Generic MitM Attack Frameworks on Sponge Constructions

Xiaoyang Dong[1,2,3,9], Boxin Zhao[3] (✉), Lingyue Qin[4,3], Qingliang Hou[5,7], Shun Zhang[8], and Xiaoyun Wang[3,5,6,7]

[1] Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, P.R.China, xiaoyangdong@tsinghua.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, 100878, P.R.China
[3] Zhongguancun Laboratory, Beijing, P.R.China, zhaobx@mail.zgclab.edu.cn
[4] BNRist, Tsinghua University, Beijing, P.R.China, qinly@tsinghua.edu.cn
[5] School of Cyber Science and Technology, Shandong University, Qingdao, P.R.China qinglianghou@mail.sdu.edu.cn
[6] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, P.R.China xiaoyunwang@tsinghua.edu.cn
[7] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, P.R.China
[8] PLA Strategic Support Force Information Engineering University, Zhengzhou, P.R.China
[9] Shandong Institute of Blockchain, Jinan, P.R.China

Abstract. This paper proposes general meet-in-the-middle (MitM) attack frameworks for preimage and collision attacks on hash functions based on (generalized) sponge construction. As the first contribution, our MitM preimage attack framework covers a wide range of sponge-based hash functions, especially those with lower claimed security level for preimage compared to their output size. Those hash functions have been very widely standardized (e.g., `Ascon-Hash`, `PHOTON`, etc.), but are rarely studied against preimage attacks. Even the recent MitM attack framework on sponge construction by Qin et al. (EUROCRYPT 2023) cannot attack those hash functions. As the second contribution, our MitM collision attack framework shows a different tool for the collision cryptanalysis on sponge construction, while previous collision attacks on sponge construction are mainly based on differential attacks. Most of the results in this paper are the first third-party cryptanalysis results. If cryptanalysis previously existed, our new results significantly improve the previous results, such as improving the previous 2-round collision attack on `Ascon-Hash` to the current 4 rounds, improving the previous 3.5-round quantum preimage attack on SPHINCS$^+$-`Haraka` to our 4-round classical preimage attack, etc.

Keywords: Sponge · Hash Function · MitM · Collision · Preimage

1 Introduction

A cryptographic hash function H, that maps a message M of arbitrary length into a short fixed-length n-bit target T, should satisfy at least three basic security properties, *i.e.*, (2nd-) preimage resistance and collision resistance. Due to the breakthrough attacks by Wang et al. [58,57] on `MD5` and `SHA-1`, the U.S. National Institute of Standards and Technology (NIST) started new standardization of hash functions in October 2008, *i.e.*, the `SHA-3` competition. After intense competition, the `Keccak` sponge function family [9] designed by Bertoni *et al.* won the competition in October 2012 and was subsequently standardized by the NIST as Secure Hash Algorithm-3 [47] (`SHA-3`) in August 2015. Instead of using the classical Merkle-Damgård construction [45,19], `Keccak` adopts a new construction called *sponge construction*. Due to the high efficiency and security, the sponge construction or its variants become widely used to build hash functions and other primitives, such as `Ascon-Hash` [27], `Xoodyak` [17], `PHOTON` [36], `SPONGENT` [11], SPHINCS+-`Haraka` [7], `ACE`-$\mathcal{H}$-256 [1], etc. Notably, `Ascon` family has been selected as the winner of the NIST Lightweight Cryptography (LWC), and `PHOTON` and `SPONGENT` are currently the ISO standard lightweight hash. For most of the hash functions except `Keccak`, there is almost no cryptanalytic result on preimage attack. The reason may be that they (except `Keccak`) all have a lower claimed security level for preimage compared to their output size. Traditionally, for a hash function like `SHA-2/3` with a n-bit digest, the security claim against preimage attacks is 2^n. However, for many new hash functions, the claimed security for preimage is lower than 2^n, e.g., `Ascon-Hash` and `ACE`-$\mathcal{H}$-256, the size of output is 256, but the security claim is just 2^{128} or 2^{192} by their designers. At CRYPTO 2022, Lefevre and Mennink [42] proved a tight preimage security for those sponge constructions, and increase security level of `Ascon-Hash` from the 2^{128} preimage security as claimed by designers, to 2^{192} preimage security.

The Meet-in-the-Middle (MitM) attack proposed by Diffie and Hellman in 1977 [22] is a time-memory trade-off cryptanalysis of symmetric-key primitives. During the last decades, it has been improved by more refined techniques and exploiting additional freedoms and structures, such as the internal state guessing [29], splice-and-cut [2], initial structure [52], bicliques [10,39], 3-subset MitM [13], (indirect-)partial matching [2,52], guess-and-determine [53,37], sieve-in-the-middle [16], match-box [32], dissection [24], MitM in differential view [40,30], and differential MitM [15], etc. Automating the MitM attacks with computer-aided tools may discover more advanced attack configurations, which was first tried in [14,21] at CRYPTO 2011 and 2016 for `AES` and `AES`-like ciphers. At IWSEC 2018, Sasaki [50] introduced the 3-subset MitM attacks on `GIFT` block cipher with Mixed Integer Linear Programming (MILP). At EUROCRYPT 2021, Bao *et al.* [4] fully automated MitM preimage attacks with MILP on `AES`-like hashing, which is built from `AES`-like structures. Later on, this model was further developed into models of key-recovery and collision attacks by Dong *et al.* [28] and Bao *et al.* [5]. Schrottenloher and Stevens [54,55] simplified the language of the automatic model and applied it in both classical and quantum settings.

MitM attack on sponge-based hash functions. The MitM attack has been widely used to attack Merkle-Damgård hash functions [52,2,35], whose compression function is usually built from a block cipher and the PGV hashing modes [48].

However, it was rarely used to attack sponge-based hash functions. At CRYPTO 2022, Schrottenloher and Stevens [54] first built several MitM attacks on sponge-based hash functions, *i.e.*, SPHINCS$^+$-`Haraka` [7] and `Sparkle` [6]. At EUROCRYPT 2023, Qin et al. [49] introduced a generic framework of MitM preimage attacks on sponge-based hashing and first built the MitM attacks on 4-round `Keccak-512` [9], 3-/4-round `Ascon-XOF` [27], and 3-round `Xoodyak-XOF` [17].

Our Contribution.

Generic MitM Preimage Attack Framework for Sponge Constructions. The sponge construction [9] with b-bit internal state ($b = c + r$ with c-bit capacity and r-bit rate) includes two phases: the first phase is the absorbing phase, which XORs r-bit message block into the state and interleaves with an application of the permutation; the second phase is the squeezing phase, which returns r-bit state bits as output, and interleaves with application of the permutation, until n bits are returned as the digest T. For `SHA-3`, the designers choose $c/2 = n$ and provide an n-bit security for preimage resistance. At CRYPTO 2011, the developers of `PHOTON` introduced the generalized sponge [36], that squeezes r'-bit ($r' \geq r$) state at a time. At CRYPTO 2022, Lefevre and Mennink [42] formally proved that the preimage security for generalized sponge is $q \approx \min\{\max\{2^{n-r'}, 2^{c/2}\}, 2^n\}$. Therefore, the bit security of preimage may not be equal to the size of digest n. For example, the length of the digest of SPHINCS$^+$-`Haraka` [7] ($b = 512, n = c = r = r' = 256$) is n=256 bits, however, the security claim against preimage attack is only 2^{128}. In this case, we prove that Qin et al.'s MitM framework [49] can not achieve better preimage attack with time complexity lower than the proved bound q. In fact, there exist many such hash functions (such as `Ascon-Hash` [27], `PHOTON` [36], `SPONGENT` [11], etc.) that previous MitM attacks can not work for preimage attacks. In this paper, we propose a new and general MitM preimage attack framework, that is applicable to any (generalized) sponge construction. Moreover, we invent more MILP modellings by exploiting the new features of the framework and dedicated hash functions, like new matching strategies for `Ascon-Hash`, `Xoodyak-Hash`. When applying to `ACE`-$\mathcal{H}$-256 and `SPONGENT`, the sieve-in-the-middle technique [16] is for the first time applied in the MitM automatic model, which is never applied in previous automatic tools [4,28].

We build the first preimage attacks on round-reduced `Ascon-Hash` (winner of NIST LWC), `PHOTON` (ISO standard), `PHOTON-Beetle-Hash` (finalist of NIST LWC), `ACE`-$\mathcal{H}$-256 (2nd round candidate of NIST LWC), `SPONGENT` (ISO standard), etc. Most attacks are the first cryptanalytic results since the primitives were designed, such as `PHOTON` and `SPONGENT`, which were designed in 2011. For SPHINCS$^+$-`Haraka`, our preimage attack in the classical setting (4-round attack) covers even more rounds than the previous quantum preimage attack (3.5-round attack) [54]. Please find a summary in Table 1.

MitM Collision Attack Frameworks for Sponge Constructions. For sponge constructions, the collision attacks are usually built with differential at-

Table 1: A Summary of the Attacks. Q: quantum attack. †: The attack is invalid, since its complexity is higher than the birthday attack.

Target	Attacks	Methods	Rounds	Time	Memory	Claim	Generic	Ref.
Ascon-Hash	Preimage	MitM	3/12	$2^{162.80}$	2^{160}	2^{128}	2^{192}	Sect. 6.1
		MitM	4/12	$2^{184.85}$	2^{178}			Sect. 6.1
		MitM	5/12	$2^{191.31}$	2^{190}			Sect. 6.1
	Collision	Diff.	2/12	2^{125}	-	2^{128}	2^{128}	[61]
		Diff.	2/12	2^{103}	-			[33]
		MitM	3/12	$2^{116.74}$	2^{116}			Sect. 6.2
		MitM	4/12	$2^{124.85}$	2^{124}			Sect. 6.2
SPHINCS$^+$-Haraka	Preimage	MitM	3.5/5	$2^{64.6}$ Q	-	-	$2^{85.3}$ Q	[54]
		MitM	4/5	2^{98}	2^{96}	2^{128}	2^{128}	Sect. 5
PHOTON-80/20/16	Preimage	MitM	4.5/12	2^{60}	2^{24}	2^{64}	2^{64}	Sect. 9.1
PHOTON-128/16/16	Preimage	MitM	4.5/12	2^{104}	2^{56}	2^{112}	2^{112}	Sect. 9.1
PHOTON-160/36/36	Preimage	MitM	4.5/12	2^{116}	2^{36}	2^{124}	2^{124}	Sect. 9.1
PHOTON-224/32/32	Preimage	MitM	4.5/12	2^{184}	2^{72}	2^{192}	2^{192}	Sect. 9.1
PHOTON-256/32/32	Preimage	MitM	4.5/12	2^{208}	2^{112}	2^{224}	2^{224}	Sect. 9.1
PHOTON-Beetle-Hash	Preimage	MitM	3.5/12	2^{112}	2^{65}	2^{128}	2^{128}	Sect. E.2
ACE-$\mathcal{H}$-256	Preimage	MitM	9/16	2^{160}	2^{128}	2^{192}	2^{192}	Sect. 8
SPONGENT-88	Collision	Diff.	6/45†	$2^{55.2}$	-	2^{40}	2^{40}	[11]
	Preimage	MitM	6/45	$2^{74.63}$	2^{73}	2^{80}	2^{80}	Sect. 7
	Preimage	MitM	7/45	$2^{78.59}$	2^{67}	2^{80}	2^{80}	Sect. F.1
Subterranean 2.0	Preimage	MitM	Full	2^{160}	2^{100}	2^{112}	2^{224}	Sect. 10
Xoodyak-XOF	Preimage	Neural	1/12	-	-	2^{128}	2^{128}	[44]
		MitM	3/12	$2^{125.06}$	2^{97}			[49]
		MitM	3/12	$2^{121.77}$	2^{118}			Sect. 11
Xoodyak-Hash	Collision	MitM	3/12	$2^{125.23}$	2^{124}	2^{128}	2^{128}	Sect. 11

tacks, e.g., the collision attacks on `Keccak` [25,26,34,60] and `Ascon` [33]. As indicated by [38,43], by finding partial target preimages with the MitM approach, one can build collision attacks on Merkle-Damgård hash functions. In Dinur's collision attack on `Keccak` [23], the partial target preimages were found by solving multivariate equations instead of MitM approach. To the best of our knowledge, the MitM approach has not yet been applied to the collision attack on sponge constructions. We build the first MitM collision attack framework for sponge constructions and immediately improve the collision attack on `Ascon-Hash` from the best previous 2 rounds [33] to the current 4 rounds, and also leads to the first collision attack on 3-round `Xoodyak-Hash` (finalist of NIST LWC). A summary is given in Table 1. We give a partial experiment to verify our collision attacks on `Ascon-Hash` in https://github.com/boxindev/MitM_attack_sponge.

2 Preliminaries

2.1 The Sponge Construction

The sponge construction [9] works on a b-bit internal state, which is divided into two parts: the r-bit outer part (r is called the rate), and c-bit inner part (c is called the capacity). One first initializes the b-bit state with the given value (all zero for `Keccak`). Then, pad and divide the given message into several r-bit blocks. In the absorbing phase, each r-bit message block is XOR-ed into the state and an inner permutation f built by iterating a round function is applied. In the squeezing phase, it produces the n-bit digest T. To find a collision requires at least $Time \approx \min\{2^{c/2}, 2^{n/2}\}$, and to find the preimage or second preimage requires at least $Time \approx \min\{2^{c/2}, 2^{n}\}$. Example hash function is SHA-3 hash function. A relaxation of sponge introduced in the design of `PHOTON` [36] is to squeeze a larger rate $r' \geq r$, which is called generalized sponge as shown in Figure 1. At CRYPTO 2022, Lefevre and Mennink [42] proved a tight preimage security for generalized sponge up to $Time \approx \min\{\max\{2^{n-r',2^{c/2}}\}, 2^n\}$. Therefore, the NIST Lightweight Cryptography standard `Ascon-Hash` ($b = 320, r = r' = 64, c = 256, n = 256$) does not generically achieve 2^{128} preimage security as claimed, but even 2^{192} preimage security.

2.2 The Meet-in-the-Middle Attack on Sponge Constructions

Meet-in-the-middle is a general attack paradigm against cryptographic primitives where internal states are computed along two independent chunks that are then matched to produce a complete path solution. For Merkle-Damgård hash functions or block ciphers, the two independent chunks are usually the forward computation path and the backward computation path, i.e., the popular splice-and-cut [2] or three-subset MitM techniques [13]. To illustrate how the MitM attack works, we detail the 7-round attack on AES-hashing of Sasaki [51] in Supplementary Material A as an example.

At EUROCRYPT 2023, Qin et al. [49] introduced a new MitM framework for sponge constructions as in Figure 2, where two independent forward chunks are

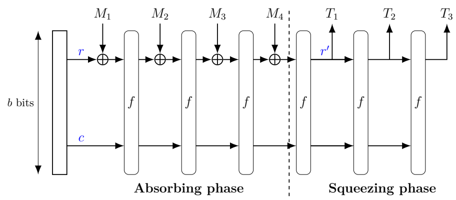


Fig. 1: The sponge construction

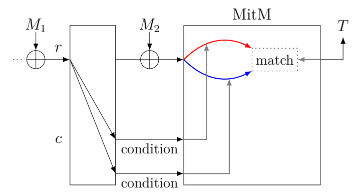


Fig. 2: Qin et al.'s MitM attack [49]

applied (without backward chunk). Starting from the r-bit outer part determined by M_2, the two neutral sets (red ■ and blue ■ neutral bits, denoted as $\mathcal{R}$ and $\mathcal{B}$, respectively) compute independently forward to the m-bit matching point, which is an m-bit deterministic relation on the two neutral sets by partially solving the inverse of the permutation from the n-bit target. Assuming the deterministic relation is $g_{\mathcal{B}} = g_{\mathcal{R}}$, where $g_{\mathcal{B}}$ is determined by the ■ and ■ bits of the starting state, and $g_{\mathcal{R}}$ is determined by ■ and ■ bits. $g_{\mathcal{B}} = g_{\mathcal{R}}$ is applied to filter the states.

When propagating the two neutral sets forward, Qin et al. found certain bit conditions can be applied to reduce the propagation of the ■ and ■ neutral bits, and therefore, improved the MitM path. Those conditions are determined by M_1. After finding one proper M_1 that satisfies all bit conditions, the attacker can perform the following MitM attack. Supposing the red and blue neutral sets of the outer part are of $2^{d_{\mathcal{R}}}$ and $2^{d_{\mathcal{B}}}$ values, ($d_{\mathcal{R}}$ and $d_{\mathcal{B}}$ are also the degrees of freedom for the two sets of neutral bits) an MitM episode is performed as follows:

1. For each of $2^{d_{\mathcal{R}}}$ values, compute forward to the matching point.
2. For each of $2^{d_{\mathcal{B}}}$ values, compute forward to the matching point.
3. Given the n-bit target, compute backward to derive an m-bit matching point.
4. Filter states.

The complexity of one MitM episode is $2^{\max(d_{\mathcal{R}},d_{\mathcal{B}})} + 2^{d_{\mathcal{R}}+d_{\mathcal{B}}-m}$, which checks a subspace of M_2 of size $2^{d_{\mathcal{R}}+d_{\mathcal{B}}}$. In order to find an n-bit target preimage, the episode should be repeated $2^{n-(d_{\mathcal{R}}+d_{\mathcal{B}})}$ times. Supposing $\mathcal{C}$ is the time complexity to find M_1, i.e., assigning proper bit conditions, then the overall time complexity to find the n-bit target preimage is

$$\mathcal{C} + 2^{n-(d_{\mathcal{R}}+d_{\mathcal{B}})} \cdot (2^{\max(d_{\mathcal{R}},d_{\mathcal{B}})} + 2^{d_{\mathcal{R}}+d_{\mathcal{B}}-m}) \approx \mathcal{C} + 2^{n-\min(d_{\mathcal{R}},d_{\mathcal{B}},m)}. \tag{1}$$

Qin et al. introduced an MILP model to find good MitM path. They introduced 5 colors to encode each bit, and each bit is represented by three binary variables $(\omega_0, \omega_1, \omega_2)$:

- `Gray` ■: $(1, 1, 1)$, global constant bits,
- `Red` ■: $(0, 1, 1)$, determined by ■ bits and ■ bits of starting state,
- `Blue` ■: $(1, 1, 0)$, determined by ■ bits and ■ bits of starting state,

– `Green` ■: $(0,1,0)$, determined by ■ bits, ■ bits and ■ bits, but the expression does not contain the product of ■ and ■ bits,
– `White` □: $(0,0,0)$, depend on the product of ■ and ■ bits, i.e., unknown bit.

We adopt most of the techniques from Qin et al.'s MILP model, but make some changes for dedicated hash functions in order to find suitable MitM paths for our new attack frameworks.

2.3 The Collision Attack based on MitM Approach

At FSE 2012, Li, Isobe, and Shibutani [43] converted the MitM-based partial target preimage attacks into pseudo collision attacks. Suppose the algorithm $\mathcal{A}$ can produce the t-bit partial target preimage. The collision finding approach works as follows:

1. Given the hash function H that produces n-bit digest, randomly fix the t-bit partial target as constant. Call $\mathcal{A}$ to produce $2^{(n-t)/2}$ different (M,T) with the same fixed t-bit partial target.
2. From the $2^{(n-t)/2}$ (M,T), find a collision on the remaining $(n-t)$ bits of the full target.

The MitM collision attacks have been applied to analyse Merkle–Damgård hash functions, such as `SHA-2` [43], `Whirlpool` [28], `AES-MMO` [5]. However, this paper is the first time to apply it to sponge constructions.

3 Generic MitM Preimage Attack Framework on Sponge

In many sponge constructions, the security claim does not match the length of digest. For example, the length of the digest of SPHINCS$^+$-`Haraka` [7] (one of 4 selected algorithms in NIST PQC process) is n=256 bits, however, the security claim against preimage attack is only 2^{128}. Qin et al.'s MitM framework [49] can not achieve preimage attack with complexity better than 2^{128}. The reason is shown in Equation (1). The complexity of Qin et al.'s MitM attack is at least $2^{n-\min(d_{\mathcal{R}},d_{\mathcal{B}},m)}$. In fact, in Qin et al.'s MitM attack, at least one MitM episode should be performed, whose complexity is $2^{\max(d_{\mathcal{R}},d_{\mathcal{B}})} + 2^{d_{\mathcal{R}}+d_{\mathcal{B}}-m} = 2^{\max(d_{\mathcal{R}},d_{\mathcal{B}},d_{\mathcal{R}}+d_{\mathcal{B}}-m)}$. Therefore, considering both $2^{n-\min(d_{\mathcal{R}},d_{\mathcal{B}},m)}$ and $2^{\max(d_{\mathcal{R}},d_{\mathcal{B}},d_{\mathcal{R}}+d_{\mathcal{B}}-m)}$, the optimal complexity is achieved when $d_{\mathcal{R}} = d_{\mathcal{B}} = m = n/2$, which is at least $2^{n/2}$. Therefore, for preimage attack with $n=256$-bit target, Qin et al.'s MitM attack framework can only achieve 2^{128} complexity at best.

We propose a generic MitM preimage attack framework for all hash functions based on sponge constructions as shown in Figure 3. Suppose sponge hashing is with parameters: n-bit target size, b-bit state size, r-bit absorbing rate, c-bit capacity, r'-bit squeezing rate. Suppose the target is $T = T_1\|T_2\|...$, where $|T_i| = r'$ and $|T| = n$. Figure 3 only includes the first two blocks of target for briefness, i.e., $T_1\|T_2$. The new framework consists of two phases:

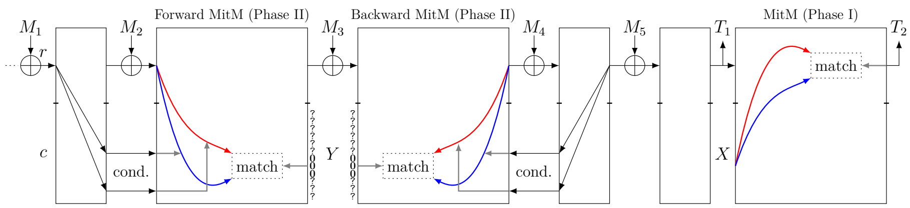


Fig. 3: General MitM Preimage Attack Framework

- **Phase I:** Given target blocks $T = T_1\|T_2\|T_3\|...$, we first find a capacity state X, such that $squeeze(T_1\|X) = T_2\|T_3\|....$ This is the so-called constrained-input constrained-output (CICO) problem [9], and $2^{n-r'}$ time is needed to find X by brute-force search.
  Here, we apply Qin et al.'s MitM framework [49] and place a MitM attack in the permutation to squeeze T_2. According to Eq. (1), the time to find X is
  $$2^{|T|-|T_1|-\min(d^I_{\mathcal{R}},d^I_{\mathcal{B}},m^I)} = 2^{n-r'-\min(d^I_{\mathcal{R}},d^I_{\mathcal{B}},m^I)}, \tag{2}$$
  A valid MitM attack in this phase can find X with time less than $2^{n-r'}$.
- **Phase II:** After we find X in Phase I, we compute $S' = f^{-1}(T_1\|X)$. As shown in Figure 3, we try to find a collision on the c-bit capacity, i.e., $Trunc(f(f(M_1 \oplus S) \oplus M_2))_c = Trunc(f^{-1}(f^{-1}(M_5 \oplus S') \oplus M_4))_c = Y$, which is called the inner collision. The Floyd's cycle finding algorithm [31] can trivially find it with time complexity of $2^{c/2}$ with negligible memory. Here, we introduce another attack that may find the inner collision with time below $2^{c/2}$.
  We fix t-bit of the capacity state Y as constants (e.g., zeros), then perform two MitM attacks (denoted as forward MitM and backward MitM, also following Qin et al.'s framework) to find $M_1\|M_2$ and $M_4\|M_5$, respectively, to meet those t-bit zeros. Suppose the MitM attack finds $2^{\frac{c-t}{2}}$ $M_1\|M_2$ where the corresponding t bits are zero, and store those $M_1\|M_2$ in L_1 indexed by the unfixed $c-t$ capacity bits, denoted as Y_1. Build similar table L_2 storing $M_4\|M_5$ indexed by Y_2 (unfixed $c-t$ capacity bits). Match the two tables L_1 and L_2 to produce a $M_1\|M_2\|M_4\|M_5$, and then compute the last block M_3. The time complexity to build L_1 is
  $$\mathcal{C}_1 + 2^{t-\min(d^{L1}_{\mathcal{R}},d^{L1}_{\mathcal{B}},m^{L1})} \cdot 2^{\frac{c-t}{2}} = \mathcal{C}_1 + 2^{\frac{c}{2}+\frac{t}{2}-\min(d^{L1}_{\mathcal{R}},d^{L1}_{\mathcal{B}},m^{L1})}, \tag{3}$$
  and similarly, the time complexity to build L_2 is
  $$\mathcal{C}_2 + 2^{t-\min(d^{L2}_{\mathcal{R}},d^{L2}_{\mathcal{B}},m^{L2})} \cdot 2^{\frac{c-t}{2}} = \mathcal{C}_2 + 2^{\frac{c}{2}+\frac{t}{2}-\min(d^{L2}_{\mathcal{R}},d^{L2}_{\mathcal{B}},m^{L2})}, \tag{4}$$
  where $\mathcal{C}_1$ and $\mathcal{C}_2$ are the time complexities of assigning conditions for these two MitM attacks. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are smaller enough than $2^{c/2}$, we only consider the second parts of Equation (3) and (4). When $\min(d^{L1}_{\mathcal{R}}, d^{L1}_{\mathcal{B}}, m^{L1}) > \frac{t}{2}$ and $\min(d^{L2}_{\mathcal{R}}, d^{L2}_{\mathcal{B}}, m^{L2}) > \frac{t}{2}$, the overall time complexity will be less than $2^{c/2}$.

