since $\lambda^-_{\texttt{ENC}} = 8$, the intial degrees of freedom for 🟥 cells in the encryption data path are used up. Similarly, for the intial degrees of freedom of 🟦 cells in the data encryption path, i.e., $\lambda^+_{\texttt{ENC}} = 1$ in Y_1, are used up due to $X_1[2] = 0 = TK_1^{(1)}[2] \oplus TK_2^{(1)}[2] \oplus TK_3^{(1)}[2] \oplus Y_1[2]$.

In Algorithm 3, after all the ■ cells are fixed, we compute backward and forward independently in Line 9 and 12, for the solutions derived in Line 6 and Line 7, respectively. The order of Line 9 and 12 has no essential significance. In particular, the order of the loops can be swapped in Algorithm 3. However, we note that in certain cases the order may affect the memory complexity. We should always build the tables within the loop requiring less memory.

In the matching point, we have DoM $= 2$ from the first two columns of $\#E^+ = Z_{12}$ and $\#E^- = X_{13}$ with $Z_{12}[4] \oplus Z_{12}[8] = X_{13}[8]$ and $Z_{12}[1] \oplus Z_{12}[9] = X_{13}[13]$ due to Equation (4) of MC. However, without increasing the overall complexity, we only use one equation to act as filter, i.e., $Z_{12}[4] \oplus Z_{12}[8] = X_{13}[8]$.

Here, $Z_{12}[4]$ and $Z_{12}[8]$ are computed by 🟦 and ■ cells, which are independent to the 🟥. The reasons are $Z_{12}[4] = Y_{12}[7]$ with $\texttt{SR}^{-1}$, and $Y_{12}[7] = X_{12}[7] \oplus STK_1 2[7]$ and $X_{12}[7] = Z_{11}[3]$ due to MC (Equation 4); $Z_{12}[8] = Y_{12}[10]$ by $\texttt{SR}^{-1}$, and $Y_{12}[10] = X_{12}[10]$, and $X_{12}[10] = Z_{12}[6] \oplus Z_{12}[10]$ due to MC. Similarly, we can compute $X_{13}[8]$ which only depends on the 🟥 and ■ cells.

At last, according to Equation (3), we have the time complexity $(2^w)^{3\times 16-\min\{1,1,1\}} = (2^w)^{47}$, with memory 2^w to store L, to perform the 23-round MITM key-recovery attack on SKINNY-n-3n.

B MITM attacks on Round-reduced ForkSkinny-n-3n

ForkSkinny, designed by Andreeva et al. [2], is the internal primitive of ForkAE, a 2nd round candidate in the NIST lightweight authenticated encryption standardization process [49]. The construction of ForkSkinny is shown in Figure 11. The encryption of ForkSkinny is split into two steps. The first r_{init} rounds process the input message with the round function of Skinny under modified constants. Then, the encryption procedure is forked, into $\texttt{ForkSkinny}_0$ and $\texttt{ForkSkinny}_1$, where two copies of the output from the first stage are separately processed by the two forks with r_0 and r_1 rounds, respectively. The tweakeys are generated by the tweakey schedule for in total $r_{init} + r_0 + r_1$ rounds, and used sequentially in the initial step, $\texttt{ForkSkinny}_0$ and $\texttt{ForkSkinny}_1$, for instance, the last r_1 round tweakeys are applied in $\texttt{ForkSkinny}_1$.

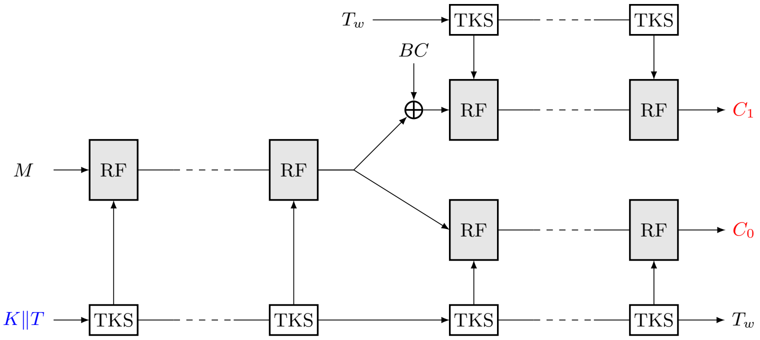


Fig. 11: The ForkSkinny framework [2].

Note that there are varies ways [8] to reduce ForkSkinny, however, the authors suggested that the number of rounds to be reduced should be the same before the fork[2], and in each of the branches. Hence, for the original versions of ForkSkinny-128-384 with $r_{init} = 25$, $r_0 = r_1 = 31$ and ForkSkinny-64-192

[2] https://www.esat.kuleuven.be/cosic/forkae/home/forkskinny-challenge/

with $r_{init} = 17, r_0 = r_1 = 23$, we reduce them to a 24-round ForkSkinny-128-384 with $r_{init} = 25-16 = 9, r_0 = r_1 = 31-16 = 15$, and a 24-round ForkSkinny-64-192 with $r_{init} = 17-8 = 9, r_0 = r_1 = 23-8 = 15$, to launch the MITM key-recovery attacks.

In the first branch $\texttt{ForkSkinny}_0$, there is no matching point as shown in Figure 12 and the matching point happens between Z_{13} and X_{14} in the second branch $\texttt{ForkSkinny}_1$, as shown in Figure 13. The degrees of freedom for blue cells and red cells are both 1 cell and the matching point provides a filter of 1 cell. We apply similar attack procedures to Algorithm 3 to perform the 24-round attack on ForkSkinny-n-$3n$. The data complexity is 2^{n-3w} and the time compleixty is 2^{3n-w}. These are the first single-key attacks on reduced ForkSkinny. Note that the NIST submission of ForAE only suggests a 128-bit key, hence, our attacks do not affect the security of ForAE and ForkSkinny.

C MITM Attacks on Round-reduced Romulus-H

Romulus-H [38] is a lightweight hash, which adopts Hirose's Double-Block-Length (DBL) compression function [32] plugged into the Merkle-Damgård with permutation domain extender as shown in Figure 14. The underline block cipher is SKINNY-128-384.

Applying the chunk separation given in Figure 8, we can find the preimage of SKINNY-128-384 with DM/MMO-modes in time $2^{128-8} = 2^{120}$. Then, for the 23-round preimage attack on the compression of Romulus-H, we place the preimage attack of SKINNY-128-384-DM at the upper block of Romulus-H and leave the partial target of the lower block to be satisfied randomly. Hence, the totally complexity is $2^{120+128} = 2^{248}$. Our preimage attack does not impact the security of Romulus-H, since the authors only claimed 128-bit security.

23-round Collision Attack on the Compression Function of Romulus-H. By applying the automatic model given in Section 6.1, we introduce a new MTIM characteristic for collision attack on reduced SKINNY-128-384 in hashing modes as shown in Figure 15, which can be converted to a 23-round collision attack on Romulus-H's compression function.

Similar to Figure 8, we have one byte degree of freedom for both blue and red cells from Figure 15. In the matching point at round 22, we have $Z_{22}[8] = (T_0[4] \oplus X_0[4]) \oplus (T_0[12] \oplus X_0[12])$ due to Equation (4), which equals to $Z_{22}[8] = (T_0[4] \oplus T_0[12]) \oplus (X_0[4] \oplus X_0[12])$. We prefix the partial target $T_0[4] \oplus T_0[12] = 0$ to get one byte filter in the matching point. Similar to the attack in Algorithm 3, we get about 2^8 partial target preimages for one MITM episode, which costs a time complexity of 2^8. Hence, for the collision attack on 23-round SKINNY-128-384 with DM/MMO-modes, we need a time complexity of about 2^{60} and a memory complexity of about 2^{60} according to Equation (9).

