

Enhancing SAT Solving to Find (Near) Collisions in 6-Round SHA-3 Variants

Yi Tu^{1,2,3}, Ling Song¹(✉), Huaijin Wu², Jian Guo³,
Jian Weng⁴, and Chaoping Xing²

¹ Jinan University, Guangzhou, China
songling.qs@gmail.com

² Shanghai Jiao Tong University, Shanghai, China
whj1201@sjtu.edu.cn, xingcp@sjtu.edu.cn

³ School of Physical and Mathematical Sciences, Nanyang Technological University,
Singapore
tuyi0002@e.ntu.sg, guojian@ntu.edu.sg

⁴ Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou
510006, China
cryptjweng@gmail.com

Abstract. The KECCAK hash function, designed by Bertoni *et al.*, was selected as the new generation of Secure Hash Algorithm (SHA-3) in 2012. For **NIST-standardized SHA-3** instances (SHA3-224/256/384/512, SHAKE128, SHAKE256), practical collision attacks have reached five rounds; six-round SHAKE128 was known only as a *theoretical* attack with cost $2^{123.5}$ six-round evaluations [GLST22]. In this paper, we present the first *classical* practical collision attack on six-round SHAKE128 with $d=160$ and total complexity about $2^{47.82}$. Building on the Dinur-framework Dinur *et al.* [DDS12] and the SAT tooling of Guo *et al.* [GLST22], we introduce a **parameterized** colliding-trail SAT model—collision length in digest and weight bounds are chosen at setup together with leaner differential encodings. Specifically, we introduce an enhanced heuristic strategy to identify suitable differential trails and employ multiple techniques to optimize the connectors: for the connector phase, we minimize weighted propagation cost of second round and third round, and penalize costly DDT=2 transitions; candidate χ -compatible equations are chosen by a bit-granular greedy routine that maximizes connector degrees of freedom. In addition to SHAKE128, we report six-round **near**-collisions on SHA3-224 and SHA3-256 (four and six differing digest bits, respectively), and practical collisions on four- and five-round SHAKE256 with the full 512-bit digest.

Keywords: SHA-3, Collision attacks, SAT-based search tool, Practical attacks

1 Introduction

The KECCAK hash function [BDPVA13] was designed by Bertoni *et al.* in 2008 and formally standardized in 2015 by the U.S. National Institute of Standards

and Technology (NIST) as the Secure Hash Algorithm-3 (SHA-3) [Dwo15]. SHA-3 consists of four main instances with fixed digest lengths, *i.e.*, SHA3-224, SHA3-256, SHA3-384 and SHA3-512, and two eXtendable-Output Functions (XOFs) SHAKE128 and SHAKE256. To promote cryptanalysis of the KECCAK hash function, in the KECCAK Contest [BDPV] the design team proposed a series of reduced variants KECCAK $[r, c, n_r, d]$ with parameters $\{r, c, n_r\}$ to be specified later and the digest length d being 80 and 160 bits for pre-image and collision resistance, respectively.