Remark. In **Phase II**, there are two independent forward and backward MitM attacks, and share a common partial fixed value (e.g., t-bit zeros) in the inner part, which should be chosen carefully so that both the two MitM attacks work efficiently.

Generally, for the generalized sponge with parameters (n, b, c, r, r') described in Section 2.1, the tight preimage security bound was proved to be $Time \approx \min\{\max\{2^{n-r'}, 2^{c/2}\}, 2^n\}$ [42]. To beat the general bound, we need to consider the following popular cases:

- Case I: If $n - r' < c/2 < n$, the general bound is $2^{c/2}$-bit. Therefore, we only need to perform MitM attack in **Phase II** to derive attack better than general bound. Typical example is SPHINCS$^+$-`Haraka` [7] with $n = 256, r = 256, c = 256, r' = 256$.
- Case II: If $n - r' = c/2$, the general bound is both reached as $2^{n-r'}$ (CICO problem in squeezing phase) and $2^{c/2}$-bit (birthday problem in absorbing phase). Therefore, we need to perform both MitM attacks in **Phase I** and **Phase II** to derive attack better than general bound. Typical examples are `Xoodyak-Hash` [17] and `Gimli-Hash` [8] with $n = 256, c = 256, r = r' = 128$.
- Case III: If $n - r' > c/2$, the general bound is $2^{n-r'}$. Therefore, we only need to perform MitM attack in **Phase I**, and perform Floyd's cycle finding algorithm in **Phase II** to get preimage attack with time lower than $2^{n-r'}$. This case is very popular in hash designs. Examples include `Ascon-Hash` [27], `PHOTON` [36], `SPONGENT` [11], `ACE`-$\mathcal{H}$-256 [1], etc.

Following the new MitM framework, we give the new preimage attacks on round-reduced SPHINCS$^+$-`Haraka`, `Ascon-Hash`, `PHOTON`, `PHOTON-Beetle-Hash`, `ACE`-$\mathcal{H}$-256, `Subterranean 2.0`, `SPONGENT`, etc. Most of the attacks are the first known cryptanalysis results on preimage attacks on the corresponding hash functions.

4 MitM Collision Attack on Sponge Constructions

For collision attack on sponge constructions, the popular method is the differential attack, e.g., collision attacks on `Keccak` [25,26,34,60] and `Ascon` [33]. In this section, we introduce a new collision attack framework for sponge constructions based on MitM approach. The new framework immediately improves the collision attack on `Ascon-Hash` from the best previous 2 rounds [33] to the current 4 rounds, and also leads to the first collision attack on 3-round `Xoodyak-Hash`. In this section, we give two MitM collision frameworks for sponge construction as shown in Figure 4 and 5.

- For **Collision Framework I** in Figure 4, the attacker tries to find two messages that output the same n-bit target T directly. Generally, the birthday attack can find the collision with complexity of $2^{n/2}$. When applying the MitM approach as shown in Section 2.3, the attacker first fixes t-bit target

and find $2^{(n-t)/2}$ t-bit partial target preimages with a MitM approach, then a collision exists among those messages. If the MitM finds one parital target preimage with time complexity $\mathcal{C}_I$ smaller than $2^{t/2}$, then the overall time complexity is $\mathcal{C}_I \cdot 2^{(n-t)/2} < 2^{n/2}$.
- For **Collision Framework II** in Figure 5, the attacker tries to find two messages that lead to a full state collision. Since the r-bit outer part can be modified freely by M_3, the full state collision can be achieved when the c-bit inner part collides, which is called inner collision. Trivially, a birthday attack can generate the inner collision in $2^{c/2}$. When applying the MitM approach, the attacker first fixes t-bit target and find $2^{(c-t)/2}$ t-bit partial target preimages with a MitM approach, then an inner collision exists among those messages. If the MitM finds one parital target preimage with time complexity $\mathcal{C}_{II}$ smaller than $2^{t/2}$, then the overall time complexity is $\mathcal{C}_{II} \cdot 2^{(c-t)/2} < 2^{c/2}$.

In the generalized sponge constructions, the collision security is $\min\{2^{c/2}, 2^{n/2}\}$.

- If $n > c$, the **Collision Framework II** is applied to find a collision with time less than the generic bound $2^{c/2}$. Example case is the eXtendable Output Function (XOF) when the output size is larger than c, like `Xoodyak-XOF`, `Ascon-XOF`, etc.
- If $n = c$, the attacker can choose **Collision Framework I** or **II** to find a better attack. Example cases are the most popular hash functions, like `Ascon-Hash`, `Xoodyak-Hash`, etc.
- If $n < c$, **Collision Framework I** should be applied. Example case is the extendable output functions (XOFs) when the output size is smaller than c.

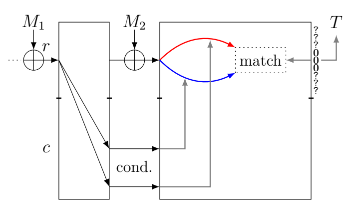


Fig. 4: Collision Framework I

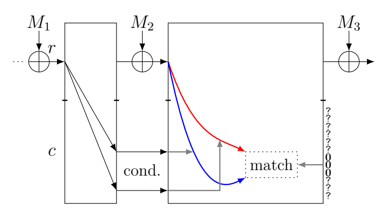


Fig. 5: Collision Framework II

Comparison between MitM preiamge and collision attack. In Qin et al.'s MitM preimage attack [49], given the n-bit target, the attacker inverts it to build the m-bit matching equation. Then, the complexity of preimage attack is about 2^{n-m} by assuming enough degrees of freedom of the two sets of neutral bits.

For our MitM collision attack, we have to choose t-bit partial target (or inner part) to be fixed and invert it to get m-bit matching equation ($m \le t < n$).

Assuming enough degrees of freedom of the neutral bits again, the attacker finds one t-bit target preimage with 2^{t-m} complexity on average. Then, to perform the birthday attack on the remaining $n-t$ bits, one needs $2^{(n-t)/2}$ t-bit target preimages, i.e., the time is at least $2^{t-m} \cdot 2^{(n-t)/2} = 2^{n/2+t/2-m}$. Therefore, $t/2$ should be smaller than m to beat the birthday bound $2^{n/2}$, i.e., $t/2 < m \leq t$, and $m = t$ is the optimal case. However, it is not trivial to find m-bit matching with $t/2 < m \leq t$ by fixing t-bit target.

For example, in Qin et al.'s attack on `Keccak` [49], their 1-bit matching equation (see equation (14) of [49]) is nonlinearly related to 4 target bits, which is of the form like $f_1(\mathcal{B}) + g_1(\mathcal{R}) + x_1 \cdot (f_2(\mathcal{B}) + g_2(\mathcal{R})) = x_2 + x_3 + x_1 \cdot x_4$, where functions $f_1(\mathcal{B})$ and $f_2(\mathcal{B})$ depend on blue neutral bits, $g_1(\mathcal{R})$ and $g_2(\mathcal{R})$ depend on red, and x_1, x_2, x_3, x_4 are four target bits. In order to derive the matching equation (a deterministic relation between red and blue bits), one can fix $x_1 = 0$ and $x_2 + x_3 = \mathit{constant}$, or $x_1 = 1$ and $x_2 + x_3 + x_4 = \mathit{constant}$, i.e., at least $t = 2$ target bits have to be fixed to derive the $m = 1$ bit matching. This is fine for the preimage attack, such as Qin et al.'s successful preimage attack on 4-round `Keccak` [49]. But it is infeasible for the collision attack, since $t/2 = 1 = m$. This is the difference between MitM preimage and collision attack, i.e., one good MitM preimage configuration may not trivially be converted into MitM collision attack.

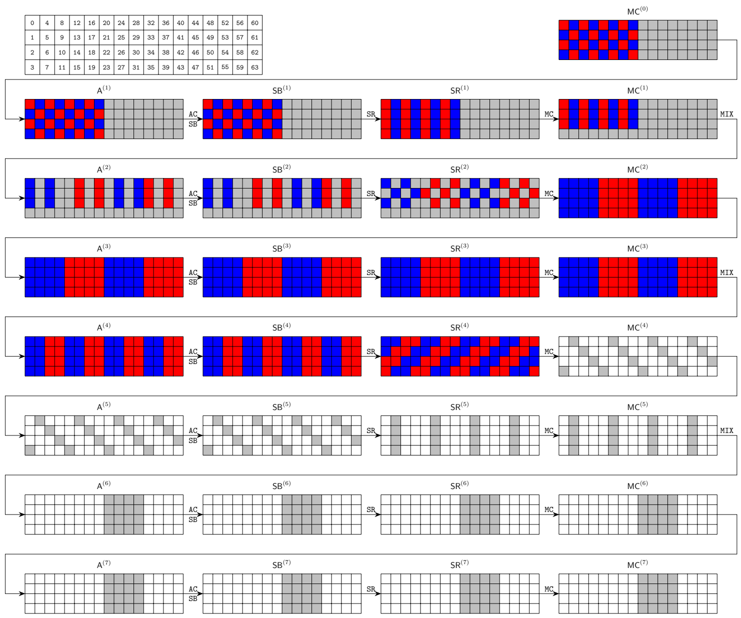
5 Preimage Attack on 4-round SPHINCS$^+$-`Haraka`

`Haraka v2` [41] is a short-input AES-like hash function, which is built by employing a 256 or 512-bit permutation and the DM construction. However, in practical application of SPHINCS$^+$ [7], one of the 4 selected algorithms of the NIST post-quantum standardization process, `Haraka v2` is integrated as SPHINCS$^+$-`Haraka`, which is a sponge-based hashing based on the 512-bit permutation of `Haraka v2`, but it was removed from the latest standardization document. The internal state of the 512-bit permutation is the concatenation of 4 AES states, and each round (total 5) applies two AES rounds individually on the states, followed by a MIX operation:

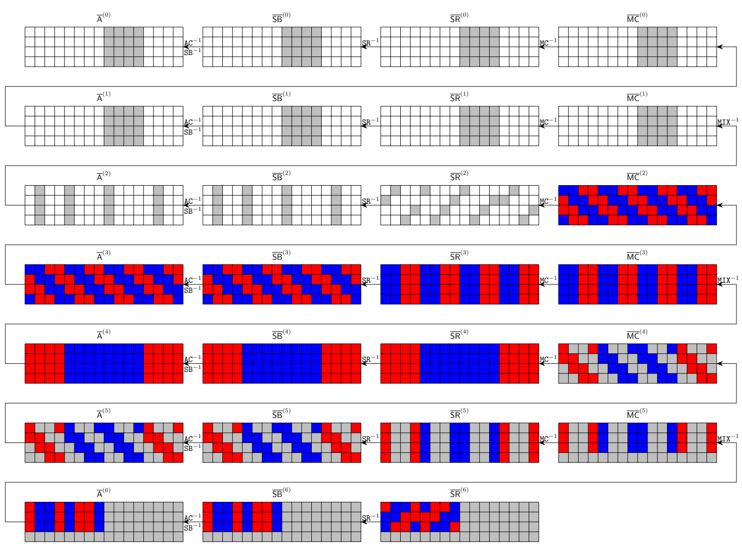
$$0, \cdots, 15 \mapsto (3, 11, 7, 15), (8, 0, 12, 4), (9, 1, 13, 5), (2, 10, 6, 14).$$

SPHINCS$^+$-`Haraka` has the configuration $(b = 512, n = c = r = r' = 256)$. The claimed security level is 128 bits preimage attack. At CRYPTO 2022, Schrottenloher and Stevens [54] proposed a 3.5-round (out of the full 5 rounds) quantum preimage attack. However, no classical attack exists for reduced SPHINCS$^+$-`Haraka` till now. In this section, we mount the first classical preimage attack on 4-round SPHINCS$^+$-`Haraka` without the last MIX operation based on our new MitM preimage framework in Section 3. We recall the new MitM framework in Figure 3. The attack procedures of two phases are given as follows.

- **Phase I:** Since the 256-bit digest is produced at once, i.e. T_1, we only need randomly set the value of 256-bit inner part X, and inversely compute $f^{-1}(T_1 \| X)$.



(a) Forward MitM



(b) Backward MitM

Fig. 6: The 4-round MitM attack on SPHINCS^{+}-`Haraka`

– **Phase II:** In the forward MitM of Phase II (Figure 6(a)), the starting state is $\mathsf{A}^{(2)}$=$\mathsf{MIX}(\mathsf{R}_{\texttt{AES}}(\mathsf{R}_{\texttt{AES}}(\mathsf{A}^{(0)})))$, where the message block (M_2 in Figure 3) is absorbed in $\mathsf{A}^{(0)}[0-31]$. In the backward MitM of Phase II (Figure 6(b)), the starting state is $\overline{\mathsf{MC}}^{(5)}$=$\mathsf{MIX}^{-1}(\mathsf{R}^{-1}_{\texttt{AES}}(\mathsf{R}^{-1}_{\texttt{AES}}(\overline{\mathsf{MC}}^{(7)})))$, where another message block (M_4 in Figure 3) is absorbed in $\overline{\mathsf{MC}}^{(7)}[0-31]$. After the MitM attacks find proper states for the starting states $\mathsf{A}^{(2)}$ (forward) and $\overline{\mathsf{MC}}^{(5)}$ (backward) , we can compute the corresponding message blocks M_2 and M_4. Note that the ■ bytes of the $\mathsf{A}^{(2)}$ and $\overline{\mathsf{MC}}^{(5)}$ can be deduced from the inner part of $\mathsf{A}^{(0)}$ and $\overline{\mathsf{MC}}^{(7)}$, respectively. Fix the 16-byte $\mathsf{MC}^{(7)}[32-47]$ and $\overline{\mathsf{A}}^{(0)}[32-47]$ as zeros. Then, perform the forward and backward MitM attacks to find $M_1||M_2$ stored in L_1 and $M_4||M_5$ stored in L_2 to satisfy those 16 zero bytes. We detail the forward MitM in Algorithm 1 with a time complexity of about 2^{97} and a memory complexity of 2^{96}. The time and memory is the same for the backward MitM. Then find a match between L_1 and L_2 to get a right $M_1\|M_2\|M_4\|M_5$, and then compute M_3. The memory of storing L_1 or L_2 is 2^{64}. The time complexity of Algorithm 1 is analyzed as follows:
  - In Line 5, the time complexity to build U is $2^{12\times 8=96}$.
  - In Line 9, the time complexity to traverse 12 ■ bytes, derive M_2 and build L_1 is 2^{96}. The memory to store L_1 is 2^{64}.

The overall time complexity to perform the preimage attack on 4-round SPHINCS$^+$-`Haraka` is 2^{98} and the memory is 2^{96}.

Algorithm 1: Forward MitM in the Attack on SPHINCS$^+$-`Haraka`

---

1 Set $\mathsf{MC}^{(7)}[32-47]=0$.
2 Given the value of the 8 columns ■ bytes of $\mathsf{A}^{(2)}$ determined by M_1 and fixing the other 8 ■ bytes as 0
3 Compute the 16 ■ bytes of $\mathsf{MC}^{(4)}$ from $\mathsf{MC}^{(7)}[32-47]$, denoted by $\mathsf{MC}^{(4)}_m$.
4 $U \leftarrow [\ ]$
5 **for** *each value of 12* ■ *bytes in* $\mathsf{A}^{(2)}[20-22, 28-30, 48-50, 56-58]$ **do**
6   Compute forward to $\mathsf{SR}^{(4)}$, and derive the 16 matching bytes, i.e., $\mathsf{MC}^{(4)}[3, 4, 9, 14, 19, 20, 25, 30, 35, 36, 41, 46, 51, 52, 57, 62]$, denoted by $\mathsf{MC}^{(4)}_{mr}$.
7     $U[\mathsf{MC}^{(4)}_{mr}] \leftarrow \mathsf{A}^{(2)}[20-22, 28-30, 48-50, 56-58]$.
8 **end**
9 **for** *each value of 12* ■ *bytes in* $\mathsf{A}^{(2)}[0-2, 8-10, 36-38, 44-46]$ **do**
10   Compute forward to $\mathsf{SR}^{(4)}$, and derive the 16 matching bytes in $\mathsf{MC}^{(4)}$, denoted by $\mathsf{MC}^{(4)}_{mb}$
11   `/* The matching equation is` $\mathsf{MC}^{(4)}_m = \mathsf{MC}^{(4)}_{mb} \oplus \mathsf{MC}^{(4)}_{mr}$ `*/`
12   Look up $U[\mathsf{MC}^{(4)}_{mb} \oplus \mathsf{MC}^{(4)}_m]$, construct M_2 and store it in L_1 indexed by unfixed 16 bytes in $\mathsf{MC}^{(7)}[48-63]$.
13 **end**

---

6 Preimage and Collision Attacks on Reduced `Ascon-Hash`

The 320-bit state A of `Ascon` is split into five 64-bit words, and denote $A^{(r)}_{\{x,y\}}$ to be the x-th (column) bit of the y-th (row) 64-bit word, where $0 \leq y \leq 4$, $0 \leq x \leq 63$. The round function consists of three operations p_C (add constants), p_S (Sbox layer), and p_L (linear layer). Denote the internal states of round r as $A^{(r)} \xrightarrow{p_S \circ p_C} S^{(r)} \xrightarrow{p_L} A^{(r+1)}$. The full description and symbols of `Ascon-Hash` are given in Supplementary Material B.

6.1 Preimage attack on Reduced `Ascon-Hash`

At CRYPTO 2022, Lefevre and Mennink [42] proved that the NIST Lightweight Cryptography standard `Ascon-Hash` ($b = 320, r = r' = 64, c = 256, n = 256$) does not generically achieve 2^{128} preimage security as claimed, but even 2^{192} preimage security. In this section, we apply the new MitM preimage attack framework introduced in Section 3 to explore round-reduced preimage attacks with complexity lower than 2^{192}. We first give an advanced matching strategy and introduce 3-/4-/5-round preimage attacks on `Ascon-Hash`.

New matching strategy for preimage. In Qin et al.'s model [49], the deterministic relations between the two neutral sets, which act as matching, are derived by the last Sbox layer. Suppose $(a_0, a_1, a_2, a_3, a_4)$ and $(b_0, b_1, b_2, b_3, b_4)$ are the input and output of Sbox. Let b_0 be the known target bit (because the first row of the state is output as target), we have $b_0 = a_4a_1 + a_3 + a_2a_1 + a_2 + a_1a_0 + a_1 + a_0$. Qin et al. gave the observation that there exists a 1-bit matching if there is no unknown $\square$ bit in $(a_0, a_1, a_2, a_3, a_4)$ and no product of $\color{blue}\blacksquare$ and $\color{red}\blacksquare$ bits, i.e., (a_1, a_4), (a_0, a_1), and (a_0, a_1) should not be pair of ($\color{blue}\blacksquare$,$\color{red}\blacksquare$), ($\color{red}\blacksquare$,$\color{green}\blacksquare$), or ($\color{green}\blacksquare$,$\color{green}\blacksquare$), etc. In this case, equation of the form $g_{\mathcal{B}} = g_{\mathcal{R}}$ can be derived by b_0, where $g_{\mathcal{B}}$ is determined by the $\color{blue}\blacksquare$ neutral set, and $g_{\mathcal{R}}$ is determined by $\color{red}\blacksquare$ neutral set. $g_{\mathcal{B}} = g_{\mathcal{R}}$ is applied as 1-bit matching.

Here we introduce a more efficient matching strategy. Let $b_0 = a_1(a_4 + a_2 + a_0) + a_3 + a_2 + a_1 + a_0$, then we give the following observation:

Observation 1 *If $(a_0, a_1, a_2, a_3, a_4)$ satisfies the following conditions, there exists a 1-bit matching:*

1. *There is no $\square$ bit in $(a_0, a_1, a_2, a_3, a_4)$.*
2. *There is no product of $\color{blue}\blacksquare$ and $\color{red}\blacksquare$, i.e., $(a_1, a_4 + a_2 + a_0)$ should not be ($\color{blue}\blacksquare$,$\color{red}\blacksquare$), ($\color{red}\blacksquare$,$\color{green}\blacksquare$), ($\color{blue}\blacksquare$,$\color{green}\blacksquare$), or ($\color{green}\blacksquare$,$\color{green}\blacksquare$), or opposite order.*

3-round Preimage Attack on `Ascon-Hash`. Following the MitM framework given in Figure 3, we perform the 3-round preimage attack on `Ascon-Hash`.

In **Phase I**, the squeezing phase outputs four 64-bit blocks (the first row of the state) as 256-bit target, i.e., $T = T_1 \| T_2 \| T_3 \| T_4$. We place a 3-round MitM phase in the permutation to squeeze T_2 (similar to Figure 3). As shown in Figure

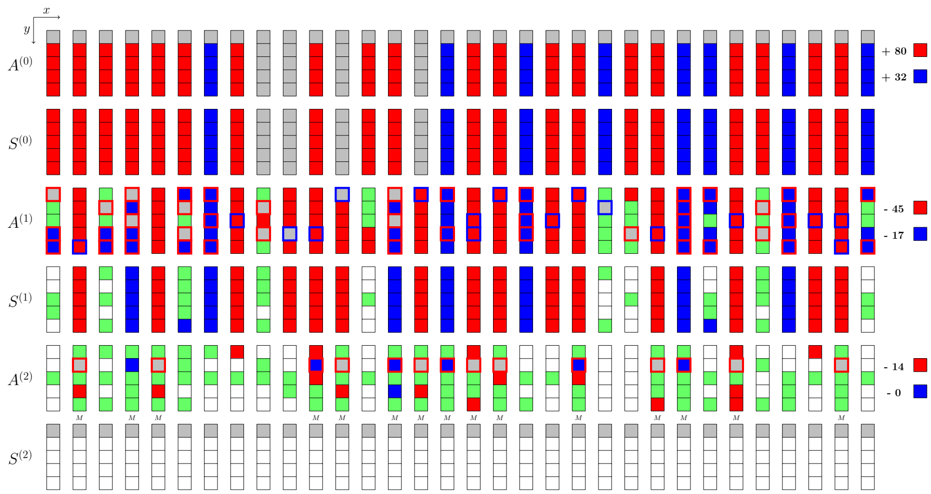


Fig. 7: The 3-round Preimage attack on `ASCON-Hash`

7, we explore the symmetry in x-axis to speedup the search by cutting the full 64-bit word into 32-bit word. Therefore, the linear operation works modular 32 instead of 64. For example, the linear operation in the second row changes from the original $A^{(r+1)}_{\{*,1\}} \leftarrow S^{(r)}_{\{*,1\}} \oplus (S^{(r)}_{\{*,1\}} \ggg 61) \oplus (S^{(r)}_{\{*,1\}} \ggg 39)$ into the current $A^{(r+1)}_{\{*,1\}} \leftarrow S^{(r)}_{\{*,1\}} \oplus (S^{(r)}_{\{*,1\}} \ggg 29) \oplus (S^{(r)}_{\{*,1\}} \ggg 7)$. It is very time consuming to solve a full MILP model. By exploring the symmetry, the model is reduced and easy to solve. But this strategy may miss paths and may be improved by different searching algorithm, e.g., [20].

Our automatic search finds a 3-round MitM characteristic shown in Figure 7. Due to the symmetry, in the full MitM path, the starting state $A^{(0)}$ contains 64 bits and 160 bits. The first row of $A^{(0)}$ is fixed as T_1 (marked as), and the remaining 32-bit can be freely chosen. In the computation from $A^{(0)}$ to $A^{(2)}$, the consumed degree of freedoms (DoFs) of and DoFs of are 118 and 34, respectively. Additional, there are 12 consumed DoFs of to make $a_0 + a_2 + a_4$ of $A^{(2)}$ become or for matching points. Therefore, $d_{\mathcal{B}} = d_{\mathcal{R}} = 30$, and there are 30 matching bits. The 3-round MitM attack is given in Algorithm 2.