In the free-start collision attack on 23-round Romulus-H, we place MITM characteristic of Figure 15 in the upper block of Figure 14. Suppose the 256-bit target is $T_0 \| T_1$, we find the partial target preimages for T_0 and then compute

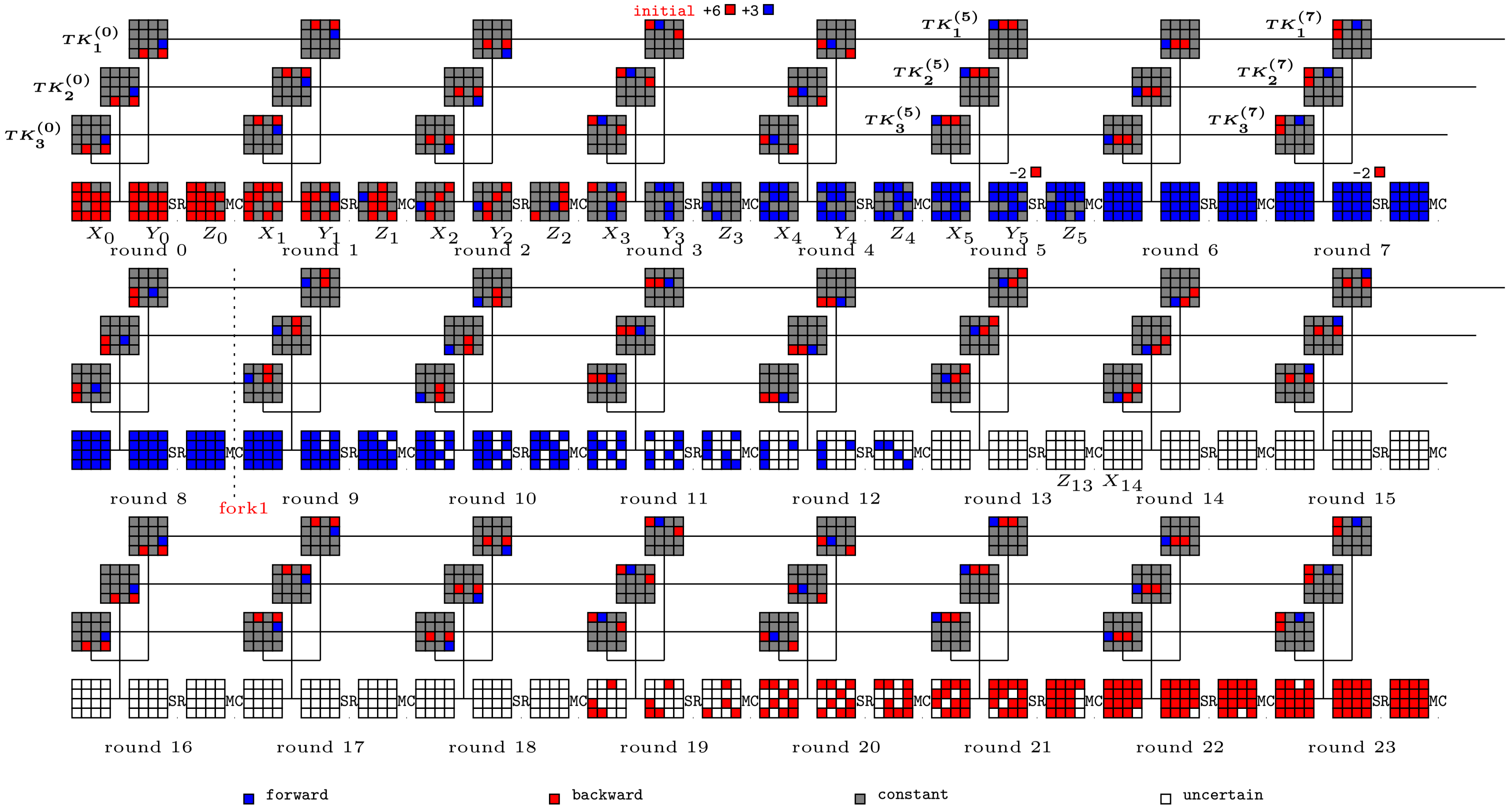


Fig. 12: Fork I: MITM key-recovery attack on 24-round `ForkSkinny-n-3n`

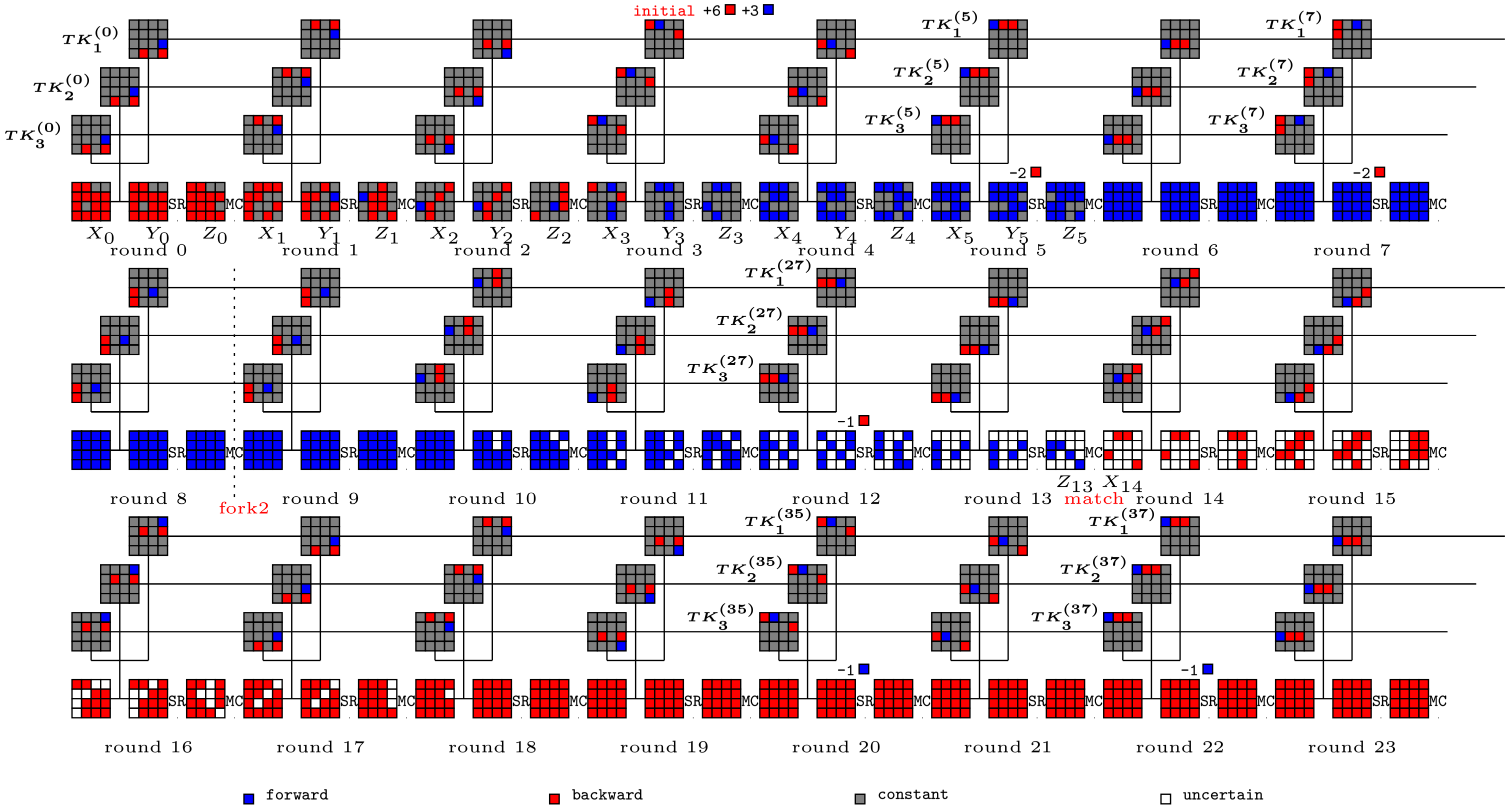


Fig. 13: Fork II: MITM key-recovery attack on 24-round ForkSkinny-n-3n

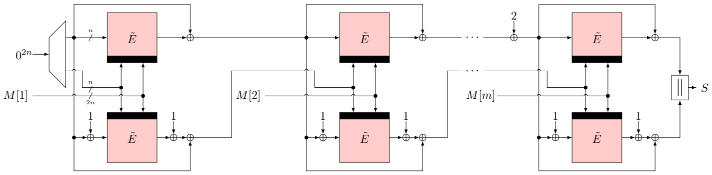


Fig. 14: `Romulus-H` [38]

T_1. We need to collect 2^{124} partial target preimages to finally find a collision, which needs 2^{124} time and 2^{124} memory.

D MITM Pseudo Preimage attacks on reduced `Grøstl-256`

`Grøstl` is a `SHA3` finalist hash function. It comes with two versions: `Grøstl-256` and `Grøstl-512`, with the trailing digits signifying the sizes of the outputs in bits. The structure of `Grøstl`-$\frac{n}{2}$ with two message blocks is depicted in Figure 16, where P and Q are two n-bit `AES`-like permutations. Before it outputs the hash value, an output transformation based on P and a truncation $\Omega : \mathbb{F}_2^n \to \mathbb{F}_2^{n/2}$ are applied to h_2. We refer the reader to [29] for more details of the design. There have been quit a few papers studying the security of `Grøstl` [48,63,54], etc.