Due to its theoretical and practical importance, SHA-3 (KECCAK) has received intensive security analysis from the community since it was made public in 2008. In this work, we focus on the collision resistance of SHA-3 with a reduced number of rounds, in particular, those practical collision attacks. In [DDS13], Dinur *et al.* generalized the internal differentials and applied them to present practical collision attacks on 3 rounds of KECCAK-384 and KECCAK-512, and theoretical collision attacks on 5/4-round KECCAK-256/KECCAK-384, respectively. Recently, Zhang *et al.* [ZHL23, ZHL24] proposed conditional internal differentials and introduced dedicated probabilistic linearization, which leads to better theoretical collision attacks on SHA-3. In [DDS12, DDS14], Dinur *et al.* also introduced a collision attack framework of combining a differential trail with a connector that links the differential trail to the initial states. With this framework, the authors presented practical collision attacks on 4 rounds of KECCAK-224 and KECCAK-256 by combining a 3-round differential trail and a 1-round connector. Following the same framework proposed by Dinur *et al.*, Qiao *et al.* [QSLG17] introduced 2-round connectors by prepending a fully linearized round to the 1-round connectors and obtained actual collisions for 5-round SHAKE128 and two 5-round instances of the KECCAK Contest. Subsequently, Song *et al.* in [SLG17] introduced the non-full S-box linearization to improve the techniques of constructing connectors, such that connectors of more degrees of freedom can be constructed. Consequently, they extended the connectors to up to three rounds and hence achieved the first practical collision attacks on 5-round SHA3-224 along with the first practical collision attacks on a 6-round instance of the KECCAK Contest. Later in [GLL⁺20], a practical collision attack against 5-round SHA3-256 was obtained by further saving more degrees of freedom in the connector construction phase. Recently, Guo *et al.* [GLST22] exploited SAT techniques in multiple intermediate phases of the attack and proposed the first theoretical collision attack on 6-round SHAKE128 with time complexity $2^{123.5}$, together with collision attacks on 6 rounds of SHA3-224 and SHA3-256 in the quantum setting. In [HBDM22], Huang *et al.* used 2-block messages instead of 1-block ones, to enjoy greater degrees of freedom and constructed 1-round connectors with the help of SAT. Combining a high-probability differential trail, a 4-round collision attack on SHA3-384 was achieved with a practical complexity $2^{59.64}$. In addition, collision attacks on the smallest instances of the KECCAK Contest were studied using algebraic methods in [BNR21].

To the best of our knowledge, there exist *no 6-round practical collision attacks* against any **NIST-standardized** instance of the SHA-3 family. The Dinur-framework pipeline was initially proposed by Dinur *et al.* [DDS12] and

further developed in subsequent studies [QSLG17, SLG17, GLL⁺20, GLST22]. This framework enables collision attacks on $n_{r_1} + n_{r_2}$ rounds and proceeds in the following three phases.

- Phase 1.** Prepare n_{r_2} -round high-probability differential trails whose output difference is zero on a chosen digest slice (length d and bit pattern fixed at search time). These trails are referred to as **colliding differential trails**.
- Phase 2.** Construct n_{r_1} -round connectors that produce a subspace of message pairs together with a differential trail that links the colliding trail. This trail is called the **connecting differential trail**.
- Phase 3.** Exhaustively search a colliding pair within the message pairs returned by the connectors.

Both the colliding differential trail and the connecting differential trail play pivotal roles in the attack. In brief, the probability associated with the colliding trail directly determines the complexity of Phase 3, while the probability of the connecting trail has a substantial impact on the degrees of freedom of the resulting message subspace, a critical factor in finding a colliding pair.

Our Contributions. We give the first explicit message pairs that collide on six-round SHAKE128 when the digest is read at length $d=160$ bits (Table 9 in the appendix); this is the first *classical* practical 6-round collision on a NIST-standardized SHAKE128 parameterization, distinct from earlier contest-scale instances at smaller $r+c$. We adopt the same Dinur-framework pipeline as Guo *et al.* [GLST22], but extend their SAT tooling to parameterized output-collision goals and obtain a classical practical six-round attack at feasible d . The same pipeline further yields practical full 512-bit digest collisions on four- and five-round SHAKE256 (Tables 11 and 12 in the appendix), improving on the previous best attacks of complexities 2^{76} [ZHL23] and $2^{163.28}$ [ZHL24], respectively. Our upgrades over [GLST22] are threefold (details in Supplementary Material, Section B). The first two are *engineering generalizations* of their SAT interface rather than new cryptographic principles; the third carries the algorithmic content.

- **Configurable digest constraints and colliding-trail SAT (engineering).** The colliding phase is parameterized at setup by effective digest length d , vanishing lanes or bit ranges, and weight or mask bounds on remaining differences, enabling truncated, near-, and full- d targets from one CNF backbone—instead of a fixed full-length zero-digest pattern per instance.
- **Lean digest-side encoding (engineering).** Digest conditions are expressed on sparse β_5 slices (or pulled back to α_5); at $d=160$ only **320** SAT variables encode the admissible last-round input differences, rather than the full β_5 state together with a duplicated α_6 layer.
- **Connector-oriented connecting trails and equation-level greedy connectors (algorithmic).** Connecting trails are ranked by $w(\beta_1) + k w(\beta_2)$ and DDT=2 counts; partial connectors use a bit-granular greedy rule on χ -compatible equations to maximize connector dimension (DF), rather than locking whole S-boxes as in [GLST22].