Analysis of Algorithm 2. In Line 13 to 21, $2^{130+30+30+\zeta}$ states are tested against the 192-bit $T_2 \| T_3 \| T_4$, therefore, $\zeta = 2$ is enough to find a preimage. In Line 8, we choose fixed $c_{\mathcal{B}}$ to eliminate their influence on the computation of $c_{\mathcal{R}} \in \mathbb{F}_2^{130}$ and the 30-bit matching point (those values will determined by / as well as /) for 2^{160} red bits in $A^{(0)}$ in Line 10.

– The Line 4 to 6, the time complexity is $2^{2+64} \times \frac{16}{3\times 64} = 2^{66} \times 2^{-3.58} = 2^{62.42}$ 3-round Ascon. The fraction $\frac{16}{3\times 64}$ means that we only need to compute the 16 Sboxes related to bits in the first round, and 3-round Ascon has a total of 3×64 Sboxes.

Algorithm 2: Preimage Attack on 3-round `Ascon-Hash`: Phase I

Fix the first row of $A^{(0)}$ as T_1
for 2^{ζ} *values of the 32-bit free gray bits in* $A^{(0)}$ **do**
    `/* Precomputation */`
    for 2^{64} *values of the* ■ *bits* $v_{\mathcal{B}}$ *in* $A^{(0)}$ **do**
        Compute forward to determine the 34-bit ■/■ (denoted as $c_{\mathcal{B}} \in \mathbb{F}_2^{34}$) in $A^{(1)}$. E.g., in the ■ bit $A^{(1)}_{\{30,2\}} = S^{(0)}_{\{30,2\}} \oplus S^{(0)}_{\{29,2\}} \oplus S^{(0)}_{\{24,2\}}$, the $S^{(0)}_{\{24,2\}}$ should be gray to make the $A^{(1)}_{\{30,2\}}$ independent of blue bits, which consumes 1 DoF of ■. This is actually done by computing the 24-th Sbox, where there are 4 input ■ bits and output one ■ bit $S^{(0)}_{\{24,2\}}$ by consumming 1 DoF of ■. Then, $S^{(0)}_{\{24,2\}}$ is one bit of the 34-bit $c_{\mathcal{B}}$
        Store the 64-bit ■ values $v_{\mathcal{B}}$ of $A^{(0)}$ in $U[c_{\mathcal{B}}]$
    end
    Choose an index $c_{\mathcal{B}}$, e.g., $c_{\mathcal{B}} = 0$, there expected 2^{64-34} elements in $U[0]$
    `/* In the following, we always fix` $c_{\mathcal{B}}$ `as 0 */`
    for 2^{160} *values of the* ■ *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
        Compute forward to determine the 130-bit ■/■ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{130}$) and the 30-bit matching point. Build the table V and store the 160-bit ■ $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 30-bit matching point in $V[c_{\mathcal{R}}]$
    end
    for $c_{\mathcal{R}} \in \mathbb{F}_2^{130}$ **do**
        Retrieve the 2^{30} elements of $V[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 30-bit matching point
        for 2^{30} *values* $v_{\mathcal{B}}$ *in* $U[0]$ **do**
            Compute forward to the 30-bit matching point and store $v_{\mathcal{B}}$ in L_2 indexed by the 30-bit matching point.
        end
        for *values matched between* L_1 *and* L_2 **do**
            if T_2 *is satisfied* **then**
                Check if $T_3 \| T_4$ is satisfied
            end
        end
    end
end

- The Line 10 to 11, the time complexity is $2^{2+160} \times \frac{90}{3\times 64} = 2^{162} \times 2^{-1.09} = 2^{160.91}$ 3-round Ascon.
- The Line 14, the time complexity is $2^{2+130+30} \times \frac{1}{3\times 64} = 2^{162} \times 2^{-7.58} = 2^{154.42}$ 3-round Ascon. The fraction $\frac{1}{3\times 64}$ is because we assume that inserting an element into L of size 2^{30} is equivalent to one Sbox operation.
- The Line 16, the time complexity is $2^{2+130+30} \times \frac{50}{3\times 64} = 2^{162} \times 2^{-1.94} = 2^{160.06}$ 3-round Ascon. The $\frac{50}{3\times 64}$ is because we only need to compute 50 Sboxes to determine the 30-bit matching point.
- The Line 21, the time complexity is $2^{2+130+30} = 2^{162}$ 3-round Ascon.

In **Phase II**, it is trivial to find an inner collision for the 256-bit capacity with the Floyd's cycle finding algorithm [31] with 2^{128} time and no memory.

Thereore, the total preimage attack on 3-round `Ascon-Hash` needs $2^{62.42} + 2^{160.91} + 2^{154.42} + 2^{160.06} + 2^{162} + 2^{128} \approx 2^{162.80}$ time and $2^{64} + 2^{160} + 2^{30} + 2^{30} \approx 2^{160}$ memory.

Fig. 8: The 4-round Preimage attack on `ASCON-Hash`

4-/5-round Preimage Attack on `Ascon-Hash`. The 4-round MitM characteristic shown in Figure 8. Due to the symmetry, in the full MitM path, the starting state $A^{(0)}$ contains 16 ■ bits and 184 ■ bits. The first row of $A^{(0)}$ is fixed as T_1 (marked as ■), and the remaining 56-bit ■ can be freely chosen. In the computation from $A^{(0)}$ to $A^{(2)}$, the consumed degree of freedoms (DoFs) of

$\color{red}{\blacksquare}$ and DoFs of $\color{blue}{\blacksquare}$ are 162 and 8, respectively. Additional, there are 8 consumed DoFs of $\color{red}{\blacksquare}$ to make $a_0 + a_2 + a_4$ become $\color{blue}{\blacksquare}$ or $\color{gray}{\blacksquare}$ for matching points. Therefore, $d_{\mathcal{B}} = 8$, $d_{\mathcal{R}} = 14$, and there are 8 matching bits. The steps for the 4-round MitM attack is very similar to the 3-round attack. We refer the readers to Supplementary Material C for detailed analysis (including a detailed 4-round attack in Algorithm 8.). The total preimage attack on 4-round `Ascon-Hash` needs about $2^{184.85}$ time and 2^{178} memory.

The 5-round Preimage Attack on `Ascon-Hash` is given in Supplementary Material C, whose complexity is about $2^{191.31}$ time and 2^{190} memory.

6.2 Collision attack on 3 and 4-round attack on `Ascon-Hash`

According to Section 4, we have two collision frameworks. In **Collision Framework I** shown in Figure 4, the first row of `Ascon` state is the output, we have to apply the Observation 1 to perform the MitM collision attack. In **Collision Framework II** shown in Figure 5, we try to find messages that collide in the 256-bit capacity part. Therefore, we can fix partial capacity bits and find its preimage. For the Sbox, we can fix the output bits of the capacity part, i.e., any of (b_1, b_2, b_3, b_4), and derive matching equations. In this case, we have a new matching strategy.

Observation 2 (Matching Strategy for Collision Framework II) *According to Eqn. (8), if we fix $b_1 = b_2 = b_3 = b_4 = 0$, then $a_0 = 1$, $a_1 \oplus a_2 = 1$, $a_3 = 0$ and $a_4 = 0$ can be derived. If there are no unknown $\square$ bit in $(a_0, a_1, a_2, a_3, a_4)$, we can immediately obtain 4 matching equations.*

Obviously, the conditions of Observation 2 is much weaker than that of Observation 1. Therefore, it is more likely to produce better results. In our practical search for MitM path, we find better results when applying the **Collision Framework II** by exploring the new matching strategy.

3-round Collision Attack on `Ascon-Hash`. The 3-round MitM path in Figure 9 can be used to build collision attacks on `Ascon-Hash`, where the starting state $A^{(0)}$ contains 24 $\color{blue}{\blacksquare}$ bits and 36 $\color{red}{\blacksquare}$ bits, $d_{\mathcal{B}} = 24$, $d_{\mathcal{R}} = 24$, and $m = t = 24$. There are totally 32 conditions on $\color{gray}{\blacksquare}$ of $A^{(0)}$, *i.e.*, $\mu = 32$, which are listed in Table 2 in Supplementary Material C.

We give the MitM collision attack on 3-round `Ascon-Hash` in Algorithm 3. We use three message blocks (M_1, M_2, M_3) to conduct the collision attack, and the MitM procedure is placed at the 3rd block. In one MitM episode in Line 12 to Line 19, $2^{d_{\mathcal{R}}+d_{\mathcal{B}}-t} = 2^{24+24-24} = 2^{24}$ partial target preimages are expected to obtain. We need $2^{(n-t)/2-(d_{\mathcal{R}}+d_{\mathcal{B}}-t)} = 2^{(256-24)/2-24} = 2^{92}$ MitM episodes to build the collision attack, i.e., $2^{\zeta-32+4+12} = 2^{92}$ and $\zeta = 108$. The time complexity of steps in Alg. 3 are analyzed below:

- In Line 3, the time complexity is $2^{108} \times 2 = 2^{109}$ 3-round `Ascon`.

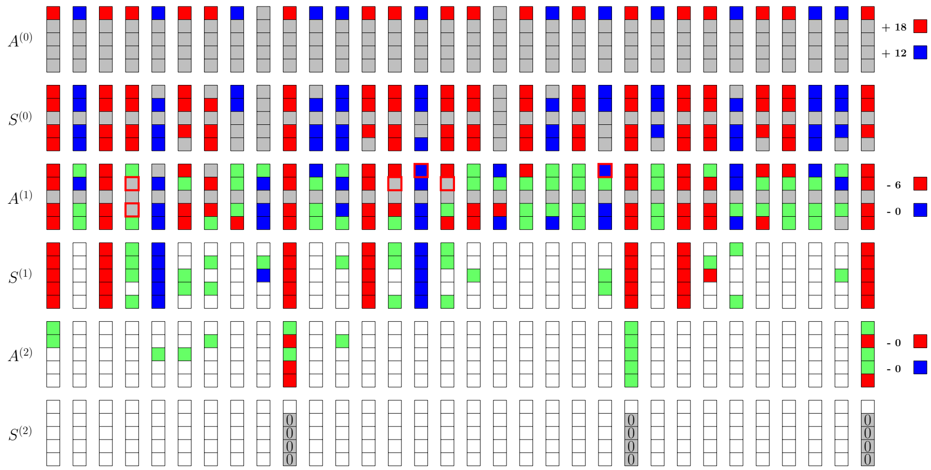


Fig. 9: The 3-round Collision attack on `ASCON-Hash`

- In Line 7, the time complexity is $2^{\zeta-32+4} \times \frac{1}{3} = 2^{80} \times 2^{-1.58} = 2^{78.42}$ 3-round `Ascon`.
- In Line 9, the time complexity is $2^{\zeta-32+4+36} \times \frac{74}{192} = 2^{116} \times 2^{-1.38} = 2^{114.62}$ 3-round `Ascon`.
- In Line 12, the time complexity is $2^{\zeta-32+4+12+24} \times \frac{1}{192} = 2^{116} \times 2^{-7.58} = 2^{108.42}$ 3-round `Ascon`.
- In Line 14, given 24 ■ bits of $A^{(0)}$, the time of computing the 24-bit matching point is $24 + 28 = 52$ Sbox applications. Therefore, the time of Line 14 is $2^{\zeta-32+4+12+24} \times \frac{(24+28)}{192} = 2^{116} \times 2^{-1.88} = 2^{114.12}$ 3-round `Ascon`.
- In Line 17, the time complexity is $2^{\zeta-40+10+6+24} = 2^{116}$ 3-round `Ascon`.

The total time complexity is $2^{109} + 2^{78.42} + 2^{114.62} + 2^{108.42} + 2^{114.12} + 2^{116} \approx 2^{116.74}$ 3-round `Ascon`. The memory is $2^6 + 2^{116} = 2^{116}$ to store U and L.

Experiment on the 3-round Collision Attack. Since the full 3-round attack has an impractical complexity, we only implement one MitM episode to verify the full attack indirectly. According to Algorithm 3, we first fix 24-bit target as 0 in Line 1. Choose all the ■ bits in $A^{(0)}$ to satisfy the 32-bit conditions directly, and fix all other ■ to zero. In Line 9, we prepare the table U, since only one MitM episode is performed, we only store $U[c_{\mathcal{R}}]$ with $c_{\mathcal{R}} = 0$, which needs 2^{24} memory cost. In the MitM episode between Line 12 to 17, 2^{24} time complexity is needed to produce 2^{24} preimages, whose 24-bit target value in $S^{(2)}$ is 0. Obviously, to find 2^{24} preimages with a 24-bit target fixed as 0, a brute-force search takes $2^{24+24=48}$ time. The source codes and results are available via https://github.com/boxindev/MitM_attack_sponge.

We deploy the experiment on a computer with i9-13900KF CPU and 32GB memory. In each experiment, the time of precomputation, i.e., building table U, is about 8500 seconds. Then, a MitM episode takes about 9 seconds, and about

$2^{24.0067}$ preimages are produced with 24-bit 0 in $S^{(2)}$. We list 10 examples in Table 4 in Supplementary Material C, since all the $\color{gray}\blacksquare$ except for 32 conditions are set to zero in this experiment, the last 4 rows of message are fixed.

Algorithm 3: Collision Attack on 3-round `Ascon-Hash`

Fix the 24 bits 0 in $S^{(2)}$ as shown in Figure 9 in order to build the matching points
for 2^{ς} *values of* (M_1, M_2) **do**
    Compute the inner part of the state after absorbing M_2 and applying the permutation
    if *the 32 conditions are satisfied* `/* probability of` 2^{-32} `*/`
    then
        for $2^{64-24-36} = 2^4$ *values of the* $\color{gray}\blacksquare$ *bits in* M_3 **do**
            Compute the $\color{gray}\blacksquare$ bits in $A^{(0)}$ to $A^{(1)}$, except those $\color{gray}\boxed{\blacksquare}$ bits
            for 2^{36} *values of the* $\color{red}\blacksquare$ *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
                Compute forward to determine the 12-bit $\color{gray}\boxed{\blacksquare}$/$\color{blue}\boxed{\blacksquare}$ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{12}$), and the 24-bit matching point. Build the table U and store the 36-bit $\color{red}\blacksquare$ $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 24-bit matching point in $U[c_{\mathcal{R}}]$
            end
            for $c_{\mathcal{R}} \in \mathbb{F}_2^{12}$ **do**
                Retrieve the 2^{24} elements of $U[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 24-bit matching point
                for 2^{24} *values of the* $\color{blue}\blacksquare$ *bits* $v_{\mathcal{B}}$ **do**
                    Compute forward to the 24-bit matching point and store $v_{\mathcal{B}}$ in L_2 indexed by the 24-bit matching point
                end
                for *values matched between* L_1 *and* L_2 **do**
                    Compute the 256-bit capacity c from the matched $\color{red}\blacksquare$ and $\color{blue}\blacksquare$ bits and store the (M_1, M_2, M_3, c) in L indexed by c
                    if *the size of* L *is* $2^{(n-t)/2} = 2^{116}$ **then**
                        Check L and return (M_1, M_2, M_3) and (M_1', M_2', M_3') with the same c
                    end
                end
            end
        end
    end
end

4-round Collision Attack on `Ascon-Hash`. The 4-round MitM path in Figure 10 can be used to build collision attacks on `Ascon-Hash`, where the starting state $A^{(0)}$ contains 8 $\color{blue}\blacksquare$ bits and 54 $\color{red}\blacksquare$ bits, $d_{\mathcal{B}} = 8$, $d_{\mathcal{R}} = 54 - 46 = 8$, and $m = t = 8$.

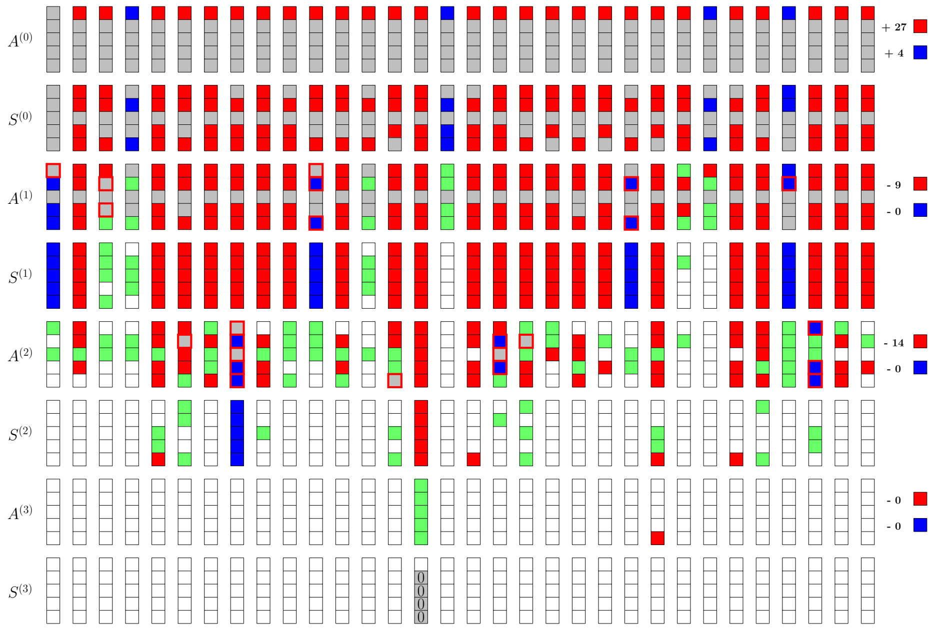


Fig. 10: The 4-round Collision attack on `ASCON-Hash`

There are totally 52 conditions on ■ of $A^{(0)}$, *i.e.*, $\mu = 52$, which are listed in Table 3 in Supplementary Material C.

We give the MitM collision attack on 4-round `Ascon-Hash` in Algorithm 4. We also use three message blocks (M_1, M_2, M_3) to conduct the collision attack, and the MitM procedure is placed at the 3rd block. In one MitM episode in Line 12 to Line 20, $2^{d_{\mathcal{R}}+d_{\mathcal{B}}-t} = 2^{8+8-8} = 2^8$ partial target preimages are expected to obtain. We need $2^{(n-t)/2-(d_{\mathcal{R}}+d_{\mathcal{B}}-t)} = 2^{(256-8)/2-8} = 2^{116}$ MitM episodes to build the collision attack, i.e., $2^{\zeta-52+2+46} = 2^{116}$ and $\zeta = 120$. The time complexity of steps in Alg. 4 are analyzed below:

- In Line 3, the time complexity is $2^{120} \times 2 = 2^{121}$ 4-round `Ascon`.
- In Line 7, the time complexity is $2^{\zeta-52+2} \times \frac{1}{3} = 2^{70} \times 2^{-1.58} = 2^{68.42}$ 4-round `Ascon`.
- In Line 9, the time complexity is $2^{\zeta-52+2+54} \times \frac{(54+52+24)}{256} = 2^{124} \times 2^{-0.98} = 2^{123.02}$ 4-round `Ascon`.
- In Line 12, the time complexity is $2^{\zeta-52+2+46+8} \times \frac{1}{256} = 2^{116}$ 4-round `Ascon`.
- In Line 14, given 8 ■ bits of $A^{(0)}$, the time of computing the 8-bit matching point is $8 + 16 + 20 = 44$ Sbox applications. Therefore, the time of Line 14 is $2^{\zeta-52+2+46+8} \times \frac{44}{256} = 2^{124} \times 2^{-2.54} = 2^{121.46}$ 4-round `Ascon`.
- In Line 17, the time complexity is $2^{\zeta-52+2+46+8} = 2^{124}$ 4-round `Ascon`.

The total time complexity is $2^{121} + 2^{68.42} + 2^{123.02} + 2^{116} + 2^{121.46} + 2^{124} \approx 2^{124.85}$ 4-round `Ascon`. The memory is $2^{54} + 2^{124} = 2^{124}$ to store U and L.

Algorithm 4: Collision Attack on 4-round `Ascon-Hash`

Fix the 8 bits, i.e. the last 4 bits of the output of 14-th, 46-th Sboxes of $A^{(3)}$, to build the matching points

for 2^{ζ} *values of* (M_1, M_2) **do**

  Compute the inner part of the 3rd block

  if *the 52 conditions are satisfied* `/* probability of` 2^{-52} `*/` **then**

    for $2^{64-8-54} = 2^2$ *values of the free* $\blacksquare$ *bits in* M_3 **do**

      Compute the $\blacksquare$ bits in $A^{(0)}$ to $A^{(1)}$, except those $\boxed{\blacksquare}$ bits

      for 2^{54} *values of the* $\blacksquare$ *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**

        Compute forward to determine the 46-bit $\boxed{\blacksquare}/\boxed{\blacksquare}$ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{46}$), and the 8-bit matching point. Build the table U and store the 54-bit $\blacksquare$ $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 8-bit matching point in $U[c_{\mathcal{R}}]$.

      end

      for $c_{\mathcal{R}} \in \mathbb{F}_2^{46}$ **do**

        Retrieve the 2^8 elements of $U[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 8-bit matching point

        for 2^8 *values of the* $\blacksquare$ *bits* $v_{\mathcal{B}}$ **do**

          Compute to the 8-bit matching point and store $v_{\mathcal{B}}$ in L_2 indexed by the 8-bit matching point.

        end

        for *values matched between* L_1 *and* L_2 **do**

          Compute the 256-bit capacity c from the matched $\blacksquare$ and $\blacksquare$ bits and store the (M_1, M_2, M_3, c) in L indexed by c

          if *the size of* L *is* $2^{(n-t)/2} = 2^{124}$ **then**

            Check L and return (M_1, M_2, M_3) and (M_1', M_2', M_3') with the same c

          end

        end

      end

    end

  end

end

Comment on the memory complexity. The straightforward version of the attack requires to store all the preimages generated as shown in Line 17 of Algorithm 3, which makes the memory complexity be of the same order as the time complexity. However, at ASIACRYPT 2012, Khovratovich [38, Section 4] implied that the memoryless collision search methods [56] may be applied, which multiply the time complexity by a small constant without storing the preimages. However, we are unable to figure out the detailed steps of the memoryless approach and leave it as an open problem.

7 Application to SPONGENT

7.1 Sieve-in-the-middle Technique

Usually, the matching of the MitM checks the compatibility between the two sets of neutral bits with a simple equality test (e.g., $g_{\mathcal{B}} = g_{\mathcal{R}}$) at a given round in the hash function [51]. In [46,16], the authors proposed the sieve-in-the-middle that leverages the valid transitions through some middle Sbox. Suppose, for the Sbox S with n-bit input and output pair (x, y), the attacker is able to compute from the one set of neutral bits (e.g., ■ bits) an $|u|$-bit vector u that corresponds to a part of input x of S. While compute from the red ■ neutal bits a $|v|$-bit vector v that corresponds to a part of input y of S. $\mathcal{R}(u, v) = 1$ if and only if (u, v) corresponds to a valid pair of input and output of S. Canteaut et al. [16] gives the following definition and proposition.

Definition 1. *Let $S : \mathbb{F}_2^n \mapsto \mathbb{F}_2^n$. Let $I, J \subset \mathbb{F}_2^n$ be two subsets with sizes $|u|$ and $|v|$, respectively. The sieving probability of (I, J) denotes by $\pi_{I,J}$ is the proportion of all elements in $\mathbb{F}_2^{|u|+|v|}$ which can be written as $(x \in I; S(x) \in J)$ for some x. The pair (I, J) is called an $(|u|, |v|)$-sieve for S if $\pi_{I,J} < 1$.*

Proposition 1. *Any pair (I, J) of sets of size $(|u|, |v|)$ with $|u| + |v| > n$ is a sieve for S with sieving probability $\pi_{I,J} \leq 2^{n-(|u|+|v|)}$.*

There are three algorithms proposed in [46], i.e., *instant matching*, *gradual matching*, and *parallel matching without memory*. In our paper, we only find application of the *instant matching*. Suppose the sizes of the two lists $L_{\mathcal{B}}$ and $L_{\mathcal{R}}$ built from the ■ and ■ neutral bits are $2^{d_{\mathcal{B}}}$ and $2^{d_{\mathcal{R}}}$, respectively. A formal description of instant matching is given in Algorithm 5, the list $L_{\mathcal{B}}$ and $L_{\mathcal{R}}$ are decomposed into l groups. The time complexity is $\pi 2^{d_{\mathcal{B}}+|u|} + \pi 2^{d_{\mathcal{B}}+d_{\mathcal{R}}}$ with memory $2^{d_{\mathcal{B}}} + 2^{d_{\mathcal{R}}}$.