Attacks on Output Transformation. As shown in Figure 17, we give a new 6-round chunk seperation for preimage attack on 6-round output transformation of `Grøstl-256`. For output transformation of `Grøstl-256`, there is no key schedule and the neutral words are all from the internal state.

The index of each cell of state X is given in Figure 17. The starting point is X_3. We have $\mathcal{B}^{\texttt{ENC}} = \{0, 2, 6, 9, 10, 11, 15, 34, 36, 38\}$, $\mathcal{R}^{\texttt{ENC}} = \{21, 22, 23, 24, 26, 30, 31, 40, 41, 42, 44, 49, 50, 51, 53, 58, 59, 60, 62\}$, $\mathcal{G}^{\texttt{ENC}} = \mathcal{N} - \mathcal{B}^{\texttt{ENC}} \cup \mathcal{R}^{\texttt{ENC}}$. Then, we have $\lambda^+ = 10$ and $\lambda^- = 19$, $w = 8$.

According to Property 1, we have $l^+ = 8$ and $\mathfrak{c}^+ = (a_1, \cdots, a_8) \in \mathbb{F}_2^{8\times 8}$. For arbitrary given $(X_3[\mathcal{G}^{\texttt{ENC}}], X_3[\mathcal{B}^{\texttt{ENC}}])$, we compute $\mathfrak{c}^+$ with Equation (14), (15) and (16), where the cells of "-" are ignored. Similarly, we have $l^- = 16$ and $\mathfrak{c}^- = (b_1, \cdots, b_{16}) \in \mathbb{F}_2^{8\times 16}$. Therefore, we can call Algorithm 4 to build V and U. $\mathrm{DoF}^+ = \lambda^+ - l^+ = 2$ and $\mathrm{DoF}^- = \lambda^- - l^- = 3$.

$$\begin{pmatrix} Z_2[0] & Z_2[8] & - & - & Z_2[32] & - & - & - \\ Z_2[1] & Z_2[9] & - & - & Z_2[33] & - & - & - \\ Z_2[2] & Z_2[10] & - & - & Z_2[34] & - & - & - \\ Z_2[3] & Z_2[11] & - & - & Z_2[35] & - & - & - \\ Z_2[4] & Z_2[12] & - & - & Z_2[36] & - & - & - \\ Z_2[5] & Z_2[13] & - & - & Z_2[37] & - & - & - \\ Z_2[6] & Z_2[14] & - & - & Z_2[38] & - & - & - \\ Z_2[7] & Z_2[15] & - & - & Z_2[39] & - & - & - \end{pmatrix} = \texttt{MC}^{-1} \cdot \begin{pmatrix} X_3[0] & X_3[8] & - & - & X_3[32] & - & - & - \\ X_3[1] & X_3[9] & - & - & X_3[33] & - & - & - \\ X_3[2] & X_3[10] & - & - & X_3[34] & - & - & - \\ X_3[3] & X_3[11] & - & - & X_3[35] & - & - & - \\ X_3[4] & X_3[12] & - & - & X_3[36] & - & - & - \\ X_3[5] & X_3[13] & - & - & X_3[37] & - & - & - \\ X_3[6] & X_3[14] & - & - & X_3[38] & - & - & - \\ X_3[7] & X_3[15] & - & - & X_3[39] & - & - & - \end{pmatrix} \tag{14}$$

$$\texttt{SR}\begin{pmatrix} X_2[0] & X_2[8] & - & - & X_2[32] & - & - & - \\ - & X_2[9] & X_2[17] & - & - & X_2[41] & - & - \\ - & - & X_2[18] & X_2[26] & - & - & X_2[50] & - \\ - & - & - & X_2[27] & X_2[35] & - & - & X_2[59] \\ X_2[4] & - & - & - & X_2[36] & X_2[44] & - & - \\ - & X_2[13] & - & - & - & X_2[45] & X_2[53] & - \\ - & - & X_2[22] & - & - & - & X_2[54] & X_2[62] \\ X_2[7] & - & - & X_2[31] & - & - & - & X_2[63] \end{pmatrix} = \texttt{SB}^{-1}\begin{pmatrix} Z_2[0] & Z_2[8] & - & - & Z_2[32] & - & - & - \\ Z_2[1] & Z_2[9] & - & - & Z_2[33] & - & - & - \\ Z_2[2] & Z_2[10] & - & - & Z_2[34] & - & - & - \\ Z_2[3] & Z_2[11] & - & - & Z_2[35] & - & - & - \\ Z_2[4] & Z_2[12] & - & - & Z_2[36] & - & - & - \\ Z_2[5] & Z_2[13] & - & - & Z_2[37] & - & - & - \\ Z_2[6] & Z_2[14] & - & - & Z_2[38] & - & - & - \\ Z_2[7] & Z_2[15] & - & - & Z_2[39] & - & - & - \end{pmatrix} \tag{15}$$