For SHAKE128, we particularly exploit its extendable digest length to execute practical collision attacks. Correspondingly, its collision resistance level can be expressed as $\min(d/2, 128)$. In our attack, we choose $d = 160$. With the application of the early-abort technique, the time complexity of our attack is approximately $2^{47.82}$, and actual collisions are found in a matter of hours using an NVIDIA A100 GPU. **Comparison with generic collision search.** The birthday bound at $d=160$ is 2^{80} calls to the *full* hash; our figure counts reduced-round (*i.e.*, six-round) evaluations along the differential attack pipeline and is therefore not directly comparable as “hash queries”—nevertheless Phase 3 falls below 2^{80} because the differential trail alone implies effective filtering stronger than generic hashing. The results of our work, along with a comparison to previous works, are presented in Table 1.

Theoretic near-collision attacks on 6-round instances. We also apply these techniques to other 6-round instances including SHA3-224 and SHA3-256. As a result, theoretic near-collision attacks are obtained with 4 and 6 difference bits for SHA3-224 and SHA3-256, respectively.

Organization. The rest of the paper is organized as follows. Section 2 gives a concise description of the SHA-3 family and reviews prior practical collision attacks. In Section 3, we present the SAT models and connector heuristics. Section 4 instantiates the Dinur-framework pipeline in three phases—colliding-trail search, connector construction, and GPU-assisted brute-force search with early abort. Section 5 reports a practical collision on six-round SHAKE128 at $d=160$, practical full-digest collisions on four- and five-round SHAKE256, near-collisions on six-round SHA3-224 and SHA3-256, and an improved collision attack on KEC-CAK[1440, 160, 6, 160]. Finally, Section 6 concludes the paper. For additional details regarding differential trails, colliding message pairs, and a detailed comparison with Guo *et al.* [GLST22], please refer to the supplementary material.

2 Preliminary

In this section, we introduce the notations used in this paper and give a description of SHA-3 as well as a review of previous collision attacks, followed by a rethinking of the previous works.

2.1 Notations

Most of the notations used in this paper are listed below.

Table 1: Summary of collision attacks against the SHA-3 family. For SHAKE128, digest size d may vary across rows (always stated explicitly). “Near” rows report Hamming distance between digests where applicable.

Target	d	Rounds	Type	Near-diff.	Complexity	GPU time	Reference
SHAKE128	256	5	Full	0	Practical	—	[GLL ⁺ 20]
	256	6	Full	0	$2^{123.5}$	—	[GLST22]
	256	6	Near	8	$2^{47.82}$	$\approx 9.5\text{--}41.5$ h	Sect. 5.1
	160	6	Full	0	$2^{47.82}$	$\approx 9.5\text{--}41.5$ h	Sect. 5.1
SHAKE256	512	4	Full	0	2^{76}	—	[ZHL23]
	512	5	Full	0	$2^{163.28}$	—	[ZHL24]
	512	6	Full	0	$2^{232.29}$	—	[ZHL24]
	512	4	Full	0	2^{16}	practical	Sect. 5.2
	512	5	Full	0	2^{36}	practical	Sect. 5.2
SHA3-224	224	5	Full	0	Practical	—	[GLL ⁺ 20]
	224	6	Near	4	$2^{90.02}$	—	Sect. 5.3
SHA3-256	256	5	Full	0	Practical	—	[GLL ⁺ 20]
	256	6	Near	6	$2^{117.85}$	—	Sect. 5.4
SHA3-384	384	3	Full	0	Practical	—	[DDS13]
	384	4	Full	0	$2^{59.64}$	—	[HBDM22]
	384	5	Full	0	$2^{172.19}$	—	[ZHL24]
SHA3-512	512	3	Full	0	Practical	—	[DDS13]
	512	4	Full	0	$2^{225.29}$	—	[ZHL24]

Standard sponge parameters (r rate, c capacity in bits): SHAKE128/SHAKE256 use (1344, 256) / (1088, 512); SHA3-224/256/384/512 use (1152, 448), (1088, 512), (832, 768), (576, 1024).