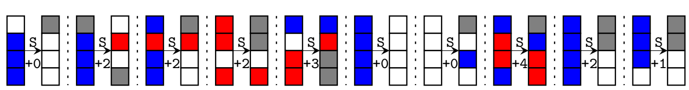


Fig. 11: Some possible colors of match

Algorithm 5: Instant matching

Input: $L_{\mathcal{B}}, L_{\mathcal{R}}$

1 Build the tables L_j such that $L_j[v_j]$ corresponds to all u_j with $R_j(u_j, v_j) = 1$

2 **for** $(b_1, b_2, \cdots, b_t) \in L_{\mathcal{B}}$ /* $b_i \in L^i_{\mathcal{B}}$ */

3 **do**

4     $L_{aux} \leftarrow \emptyset$

5   **for** *i from 1 to l* **do**

6   **if** $L_i[b_i]$ *is empty* **then**

7   go to Step 2

8   **end**

9   **end**

10   Add all tuples $(a_1, a_2, \cdots, a_l)$ with $a_j \in L_j[b_j]$, $\forall j$, to L_{aux}

11   **for** $(a_1, a_2, \cdots, a_l) \in L_{aux}$ **do**

12   **if** $(a_1, a_2, \cdots, a_l) \in L_{\mathcal{R}}$ **then**

13   Add $(a_1, a_2, \cdots, a_l, b_1, b_2, \cdots, b_l)$ to L_{sol}

14   **end**

15   **end**

16 **end**

17 **return** L_{sol}

7.2 Introducing Sieve-in-the-middle Technique into Automatic Tool

The key point for introducing sieve-in-the-middle technique into MILP automatic tool is to count the degree of matching. According to Proposition 1 and take the 4×4 Sbox as example, if $|u|$ and $|v|$ bits of the input and output of Sbox are known, and $|u| + |v| > 4$, then the sieve probability $\pi = 2^{4-(|u|+|v|)}$. Therefore, in the MILP model, we let the degree of matching (DoM) for one Sbox be $\delta_{\mathcal{M}} = (|u| + |v|) - 4$ in bits, and the full DoM= $\sum \delta_{\mathcal{M}}$. Figure 11 shows some possible colors of match.

7.3 Description of SPONGENT

SPONGENT lightweight hash function family [12] proposed by Bogdanov et al. at CHES 2011 is based on the generalized sponge construction instantiated with b-bit PRESENT-type permutations. SPONGENT has been selected as ISO standard. The round function consist of Sbox layer with 4-bit Sbox and a bit permutation layer P_b moves bit j of the b-bit state to bit position $P_b(j)$:

$$P_b(j) = \begin{cases} j \cdot b/4 \bmod b-1, & \text{if} \in \{0, ..., b-2\} \\ b-1, & \text{if} j = b-1 \end{cases}$$

The internal states of round r are denoted as $A^{(r)} \xrightarrow{Sbox} S^{(r)} \xrightarrow{P_b} A^{(r+1)}$. We only focus on SPONGENT-88, which adopts $b = 88$-bit permutation with capacity $c = 80$, rate $r = r' = 8$, and digest $n = 88$. The total number of rounds is 45. The security against preimage and collision is claimed to be 80-bit and

40-bit, respectively. The only cryptanalysis result against one of the 3 basic security properties (preimage, 2nd preimage, or collision) is the designer's self-cryptanalysis, a rebound attack designed to build a 6-round collision attack on `SPONGENT-88`. However, the time complexity of the rebound attack is at least $2^{55.2}$ [11, Section 3.2], which is even higher than that of the trivial birthday attack with 2^{40}.

6-round Preimage Attack on `SPONGENT-88`. The squeezing phase of `SPONGENT-88` outputs eleven 8-bit blocks as 88-bit target, i.e., $T = T_1 \| T_2 \| \cdots \| T_{11}$. Different from the MitM framework given in Figure 3, we apply inverse permutation for the attack. Denote the output of the permutation when squeezing T_2 is $T_2 \| Y$, then we let state Y be free state and T_1 be fixed as target. Our MitM attack is to find the preimage Y of the target T_1.

The 6-round MitM characteristic is shown in Figure 12. The first 8 bits of $A^{(0)}$ is fixed as T_1 (marked as ■), and the remaining 80-bit □ are capacity which are not known. Compute the partial state of the first round forward, 8-bit of $A^{(1)}$ can be deduced (marked as ■), which can be used for the matching. Since we carry out the MitM attack inversely, the starting state is $A^{(6)}$, which contains 73 ■ bits and 6 ■ bits. In the inverse computation from $A^{(6)}$ to $S^{(1)}$, the consumed degree of freedoms (DoFs) of ■ and DoFs of ■ are 67 and 0, respectively. Therefore, $d_{\mathcal{B}} = 6$, $d_{\mathcal{R}} = 6$. Accoring to the sieve-in-the-middle, the degree of matching is 6, i.e., the DoM=2+2+1+1=6, includes two Sboxes with 4 ■ bits input and 2 ■ bits output (i.e., $S^{(1)}[0,1,2,3] \mapsto A^{(1)}[0,1,2,3]$, $S^{(1)}[20,21,22,23] \mapsto A^{(1)}[20,21,22,23]$), and two Sboxes with 3 ■ bits input and 2 ■ bits output (i.e., $S^{(1)}[44,45,46,47] \mapsto A^{(1)}[44,45,46,47]$, $S^{(1)}[64,65,66,67] \mapsto A^{(1)}[64,65,66,67]$), which are marked by black boxes. The 6-round MitM attack is given in Algorithm 6.

Analysis of Algorithm 6. In Line 10 to 17, $2^{67+6+6+\zeta}$ states are tested against the 80-bit $T_1 \| T_3 \| \cdots \| T_{11}$, therefore, $\zeta = 1$ is enough to find a preimage.

- In Line 7, the time complexity is $2^{1+73} \times \frac{20+11+6}{6\times 22} = 2^{74} \times 2^{-1.83} = 2^{72.17}$ 6-round `SPONGENT`.
- In Line 12, the time complexity is $2^{1+67+6} \times \frac{35}{6\times 22} = 2^{74} \times 2^{-1.92} = 2^{72.08}$ 6-round `SPONGENT`.
- In Line 17, the time complexity is $2^{1+67+6} = 2^{74}$ 6-round `SPONGENT`.

In **Phase II**, it is trivial to find an inner collision for the 80-bit capacity with the Floyd's cycle finding algorithm [31] with 2^{40} time and no memory.

Thereore, the total preimage attack on 6-round `SPONGENT-88` is $2^{72.17} + 2^{72.08} + 2^{74} + 2^{40} \approx 2^{74.63}$ time and $2^{73} + 2^{6} + 2^{6} \approx 2^{73}$ memory. We also find a 7-round preimage attack on `SPONGENT-88` with $2^{78.59}$ time and 2^{67} memory. Please check the details in Supplementary Material F.

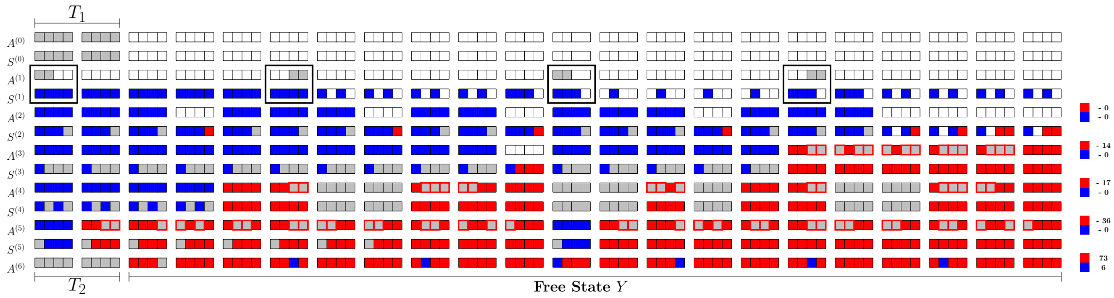


Fig. 12: The MitM preimage attack on 6-round `SPONGENT-88`

Algorithm 6: Preimage Attack on 6-round `SPONGENT-88`: Phase I

Fix the first 8 bits of $A^{(0)}$ as T_1, and compute forward to get the 8-bit of $A^{(1)}$, which are marked as ■

`/* According to Alg. 5, prepare tables with valid elements through Sbox matching for the 4 Sbox. For simplicity, we build a full table containing all valid elements. */`

Build the table L, stored $2^{4-2} \times 2^{4-2} \times 2^{3+1-2} \times 2^{3+1-2} = 2^8$ valid elements indexed by $4+4+3+3 = 14$ blue bits, i.e., $S^{(1)}[0, 1, 2, 3, 20, 21, 22, 23, 44, 45, 46, 64, 65, 66]$

`/* Therefore, under each 14-bit index, there are` $2^8/2^{14} = 2^{-6}$ `elements. */`

for 2^{ζ} *values of the 1-bit free gray bits in* $A^{(6)}$ **do**
    for 2^{73} *values of the* ■ *bits* $v_{\mathcal{R}}$ *in* $A^{(6)}$ **do**
        Compute backward to determine the 67-bit ■/■ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{67}$) and store the 73-bit ■ $v_{\mathcal{R}}$ of $A^{(6)}$ in $V[c_{\mathcal{R}}]$
    end
    for $c_{\mathcal{R}} \in \mathbb{F}_2^{67}$ **do**
        Retrieve the 2^6 $v_{\mathcal{R}}$ from $V[c_{\mathcal{R}}]$
        for 2^6 *values of* ■ *bits* $v_{\mathcal{B}}$ **do**
            Compute to derive the $4+4+3+3 = 14$ ■ bits $\xi = S^{(1)}[0, 1, 2, 3, 20, 21, 22, 23, 44, 45, 46, 64, 65, 66]$
            if $L[\xi]$ *is not empty* `/* Probability of` 2^{-6} `*/`
            then
                Combine $v_{\mathcal{B}}$ and 2^6 $v_{\mathcal{R}}$ to construct the full state Y
                if T_1 *is satisfied* **then**
                    Check if $T_3 \| \cdots \| T_{11}$ is satisfied
                end
            end
        end
    end
end

---

8 The 9-round Preimage Attack on ACE-$\mathcal{H}$-256

ACE [1] is one of the second round NIST LWC candidates. It has a 320-bit permutation and 16 iterations of round function, which is shown in Figure 13 by ignoring round constants addition. SB-64 is an 8-round unkeyed Simeck block cipher [59] with 8-byte block. In [1], Aagaard et.al. offered a sponge-based hash algorithm with ACE premutation, denoted by ACE-$\mathcal{H}$-256 with $b = 320$, $c = 256$, $r = r' = 64$, and $n = 256$. The security claim by the designers on preimage attack is 2^{192}.

As shown in Figure 13, $A^{(0)}[0,1,2,3]$ and $C^{(0)}[0,1,2,3]$ are used for both absorbing and squeezing. We denote the four squeezed 8-byte blocks as $T_1\|T_2\|T_3\|T_4$, with the new attack framework as shown in Figure 3, a 9-round (out of 16 rounds) MitM preimage attack on ACE-$\mathcal{H}$-256 is found:

- **Phase I:** As shown in Figure 13, given $T_1 = A^{(0)}[0\text{-}3]\|C^{(0)}[0\text{-}3]$ and $T_2 = A^{(9)}[0\text{-}3]\|C^{(9)}[0\text{-}3]$, the 32-byte capacity state X is separated into two neutral sets, i.e., 12 ■ bytes and 16 ■ bytes. In the forward computation, 8 ■ bytes and 12 ■ bytes are consumed. Therefore, the DoFs of ■ and ■ are both 4 bytes. In the matching phase as shown in Figure 14, we use the *instant matching* strategy in [46] by seeing the four consecutive SB-64 as one Big-SBox. Then, 4-byte degree of match can be gotten with an addition table L. The detailed attack is given in Algorithm 7. The precomputation time to build L is about 2^{160} with a memory 2^{128} to store L. The time of MitM procedure is about $2^{128+32} = 2^{160}$ with a memory of 2^{32} to store $L_{\mathcal{R}}$.
- **Phase II:** Once getting a valid X, the Floyd's cycle finding algorithm is applied to find a collision at $c = 256$ bits inner part with time 2^{128}.

Hence, the time and memory complexity are both dominated by Phase I, i.e. 2^{160} and 2^{128}.

9 Applications to PHOTON and PHOTON-Beetle-Hash

PHOTON lightweight hash function family [36] was proposed by Guo et al. at CRYPTO 2011, which is also one of the ISO standards. It adopts 12-round AES-like permutations P with an internal state of d^2 elements of s bits each (arranged as a $d \times d$ matrix). The operations in each round are AddConstants (AC), SubCells (SC), ShiftRows (SR), and MixColumnsSerial (MC). In [36], the authors offered five different hash variants and we focus on the PHOTON-160/36/36 with $b = 196$-bit P_{196}, $d = 7$, $s = 4$, $r = r' = 36$, $n = c = 160$, and a 124-bit preimage security, and keep other hash variants in Supplementary Material E.

9.1 Preimage Attack on 4.5-round PHOTON-160/36/36

In this section, we take PHOTON-160/36/36 as an example to show the detailed attack procedure. Other variants can be analysed in the same way and the corresponding attack complexities are given in Table 1. With the new attack framework as shown in Figure 3, we give a MitM preimage attack on 4.5-round without the MC in the last round in Algorithm 12 in Supplementary Material E.1.

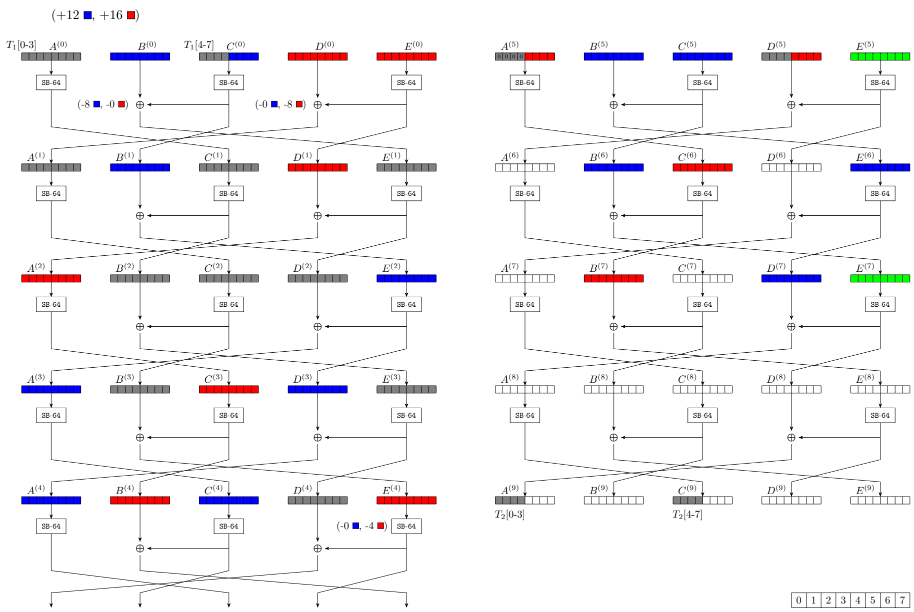


Fig. 13: The MitM preimage attack on 9-round ACE-$\mathcal{H}$-256

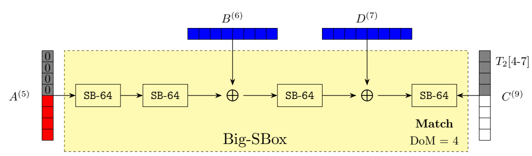


Fig. 14: The Big-SBox matching for 9-round preimage attack on ACE-$\mathcal{H}$-256

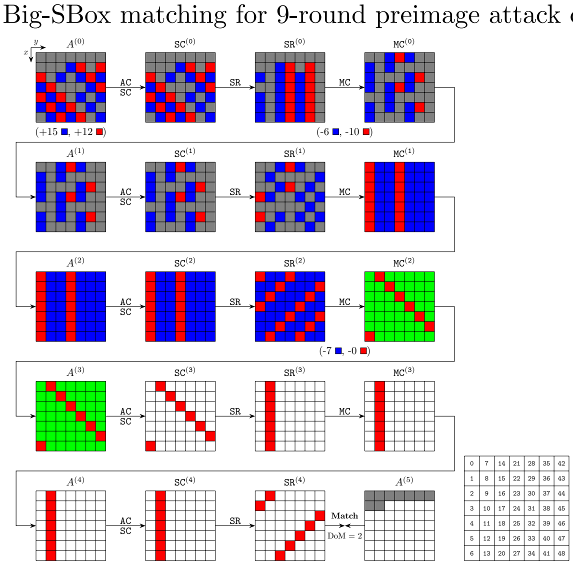


Fig. 15: The 4.5-round MitM attack on PHOTON-160/36/36

Algorithm 7: MitM Preimage Attack on 9-round ACE-$\mathcal{H}$-256

Let $A^{(0)}[0,1,2,3] \| C^{(0)}[0,1,2,3]$ be T_1
`/* Precomputation of instant matching */`
Set $A^{(5)}[0,1,2,3] = 0$
$L \leftarrow [\ ]$
for $2^{16\times 8=128}$ *values for* ■ *bytes of* $\tau = B^{(6)} \| D^{(7)}$ **do**
  for $2^{4\times 8=32}$ *values for* ■ *bytes of* $\xi = A^{(5)}[4,5,6,7]$ **do**
    if *Big-SBox* $(A^{(5)}, B^{(6)}, D^{(7)})[0\text{-}3]$ *equals to* $T_2[4\text{-}7]$ **then**
      $L[\tau] \leftarrow \xi$ `/* There exists` $2^{128+32-32}/2^{128} = 1$ `element under each` τ `on average, therefore, there are totally` $2^{128+32-32=128}$ `elements stored in` L `*/`
    end
  end
end
`/* Main procedure of MitM preimage attack */`
Set $A^{(0)}[4,5,6,7]$ to be constant value
for $2^{16\times 8=128}$ *possible values of* ■ *bytes in* $A^{(1)}$ *and* $E^{(1)}$ **do**
  $L_{\mathcal{R}} \leftarrow [\ ]$
  Compute forward to the ■ words $B^{(2)}$, $C^{(2)}$, $D^{(2)}$, $B^{(3)}$, $E^{(3)}$, $D^{(4)}$
  $D^{(5)}[0\text{-}3] \leftarrow D^{(4)}[0\text{-}3]$
  for $2^{4\times 8=32}$ ■ *values of* $D^{(5)}[4,5,6,7]$ **do**
    $A^{(5)} \leftarrow D^{(4)} \oplus D^{(5)}$ `/* This step makes` $A^{(5)}[0,1,2,3] = 0$ `*/`
    Compute backward to the 16-byte values of $v_{\mathcal{R}} = D^{(0)} \| E^{(0)}$
    Store $v_{\mathcal{R}}$ in $L_{\mathcal{R}}$ indexed by $D^{(5)}[4,5,6,7]$
    `/* There is 1 element under each index */`
  end
  for $2^{4\times 8=32}$ *possible values of* ■ *bytes in* $C^{(0)}[4,5,6,7]$ **do**
    Compute $B^{(0)}$ by SB-64$(C^{(0)}) \oplus E^{(1)}$
    Compute forward to $\tau = B^{(6)} \| D^{(7)}$
    Retrieve one element in $L[\tau]$ as ξ'
    Reconstruct the (candidate) state X by $B^{(0)}$, $C^{(0)}$, and $L_{\mathcal{R}}[\xi']$
    if X *satisfies 192-bit* $T_2 \| T_3 \| T_4$ **then**
      Output X and stop
    end
  end
end

- **Phase I:** Slightly different from the Figure 15, we could see $\texttt{SR}^{(0)}$ as the starting state. There are 12 $\textcolor{red}{\blacksquare}$ cells and 15 $\textcolor{blue}{\blacksquare}$ cells. In the forward computation, 10 $\textcolor{red}{\blacksquare}$ and 13 $\textcolor{blue}{\blacksquare}$ are consumed. The DoFs of $\textcolor{red}{\blacksquare}$, $\textcolor{blue}{\blacksquare}$, and matching are 2 cells. The detail of the MitM attack is given in Algorithm 12. The time complexity is about $2^{124-\min\{8,8,8\}} = 2^{116}$. The memory is 2^{36} to store U.
- **Phase II:** Since the capacity is $c = 160$ bits, and the Floyd's cycle finding algorithm finds a collision in the capacity with 2^{80} time and without memory.

Hence, the total time complexity is 2^{116} and memory complexity is 2^{36}.

Additionally, we give the 4.5-round attacks on all other versions of `PHOTON`-$n/r/r'$ as shown in Figure 25, 26, 27, 28 in Supplementary Material E.1, and we also give a 3.5-round attack on `PHOTON-Beetle-Hash` in Supplementary Material E.2. The complexities are summarized in Table 1.

10 Application to `Subterranean 2.0`

`Subterranean 2.0` family, designed by Daemen et al. [18], is a second round candidate of NIST LWC. In the family, the designers also provide a hash function `Subterranean-XOF` with a 112-bit security claim [18]. The designers have performed good study to support the 112-bit security. However, their analysis is about generating collision on the (257-9=) 248-bit inner part in the absorbing phase (see Section 3.4 of [18]). This may finally lead to collision attack and there is no cryptanalysis against preimage attack. At CRYPTO 2022, Lefevre and Mennink proved a tight bound of 224-bit preimage security [42] under ideal permutation model for `Subterranean-XOF` with 256-bit digest ($n = 256$, $b = 257$, $r = 9$, $r' = 32$, $c = 248$). However, the permutation of `Subterranean-XOF` only consists of 2 rounds (far from ideal). Therefore, it arises a natural open question how the preimage attack will work for `Subterranean-XOF` if the permutation is not ideal. In this section, we give the first preimage attack with complexity 2^{160} instead of the generic 2^{224}. We emphasize our attack does not break the designers' security claim.

The internal state of `Subterranean-XOF` is 257 bits, and the round function contains four operations: $\chi : s_i \leftarrow s_i + (s_{i+1} + 1)s_{i+2}$, ι : constant addition, $\theta : s_i \leftarrow s_i + s_{i+3} + s_{i+8}$, and $\pi : s_i \leftarrow s_{12i}$. The internal state of each round is updated as : $A^{(r)} \xrightarrow{\chi\circ\iota} S^{(r)} \xrightarrow{\theta} \theta^{(r+1)} \xrightarrow{\pi} A^{(r+1)}$. The output function is $z_i = s_{12^{4i}} + s_{-12^{4i}}$, $(0 \leqslant i < 32)$ as shown in Table 8 in Supplementary Material G, e.g., when $i = 0$, $z_0 = s_1 + s_{256}$. After each 32-bit digest is squeezed out, 1-round function is executed to update the internal state.

Under the attack framework of Figure 3, assume that the 256-bit hash value is T, which consists of T_i $(1 \leq i \leq 8)$. In **Phase I** of Figure 3, taking the internal state as the starting point after outputting T_1, denoted as $A^{(0)}$. When $(s_{12^{4i}}, s_{-12^{4i}})$ of $A^{(0)}$ is ($\textcolor{gray}{\blacksquare}\textcolor{gray}{\blacksquare}$), ($\textcolor{red}{\blacksquare}$,$\textcolor{red}{\blacksquare}$), or ($\textcolor{blue}{\blacksquare}$,$\textcolor{blue}{\blacksquare}$), the DoF for $(s_{12^{4i}}, s_{-12^{4i}})$ is only 1, since the output bit $z_i = s_{12^{4i}} + s_{-12^{4i}}$ is fixed as T_1. Therefore, when counting number of $\textcolor{red}{\blacksquare}$ bits of $(s_{12^{4i}}, s_{-12^{4i}})$ in $A^{(0)}$, ($\textcolor{red}{\blacksquare}$,$\textcolor{red}{\blacksquare}$) is only counted as 1-bit $\textcolor{red}{\blacksquare}$. Similar to $\textcolor{blue}{\blacksquare}$ and $\textcolor{gray}{\blacksquare}$ bits. In the matching phase, when $(s_{12^{4i}}, s_{-12^{4i}})$ for the digest output has no unknown □ bit, 1-bit matching point is derived. The matching points can be constructed by the digest output of $A^{(r)}$ $(r \geq 1)$.

The color pattern of the MitM preimage attack for **Phase I** is shown in Figure 31. There are 64 ■ bits and 100 ■ bits in $A^{(0)}$. It consumes 36 DoFs of ■, so that $d_B = 64$, $d_R = 64$. The final matching points are 65 bits (DoM= 65) which are marked with m in Figure 31 (note that two bits marked by m in $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$ are served as 1-bit matching). We give the Algorithm 15 in Supplementary Material G. The complexity is about 2^{160} 3-round `Subterranean-XOF` with a memory of 2^{100} to store U. In **Phase II**, since the size of capacity is 248 and the Floyd's cycle finding algorithm finds a collision in the capacity with 2^{124} time. Therefore, the overall complexity to find the 256-bit target preimage is about 2^{160} time and 2^{100} memory.