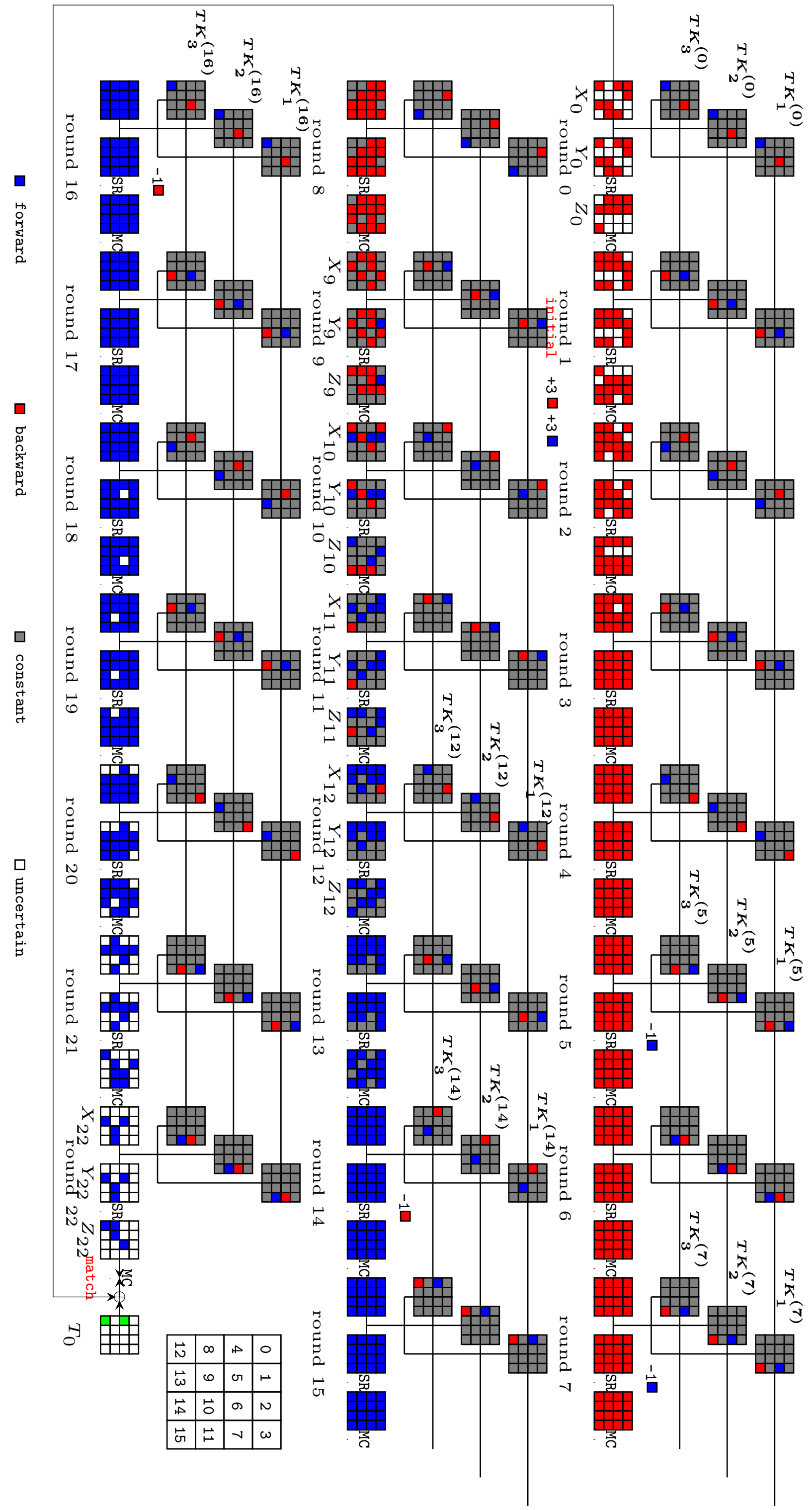


Fig. 15: Free-start collision attack on 23-round Romulus-H

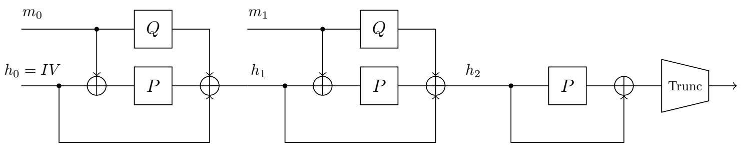
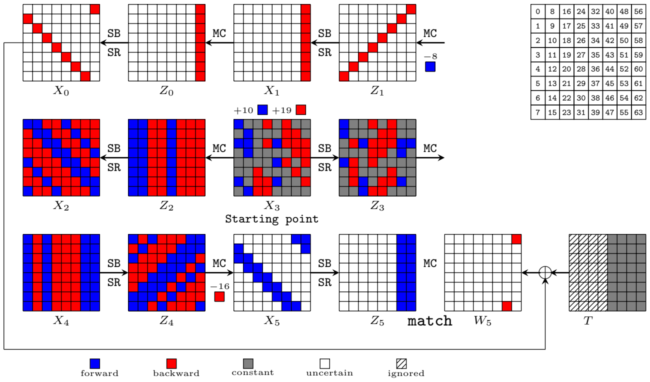


Fig. 16: `Grøstl-n/2` with two message blocks



0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

Fig. 17: Preimage attack on 6-round output transformation of `Grøstl-256`

$$\begin{pmatrix} - & - & - & - & - & - & - & a_8 \\ - & - & - & - & - & - & a_7 & - \\ - & - & - & - & - & a_6 & - & - \\ - & - & - & - & a_5 & - & - & - \\ - & - & - & a_4 & - & - & - & - \\ - & - & a_3 & - & - & - & - & - \\ - & a_2 & - & - & - & - & - & - \\ a_1 & - & - & - & - & - & - & - \end{pmatrix} = \mathtt{MC}^{-1} \cdot \begin{pmatrix} X_2[0] & X_2[8] & - & - & X_2[32] & - & - & - \\ - & X_2[9] & X_2[17] & - & - & X_2[41] & - & - \\ - & - & X_2[18] & X_2[26] & - & - & X_2[50] & - \\ - & - & - & X_2[27] & X_2[35] & - & - & X_2[59] \\ X_2[4] & - & - & - & X_2[36] & X_2[44] & - & - \\ - & X_2[13] & - & - & - & X_2[45] & X_2[53] & - \\ - & - & X_2[22] & - & - & - & X_2[54] & X_2[62] \\ X_2[7] & - & - & X_2[31] & - & - & - & X_2[63] \end{pmatrix} \tag{16}$$

We call Algorithm 5 to perform the MITM preimage attack on the 6-round output transformation of `Grøstl`-256. As shown in Figure 17, $h = 32$ cells, $m = 2$ cells. According to Equation (6), the time complexity is

$$(2^8)^{32-10} + (2^8)^{32-19} + (2^8)^{32-\min(2,\ 3,\ 2)} \approx 2^{240}. \tag{17}$$

The memory complexity is

$$(2^8)^{10} + (2^8)^{19} + (2^8)^{\min(2,\ 3)} \approx 2^{152}. \tag{18}$$

Pseudo Preimage Attacks on the Hash Function. At FSE 2012, Wu et al. [63] converted the preimage attack on `Grøstl-256`'s output transformation into pseudo preimage attack on `Grøstl-256` hash function. Suppose given 256-bit target T, we have find 2^x preimages for the output transformation, i.e., we find 2^x X that meet $P(X) \oplus X = *\|T$. Wu et al. try to find the pseudo preimage (H, M) of `Grøstl-256` hash function that meets $(H' = H \oplus M)$:

$$(P(H') \oplus H') \oplus (Q(M) \oplus M) \oplus X = 0. \tag{19}$$

We set $x = 10$, and the time complexity to find 2^{10} preimages X for the output transformation is 2^{250}. We store the 2^{10} X in table L_1. Firstly, we select 2^{251} M and compute $Q(M) \oplus M$ and check L_1 to find 10-bit partial collisions with X. Hence, we find $2^{251+10}/2^{10} = 2^{251}$ $(Q(M) \oplus M, X)$ pairs with a 10-bit zero in $Q(M) \oplus M \oplus X$ and store them in table L_2. Next, we find partial target preimages for $P(H') \oplus H'$ with 10-bit fixed zero as partial target. We reuse Figure 17 to find H'. We can just fix $T[55] = T[56] = 0$ (just assume the 10-bit fixed zero bits are among the two bytes). We find 2^{251} such H' with time complexity of 2^{251} according to Equation (6)[3]. We use $P(H') \oplus H'$ to check L_2, and it is expected to find $2^{251+251}/2^{512-10} = 1$ collision, which is just the pseudo preimage (H, M) for `Grøstl`-256 hash function. The time complexity is 2^{252} with memory 2^{251}.

E MITM Attacks on Round-reduced `Saturnin-Hash`

`Saturnin` is a suite of lightweight symmetric algorithms proposed by Canteaut et al. [15]. It is among the 2nd round candidates of the NIST Lightweight cryptography (LWC) standardization process [49]. Based on a 256-bit block cipher with 256-bit key, two authenticated ciphers and a hash function are designed. In this section, we focus on its hash function, called `Saturnin-Hash`.

A 7-round preimage attack on `Saturnin-Hash`. We first give a 7-round chunk separation on the compression of `Saturnin-Hash` as shown in Figure 18, the operations in round function work on 4×4 square of 16-bit "supernibbles". The arrangement of "supernibbles" is

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix}.$$

The `SB` operation applies 16-bit Super-Sbox S to each supernibble. `MC` is Super-Mixcolumns, which acts as the same role to the `Mixcolumns` of `AES`. `MR` is Super-Mixrows. `Saturnin` has a very simple key-schedule. In even rounds, the 256-bit key K is XORed to the internal state; in odd rounds, it is rotated by 5 supernibble-positions to get K'.

The starting states are X_4 and K'. We have $\lambda^+ = 24$ and $\lambda^- = 7$. In the backward direction, we apply the property of `XOR then MixColumns (XOR-MC)` by Bao et al. [7] to ignore the state Z_3 and Z_2.

[3] With $h = 2$, $m = 2$, $\mathrm{DoF}^+ = 2$ and $\mathrm{DoF}^- = 3$, we have $(2^w)^{h-\min(\mathrm{DoF}^+,\ \mathrm{DoF}^-,\ m)} = 1$, which means we get one partial target preimage with one computation of `Grøstl-256` on average.

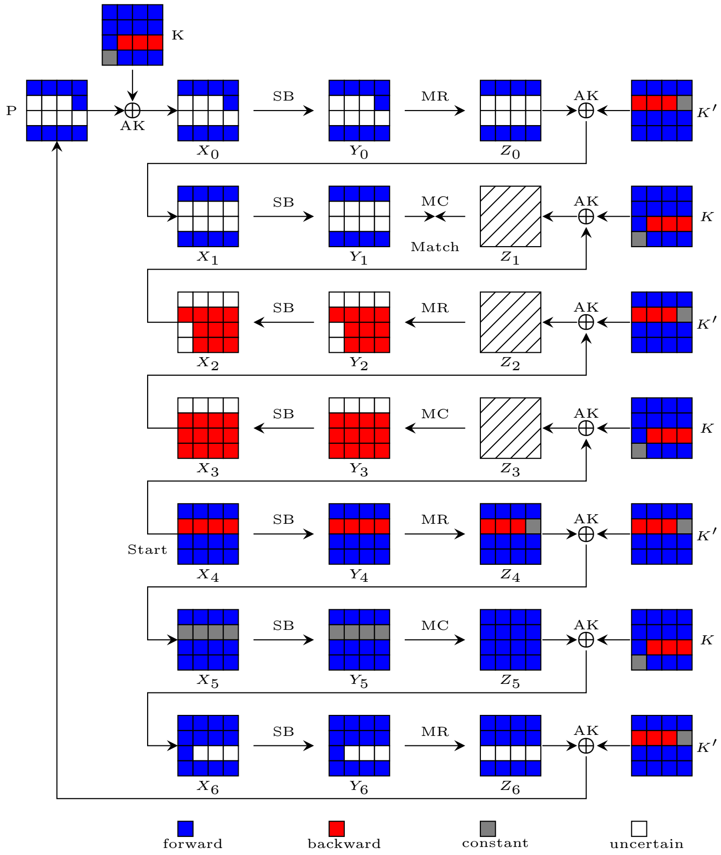


Fig. 18: The 7-round MITM attack on `Saturnin-Hash`

Then, from X_4 to Y_3, we have

$$\mathtt{MC}^{-1}\begin{pmatrix} X_4[0]\oplus K[0] & X_4[4]\oplus K[4] & X_4[8]\oplus K[8] & X_4[12]\oplus K[12] \\ K[1] & K[5] & K[9] & K[13] \\ X_4[2]\oplus K[2] & X_4[6] & X_4[10] & X_4[14] \\ X_4[3]\oplus K[3] & X_4[7]\oplus K[7] & X_4[11]\oplus K[11] & X_4[15]\oplus K[15] \end{pmatrix} = \begin{pmatrix} \text{-} & \text{-} & \text{-} & \text{-} \\ a_1 & a_4 & a_7 & a_{10} \\ a_2 & a_5 & a_8 & a_{11} \\ a_3 & a_6 & a_9 & a_{12} \end{pmatrix} \tag{20}$$