“GPU time” lists observed wall-clock on one NVIDIA A100 for our new practical rows; earlier rows cite published CPU/GPU mixes and omit this column.

The two bottom SHAKE128 rows report the same practical attack: a full collision at $d=160$ and an 8-bit near-collision at $d=256$ (Table 9), both with total complexity $2^{47.82}$ six-round evaluations (Table 13). A separate *theoretical* full- $d=256$ accounting gives 2^{121} evaluations (“When $d=256$ ” in Sect. 5.1); we did not exhibit a full 256-bit collision.

Extrapolating [GLST22] to $d=160$ with their connector model (but a $d=160$ colliding trail) plausibly remains at $\geq 2^{115.5}$ evaluations and is still theoretical; see Paragraph 5.6.

c	Capacity of a sponge function
r	Rate of a sponge function
d	Length of the digest in bits
p	Number of fixed bits in the initial state due to padding
n_r	Number of rounds
$Keccak-f$	The underlying permutation of SHA-3 hashes functions
$\theta, \rho, \pi, \chi, \iota$	The five operations of the round function of $Keccak-f$. A subscript i denotes the operation at the i -th round, <i>e.g.</i> , χ_i denotes the χ layer at the i -th round where $i = 0, 1, 2, \dots$
λ	Composition of θ, ρ, π and its inverse denoted by λ^{-1}
RC_i	Round constant of the i -th round, where $i = 0, 1, 2, \dots$
$R^i(\cdot)$	$Keccak-f$ permutation reduced to the first i rounds
$S(\cdot)$	5-bit S-box operating on each row of $Keccak-f$ state
$\delta_{in}, \delta_{out}$	5-bit input and output differences of an S-box
DDT	Differential distribution table, and $DDT(\delta_{in}, \delta_{out}) = \{x : S(x) + S(x + \delta_{in}) = \delta_{out}\} $, where $ \cdot $ denotes the size of a set
α_i	Input difference of the i -th round, where $i = 0, 1, 2, \dots$
β_i	Input difference of χ in the i -th round, where $i = 0, 1, 2, \dots$
w_i	Sum of per-S-box propagation weights in round i (cf. Equation 4); <i>not</i> the Hamming weight of the full 1600-bit state
$w(\beta_i)$	Propagation weight of β_i (sum of per-S-box weights via Equation 4), <i>not</i> the Hamming weight of the 1600-bit mask
$w^{rev}(\alpha_i)$	Minimal reverse propagation weight through χ : $w^{rev}(\alpha_i) = \min\{w(\gamma) \mid \gamma \xrightarrow{\chi} \alpha_i \text{ compatible}\}$ (see [DVA12, GLST22])
DF	Degree of freedom of the solution space of connectors
$\overline{M}$	Padded message of M . Note that $\overline{M}$ is of one block in our attacks.
$M_1 M_2$	Concatenation of strings M_1 and M_2
x_i	Bit value vector before λ of each round, where $i = 0, 1, 2, \dots$
y_i	Bit value vector before χ of each round, where $i = 0, 1, 2, \dots$
E_{y_i}	System of equations on y_i of each round, where $i = 0, 1, 2, \dots$

2.2 Description of SHA-3 family

The **SHA-3** family, as specified in [Dwo15], encompasses a specific set of hash functions derived from **KECCAK** [BDPVA13]. These functions are built using the sponge construction [BDPVA07, GJMG11] with an internal permutation called $Keccak-f$.

The Sponge Function. The sponge construction, illustrated in Figure 1, provides a framework for building hash functions from permutations. It comprises three key components: a b -bit permutation denoted as f , a parameter r referred to as the rate, and a padding rule. The difference between the block size b and the rate r defines the capacity $c = b - r$. Utilizing this construction, a hash function

accepts a message M as input and generates a digest of d bits. When presented with a message M , the sponge construction first pads and divides it into r -bit blocks. The b -bit state is initialized to all zeros. Subsequently, the construction proceeds through two phases. During the absorbing phase, each message block is XORed into the first r bits of the state, followed by the application of the permutation f . This process iterates until all message blocks are processed. Subsequently, the construction transitions to the squeezing phase. Here, each iteration yields the first r bits of the state as part of the output, followed by the application of the permutation f to the current state. This cycle continues until all d bits of the digest are obtained.