11 Application to `Xoodyak`

The specification of `Xoodyak` [17] (one of the finalists of NIST LWC) is given in Supplementary Material B.2. We focus on `Xoodyak-XOF` and `Xoodyak-Hash`. For `Xoodyak-Hash` $(b = 384, c = 256, r = r' = 128, n = 256)$, the security claims are 2^{128} against both preimage and collision attacks, we only study it against the collision attacks. The `Xoodyak-XOF` offers an arbitrary output length l and the preimage resistance is $\min(2^{128}, 2^l)$. We target on `Xoodyak-XOF` with a 128-bit digest against the preimage attack.

Collision Attack on 3-round `Xoodyak-Hash`. By applying the **Collision Framework II** (Figure 5), we find the following new matching strategy.

Observation 3 (Matching Strategy of `Xoodyak` for Collision) *Suppose the input and output of the Sbox are (a_0, a_1, a_2) and (b_0, b_1, b_2), we have $b_i = a_i \oplus (a_{i+1} \oplus 1) \cdot a_{i+2}$, where $0 \leq i \leq 2$ according to Eqn. (9). If we fix $b_1 = b_2 = 0$, then $a_0 = a_1$ and $a_2 = 0$ can be derived. If there are no unknown □ bit in (a_0, a_1, a_2), we can immediately obtain 2 matching equations.*

A new 3-round MitM characteristic in Figure 19 is found in Supplementary Material D.1. With the MitM characteristic, we can build collision attack on `Xoodyak-Hash`, which is given in Algorithm 10 in Supplementary Material D.1. The starting state $A^{(0)}$ contains 8 ■ bits and 118 ■ bits. There are totally 51 conditions on ■ bits of $\iota^{(0)}$, which are listed in Table 5. In the computation from $A^{(0)}$ to $\iota^{(2)}$, the consumed DoFs of ■ is 110 and the consumed DoFs of ■ is 0. Therefore, $d_{\mathcal{B}} = 8$, $d_{\mathcal{R}} = 118 - 110 = 8$. We get $m = t = 8$ matching equations with the deterministic relations of $\iota^{(2)}$. The time complexity is about $2^{125.23}$ 3-round `Xoodyak-Hash` with the memory about 2^{124}.

MITM preimage attack on 3-round `Xoodyak-XOF`. Solving with the MILP model for `Xoodyak`, we get a new 3-round MITM preimage attack. The attack Figure 20 and other details are given in Supplementary Material D.2. The attack parameters are $d_{\mathcal{B}} = 8,\ d_{\mathcal{R}} = 118-111 = 7$, $\mu = 53$, $m = 7$. The time complexity of the 3-round preimage attack is $2^{121.77}$ 3-round `Xoodyak-XOF`, and the memory is 2^{118}.

12 Conclusion

In this paper, we propose the generic MitM attack frameworks for preimage and collision attacks on sponge constructions. In the last decade, the sponge-based hash functions with lower claimed security level for preimages compared to their output size have been widely used and standardized. However, cryptanalysis tools regarding the preimage attacks against those hash functions are absent. This paper proposes the first generic cryptanalysis tool for preimage attacks against those hash functions. Most of our results are the first preimage cryptanalysis results. For example, the ISO standard `PHOTON` were designed in 2011, however, no result on round-reduced preimage attack ever proposed by the community before our results. Moreover, our MitM collision attack framework provides a different method to build collisions on sponge construction besides the method of differential attack.

Acknowledgements. We thank the anonymous reviewers from CRYPTO 2024 for their insightful comments. This work is supported by the Natural Science Foundation of China (62272257, 62072270, 62302250), the Major Program of Guangdong Basic and Applied Research (2019B030302008), the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

References

1. Mark Aagaard, Riham AlTawy, Guang Gong, Kalikinkar Mandal, and Raghvendra Rohit. ACE: An authenticated encryption and hash algorithm. *Submission to NIST-LWC (announced as round 2 candidate on August 30, 2019)*, 2019.
2. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *SAC 2008*, volume 5381, pages 103–119. Springer, 2008.
3. Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. PHOTON-Beetle authenticated encryption and hash family. 2021.
4. Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In *EUROCRYPT 2021, Part I*, volume 12696, pages 771–804.
5. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. Superposition meet-in-the-middle attacks: Updates on fundamental security of AES-like hashing. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Proceedings, Part I*, volume 13507 of *LNCS*, pages 64–93. Springer, 2022.
6. Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Schwaemm and Esch: lightweight authenticated encryption and hashing using the Sparkle permutation family. *NIST Lightweight Cryptography*, 2019.

7. Daniel J Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. SPHINCS+: Submission to the NIST post-quantum project. 2017.
8. Daniel J Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke Todo, et al. Gimli. *Submission to the NIST Lightweight Cryptography project. Available online https://csrc. nist. gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/gimli-spec. pdf*, 2019.
9. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST*, 3(30):320–337, 2009.
10. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *ASIACRYPT 2011, Proceedings*, pages 344–371.
11. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. spongent: A lightweight hash function. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2011.
12. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. spongent: A lightweight hash function. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2011.
13. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *SAC 2010*, volume 6544, pages 229–240. Springer, 2010.
14. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In *CRYPTO 2011, Proceedings*, volume 6841, pages 169–187. Springer, 2011.
15. Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 240–272. Springer, 2023.
16. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In *CRYPTO 2013, Proceedings, Part I*, pages 222–240.
17. Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.
18. Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The subterranean 2.0 cipher suite. *IACR Trans. Symmetric Cryptol.*, 2020(S1):262–294, 2020.
19. Ivan Damgård. A design principle for hash functions. In *CRYPTO '89*, pages 416–427.
20. Mathieu Degré, Patrick Derbez, Lucie Lahaye, and André Schrottenloher. New models for the cryptanalysis of ASCON. *IACR Cryptol. ePrint Arch.*, page 298, 2024.

21. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In *CRYPTO 2016, Proceedings, Part II*, volume 9815, pages 157–184. Springer, 2016.
22. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.
23. Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In *EUROCRYPT 2021, Proceedings, Part I*, volume 12696, pages 374–403. Springer, 2021.
24. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *CRYPTO 2012*, volume 7417, pages 719–740.
25. Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on keccak-224 and keccak-256. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 442–461. Springer, 2012.
26. Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 219–240. Springer, 2013.
27. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.
28. Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO 2021, Proceedings, Part III*, volume 12827, pages 278–308. Springer.
29. Orr Dunkelman, Gautham Sekar, and Bart Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In *INDOCRYPT 2007, Proceedings*, volume 4859, pages 86–100. Springer, 2007.
30. Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman. Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 683–701. Springer, 2015.
31. Robert W. Floyd. Nondeterministic algorithms. *J. ACM*, 14(4):636–644, 1967.
32. Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In *FSE 2014*, pages 61–81, 2014.
33. David Gérault, Thomas Peyrin, and Quan Quan Tan. Exploring differential-based distinguishers and forgeries for ASCON. *IACR Trans. Symmetric Cryptol.*, 2021(3):102–136, 2021.
34. Jian Guo, Guohong Liao, Guozhen Liu, Meicheng Liu, Kexin Qiao, and Ling Song. Practical collision attacks against round-reduced SHA-3. *J. Cryptol.*, 33(1):228–270, 2020.
35. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In *ASIACRYPT 2010, Proceedings*, volume 6477, pages 56–75.
36. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

37. Hosein Hadipour and Maria Eichlseder. Autoguess: A tool for finding guess-and-determine attacks and key bridges. In Giuseppe Ateniese and Daniele Venturi, editors, *Applied Cryptography and Network Security - 20th International Conference, ACNS 2022, Rome, Italy, June 20-23, 2022, Proceedings*, volume 13269 of *Lecture Notes in Computer Science*, pages 230–250. Springer, 2022.
38. Dmitry Khovratovich. Bicliques for permutations: Collision and preimage attacks in stronger settings. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 544–561. Springer, 2012.
39. Dmitry Khovratovich, Gaëtan Leurent, and Christian Rechberger. Narrow-bicliques: Cryptanalysis of full IDEA. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012, Proceedings*, volume 7237, pages 392–410.
40. Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced SHA-1. In *CRYPTO 2012, Proceedings*, volume 7417, pages 367–383.
41. 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.
42. Charlotte Lefevre and Bart Mennink. Tight preimage resistance of the sponge construction. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.
43. 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*, volume 7549, pages 264–286. Springer, 2012.
44. Guozhen Liu, Jingwen Lu, Huina Li, Peng Tang, and Weidong Qiu. Preimage attacks against lightweight scheme Xoodyak based on deep learning. In *Future of Information and Communication Conference*, pages 637–648. Springer, 2021.
45. Ralph C. Merkle. A certified digital signature. In *CRYPTO 1989, Proceedings*, pages 218–238, 1989.
46. María Naya-Plasencia. How to improve rebound attacks. In *CRYPTO 2011, Proceedings*, pages 188–205, 2011.
47. The U.S. National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. *Federal Information Processing Standard, FIPS 202, 5th August 2015*.
48. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93*, volume 773, pages 368–378.
49. Lingyue Qin, Jialiang Hua, Xiaoyang Dong, Hailun Yan, and Xiaoyun Wang. Meet-in-the-middle preimage attacks on sponge-based hashing. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 158–188. Springer, 2023.
50. Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In *IWSEC 2018*, volume 11049, pages 227–243.
51. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In Antoine Joux, editor, *Fast Software Encryption -*

18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers, volume 6733 of *Lecture Notes in Computer Science*, pages 378–396. Springer, 2011.
52. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In *EUROCRYPT 2009, Proceedings*, volume 5479, pages 134–152.
53. 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, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 562–579. Springer, 2012.
54. André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. In *CRYPTO 2022, Proceedings, Part III*, volume 13509, pages 717–747. Springer, 2022.
55. André Schrottenloher and Marc Stevens. Simplified modeling of MITM attacks for block ciphers: new (quantum) attacks. *IACR Cryptol. ePrint Arch.*, page 816, 2023.
56. Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptol.*, 12(1):1–28, 1999.
57. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *CRYPTO 2005, Proceedings*, volume 3621, pages 17–36. Springer, 2005.
58. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT 2005, Proceedings*, volume 3494, pages 19–35. Springer, 2005.
59. Gangqiang Yang, Bo Zhu, Valentin Suder, Mark Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In *Workshop on Cryptographic Hardware and Embedded Systems*, 2015.
60. Zhongyi Zhang, Chengan Hou, and Meicheng Liu. Collision attacks on round-reduced SHA-3 using conditional internal differentials. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 220–251. Springer, 2023.
61. Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Collision attacks on round-reduced Gimli-Hash/Ascon-Xof/Ascon-Hash. *Cryptol. ePrint Arch., Paper 2019/111.*

Supplementary Material

A Sasaki's MitM attack on 7-round AES hashing Mode

We take the MitM preimage attack on 7-round AES-hashing in [51] as an example.

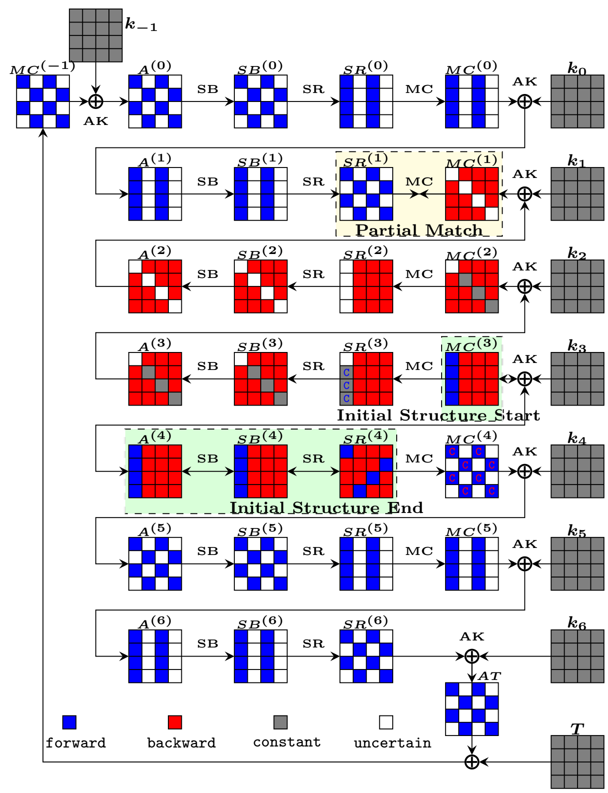


Fig. 16: The MitM preimage attack on 7-round AES-hashing.

Denote the internal states of round r as

$$A^{(r)} \xrightarrow{SB} SB^{(r)} \xrightarrow{SR} SR^{(r)} \xrightarrow{MC} MC^{(r)} \xrightarrow{AK} A^{(r+1)}.$$

$A^{(r)}_{\{i\}}$ represents the i-th ($0 \le i \le 15$) byte of state $A^{(r)}$ numbered from up to bottom, left to right. $A^{(r)}_{\{i-j\}}$ represents the i-th byte to j-th byte of state $A^{(r)}$.

Chunk Separation: As shown in Figure 16, the initial structure involves a few consecutive starting steps, i.e. $\{MC^{(3)}, A^{(4)}, SB^{(4)}, SR^{(4)}\}$. The $MC^{(3)}_{\{0-3\}}$

are chosen as neutral bytes (marked by blue) for the forward chunk and the $SR^{(4)}_{\{1-6,8,9,11,12,14,15\}}$ (marked by red) are chosen as neutral bytes for the backward chunk. Results from two chunks will match at $SR^{(1)}$ and $MC^{(1)}$ for a partial match.

Constraints on Initial Structure: To make the initial structure work, one needs to add 3 constraints on the neutral bytes for the forward chunk $MC^{(3)}_{\{0-3\}}$ to avoid the impacts on the backward chunk. The bytes $SR^{(3)}_{\{1,2,3\}}$ can be predetermined constant values as follows:

$$\begin{bmatrix} \mathtt{c}_0 = 9 \cdot MC^{(3)}_{\{0\}} \oplus e \cdot MC^{(3)}_{\{1\}} \oplus b \cdot MC^{(3)}_{\{2\}} \oplus d \cdot MC^{(3)}_{\{3\}} \\ \mathtt{c}_1 = d \cdot MC^{(3)}_{\{0\}} \oplus 9 \cdot MC^{(3)}_{\{1\}} \oplus e \cdot MC^{(3)}_{\{2\}} \oplus b \cdot MC^{(3)}_{\{3\}} \\ \mathtt{c}_2 = b \cdot MC^{(3)}_{\{0\}} \oplus d \cdot MC^{(3)}_{\{1\}} \oplus 9 \cdot MC^{(3)}_{\{2\}} \oplus e \cdot MC^{(3)}_{\{3\}} \end{bmatrix}. \tag{5}$$

There are 2^8 values of $MC^{(3)}_{\{0-3\}}$ when the constants $\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2$ are determined. Similarly, adding 8 constraints on the neutral bytes for the backward chunk to avoid the impacts on 8 bytes $MC^{(4)}_{\{0,2,5,7,8,10,13,15\}}$.

Matching through MC: According to the property of the MC operation, the match is tested column by column. There are totally five bytes known in each column of $SR^{(1)}$ and $MC^{(1)}$. So there is one byte matching for each column. Taking the match for first column as an example. The $SR^{(1)}_{\{0,2\}}$ are deduced in the forward computation and $MC^{(1)}_{\{1,2,3\}}$ are deduced in the backward computation. There is

$$\begin{aligned} &d \cdot SR^{(1)}_{\{0\}} \oplus e \cdot SR^{(1)}_{\{2\}} \\ =&d \cdot (b \cdot MC^{(1)}_{\{1\}} \oplus d \cdot MC^{(1)}_{\{2\}} \oplus 9 \cdot MC^{(1)}_{\{3\}}) \oplus e \cdot (9 \cdot MC^{(1)}_{\{1\}} \oplus e \cdot MC^{(1)}_{\{2\}} \oplus b \cdot MC^{(1)}_{\{3\}}). \end{aligned} \tag{6}$$

Forward and Backwork Computation: The forward computation list contains the blue neutral bits in $MC^{(3)}$ to $SR^{(1)}$. When accounting for the constraints, one can compute the neutral bytes in the forward chunk by traversing 2^8 possible values of $MC^{(3)}_{\{0-3\}}$. Then store $MC^{(3)}_{\{0-3\}}$ in table L_1 indexed by the value of $SR^{(1)}$ as the left part of Equ. (6) (i.e. $d \cdot SR^{(1)}_{\{0\}} \oplus e \cdot SR^{(1)}_{\{2\}}$). Similarly, the backward computation list contains the red neutral bits in $SR^{(4)}$ to $MC^{(1)}$. Store them in table L_2 indexed by the value of $MC^{(1)}$ as the right part of Equ. (6). Then one can use L_1 and L_2 for a 32-bit partial match on the indices.

B Details of Specifications on `Ascon` and `Xoodyak` Hash functions

B.1 `Ascon-Hash` and `Ascon-XOF`

The **`Ascon`** family [27] includes the hash functions **`Ascon-Hash`** and **`Ascon-Hasha`** as well as the extendable output functions **`Ascon-XOF`** and **`Ascon-XOFa`** with sponge-based modes of operations.

`Ascon Permutation.` The inner permutation applies 12 round functions to a 320-bit state. The state A is split into five 64-bit words, and denote $A^{(r)}_{\{x,y\}}$ to be the x-th (column) bit of the y-th (row) 64-bit word, where $0 \le y \le 4$, $0 \le x \le 63$. The round function consists of three operations p_C, p_S and p_L. Denote the internal states of round r as $A^{(r)} \xrightarrow{p_S \circ p_C} S^{(r)} \xrightarrow{p_L} A^{(r+1)}$.

– **Addition of Constants** p_C**:** $A^{(r)}_{\{*,2\}} = A^{(r)}_{\{*,2\}} \oplus RC_r$.
– **Substitution Layer** p_S**:** For each x, this step updates the columns $A^{(r)}_{\{x,*\}}$ using the 5-bit Sbox. Assume the S-box maps $(a_0, a_1, a_2, a_3, a_4) \in \mathbb{F}_2^5$ to $(b_0, b_1, b_2, b_3, b_4) \in \mathbb{F}_2^5$, where a_0 is the most significant bit. The algebraic normal form (ANF) of the Sbox is as follows:

$$\begin{cases} b_0 = a_4a_1 + a_3 + a_2a_1 + a_2 + a_1a_0 + a_1 + a_0 \\ b_1 = a_4 + a_3a_2 + a_3a_1 + a_3 + a_2a_1 + a_2 + a_1 + a_0 \\ b_2 = a_4a_3 + a_4 + a_2 + a_1 + 1 \\ b_3 = a_4a_0 + a_4 + a_3a_0 + a_3 + a_2 + a_1 + a_0 \\ b_4 = a_4a_1 + a_4 + a_3 + a_1a_0 + a_1 \end{cases} \tag{7}$$

The algebraic normal form (ANF) of the inverse Sbox is as follows:

$$\begin{cases} a_0 = b_4b_3b_2 + b_4b_3b_1 + b_4b_3b_0 + b_3b_2b_0 + b_3b_2 + b_3 + b_2 + b_1b_0 + b_1 + 1 \\ a_1 = b_4b_2b_0 + b_4 + b_3b_2 + b_2b_0 + b_1 + b_0 \\ a_2 = b_4b_3b_1 + b_4b_3 + b_4b_2b_1 + b_4b_2b_0 + b_4b_2 + b_4 + b_3b_2 + b_3b_1b_0 + b_3b_1 \\ \qquad + b_2b_1b_0 + b_2b_1 + b_2b_0 + b_2 + b_1 + b_0 + 1 \\ a_3 = b_4b_2b_1 + b_4b_2b_0 + b_4b_2 + b_4b_1 + b_4 + b_3 + b_2b_1 + b_2b_0 + b_1 \\ a_4 = b_4b_3b_2 + b_4b_2b_1 + b_4b_2b_0 + b_4b_2 + b_3b_2b_0 + b_3b_2 + b_3 + b_2b_1 + b_2b_0 + b_1b_0 \end{cases} \tag{8}$$

– **Linear Diffusion Layer** p_L**:**

$$\begin{aligned} A^{(r+1)}_{\{*,0\}} &\leftarrow S^{(r)}_{\{*,0\}} \oplus (S^{(r)}_{\{*,0\}} \ggg 19) \oplus (S^{(r)}_{\{*,0\}} \ggg 28), \\ A^{(r+1)}_{\{*,1\}} &\leftarrow S^{(r)}_{\{*,1\}} \oplus (S^{(r)}_{\{*,1\}} \ggg 61) \oplus (S^{(r)}_{\{*,1\}} \ggg 39), \\ A^{(r+1)}_{\{*,2\}} &\leftarrow S^{(r)}_{\{*,2\}} \oplus (S^{(r)}_{\{*,2\}} \ggg 1) \oplus (S^{(r)}_{\{*,2\}} \ggg 6), \\ A^{(r+1)}_{\{*,3\}} &\leftarrow S^{(r)}_{\{*,3\}} \oplus (S^{(r)}_{\{*,3\}} \ggg 10) \oplus (S^{(r)}_{\{*,3\}} \ggg 17), \\ A^{(r+1)}_{\{*,4\}} &\leftarrow S^{(r)}_{\{*,4\}} \oplus (S^{(r)}_{\{*,4\}} \ggg 7) \oplus (S^{(r)}_{\{*,4\}} \ggg 41). \end{aligned}$$

`Ascon-Hash` and `Ascon-XOF`. The state A is composed of the outer part with 64 bits $A_{\{*,0\}}$ and the inner part 256 bits $A_{\{*,i\}}$ ($i = 1, 2, 3, 4$). For `Ascon-Hash`, the output size is 256 bits, and the security claim is 2^{128}. For `Ascon-XOF`, the output can have arbitrary length and the security claim against preimage attack is $\min(2^{128}, 2^l)$, where l is the output length. In this paper, we target on `Ascon-XOF` with a 128-bit hash value and a 128-bit security claim against preimage attack.

B.2 `Xoodyak` and `Xoodoo` Permutation

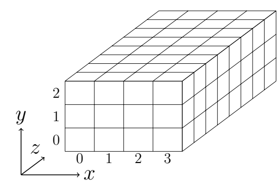


Fig. 17: Toy version of the `Xoodoo` state. The order in y is opposite to `Keccak`

Internally, `Xoodyak` makes use of the `Xoodoo` permutation [17], whose state (shown in Figure 17) bit denoted by $A^{(r)}_{\{x,y,z\}}$ is located at the x-th column, y-th row and z-th lane in the round r, where $0 \le x \le 3,\ 0 \le y \le 2,\ 0 \le z \le 31$. For `Xoodoo`, all the coordinates are considered modulo 4 for x, modulo 3 for y and modulo 32 for z. The permutation consists of the iteration of a round function $R = \rho_{\text{east}} \circ \chi \circ \iota \circ \rho_{\text{west}} \circ \theta$. The number of rounds is a parameter, which is 12 in `Xoodyak`. Denote the internal states of the round r as

$$A^{(r)} \xrightarrow{\theta} \theta^{(r)} \xrightarrow{\rho_{\text{west}}} \rho^{(r)} \xrightarrow{\iota} \iota^{(r)} \xrightarrow{\chi} \chi^{(r)} \xrightarrow{\rho_{\text{east}}} A^{(r+1)}.$$

$$\begin{aligned}
\theta:\ & \theta^{(r)}_{\{x,y,z\}} = A^{(r)}_{\{x,y,z\}} \oplus \sum\nolimits_{y'=0}^{2} (A^{(r)}_{\{x-1,y',z-5\}} \oplus A^{(r)}_{\{x-1,y',z-14\}}),\\
\rho_{\text{west}}:\ & \rho^{(r)}_{\{x,0,z\}} = \theta^{(r)}_{\{x,0,z\}},\ \rho^{(r)}_{\{x,1,z\}} = \theta^{(r)}_{\{x-1,1,z\}},\ \rho^{(r)}_{\{x,2,z\}} = \theta^{(r)}_{\{x,2,z-11\}},\\
\iota:\ & \iota^{(r)}_{\{0,0,z\}} = \rho^{(r)}_{\{0,0,z\}} \oplus RC_r,\ \text{where } RC_r \text{ is round-dependent constant},\\
\chi:\ & \chi^{(r)}_{\{x,y,z\}} = \iota^{(r)}_{\{x,y,z\}} \oplus (\iota^{(r)}_{\{x,y+1,z\}} \oplus 1) \cdot \iota^{(r)}_{\{x,y+2,z\}},\\
\rho_{\text{east}}:\ & A^{(r+1)}_{\{x,0,z\}} = \chi^{(r)}_{\{x,0,z\}},\ A^{(r+1)}_{\{x,1,z\}} = \chi^{(r)}_{\{x,1,z-1\}},\ A^{(r+1)}_{\{x,2,z\}} = \chi^{(r)}_{\{x-2,2,z-8\}}.
\end{aligned} \tag{9}$$

`Xoodyak` can serve as a `XOF`, i.e. `Xoodyak-XOF`, which offers arbitrary output length l. The preimage resistance is $\min(2^{128}, 2^l)$. We target on `Xoodyak-XOF` with output of 128-bit hash value and 128-bit absorbed message block.