From X_3 to Y_2, we have

$$\mathtt{MR}^{-1}\begin{pmatrix} \text{-} & \text{-} & \text{-} & \text{-} \\ \text{-} & \text{-} & \text{-} & \text{-} \\ K'[2] & K'[6] & K'[10] & K'[14] \\ K'[3] & K'[7] & K'[11] & K'[15] \end{pmatrix} = \begin{pmatrix} \text{-} & \text{-} & \text{-} & \text{-} \\ \text{-} & \text{-} & \text{-} & \text{-} \\ \text{-} & a_{13} & a_{14} & a_{15} \\ \text{-} & a_{16} & a_{17} & a_{18} \end{pmatrix} \tag{21}$$

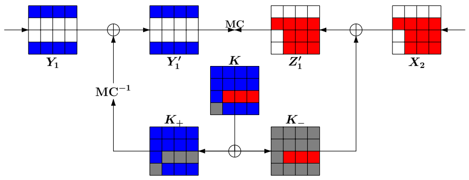


Fig. 19: Matching in the 7-round MITM attack on `Saturnin-Hash`

For a given $\mathfrak{c}^+ = (a_1, \cdots, a_{18}) \in \mathbb{F}_2^{16\times 18}$ with $l^+ = 18$, we solve the two linear Equations (20) and (21) to compute the neutral words of blue. Hence, $\mathrm{DoF}^+ = \lambda^+ - l^+ = 6$.

The neutral values for red cells are easy to obtain by solving a linear system of equations from Y_4 to X_5. We have $\lambda^- = 7$, $l^- = 4$, $\mathfrak{c}^- = (b_1, \cdots, b_4) \in \mathbb{F}_2^{16\times 4}$ and $\mathrm{DoF}^- = \lambda^- - l^- = 3$. In the matching point, we use tricks for matching the ending states by Bao et al. The matching point is decomposed as Figure 19 and we get a filter of $\mathrm{DoM} = 3$ cells. We have time complexity

$$(2^{16})^{16-\min(6,\ 3,\ 3)} \approx 2^{208}. \tag{22}$$

The memory complexity is about $(2^{16})^3 = 2^{48}$.

Converting the preimage attack on the compression function into a preimage attack on 7-round `Saturnin-hash`, we get the time complexity $2^{256-24} = 2^{232}$. Since the authors of `Saturnin-hash` only claimed a security of 2^{224}, our attack does not impact the security `Saturnin-hash`, but only gives a better understanding on `Saturnin-hash` against MITM attack.

F Conllision Attacks on Round-reduced `Grøstl`

F.1 Collision attack on 6-round `Grøstl-256`'s output transformation

With the automatic model of MITM collision attack in Sect. 6.1, we find a new 6-round chunk seperation as shown in Figure 20. Based on it, we give the 6-round collision attack on the output transformation of `Grøstl`-256. The starting point is X_3 and $\lambda^+ = \lambda^- = 10$. From Z_1 and X_5, we get $l^+ = l^- = 8$. Hence, $\mathrm{DoF}^+ = \lambda^+ - l^+ = 2$, $\mathrm{DoF}^- = \lambda^- - l^- = 2$, $m = 1$, and $t = 1$. By applying Algorithm 6, we have time complexity according to Equation (9).

$$(2^8)^{10} + (2^8)^{10} + (2^8)^{\frac{32}{2} - \min\{2-\frac{1}{2},\ 2-\frac{1}{2},\ 1-\frac{1}{2},\ \frac{1}{2}\}} \approx 2^{124}. \tag{23}$$

The memory complexity is about 2^{124} according to Equation (11).

F.2 Collision attack on 8-round `Grøstl-512`'s output transformation

As shown in Figure 21, the starting state is W_3. Here, since in the computation from W_3 to Z_3, the degrees of freedom of the blue bytes are reduced from 31

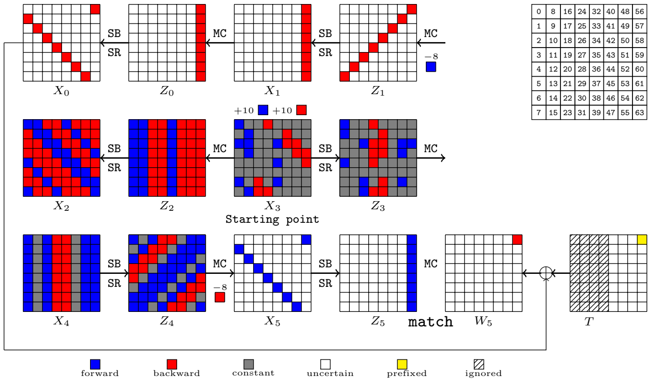


Fig. 20: The Meet-in-the-Middle attack on 6-round output transformation of `Grøstl-256`

bytes to $31 - 17 = 14$ bytes. By solving a linear system of equations, we can obtain the space of $(\lambda^+ =)$ 14 bytes with fixed gray constants. For each value of the 14 bytes, we apply the table-based method of computing the solution space of the neutral words (Algorithm 4), where $l^+ = 12$ and $\mathfrak{c}^+ = (a_1, \cdots, a_{12}) \in \mathbb{F}_2^{8\times 12}$. Hence, $\text{DoF}^+ = \lambda^+ - l^+ = 2$. Similarly, we have $\lambda^- = 42 - 26 = 16$, $l^- = 14$ and $\mathfrak{c}^- = (b_1, \cdots, b_{14}) \in \mathbb{F}_2^{8\times 14}$. Hence, $\text{DoF}^- = \lambda^- - l^- = 2$. In addition, we have $m = 2$ and $t = 2$. By applying Algorithm 6, we have the time complexity according to Equation (10)

$$(2^8)^{\frac{64}{2}-(14-\frac{2}{2})} + (2^8)^{\frac{64}{2}-(16-\frac{2}{2})} + (2^8)^{\frac{64}{2}-\min\{2-\frac{2}{2},\ 2-\frac{2}{2},\ 2-\frac{2}{2},\ \frac{2}{2}\}} \approx 2^{248}. \qquad (24)$$

The memory cost is 2^{248} to store the partial target preimages.

G Preimage Attacks on Some Hashing Modes with 10-round `AES-256`

At EUROCRYPT 2021, Bao et al. [7] introduced the MITM preimage attacks on 8-/9-/9-round `AES-128/-192/-256` in PGV hashing modes. At EUROCRYPT 2021, Leurent and Pernot [45] introduced new representations of the `AES` key schedules. Taking the new representations into account, we introduce an updated automatic model for MITM preimage attacks on `AES` hashing modes. Finally, we find an MITM preimage attack on 10-round `AES-256` as shown in Figure 22 and improve the best previous attack by one round.

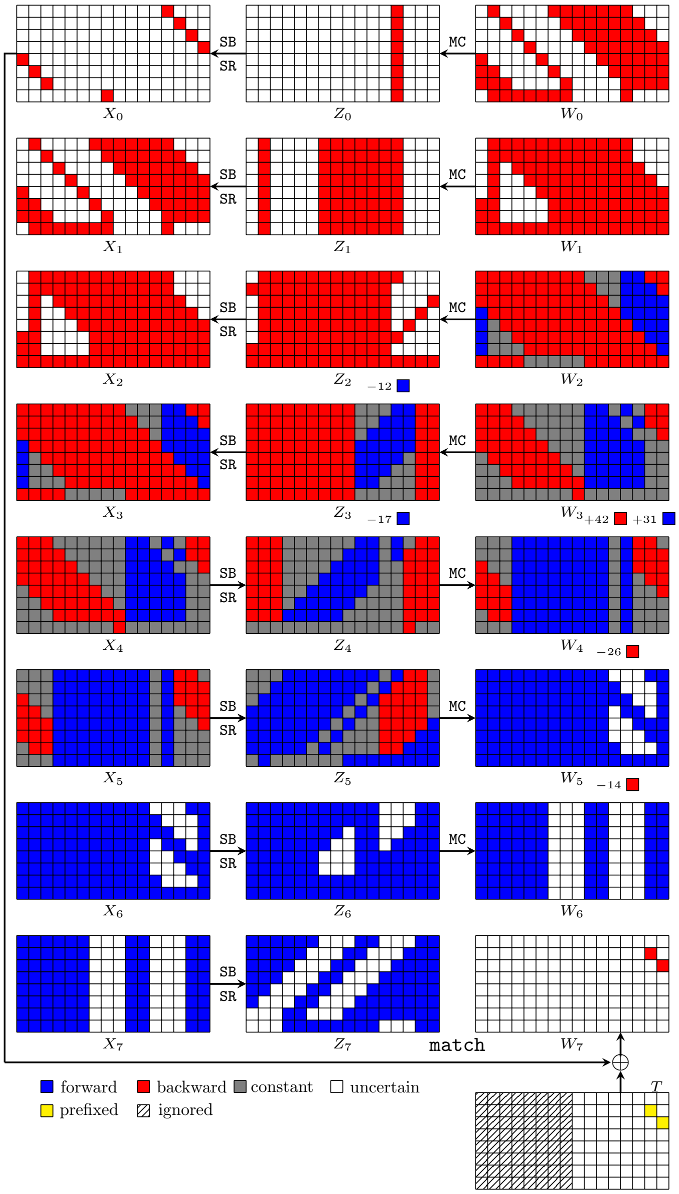
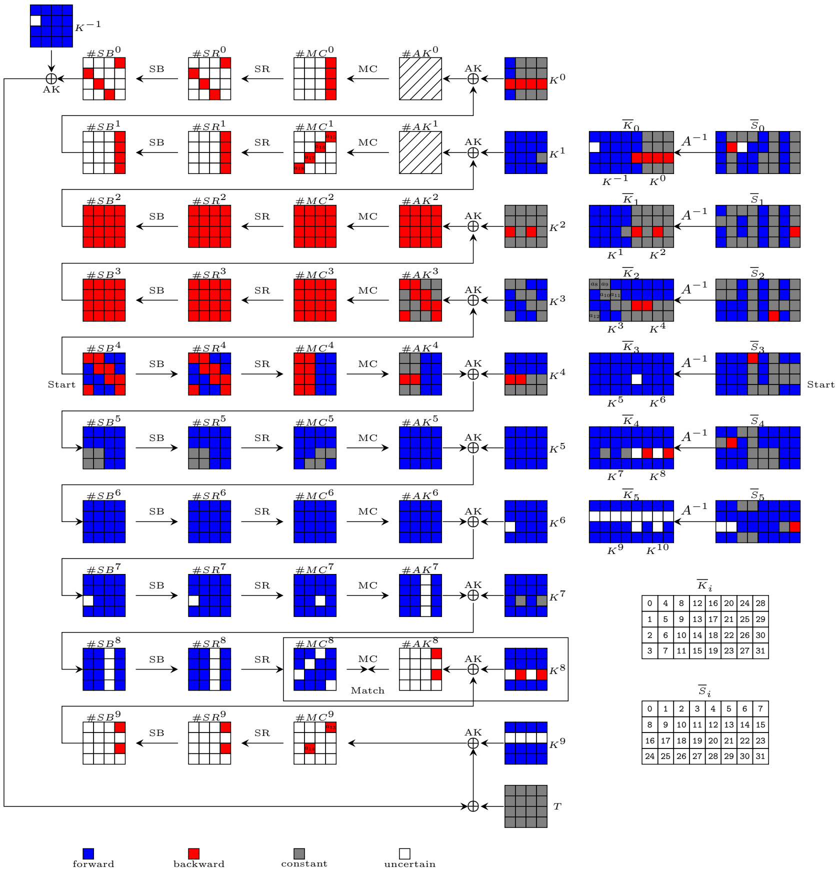


Fig. 21: The Meet-in-the-Middle attack on 8-round output transformation of `Grøstl-512`


K^{-1}
AK
$\#SB^0$ SB $\#SR^0$ SR $\#MC^0$ MC $\#AK^0$ AK K^0
$\#SB^1$ SB $\#SR^1$ SR $\#MC^1$ MC $\#AK^1$ AK K^1
$\#SB^2$ SB $\#SR^2$ SR $\#MC^2$ MC $\#AK^2$ AK K^2
$\#SB^3$ SB $\#SR^3$ SR $\#MC^3$ MC $\#AK^3$ AK K^3
Start
$\#SB^4$ SB $\#SR^4$ SR $\#MC^4$ MC $\#AK^4$ AK K^4
$\#SB^5$ SB $\#SR^5$ SR $\#MC^5$ MC $\#AK^5$ AK K^5
$\#SB^6$ SB $\#SR^6$ SR $\#MC^6$ MC $\#AK^6$ AK K^6
$\#SB^7$ SB $\#SR^7$ SR $\#MC^7$ MC $\#AK^7$ AK K^7
$\#SB^8$ SB $\#SR^8$ SR $\#MC^8$ MC $\#AK^8$ AK K^8
Match
$\#SB^9$ SB $\#SR^9$ SR $\#MC^9$ AK K^9
T
$\overline{K}_0$ A^{-1} $\overline{S}_0$
K^{-1} K^0
$\overline{K}_1$ A^{-1} $\overline{S}_1$
K^1 K^2
$\overline{K}_2$ A^{-1} $\overline{S}_2$
K^3 K^4
$\overline{K}_3$ A^{-1} $\overline{S}_3$ Start
K^5 K^6
$\overline{K}_4$ A^{-1} $\overline{S}_4$
K^7 K^8
$\overline{K}_5$ A^{-1} $\overline{S}_5$
K^9 K^{10}
🟦 forward
🟥 backward
⬛ constant
□ uncertain

$\overline{K}_i$

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

$\overline{S}_i$

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

Fig. 22: An MITM attack on 10-round AES-256. Note that the cell order for $\bar{K}_i$ and $\bar{S}_i$ is different.

$$A^{-1} = \begin{bmatrix}
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix} \qquad (25)$$

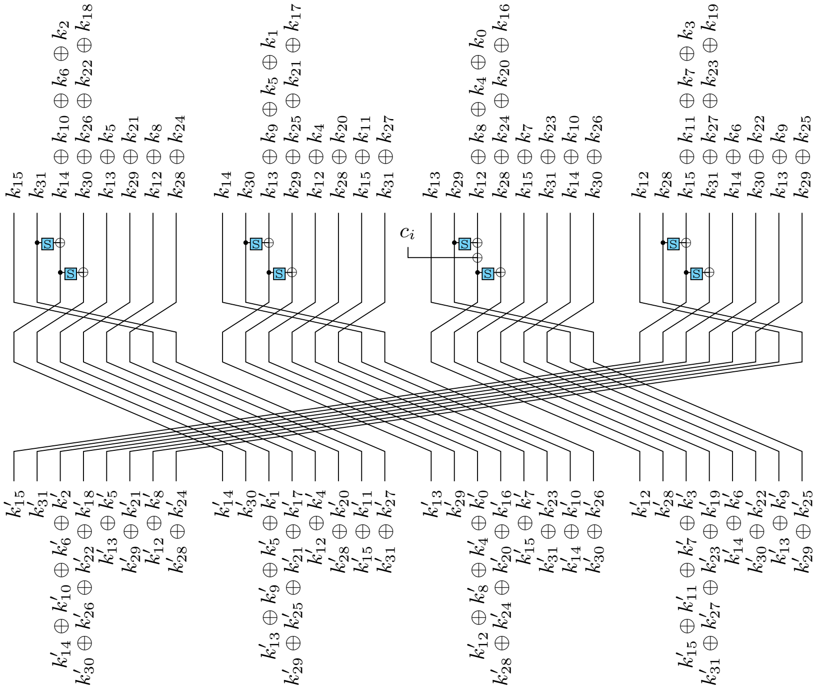


Fig. 23: A new representation of `AES-256`'s key schedule from Leurent and Pernot [45]