C The Details of Attacks on `Ascon-Hash`

C.1 Details on the 4-round Preimage attack on `Ascon-Hash`

The attack is given in Algorithm 8.

Analysis of Algorithm 8. In Line 14 to 21, $2^{170+8+8+\zeta}$ states are tested against the 192-bit $T_2\|T_3\|T_4$, therefore, $\zeta = 6$ is enough to find a preimage. In Line 8, we choose fixed $c_{\mathcal{B}}$ to eliminate their influence on the computation of $c_{\mathcal{R}} \in \mathbb{F}_2^{170}$ and the 8-bit matching point (those values will determined by ${\color{red}\blacksquare}/{\color{gray}\blacksquare}$ as well as ${\color{blue}\boxed{\color{gray}\blacksquare}}/{\color{blue}\boxed{\color{red}\blacksquare}}$) for 2^{178} red bits in $A^{(0)}$ in Line 10.

- The Line 4 to 6, the time complexity is $2^{16} \times \frac{4}{4\times 64} = 2^{16} \times 2^{-6} = 2^{10}$ 4-round Ascon. The fraction $\frac{4}{4\times 64}$ means that we only need to compute the 4 Sboxes related to ${\color{blue}\blacksquare}$ bits in the first round, and 4-round Ascon has a total of 4×64 Sboxes.
- The Line 10 to 11, the time complexity is $2^{6+178} \times \frac{142}{4\times 64} = 2^{184} \times 2^{-0.85} = 2^{183.15}$ 4-round Ascon.
- The Line 14, the time complexity is $2^{6+170+8} \times \frac{1}{4\times 64} = 2^{184} \times 2^{-8} = 2^{176}$ 4-round Ascon.
- The Line 16, the time complexity is $2^{6+170+8} \times \frac{62}{4\times 64} = 2^{184} \times 2^{-2.05} = 2^{181.95}$ 4-round Ascon.
- The Line 20, the time complexity is $2^{6+170+8} = 2^{184}$ 4-round Ascon.

In **Phase II**, it is trivial to find an inner collision for the 256-bit capacity with the Floyd's cycle finding algorithm [31] with 2^{128} time and no memory.

Thereore, the total preimage attack on 4-round `Ascon-Hash` is $2^{10} + 2^{183.15} + 2^{176} + 2^{181.95} + 2^{184} + 2^{128} \approx 2^{184.85}$ time and $2^{16} + 2^{178} + 2^{14} + 2^{8} \approx 2^{178}$ memory.

C.2 Details on the 5-round Preimage attack on `Ascon-Hash`

The 5-round MitM characteristic shown in Figure 18. The starting state $A^{(0)}$ contains 1 ${\color{blue}\blacksquare}$ bits and 190 ${\color{red}\blacksquare}$ bits. The first row of $A^{(0)}$ is fixed as T_1 (marked as ${\color{gray}\blacksquare}$), and the remaining 65-bit ${\color{gray}\blacksquare}$ can be freely chosen. In the computation from $A^{(0)}$ to $A^{(4)}$, the consumed degree of freedoms (DoFs) of ${\color{red}\blacksquare}$ and DoFs of ${\color{blue}\blacksquare}$ are 187 and 0, respectively. Additional, there are 2 consumed DoFs of ${\color{red}\blacksquare}$ to make $a_0 + a_2 + a_4$ become ${\color{blue}\blacksquare}$ or ${\color{gray}\blacksquare}$ for matching points. Therefore, $d_{\mathcal{B}} = 1$, $d_{\mathcal{R}} = 1$, and there are 2 matching bits.

The 5-round attack is given in Algorithm 9.

Analysis of Algorithm 9. In Line 7 to 14, $2^{189+1+1+\zeta}$ states are tested against the 192-bit $T_2\|T_3\|T_4$, therefore, $\zeta = 1$ is enough to find a preimage.

- The Line 4, the time complexity is $2^{1+190} \times \frac{54+62+48+11}{5\times 64} = 2^{191} \times 2^{-0.87} = 2^{190.13}$ 5-round Ascon.
- The Line 7, the time complexity is $2^{1+189+1} \times \frac{1}{5\times 64} = 2^{191} \times 2^{-8.32} = 2^{182.68}$ 5-round Ascon.
- The Line 9, the time complexity is $2^{1+189+1} \times \frac{61}{5\times 64} = 2^{191} \times 2^{-2.39} = 2^{188.61}$ 5-round Ascon.
- The Line 13, the time complexity is $2^{1+189+0} = 2^{190}$ 5-round Ascon.

In **Phase II**, it is trivial to find an inner collision for the 256-bit capacity with the Floyd's cycle finding algorithm [31] with 2^{128} time and no memory.

Thereore, the total preimage attack on 4-round `Ascon-Hash` is $2^{190.13} + 2^{182.68} + 2^{188.61} + 2^{190} + 2^{128} \approx 2^{191.31}$ time and $2^{190} + 2^{1} + 2^{1} \approx 2^{190}$ memory.

Algorithm 8: Preimage Attack on 4-round `Ascon-Hash`: Phase I

Fix the first row of $A^{(0)}$ as T_1
for 2^{ζ} *values of the 32-bit free gray bits in* $A^{(0)}$ **do**
  /* Precomputation */
  for 2^{16} *values of the* $\blacksquare$ *bits* $v_{\mathcal{B}}$ *in* $A^{(0)}$ **do**
    Compute forward to determine the 8-bit $\blacksquare$ (denoted as $c_{\mathcal{B}} \in \mathbb{F}_2^8$) in $A^{(1)}$. E.g., in the $\blacksquare$ bit $A^{(1)}_{\{25,0\}} = S^{(0)}_{\{25,0\}} \oplus S^{(0)}_{\{6,0\}} \oplus S^{(0)}_{\{61,0\}}$, the $S^{(0)}_{\{61,0\}}$ should be gray to make the $A^{(1)}_{\{25,0\}}$ independent of blue bits, which consumes 1 DoF of $\blacksquare$. This is actually done by computing the 61-th Sbox, where there are 4 input $\blacksquare$ bits and output one $\blacksquare$ bit $S^{(0)}_{\{61,0\}}$ by consumming 1 DoF of $\blacksquare$. Then, $S^{(0)}_{\{61,0\}}$ is one bit of the 8-bit $c_{\mathcal{B}}$
    Store the 16-bit $\blacksquare$ values $v_{\mathcal{B}}$ of $A^{(0)}$ in $U[c_{\mathcal{B}}]$
  end
  Choose an index $c_{\mathcal{B}}$, e.g., $c_{\mathcal{B}} = 0$, there expected $2^{16-8} = 2^8$ elements in $U[0]$
  /* In the following, we always fix $c_{\mathcal{B}}$ as 0 */
  for 2^{178} *values of the 184* $\blacksquare$ *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
    Compute forward to determine the 170-bit $\blacksquare$/$\blacksquare$ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{170}$) and the 8-bit matching point. Build the table V and store the 184-bit $\blacksquare$ $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 8-bit matching point in $V[c_{\mathcal{R}}]$
  end
  for $c_{\mathcal{R}} \in \mathbb{F}_2^{170}$ **do**
    Retrieve the 2^8 elements of $V[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 8-bit matching point
    for 2^8 *values* $v_{\mathcal{B}}$ *in* $U[0]$ **do**
      Compute to the 8-bit matching point and store $\blacksquare$ $v_{\mathcal{B}}$ in L_2 indexed by the 8-bit matching point.
    end
    for *values matched between* L_1 *and* L_2 **do**
      if T_2 *is satisfied* **then**
        Check if $T_3 \| T_4$ is satisfied
      end
    end
  end
end

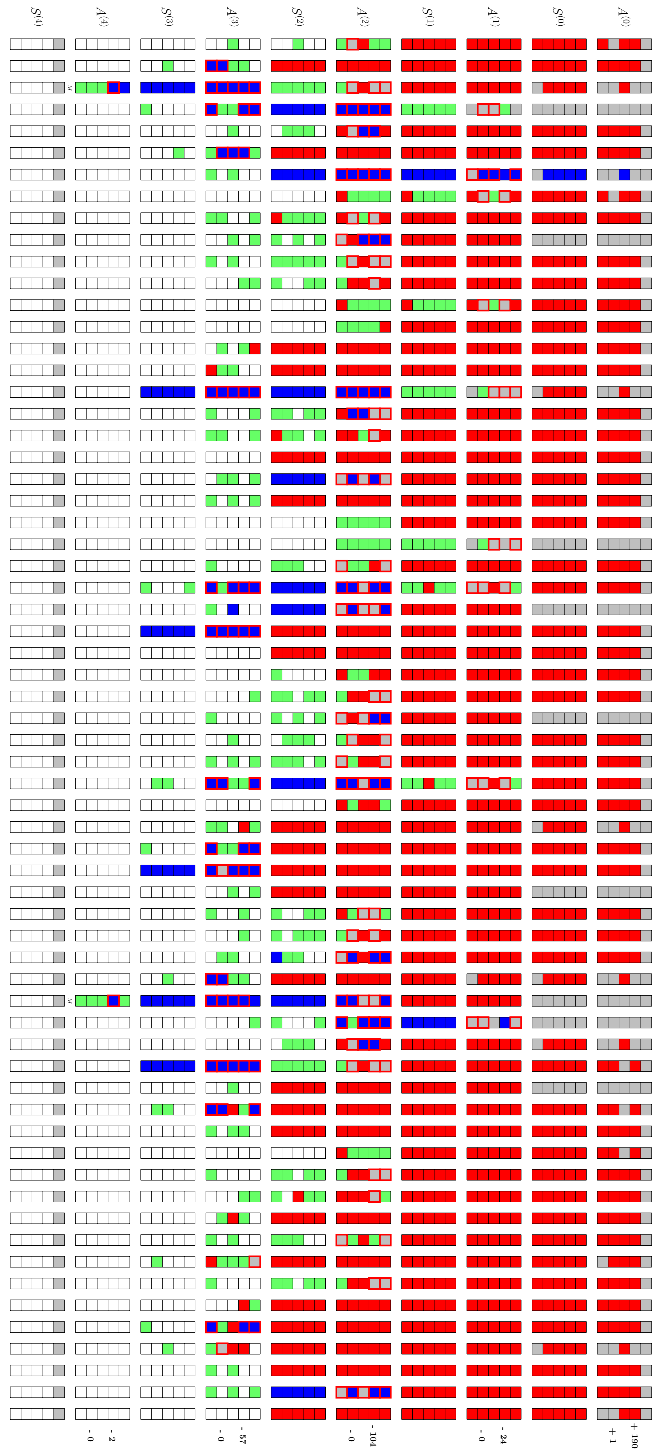


Fig. 18: The 5-round Preimage attack on ASCON-Hash

Algorithm 9: Preimage Attack on 5-round `Ascon-Hash`: Phase I

```
Fix the first row of A^(0) as T_1
for 2^ζ values of the 65-bit free gray bits in A^(0) do
    for 2^190 values of the ■ bits v_R in A^(0) do
        Fix ■ as 0, compute forward to determine the 189-bit ■/■ (denoted as
          c_R ∈ F_2^189), and the 2-bit matching point. Build the table V and store
          the 190-bit ■ v_R of A^(0) as well as the 2-bit matching point in V[c_R]
    end
    for c_R ∈ F_2^189 do
        Retrieve the 2^1 elements of V[c_R] and restore v_R in L_1 under the
          index of 2-bit matching point
        for 2^1 values of ■ v_B do
            Compute to the 2-bit matching point and store ■ v_B in L_2 indexed
              by the 2-bit matching point.
        end
        for values matched between L_1 and L_2 do
            if T_2 is satisfied then
                Check if T_3||T_4 is satisfied
            end
        end
    end
end
```

C.3 Conditions for 3-/4-round collision attacks on `Ascon-Hash`

The bit conditions for 3-/4-round collision attacks on `Ascon-Hash` are given in Table 2 and Table 3. Table 4 shows ten messages that produce the 24-bit 0 in $S^{(2)}$ for the partial experiment on 3-round collision attack on `Ascon-Hash`.

$$
\begin{aligned}
&A^{(0)}_{\{4,1\}} = 1, A^{(0)}_{\{5,1\}} = 0, A^{(0)}_{\{6,1\}} = 1, A^{(0)}_{\{7,3\}} + A^{(0)}_{\{7,4\}} = 1, A^{(0)}_{\{7,1\}} = 0, A^{(0)}_{\{10,1\}} = 1,\\
&A^{(0)}_{\{12,1\}} = 0, A^{(0)}_{\{14,3\}} + A^{(0)}_{\{14,4\}} = 1, A^{(0)}_{\{19,1\}} = 1, A^{(0)}_{\{21,3\}} + A^{(0)}_{\{21,4\}} = 1,\\
&A^{(0)}_{\{21,1\}} = 0, A^{(0)}_{\{23,1\}} = 0, A^{(0)}_{\{26,1\}} = 1, A^{(0)}_{\{27,1\}} = 0, A^{(0)}_{\{30,1\}} = 0, A^{(0)}_{\{31,1\}} = 0,\\
&A^{(0)}_{\{36,1\}} = 1, A^{(0)}_{\{37,1\}} = 0, A^{(0)}_{\{38,1\}} = 1, A^{(0)}_{\{39,3\}} + A^{(0)}_{\{39,4\}} = 1, A^{(0)}_{\{39,1\}} = 0, A^{(0)}_{\{42,1\}} = 1,\\
&A^{(0)}_{\{44,1\}} = 0, A^{(0)}_{\{46,3\}} + A^{(0)}_{\{46,4\}} = 1, A^{(0)}_{\{51,1\}} = 1, A^{(0)}_{\{53,3\}} + A^{(0)}_{\{53,4\}} = 1,\\
&A^{(0)}_{\{53,1\}} = 0, A^{(0)}_{\{55,1\}} = 0, A^{(0)}_{\{58,1\}} = 1, A^{(0)}_{\{59,1\}} = 0, A^{(0)}_{\{62,1\}} = 0, A^{(0)}_{\{63,1\}} = 0,
\end{aligned}
$$

Table 2: Bit Conditions in 3-round Collision Attack on `Ascon-Hash`

$A^{(0)}_{\{2,3\}} + A^{(0)}_{\{2,4\}} = 1, A^{(0)}_{\{3,1\}} = 1, A^{(0)}_{\{3,3\}} + A^{(0)}_{\{3,4\}} = 1, A^{(0)}_{\{5,3\}} + A^{(0)}_{\{5,4\}} = 1,$
$A^{(0)}_{\{7,1\}} = 1, A^{(0)}_{\{9,1\}} = 1, A^{(0)}_{\{10,3\}} + A^{(0)}_{\{10,4\}} = 1, A^{(0)}_{\{11,3\}} + A^{(0)}_{\{11,4\}} = 1, A^{(0)}_{\{12,1\}} = 1,$
$A^{(0)}_{\{12,3\}} + A^{(0)}_{\{12,4\}} = 1, A^{(0)}_{\{13,1\}} = 0, A^{(0)}_{\{15,1\}} = 1, A^{(0)}_{\{16,1\}} = 1, A^{(0)}_{\{18,3\}} + A^{(0)}_{\{18,4\}} = 1,$
$A^{(0)}_{\{19,1\}} = 0, A^{(0)}_{\{20,3\}} + A^{(0)}_{\{20,4\}} = 1, A^{(0)}_{\{21,1\}} = 0, A^{(0)}_{\{22,1\}} = 1, A^{(0)}_{\{22,3\}} + A^{(0)}_{\{22,4\}} = 1,$
$A^{(0)}_{\{23,1\}} = 0, A^{(0)}_{\{25,1\}} = 1, A^{(0)}_{\{25,3\}} + A^{(0)}_{\{25,4\}} = 1, A^{(0)}_{\{26,1\}} = 1, , A^{(0)}_{\{27,3\}} + A^{(0)}_{\{27,4\}} = 1,$
$A^{(0)}_{\{28,3\}} + A^{(0)}_{\{28,4\}} = 1, A^{(0)}_{\{28,1\}} = 1,$
$A^{(0)}_{\{34,3\}} + A^{(0)}_{\{34,4\}} = 1, A^{(0)}_{\{35,1\}} = 1, A^{(0)}_{\{35,3\}} + A^{(0)}_{\{35,4\}} = 1, A^{(0)}_{\{37,3\}} + A^{(0)}_{\{37,4\}} = 1,$
$A^{(0)}_{\{39,1\}} = 1, A^{(0)}_{\{41,1\}} = 1, A^{(0)}_{\{42,3\}} + A^{(0)}_{\{42,4\}} = 1, A^{(0)}_{\{43,3\}} + A^{(0)}_{\{43,4\}} = 1, A^{(0)}_{\{44,1\}} = 1,$
$A^{(0)}_{\{44,3\}} + A^{(0)}_{\{44,4\}} = 1, A^{(0)}_{\{45,1\}} = 0, A^{(0)}_{\{47,1\}} = 1, A^{(0)}_{\{48,1\}} = 1, A^{(0)}_{\{50,3\}} + A^{(0)}_{\{50,4\}} = 1,$
$A^{(0)}_{\{51,1\}} = 0, A^{(0)}_{\{52,3\}} + A^{(0)}_{\{52,4\}} = 1, A^{(0)}_{\{53,1\}} = 0, A^{(0)}_{\{54,1\}} = 1, A^{(0)}_{\{54,3\}} + A^{(0)}_{\{54,4\}} = 1,$
$A^{(0)}_{\{55,1\}} = 0, A^{(0)}_{\{57,1\}} = 1, A^{(0)}_{\{57,3\}} + A^{(0)}_{\{57,4\}} = 1, A^{(0)}_{\{58,1\}} = 1, , A^{(0)}_{\{59,3\}} + A^{(0)}_{\{59,4\}} = 1,$
$A^{(0)}_{\{60,3\}} + A^{(0)}_{\{60,4\}} = 1, A^{(0)}_{\{60,1\}} = 1,$

Table 3: Bit Conditions in 4-round Collision Attack on `Ascon-Hash`

Round	Message (first row)	Message (last four rows)
3	0002005b173f21cd	0a2010200a201020 0000000000000000 0102040001020400 0000000000000000
	0010005f570f34ed	
	0010015bd71f2ccd	
	0020045b173d25cd	
	0020115b173f2dc9	
	0110011feb353c42	
	0030005f572f34ed	
	0030045bd70f24e9	
	0100001bea0f2ae3	
	0100111bab253966	

Table 4: Preimages of 3-round `Ascon-Hash` in collision attack

D The Details of Attacks on 3-round `Xoodyak-XOF` and `Xoodyak-Hash`

D.1 Details of the 3-round Collision Attack on `Xoodyak-Hash`

A new 3-round MitM characteristic in Figure 19 is found. With the MitM characteristic, we can build collision attack on `Xoodyak-Hash`, which is given in Algorithm 10. The starting state $A^{(0)}$ contains 8 ■ bits and 118 ■ bits. There are totally 51 conditions on ■ bits of $\iota^{(0)}$, which are listed in Table 5. In the computation from $A^{(0)}$ to $\iota^{(2)}$, the consumed DoFs of ■ is 110 and the consumed DoFs of ■ is 0. Therefore, $d_{\mathcal{B}} = 8$, $d_{\mathcal{R}} = 118 - 110 = 8$. We get $m = t = 8$ matching equations with the deterministic relations of $\iota^{(2)}$.

In one MitM episode in Line 10 to 15, $2^{8+8-8} = 2^8$ partial target preimages are expected to obtain. We need $2^{(n-t)/2-8} = 2^{116}$ MitM episodes to build the collision attack, *i.e.*, $2^{\zeta-51+110} = 2^{116}$, i.e., $\zeta = 57$. Each step of Algorithm 10 is analyzed below:

- In Line 3, the time complexity is $2^{57} \times 51^3 = 2^{74.02}$ bit operations and 2^{57} 3-round `Xoodyak`.
- In Line 7, the time complexity is $2^{57-51+118} \times \frac{128+128+4}{128\times 3} = 2^{123.44}$ 3-round `Xoodyak`. The fraction $\frac{128+128+4}{128\times 3}$ is because that in the last round only 4 Sboxes with matching point are computed, while there are totally 128×3 Sboxes applications in the 3-round `Xoodyak`.
- In Line 10, the time is $2^{57-51+110+8} \times \frac{1}{384} = 2^{115.42}$ 3-round `Xoodyak`. This step is just to retrieve the values $U[c_{\mathcal{R}}]$ and restore it in L_1. Assuming one table access is about one Sbox application, we get the fraction $\frac{1}{384}$.
- In Line 12, the time is $2^{57-51+110+8} \times \frac{128+128+4}{128\times 3} = 2^{123.44}$ 3-round `Xoodyak`.
- In Line 15, the time complexity is $2^{57-51+110+16-8} = 2^{124}$ 3-round `Xoodyak`.