As shown in Figure 22, $\bar{K}_i = A^{-1}(\bar{S}_i)$ with $i = 0, 1, 2, 3, 4, 5$. From Figure 23, we derive A^{-1} given in Equation (25). In the new representation of `AES-256`'s key schedule shown in Figure 24, the starting state is $\bar{S}_3$ and $\#SB^4$. In $\bar{S}_3$, there are 19 🟦 cells, 1 🟥 cells and 12 ▨ cells. In $\#SB^4$, there are 8 🟦 cells, 8 🟥 cells. Hence, the total initial degrees of freedom are $\lambda^+ = 19 + 8 = 27$ cells for 🟦 cells, and $\lambda^- = 1 + 8 = 9$ for 🟥 cells. We first consider the reduction of degrees of freedom for the 🟦 cells in $\bar{S}_3$. As shown in Figure 24, the 🟦 cells in state $\bar{S}_3$ have to meet the Equation (26), where $(a_1, a_2, .., a_7)$ are constants and symbols marked by blue are 🟦 cells and marked by black are ▨ cells.

$$\begin{cases} \texttt{S}(\bar{S}_3[1]) \oplus \bar{S}_3[2] = a_1 \\ \texttt{S}(\bar{S}_3[8]) \oplus \bar{S}_3[9] = a_2 \\ \texttt{S}(\bar{S}_3[16]) \oplus \bar{S}_3[17] = a_3 \\ \texttt{S}(\bar{S}_3[24]) \oplus \bar{S}_3[25] = a_4 \\ \texttt{S}(\bar{S}_3[0]) \oplus \bar{S}_3[1] = a_5 \\ \texttt{S}(\bar{S}_3[30]) \oplus \bar{S}_3[31] = a_6 \\ \texttt{S}(\bar{S}_3[6]) \oplus \bar{S}_3[7] = a_7. \end{cases} \tag{26}$$

In addition, as shown in Figure 22, the 🟦 and ▨ cells of $\bar{S}_3$ propagate to $\bar{S}_2$ and then compute $\bar{K}_2$ by applying A^{-1} to $\bar{S}_2$. The 🟦 and ▨ cells in $\bar{K}_2$ and $\bar{S}_3$ have to meet Equation (27), where $(a_8, a_9, a_{10}, a_{11}, a_{12})$ are constants. For example, in the first equation of Equation (27), $\bar{K}_2[4] = \bar{S}_2[12] \oplus \bar{S}_2[24]$ by $\bar{K}_2 = A^{-1}(\bar{S}_2)$, and $\bar{S}_2[12] \oplus \bar{S}_2[24] = \bar{S}_3[18] \oplus \bar{S}_3[6]$ by considering the propagation from $\bar{S}_3$ to $\bar{S}_2$ in Figure 24.

$$\begin{cases} \bar{K}_2[0] = \bar{S}_2[6] \oplus \bar{S}_2[12] \oplus \bar{S}_2[18] \oplus \bar{S}_2[24] \\ \qquad\quad = \bar{S}_3[12] \oplus \bar{S}_3[18] \oplus \bar{S}_3[24] \oplus \mathtt{S}(\bar{S}_3[31]) \oplus \bar{S}_3[6] = a_8, \\ \bar{K}_2[4] = \bar{S}_2[12] \oplus \bar{S}_2[24] = \bar{S}_3[18] \oplus \bar{S}_3[6] = a_9, \\ \bar{K}_2[5] = \bar{S}_2[4] \oplus \bar{S}_2[16] = \bar{S}_3[10] \oplus \bar{S}_3[30] = a_{10}, \\ \bar{K}_2[9] = \bar{S}_2[16] \oplus \bar{S}_2[30] = \bar{S}_3[30] \oplus \bar{S}_3[4] = a_{11}, \\ \bar{K}_2[3] = \bar{S}_2[0] \oplus \bar{S}_2[14] \oplus \bar{S}_2[20] \oplus \bar{S}_2[26] \\ \qquad\quad = \bar{S}_3[14] \oplus \bar{S}_3[20] \oplus \bar{S}_3[26] \oplus \mathtt{S}(\bar{S}_3[7]) \oplus \bar{S}_3[0] = a_{12}. \end{cases} \tag{27}$$

In Figure 22, the subkeys meet $K^{-1}[12] \oplus K^9[12] = a_{13}$ and $K^{-1}[6] \oplus K^9[6] = a_{14}$, where the subkeys K^{-1} and K^9 are the left 16 cells of $\bar{K}_0$ and $\bar{K}_5$, respectively. Since $\bar{K}_0 = A^{-1}(\bar{S}_0)$ and $\bar{K}_5 = A^{-1}(\bar{S}_5)$, we have Equation (28) with Equation (25).

$$\begin{cases} K^9[12] = \bar{K}_5[12] = \bar{S}_5[24], \\ K^{-1}[12] = \bar{K}_0[12] = \bar{S}_0[24], \\ K^9[6] = \bar{K}_5[6] = \bar{S}_5[8] \oplus \bar{S}_5[28], \\ K^{-1}[6] = \bar{K}_0[6] = \bar{S}_0[8] \oplus \bar{S}_0[28]. \end{cases} \tag{28}$$

Further changing of variables of Equation (28) by cells of $\bar{S}_3$, together with the conditions $K^{-1}[12] \oplus K^9[12] = a_{13}$ and $K^{-1}[6] \oplus K^9[6] = a_{14}$, we have

$$\begin{cases} \underbrace{\mathtt{S}(\mathtt{S}(\mathtt{S}(\bar{S}_3[9]) \oplus \bar{S}_3[10]) \oplus \bar{S}_3[11]) \oplus \bar{S}_3[12]}_{\bar{S}_5[24]} \oplus \underbrace{\bar{S}_3[18]}_{\bar{S}_0[24]} = a_{13}, \\ \underbrace{\mathtt{S}(\mathtt{S}(\mathtt{S}(\bar{S}_3[25]) \oplus \bar{S}_3[26]) \oplus \bar{S}_3[27]) \oplus \bar{S}_3[28]}_{\bar{S}_5[8]} \oplus \underbrace{\bar{S}_3[8]}_{\bar{S}_5[28]} \oplus \underbrace{\bar{S}_3[2]}_{\bar{S}_0[8]} \oplus \underbrace{\mathtt{S}(\bar{S}_3[21]) \oplus \bar{S}_3[22]}_{\bar{S}_0[28]} = a_{14}. \end{cases} \tag{29}$$

In round 1, we apply the property of `XOR then MixColumns (XOR-MC)` by Bao et al. [7] to ignore the state $\#AK^1$, and get Equation (30).

$$\begin{pmatrix} - & - & - & a_{15} \\ - & - & a_{16} & - \\ - & a_{17} & - & - \\ a_{18} & - & - & - \end{pmatrix} = \mathtt{MC}^{-1}(K^1). \tag{30}$$

Equation (26), (27), (29) and (30) introduce a (7+5+2+4 =)18-cell reduction of degrees of freedom for 🟦 cells. In addition, from $\#SB^4$ to $\#AK^3$, there is an 8-cell reduction of degrees of freedom for 🟦 cells. Totally, the degree of freedom for 🟦 cells is $\mathrm{DoF}^+ = \lambda^+ - 18 - 8 = 1$.

For the 🟥 cells, the initial degree of freedom $\lambda^- = 1+8 = 9$. The computation from $\#MC^4$ to $\#SB^5$ consume 8-cell degree of freedom. Hence, $\mathrm{DoF}^- = \lambda^- - 8 = 1$.

We introduce the preimage attack on the hashing modes with 10-round `AES-256` in Algorithm 7. Since, we are going to find a 128-bit preimage attack, the encryption data path (with 128-bit internal state) and key schedule

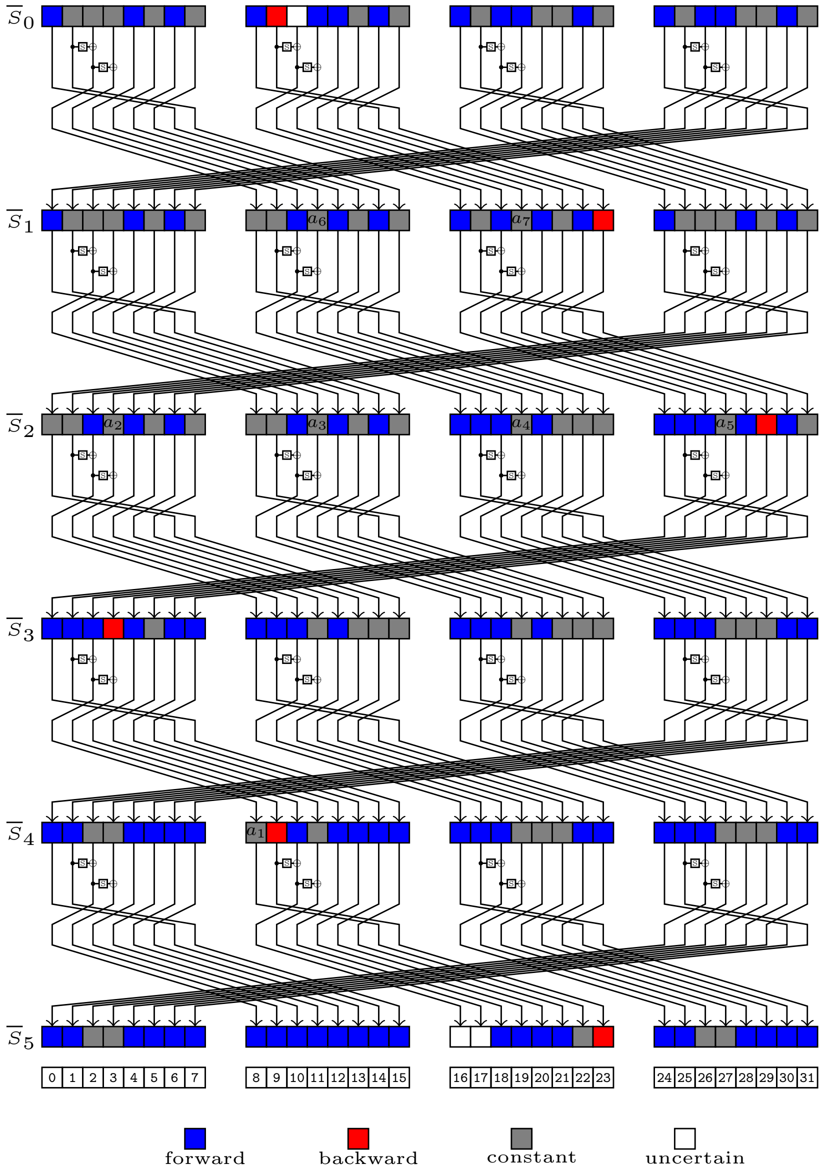


Fig. 24: Chunk separations of `AES-256`'s key schedule with Leurent and Pernot's representations

data path (with 256-bit internal state) provide enough degrees of freedom, and we do not need to traverse all the ■ cells to find the 128-bit preimage. Hence, for briefness, we fixed all the ■ as zero in Line 1, as well as $(a_1, a_2, ..., a_{12})$ in Line 2. Then for each 7-cell 🟦 values $(\bar{S}_3[1], \bar{S}_3[9], \bar{S}_3[17], \bar{S}_3[25], \bar{S}_3[30], \bar{S}_3[6], \bar{S}_3[20])$

Algorithm 7: The MITM preimage attack on 10-round `AES-256`

```
1  $\bar{S}_3[5, 11, 13, 14, 15, 19, 21, 22, 23, 27, 28, 29] \leftarrow 0$,
2  $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{12}) \leftarrow 0$,
3  $V \leftarrow [\ ]$
4  for 7-cell 🟦 values $(\bar{S}_3[1], \bar{S}_3[9], \bar{S}_3[17], \bar{S}_3[25], \bar{S}_3[30], \bar{S}_3[6], \bar{S}_3[20])$ do
5      Deduce the 12 🟦 cells from Equation (26) and (27):
         $(\bar{S}_3[2], \bar{S}_3[8], \bar{S}_3[16], \bar{S}_3[24], \bar{S}_3[0], \bar{S}_3[31], \bar{S}_3[7], \bar{S}_3[18], \bar{S}_3[10], \bar{S}_3[4], \bar{S}_3[26], \bar{S}_3[12])$
6      /* All 🟦 and ◼ cells in $\bar{S}_3$ are known.                    */
7      Compute Equation (29) and (30) to get the 6-cell value
         $(a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18})$, and store the values for the 🟦 cells of $\bar{S}_3$ in
         $V[a_{13}, ..., a_{17}, a_{18}]$
8      /* When computing Equation (30), only 🟦 and ◼ cells in $K^1$ are
          considered, which is derived following the path:
          $\bar{S}_3 \rightarrow \bar{S}_2 \rightarrow \bar{S}_1 \xrightarrow{A^{-1}} \bar{K}_1 \rightarrow K^1$         */
9      /* In $V[a_{13}, ..., a_{17}, a_{18}]$, there are $2^8$ values of 🟦 on average.   */
10 Randomly pick one $X = (a_{13}, ..., a_{17}, a_{18})$ whose $V[X]$ is of about $2^8$ values.
11 for All values of ◼ in $\#AK^3$, $\#AK^4$ and $\#SB^5[2, 6]$
12 /* There are 8+6+2=16 independent gray cells in the data encryption
      path, which provide enough degrees of freedom to finally find a
      16-byte preimage.                                                */
13 do
14     for all values of the one 🟥 cell in $\bar{S}_3$ do
15         Compute backward to the matching point, store it in $L[\ ]$
16     for each blue value in $V[X]$ do
17         Compute forward to the matching point to match
18         /* use the trick by Bao et al.'s [7] in the matching point
              and there is one cell filter in the matching phase.      */
19         For each match, test the full preimage.
```

in Line 4, we deduce other unfixed 12-cell values by Equation (26) and (27) in Line 5. Here, we actually do not try to solve the nonlinear equations in Equation (26) and (27), but deduce the 12-cell values one by one. For example, for the first equation of Equation (26), i.e., $\mathtt{S}(\bar{S}_3[1]) \oplus \bar{S}_3[2] = a_1$, with knowledge of $\bar{S}_3[1]$ and $a_1 = 0$, we get the value for $\bar{S}_3[2]$. Thereafter, all the 19 🟦 cells in $\bar{S}_3$ are determined. Hence, all the 🟦 and ◼ cells in K^1 can be derived following the $\bar{S}_3 \rightarrow \bar{S}_2 \rightarrow \bar{S}_1 \xrightarrow{A^{-1}} \bar{K}_1 \rightarrow K^1$. Then, in Line 7, we compute Equation (29) and (30) to get the 6-cell value $(a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18})$, and store the blue values of $\bar{S}_3$ in $V[a_{13}, ..., a_{17}, a_{18}]$. Since in Line 4 we traverse 7-cell values, and at last store them in V under the 6-cell index, there will be about 2^8 🟦 values under each index of V. Here, it is a tweaked nonlinear-constrained technique introduced in Section 5.

Since we have large enough degrees of freedom to find the 128-bit preimage when taken the key schedule into consideration, we only pick one subset of V

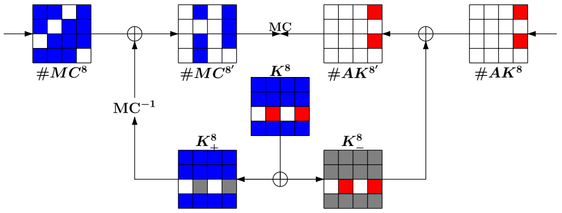


Fig. 25: Decomposition of the matching point of the MITM attack on 10-round `AES-256`

indexed by a given $X = (a_{13}, ..., a_{17}, a_{18})$ to perform the MITM attack, where the number of elements in $V[X]$ is about 2^8.

In the MITM attack, we traverse the ◼ cells in $\#AK^3$, $\#AK^4$ and $\#SB^5[2,6]$ until a preimage is found. There are 8+6+2=16 independent gray cells in the data encryption path, which provide enough degrees of freedom to finally find a 128-bit preimage.

In Line 15, together with the ◼ cells fixed as 0 in Line 1, all the 🟥 and ◼ cells in the subkeys K^i $(-1 \le i \le 9)$ are computed. Then, from $\#SB^5$ to $\#AK^4$, with the knowledge of $\#SB^5[2,6]$ and 🟥 and ◼ in K^4, we compute the 🟥 cells backward in $\#AK^4$. Further compute backward to the matching point. We use the trick by Bao et al.'s [7] in the matching point and decompose the matching point as shown in Figure 25. There is one cell of filter at the last column.

In forward direction, for each value in $V[X]$, all the 🟦 and ◼ cells in the subkeys K^i $(-1 \le i \le 9)$ are derived. Then from $\#AK^3$ to $\#SB^4$, with the knowledge of the ◼ cells of $\#AK^3$, we compute the 🟦 cells forward in $\#SB^4$. Further, compute the 🟦 cells forward to the matching point. For each match of the 🟥 and 🟦 values, check for a full 128-bit target match to find the preimage.

Complexity. The memory complexity is about 2^{56} to store V. The time complexity is about 2^{120}.