The total complexity of the 3-round attack is $2^{74.02} + 2^{57} + 2^{123.44} + 2^{115.42} + 2^{123.44} + 2^{124} = 2^{125.23}$ 3-round `Xoodyak-Hash`, and the memory to store U and L is $2^{118} + 2^{124} = 2^{124.02}$.

$$
\begin{aligned}
&\iota^{(0)}_{\{1,0,0\}} = 0, \iota^{(0)}_{\{1,1,0\}} = 1, \iota^{(0)}_{\{2,0,0\}} = 1, \iota^{(0)}_{\{2,2,0\}} = 0, \iota^{(0)}_{\{2,0,2\}} = 0, \iota^{(0)}_{\{2,1,2\}} = 1, \iota^{(0)}_{\{3,0,2\}} = 0,\\
&\iota^{(0)}_{\{3,1,2\}} = 1, \iota^{(0)}_{\{2,0,3\}} = 1, \iota^{(0)}_{\{2,2,3\}} = 0, \iota^{(0)}_{\{0,1,4\}} = 0, \iota^{(0)}_{\{0,2,4\}} = 1, \iota^{(0)}_{\{2,2,4\}} = 0, \iota^{(0)}_{\{0,0,7\}} = 0,\\
&\iota^{(0)}_{\{0,1,7\}} = 1, \iota^{(0)}_{\{3,0,7\}} = 0, \iota^{(0)}_{\{3,1,7\}} = 1, \iota^{(0)}_{\{1,2,8\}} = 0, \iota^{(0)}_{\{0,1,9\}} = 0, \iota^{(0)}_{\{1,0,9\}} = 0, \iota^{(0)}_{\{2,0,9\}} = 1,\\
&\iota^{(0)}_{\{2,2,9\}} = 0, \iota^{(0)}_{\{0,0,16\}} = 0, \iota^{(0)}_{\{0,1,16\}} = 1, \iota^{(0)}_{\{1,0,16\}} = 0, \iota^{(0)}_{\{1,1,16\}} = 1, \iota^{(0)}_{\{2,0,16\}} = 0, \iota^{(0)}_{\{2,1,16\}} = 1,\\
&\iota^{(0)}_{\{3,0,16\}} = 0, \iota^{(0)}_{\{3,1,16\}} = 1, \iota^{(0)}_{\{1,0,17\}} = 0, \iota^{(0)}_{\{1,1,17\}} = 1, \iota^{(0)}_{\{1,2,18\}} = 1, \iota^{(0)}_{\{2,2,18\}} = 0, \iota^{(0)}_{\{3,1,21\}} = 1,\\
&\iota^{(0)}_{\{2,0,23\}} = 0, \iota^{(0)}_{\{0,0,25\}} = 0, \iota^{(0)}_{\{0,1,25\}} = 1, \iota^{(0)}_{\{2,0,25\}} = 0, \iota^{(0)}_{\{2,1,25\}} = 1, \iota^{(0)}_{\{3,0,25\}} = 0, \iota^{(0)}_{\{3,1,25\}} = 1,\\
&\iota^{(0)}_{\{2,1,26\}} = 1, \iota^{(0)}_{\{0,2,27\}} = 1, \iota^{(0)}_{\{1,0,27\}} = 0, \iota^{(0)}_{\{2,0,27\}} = 1, \iota^{(0)}_{\{2,2,27\}} = 0, \iota^{(0)}_{\{2,0,28\}} = 1, \iota^{(0)}_{\{0,1,30\}} = 1,\\
&\iota^{(0)}_{\{3,0,30\}} = 0, \iota^{(0)}_{\{3,1,30\}} = 1,
\end{aligned}
$$

Table 5: Bit Conditions in 3-round Collision Attack on `Xoodyak-Hash`

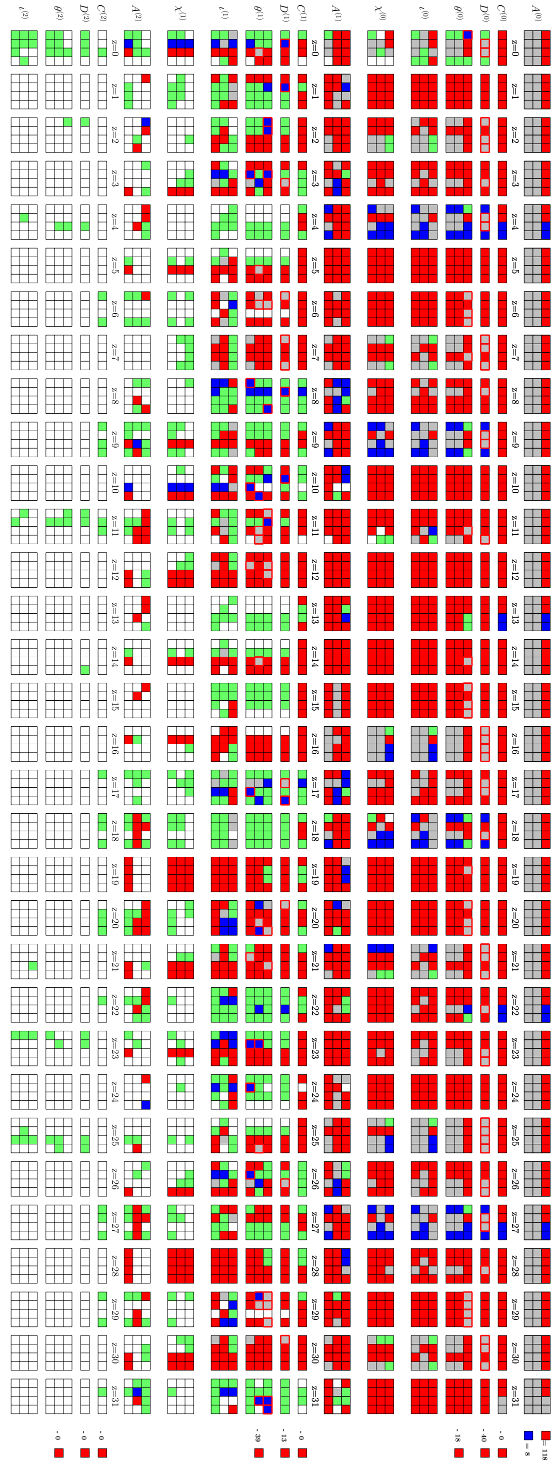


Fig. 19: The MitM collision attack on 3-round Xoodyak-Hash

Algorithm 10: Collision Attack on 3-round `Xoodyak-Hash`

Fixed $t = 8$ bits of $\chi^{(2)}$ as zero.
for 2^{ζ} *values of* M_1 **do**
  Compute the inner part of the 2nd block and solve the system of 51 linear equations
  if *the equations have solutions* `/* with probability of` 2^{-51} `*/`
  then
    for 2^{118} *values of the* *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
      Compute forward to determine 110-bit / bits (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{110}$), and the 8-bit matching point. Build the table U and store the 118-bit $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 8-bit matching point in $U[c_{\mathcal{R}}]$.
    end
    for $c_{\mathcal{R}} \in \mathbb{F}_2^{110}$ **do**
      Retrieve the 2^8 elements of $V[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 8-bit matching point
      for 2^8 *values of* $v_{\mathcal{B}}$ **do**
        Compute forward to the 8 matching point and store $v_{\mathcal{B}}$ in L_2 indexed by the 8 matching point.
      end
      for *values matched between* L_1 *and* L_2 **do**
        Compute the 256-bit capacity c from the matched and cells and store the (M_1, M_2, c) in L indexed by c
        if *the size of* L *is* $2^{(n-t)/2} = 2^{124}$ **then**
          Check L and return (M_1, M_2) and (M_1', M_2') with the same c
        end
      end
    end
  end
end

D.2 New MITM preimage attack on 3-round Xoodyak-XOF

The 3-round MITM characteristic is shown in Figure 20. The starting state $A^{(0)}$ contains 8 ■ bits and 118 ■ bits. There are totally 53 conditions on ■ bits of $\iota^{(0)}$, which are listed in Table 6. In the computation from $A^{(0)}$ to $\iota^{(2)}$, the consumed DoFs of ■ is 111 and the consumed DoFs of ■ is 0. Therefore, $d_{\mathcal{B}} = 8$, $d_{\mathcal{R}} = 118 - 111 = 7$. We get $m = 7$ matching equations as Equ. (10) with the deterministic relations of $\iota^{(2)}$.

$$\begin{aligned}
&\chi^{(2)}_{\{1,2,8\}} = \iota^{(2)}_{\{1,2,8\}} \oplus (\iota^{(2)}_{\{1,0,8\}} \oplus 1) \cdot \iota^{(2)}_{\{1,1,8\}}, \chi^{(2)}_{\{1,2,10\}} = \iota^{(2)}_{\{1,2,10\}} \oplus (\iota^{(2)}_{\{1,0,10\}} \oplus 1) \cdot \iota^{(2)}_{\{1,1,10\}},\\
&\chi^{(2)}_{\{2,2,10\}} = \iota^{(2)}_{\{2,2,10\}} \oplus (\iota^{(2)}_{\{2,0,10\}} \oplus 1) \cdot \iota^{(2)}_{\{2,1,10\}}, \chi^{(2)}_{\{1,2,17\}} = \iota^{(2)}_{\{1,2,17\}} \oplus (\iota^{(2)}_{\{1,0,17\}} \oplus 1) \cdot \iota^{(2)}_{\{1,1,17\}},\\
&\chi^{(2)}_{\{1,2,26\}} = \iota^{(2)}_{\{1,2,26\}} \oplus (\iota^{(2)}_{\{1,0,26\}} \oplus 1) \cdot \iota^{(2)}_{\{1,1,26\}}, \chi^{(2)}_{\{1,2,31\}} = \iota^{(2)}_{\{1,2,31\}} \oplus (\iota^{(2)}_{\{1,0,31\}} \oplus 1) \cdot \iota^{(2)}_{\{1,1,31\}},\\
&\chi^{(2)}_{\{2,2,31\}} = \iota^{(2)}_{\{2,2,31\}} \oplus (\iota^{(2)}_{\{2,0,31\}} \oplus 1) \cdot \iota^{(2)}_{\{2,1,31\}}.
\end{aligned} \tag{10}$$

We give the attack procedure in Algorithm 11. In the MitM episode in Line 9 to 19, a space of $2^{7+8} = 2^{15}$ is searched. In order to search a 128-bit preimage, we have to search a space of $2^{\zeta-53+111+15} = 2^{128}$, i.e., $\zeta = 55$. Each step of Algorithm 11 is analyzed below:

- In Line 2, the time complexity is $2^{55} \times 53^3 = 2^{72.2}$ bit operations and 2^{55} 3-round Xoodyak.
- In Line 6, the time complexity is $2^{55-53+118} \times \frac{128+128+7}{128\times 3} = 2^{119.45}$ 3-round Xoodyak. The fraction $\frac{128+128+7}{128\times 3}$ is because that in the last round only 7 Sboxes with matching point are computed, while there are totally 128×3 Sboxes applications in the 3-round Xoodyak.
- In Line 9, the time is $2^{55-53+111+7} \times \frac{1}{384} = 2^{111.41}$ 3-round Xoodyak. This step is just to retrieve the values $U[c_{\mathcal{R}}]$ and restore it in L_1. Assuming one table access is about one Sbox application, we get the fraction $\frac{1}{384}$.
- In Line 11, the time is $2^{55-53+111+8} \times \frac{128+128+7}{128\times 3} = 2^{120.45}$ 3-round Xoodyak.
- In Line 14, the time complexity is $2^{55-53+111+15-7} \times \frac{2}{3} = 2^{120.41}$ 3-round Xoodyak.
- In Line 15, we only compute 5 Sboxes with $\rho^{(2)}$ to gain a filter of 2^{-5}, whose time complexity is $2^{55-53+111+15-7} \times \frac{5}{128\times 3} = 2^{114.73}$ 3-round Xoodyak.
- In Line 18, we check the remaining states with the remaining $128-7-5 = 116$ Sboxes, which is $2^{55-53+111+15-7-5} \times \frac{116}{384} = 2^{114.27}$ 3-round Xoodyak.

The total complexity of the 3-round attack is $2^{72.2} + 2^{55} + 2^{119.45} + 2^{111.41} + 2^{120.45} + 2^{120.41} + 2^{114.73} + 2^{114.27} = 2^{121.77}$ 3-round Xoodyak-XOF, and the memory to store U is 2^{118}.

E Attacks on PHOTON and PHOTON-Beetle-Hash

E.1 Preimage Attacks on round-reduced PHOTON

All hash variants of PHOTON are given in Table 7.

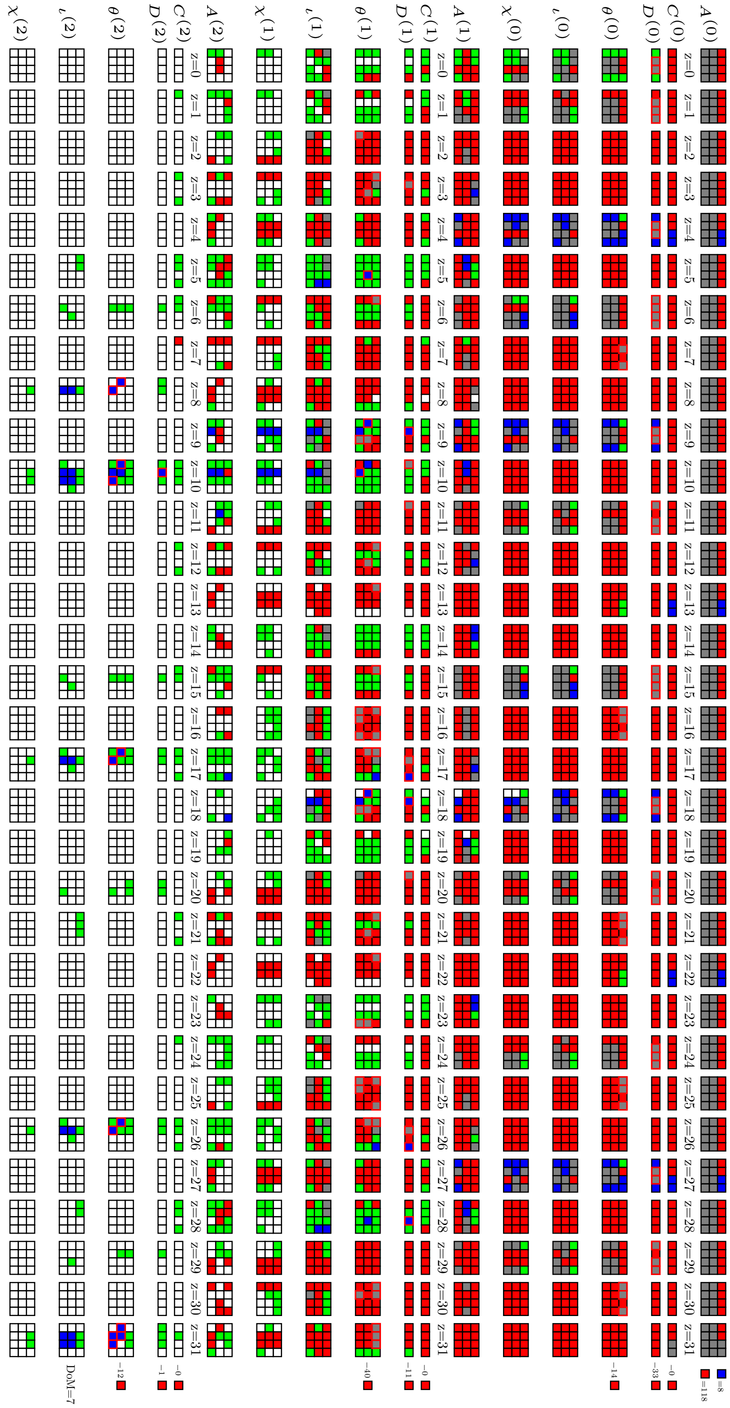


Fig. 20: The MitM preimage attack on 3-round Xoodyak-XOF

$\iota^{(0)}_{\{1,0,0\}} = 1; \iota^{(0)}_{\{3,1,0\}} = 0; \iota^{(0)}_{\{3,2,0\}} = 1; \iota^{(0)}_{\{2,0,1\}} = 0; \iota^{(0)}_{\{2,1,1\}} = 1; \iota^{(0)}_{\{3,0,1\}} = 0;$
$\iota^{(0)}_{\{3,1,1\}} = 1; \iota^{(0)}_{\{1,0,4\}} = 1; \iota^{(0)}_{\{1,2,4\}} = 0; \iota^{(0)}_{\{3,2,4\}} = 1; \iota^{(0)}_{\{3,1,4\}} = 0; \iota^{(0)}_{\{0,1,6\}} = 1; \iota^{(0)}_{\{2,0,6\}} = 0;$
$\iota^{(0)}_{\{2,1,6\}} = 1; \iota^{(0)}_{\{3,0,6\}} = 0; \iota^{(0)}_{\{3,1,6\}} = 1; \iota^{(0)}_{\{1,0,9\}} = 1; \iota^{(0)}_{\{1,2,9\}} = 0; \iota^{(0)}_{\{3,2,9\}} = 1; \iota^{(0)}_{\{3,1,9\}} = 0;$
$\iota^{(0)}_{\{0,0,11\}} = 0; \iota^{(0)}_{\{0,1,11\}} = 1; \iota^{(0)}_{\{3,0,11\}} = 0; \iota^{(0)}_{\{3,1,11\}} = 1; \iota^{(0)}_{\{0,0,15\}} = 0; \iota^{(0)}_{\{0,1,15\}} = 1;$
$\iota^{(0)}_{\{1,0,15\}} = 0; \iota^{(0)}_{\{1,1,15\}} = 1; \iota^{(0)}_{\{2,0,15\}} = 0; \iota^{(0)}_{\{2,1,15\}} = 1; \iota^{(0)}_{\{3,0,15\}} = 0; \iota^{(0)}_{\{3,1,15\}} = 1; \iota^{(0)}_{\{1,0,18\}} = 1;$
$\iota^{(0)}_{\{2,0,18\}} = 0; \iota^{(0)}_{\{3,1,18\}} = 0; \iota^{(0)}_{\{3,2,18\}} = 1; \iota^{(0)}_{\{0,0,20\}} = 0; \iota^{(0)}_{\{0,1,20\}} = 1; \iota^{(0)}_{\{3,0,20\}} = 0; \iota^{(0)}_{\{3,1,20\}} = 1;$
$\iota^{(0)}_{\{2,0,24\}} = 0; \iota^{(0)}_{\{2,1,24\}} = 1; \iota^{(0)}_{\{3,0,24\}} = 0; \iota^{(0)}_{\{3,1,24\}} = 1; \iota^{(0)}_{\{1,0,27\}} = 1; \iota^{(0)}_{\{1,2,27\}} = 0;$
$\iota^{(0)}_{\{2,0,27\}} = 0; \iota^{(0)}_{\{3,1,27\}} = 0; \iota^{(0)}_{\{3,2,27\}} = 1; \iota^{(0)}_{\{0,0,29\}} = 0; \iota^{(0)}_{\{0,1,29\}} = 1; \iota^{(0)}_{\{3,0,29\}} = 0; \iota^{(0)}_{\{3,1,29\}} = 1$

Table 6: Bit Conditions in 3-round Attack on `Xoodyak-XOF`

Algorithm 11: New Preimage Attack on 3-round `Xoodyak-XOF`

for 2^{ζ} *values of* M_1 **do**
    Compute the inner part of the 2nd block and solve the system of 53 linear equations
    if *the equations have solutions* `/* with probability of` 2^{-53} `*/` **then**
        for 2^{118} *values of the* 🟥 *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
            Fix 🟦 as 0, compute forward to determine 111-bit 🔲/🟦 bits (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{111}$), and the 7-bit matching point in Equ. (10), *i.e.*, compute 7 bits $f'_{\mathcal{M}} = f_{\mathcal{R}} \oplus f_{\mathcal{G}}$. Build the table U and store the 118-bit 🟥 bits $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the 7-bit matching point in $U[c_{\mathcal{R}}]$.
        end
        for $c_{\mathcal{R}} \in \mathbb{F}_2^{111}$ **do**
            Retrieve the 2^7 elements of $V[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 7-bit matching point
            for 2^8 *values of* 🟦 $v_{\mathcal{B}}$ **do**
                Compute to the 7 matching points and store 🟦 $v_{\mathcal{B}}$ in L_2 indexed by 7 matching points.
            end
            for *values matched between* L_1 *and* L_2 **do**
                Compute $\iota^{(2)}$ from the matched 🟥 and 🟦 cells
                if $\iota^{(2)}$ *satisfy the first 5 Sbox* `/* Probability of` 2^{-5}`. This step is to avoid computing all Sboxes of` $\iota^{(2)}$ `and only use partial Sboxes to filter first. */` **then**
                    Check $\iota^{(2)}$ against the remaining 116 Sboxes
                    if *it leads to the given hash value* **then**
                        Output the preimage
                    end
                end
            end
        end
    end
end

Table 7: Five different flavors of PHOTON

PHOTON-$n/r/r'$	Permutation	State	d	s	N_r	Preimage
PHOTON-80/20/16	P_{100}	100	5	4	12	64
PHOTON-128/16/16	P_{144}	144	6	4	12	112
PHOTON-160/36/36	P_{196}	196	7	4	12	124
PHOTON-224/32/32	P_{256}	256	8	4	12	192
PHOTON-256/32/32	P_{288}	288	6	8	12	224

Algorithm 12: MitM of Phase I in the Attack on PHOTON-160/36/36

Set the value 9 $\blacksquare$ nibbles $\mathtt{SR}^{(0)}[0,7,14,21,28,35,42,1,8]$ as $\mathtt{SC}\circ\mathtt{AC}(T_1)$
for 2^{80} *values of 13* $\blacksquare$ *cells in* $\mathtt{SR}^{(0)}$, *6-cell* $\zeta = \mathtt{MC}^{(0)}[0,4,16,20,29,32]$, *and 10-cell* $\xi = \mathtt{MC}^{(0)}[22,23,25,26,27,35,36,38,39,41]$ **do**
    Solve linear system on the 15 $\blacksquare$ cells of $\mathtt{SR}^{(0)}$ with ζ to derive the $2^{(15-6)\times 4=36}$ solutions $v_{\mathcal{B}}$
    for *each of* 2^{36} *solutions* $v_{\mathcal{B}}$ **do**
        Compute forward to $\mathtt{SR}^{(2)}$ and let the $\blacksquare$ in $\mathtt{SR}^{(2)}$ be 0
        $\varphi \leftarrow \mathtt{MC}\left(\mathtt{SR}^{(2)}\right)[6,7,15,23,31,39,47]$
        $U[\varphi] \leftarrow v_{\mathcal{B}}$ /* There are 2^8 elements on average */
    end
    Solve linear system on the 12 $\blacksquare$ cells of $\mathtt{SR}^{(0)}$ with ξ to derive the $2^{(12-10)\times 4=8}$ solutions $v_{\mathcal{R}}$
    for 2^{28} *values of* φ **do**
        for 2^8 *values of* $v_{\mathcal{R}}$ **do**
            Compute forward to $\mathtt{SR}^{(4)}[1,7]$
            if $\mathtt{SR}^{(4)}[1,7] = \mathtt{A}^{(5)}[1,7]$ **then**
                for $v_{\mathcal{B}} \in U[\varphi]$ **do**
                    Reconstruct the state X by $v_{\mathcal{B}}$ and $v_{\mathcal{R}}$
                    if X *satisfies the full* $160-36=124$ *bits target* **then**
                        Output X and stop
                    end
                end
            end
        end
    end
end

The 4.5-round MitM Attack on PHOTON-160/36/36 is given in Algorithm 12.

We give the 4.5-round attacks on all other versions of PHOTON-$n/r/r'$ as shown in Figure 25, 26, 27, 28, the attacks are very similar to the attack on PHOTON-160/36/36 in Section 9.1 and the complexities are summarized in Table 1.

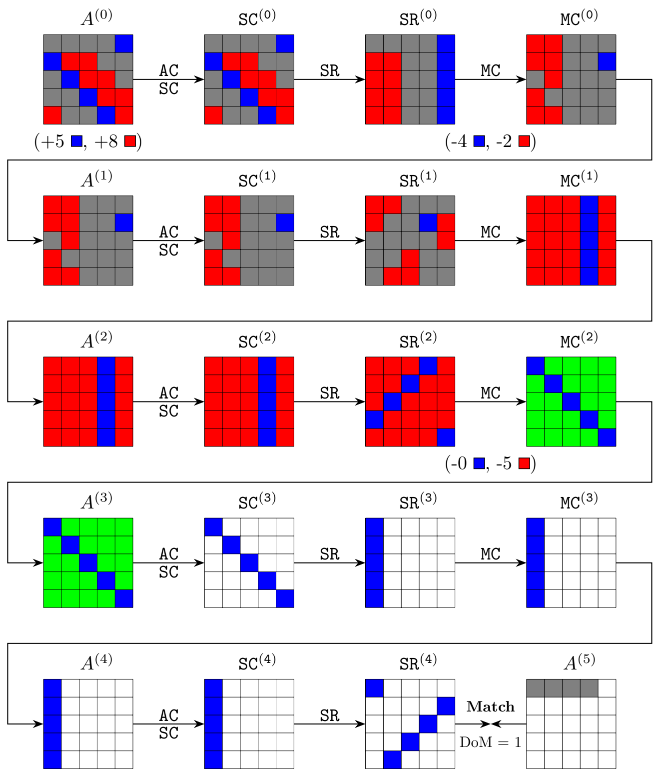


Fig. 21: The 4.5-round MitM attack on PHOTON-80/20/16

E.2 Preimage Attack on 3-round PHOTON-Beetle-Hash

The family of PHOTON-Beetle [3] designed by Bao et al. is one of the finalists of NIST LWC project. The AES-like P_{256} permutation is applied. In the PHOTON-Beetle family, the authors define a hash function PHOTON-Beetle-Hash. The message with arbitrary length is divided into one 128-bit block (M_1) and several $r = 32$-bit blocks. The 256-bit digest $T = T_1 \| T_2$ is squeezed with two $T_1 = T_2 = 128$ bits, i.e., $r' = 128$. The security level against preimage attack claimed by the designers is 128-bit. We use our new attack framework to find a 3.5-round (omitting the last MC) preimage attack on PHOTON-Beetle-Hash:

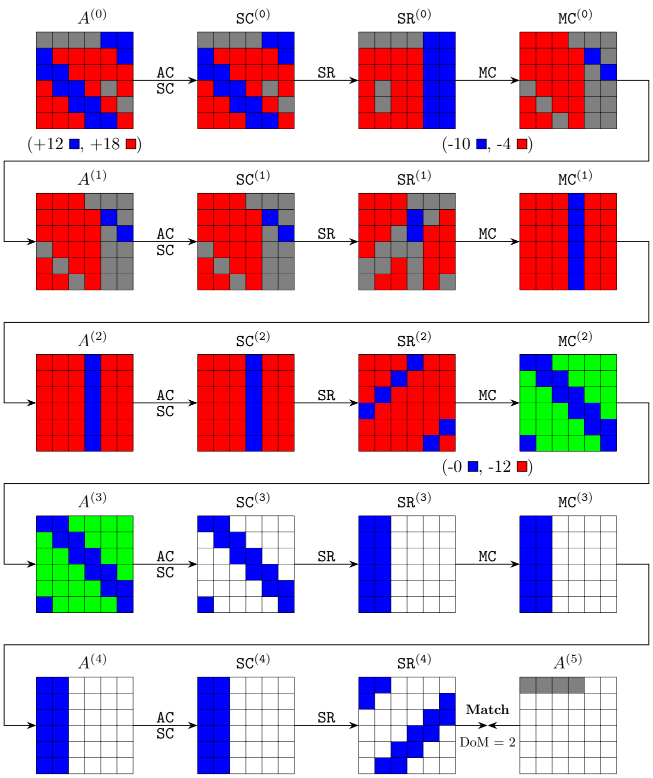


Fig. 22: The 4.5-round MitM attack on PHOTON-128/16/16

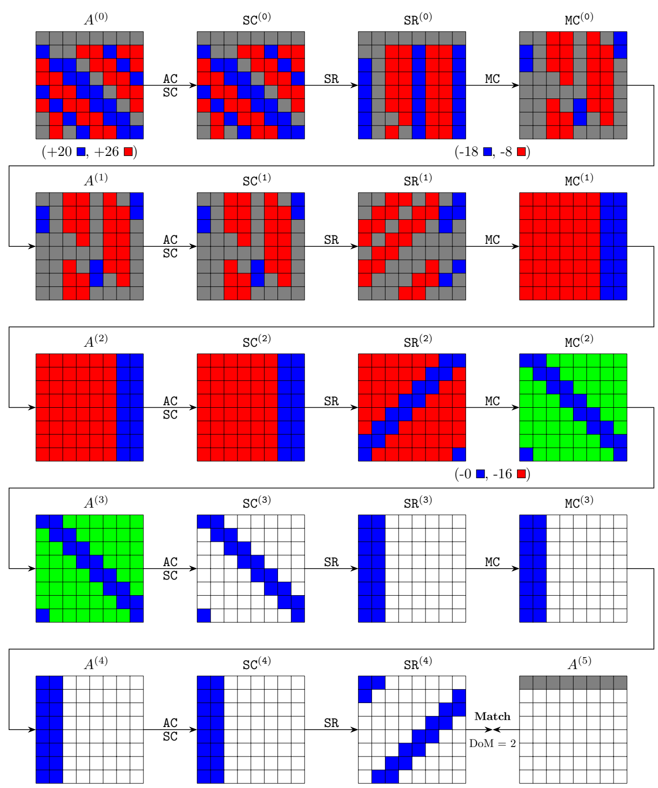


Fig. 23: The 4.5-round MitM attack on PHOTON-224/32/32

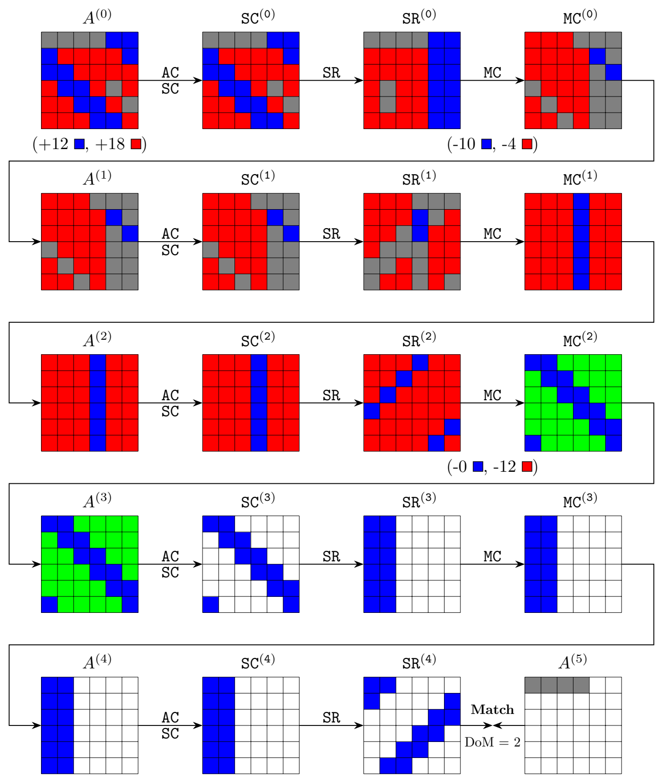


Fig. 24: The 4.5-round MitM attack on PHOTON-256/32/32

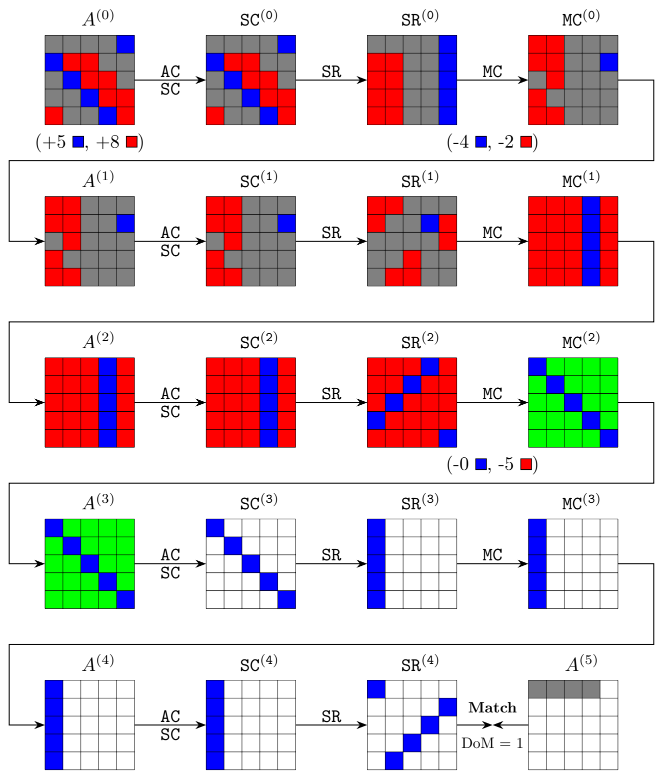


Fig. 25: The 4.5-round MitM attack on PHOTON-80/20/16

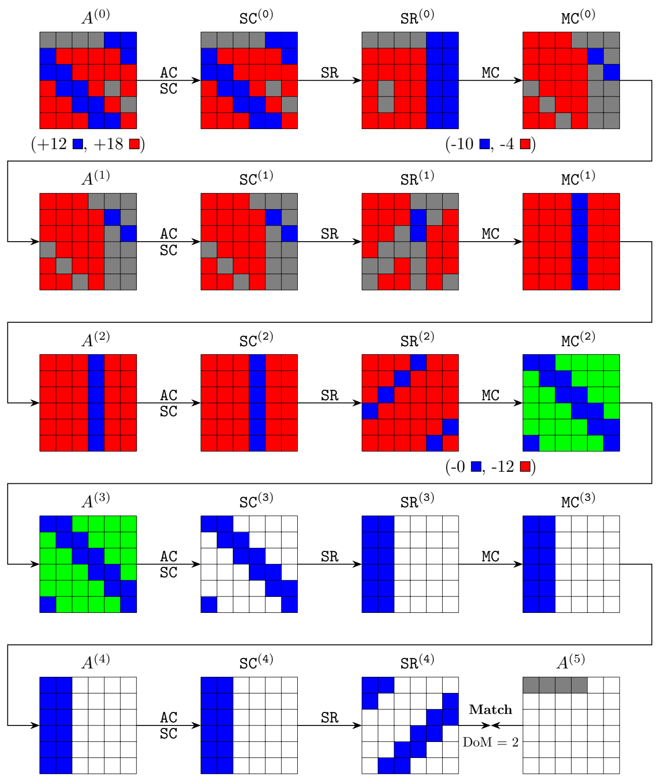


Fig. 26: The 4.5-round MitM attack on PHOTON-128/16/16

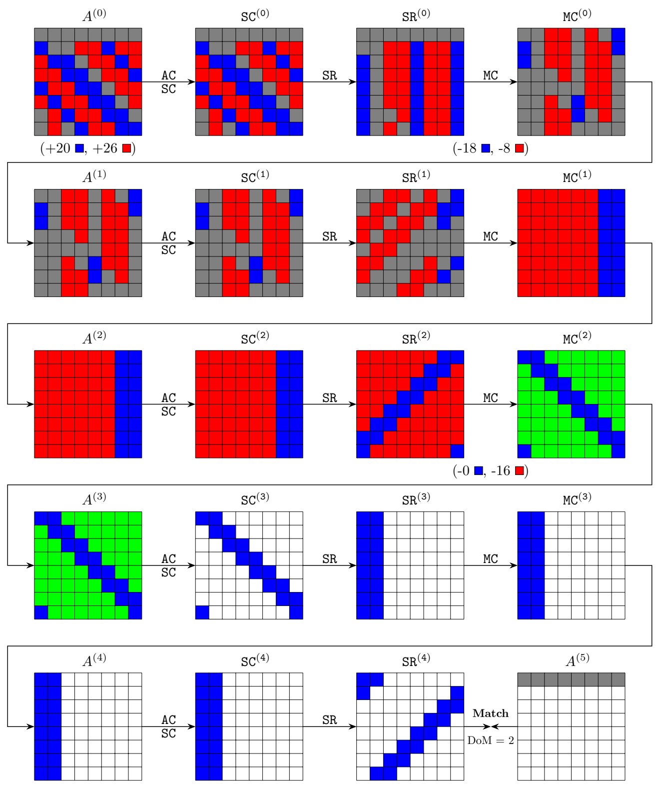


Fig. 27: The 4.5-round MitM attack on PHOTON-224/32/32

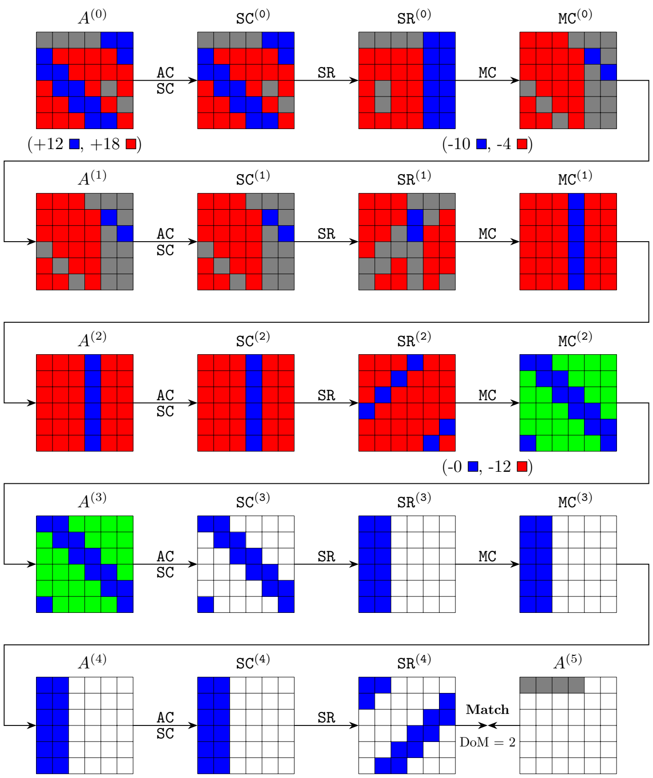


Fig. 28: The 4.5-round MitM attack on PHOTON-256/32/32

– **Phase I:** As shown in Figure 29, we conduct the MitM attack with given T_1 to find a capacity state X satisfying $f(T_1\|X) = T_2\|*$. There are 16 ■ nibbles and 16 ■ nibbles in the capacity state X. In the computation, there consumes 8 ■ nibbles and 8 ■ nibbles. After 3.5 rounds, we get 8 nibbles as matching points with the given target T_2. The detail of the MitM attack is given in Algorithm 13. The time of Phase I is 2^{97} and the memory of storing U and V is 2^{65}.
– **Phase II:** We need to find a collision at $c = 256 - 32 = 224$ bits inner part. The time complexity is 2^{112} without memory.

The total time complexity is 2^{112} and memory complexity is 2^{65}.

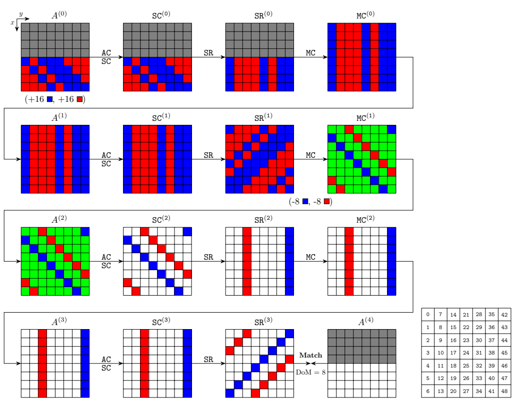


Fig. 29: The 3.5-round MitM attack on `PHOTON-Beetle-Hash`

The details of the MitM attack of phase I on 3.5-round `PHOTON-Beetle-Hash` are given in Figure 29 and Algorithm 13. There are 16 ■ nibbles and 16 ■ nibbles in the capacity state. After 3.5 rounds, we get 8 nibbles as matching points in the given target T_2. In the computation, there consumes 8 ■ nibbles and 8 ■ nibbles. So we have $d^I_{\mathcal{R}} = d^I_{\mathcal{B}} = 8 \times 4 = 32$ and $m^I = 8 \times 4 = 32$. The time complexities are analyzed as follows:

– The time to build U is 2^{64}. There are 2^{32} values for each index ϕ on average.
– The time to build V is 2^{64}. There are 2^{32} values for each index φ on average.
– There are $2^{y+32+32-32} = 2^{y+32}$ values matched. According to Section 3, we need $2^{128-8\times 4} = 2^{96}$ matched values. Therefore, $y = 96 - 32 = 64$ and the time complexity is $2^{64} \times (2^{32} + 2^{32} + 2^{32+32-32}) = 2^{97}$.

The time of Phase I is about 2^{97}. The memory complexity of storing U and V is 2^{65}.

Algorithm 13: MitM of Phase I in the 3.5-round Attack on `PHOTON-Beetle-Hash`

Set the value of the 32 ■ nibbles in $\mathsf{A}^{(0)}$ as the given T_1
$U \leftarrow [\ ]$, $V \leftarrow [\ ]$
for *each value of 16* ■ *nibbles* $v_{\mathcal{R}}$ *in* $\mathsf{A}^{(0)}$ *with* ■ *as fixed 0* **do**
  Compute forward to state $\phi = \mathtt{MC}^{(1)}_{1,9,17,25,33,41,42}$.
  $U[\phi] \leftarrow v_{\mathcal{R}}$
end
for *each value of 16* ■ *nibbles* $v_{\mathcal{B}}$ *in* $\mathsf{A}^{(0)}$ *with* ■ *as fixed 0* **do**
  Compute forward to state $\varphi = \mathtt{MC}^{(1)}_{5,13,14,22,30,38,46}$.
  $V[\varphi] \leftarrow v_{\mathcal{B}}$
end
for 2^y *values of* ϕ *and* φ /* $y \leq 32 + 32$ */
do
  for 2^{32} *values* $v_{\mathcal{R}}$ *in* $U[\phi]$ **do**
    Compute $\mathsf{SR}^{(3)}$ and store the ■ values in L_U indexed by the 8-nibble matching point
  end
  for 2^{32} *values in* $V[\varphi]$ **do**
    Compute $\mathsf{SR}^{(3)}$ and store the ■ values in L_V indexed by the 8-nibble matching point
  end
  for *values matched between* L_U *and* L_V **do**
    Compute $\mathsf{SR}^{(3)}$ from the matched ■ and ■ nibbles.
    if *it leads to the given* T_2 **then**
      Output the preimage as X
    end
  end
end

F Attack on 7-round `SPONGENT-88`

F.1 7-round Preimage Attack on `SPONGENT-88`

The 7-round MitM characteristic is shown in Figure 30. The starting state $A^{(7)}$ contains 67 🟥 bits and 2 🟦 bits. The first 8 bits of $A^{(0)}$ are fixed as T_1 (marked as ▩), and the remaining 80-bit □ are capacity which are not known. In the inverse computation from $A^{(7)}$ to $S^{(1)}$, the consumed degree of freedoms (DoFs) of 🟥 and DoFs of 🟦 are 65 and 0, respectively. Therefore, $d_{\mathcal{B}} = 2$, $d_{\mathcal{R}} = 2$, and there are 2 matching bits. The 7-round MitM attack is given in Algorithm 14.

Analysis of Algorithm 14. In Line 9 to 17, $2^{65+2+2+\zeta}$ states are tested against the 80-bit $T_1 \| T_3 \| \cdots \| T_{11}$, therefore, $\zeta = 11$ is enough to find a preimage.

- The Line 6 to 7, the time complexity is $2^{11+67} \times \frac{32}{7\times 22} = 2^{78} \times 2^{-2.27} = 2^{75.73}$ 7-round SPONGENT.
- The Line 12, the time complexity is $2^{11+65+2} \times \frac{46}{7\times 22} = 2^{78} \times 2^{-1.74} = 2^{76.26}$ 7-round SPONGENT.
- The Line 17, the time complexity is $2^{11+65+2} = 2^{78}$ 7-round SPONGENT.

In **Phase II**, it is trivial to find an inner collision for the 80-bit capacity with the Floyd's cycle finding algorithm [31] with 2^{40} time and no memory.

Thereore, the total preimage attack on 7-round SPONGENT is $2^{75.73} + 2^{76.26} + 2^{78} + 2^{40} \approx 2^{78.59}$ time and $2^{67} + 2^2 + 2^2 \approx 2^{67}$ memory.

G Parameters for `subterranean 2.0`

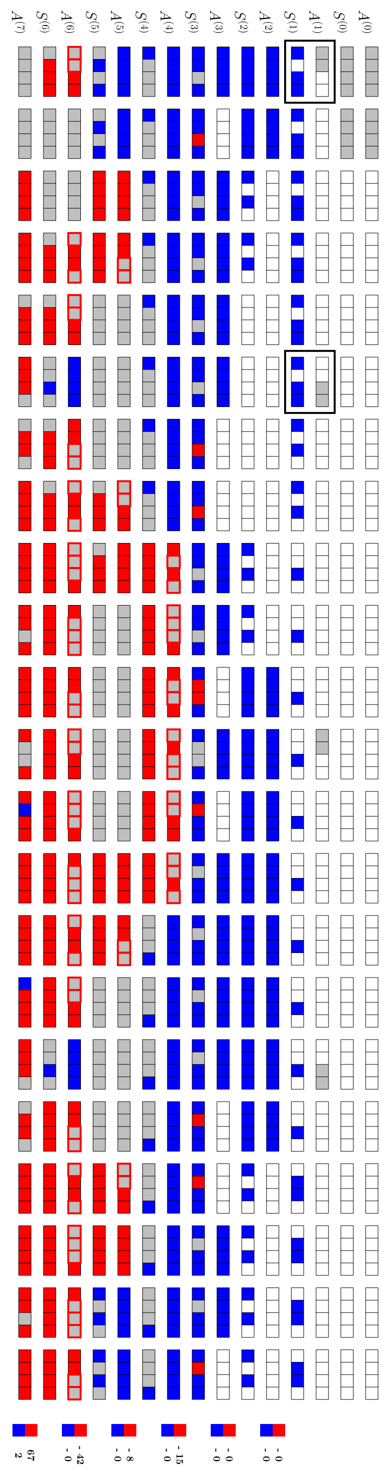


Fig. 30: The MitM preimage attack on 7-round Spongent-88

Algorithm 14: Preimage Attack on 7-round `SPONGENT-88`: Phase I

Fix the first 8 bits of $A^{(0)}$ as T_1, and compute forward to get the 8-bit of $A^{(1)}$, which are marked as ■

`/* According to Alg. 5, prepare tables with valid elements through Sbox matching for the 2 Sbox. For simplicity, we build a full table containing all valid elements. */`

Build the table L, stored $2^{3+1-2} \times 2^{3+1-2} = 2^4$ valid elements indexed by $3+3=6$ blue bits, i.e., $S^{(1)}[0,2,3,20,22,23]$

`/* Therefore, under each 6-bit index, there are` $2^4/2^6 = 2^{-2}$ `elements. */`

for 2^{ζ} *values of the 11-bit free gray bits in* $A^{(7)}$ **do**
    for 2^{67} *values of the* ■ *bits* $v_{\mathcal{R}}$ *in* $A^{(7)}$ **do**
        Compute backward to determine the 65-bit ■/■ (denoted as $c_{\mathcal{R}} \in \mathbb{F}_2^{65}$), and store the 67-bit ■ $v_{\mathcal{R}}$ of $A^{(7)}$ in $V[c_{\mathcal{R}}]$
    end
    for $c_{\mathcal{R}} \in \mathbb{F}_2^{65}$ **do**
        Retrieve the 2^2 $v_{\mathcal{R}}$ from $V[c_{\mathcal{R}}]$
        for 2^2 *values of* ■ *bits* $v_{\mathcal{B}}$ **do**
            Compute to derive the $3+3=6$ ■ bits $\xi = S^{(1)}[0,2,3,20,22,23]$
            if $L[\xi]$ *is not empty* `/* Probability of` 2^{-2} `*/`
            then
                Combine $v_{\mathcal{B}}$ and 2^2 $v_{\mathcal{R}}$ to construct the full state Y
                if T_1 *is satisfied* **then**
                    Check if $T_3 \| \cdots \| T_{11}$ is satisfied
                end
            end
        end
    end
end

i	state bits	i	state bits	i	state bits	i	state bits
0	(1, 256)	8	(64, 193)	16	(241, 16)	24	(4, 253)
1	(176, 81)	9	(213, 44)	17	(11, 246)	25	(190, 67)
2	(136, 121)	10	(223, 34)	18	(137, 120)	26	(30, 227)
3	(35, 222)	11	(184, 73)	19	(211, 46)	27	(140, 117)
4	(249, 8)	12	(2, 255)	20	(128, 129)	28	(225, 32)
5	(134, 123)	13	(95, 162)	21	(169, 88)	29	(22, 235)
6	(197, 60)	14	(15, 242)	22	(189, 68)	30	(17, 240)
7	(234, 23)	15	(70, 187)	23	(111, 146)	31	(165, 92)

Table 8: Mapping between state bits and input/output bits of `Subterranean 2.0`

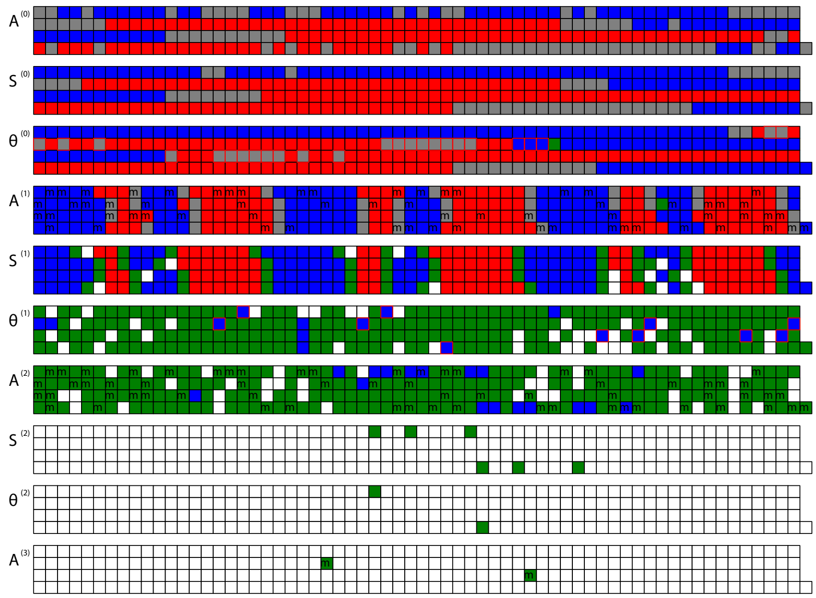


Fig. 31: MitM Preimage Attack on `Subterranean-XOF`

Algorithm 15: MitM Preimage Attack on `Subterranean-XOF`

Fix the value to generate T_0 in $A^{(0)}$.
for 2^{60} *values of* *in* $A^{(0)}$ **do**
  for 2^{100} *values of the* *bits* $v_{\mathcal{R}}$ *in* $A^{(0)}$ **do**
    Compute the 36-bit / (denoted as $c_{\mathcal{R}}$) and 65-bit matching point. Build the table U and store the 100-bit $v_{\mathcal{R}}$ of $A^{(0)}$ as well as the matching point in $U[c_{\mathcal{R}}]$.
  end
  for $c_{\mathcal{R}} \in 2^{36}$ **do**
    Retrieve the 2^{64} elements of $U[c_{\mathcal{R}}]$ and restore $v_{\mathcal{R}}$ in L_1 under the index of 65-bit matching point
    for 2^{64} *values* $v_{\mathcal{B}}$ *of* *bits in* $A^{(0)}$ **do**
      Compute forward to the 65-bit matching point and store $v_{\mathcal{B}}$ in L_2 with the 65-bit matching as index
      for *values matched in* L_1 *and* L_2 **do**
        Reconstruct the state X with $v_{\mathcal{R}}$ and $v_{\mathcal{B}}$
        if *it leads to the given 224-bit hash value* $(T_2, T_3 ... T_8)$ **then**
          Output the preimage
        end
      end
    end
  end
end