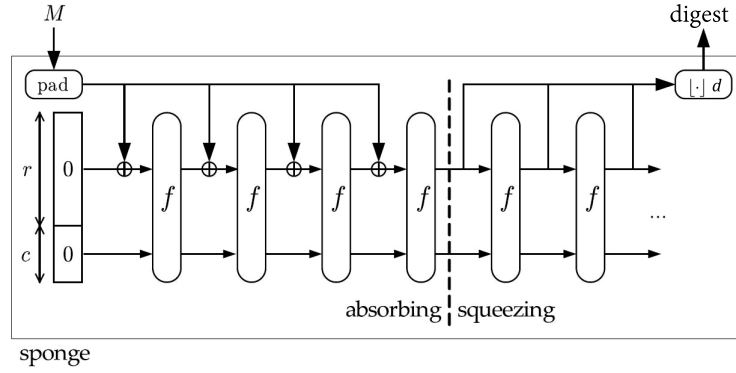


Figure 1: Sponge Construction [GJMG11]

Keccak- f Permutation. The KECCAK hash function adheres to the sponge construction. Its underlying permutation, *Keccak- f* , operates on a large state size of 1600 bits and comprises a total of 24 iterative rounds. Each round consists of five distinct operations: four linear operations denoted by θ , ρ , π , and ι , and one nonlinear operation denoted by χ . The 1600-bit state is structured as a 3-dimensional array of bits, specifically $5 \times 5 \times 64$, represented as $A[5][5][64]$. Each state bit, indexed by the coordinates (i, j, k) in the state array, is denoted as $A[i][j][k]$, where $0 \leq i, j < 5$ and $0 \leq k < 64$. The five mappings of the *Keccak- f* round are detailed by the following transformations.

$$\theta: A[i][j][k] \leftarrow A[i][j][k] \oplus \sum_{j'=0}^4 A[i-1][j'][k] \oplus \sum_{j'=0}^4 A[i+1][j'][k-1].$$

$$\rho: A[i][j] \leftarrow A[i][j] \lll T(i, j), \text{ where } T(i, j) \text{ s are constants.}$$

$$\pi: A[j][2i+3j] \leftarrow A[i][j].$$

$$\chi: A[i][j][k] \leftarrow A[i][j][k] \oplus (A[i+1][j][k] \oplus 1) \cdot A[i+2][j][k].$$

$$\iota: A[0][0] \leftarrow A[0][0] \oplus RC_{i_r}, \text{ where } RC_{i_r} \text{ is the } i_r\text{-th round constant.}$$

The multiplication used in χ operation is in $\text{GF}(2)$. As the ι operation does not impact differences, we will omit it for the remainder of the paper unless explicitly mentioned.

Instances of SHA-3 Family. The SHA-3 family comprises six instances, including four fixed-length variants denoted as SHA3-224/256/384/512 (where the numbers 224/256/384/512 indicate the digest size in bits), and two extendable output variants known as SHAKE128 and SHAKE256. A multirate padding rule, denoted as 10^*1 , is defined for all SHA-3 instances. For SHA3-224/256/384/512, a 2-bit string “01” is appended to the message before applying padding, with the capacity specified as $c = 2 \times d$. In the case of the two extendable variants, a 4-bit string “1111” is appended to the messages prior to applying the multirate padding rule. The capacity is set to 256 and 512 bits for SHAKE128 and SHAKE256, respectively. It is noteworthy that the digest size d of SHAKE128 and SHAKE256 can vary, leading to their collision resistance levels being not solely defined by $d/2$, but rather by $\min(d/2, 128)$ and $\min(d/2, 256)$, respectively.

2.3 Review of Previous Practical Collision Attacks on SHA-3

In this subsection, the associated core techniques developed in previous collision attacks on SHA-3 are discussed. Collisions of reduced-round SHA-3 are obtained by combining a differential trail and a connector that links the initial state of SHA-3 and the input of the trail, which was first proposed by Dinur *et al.* at FSE 2012 [DDS12].

Framework of Collision Attacks on SHA-3. The collision attacks on SHA-3 and KECCAK instances consist of three phases as below.

Phase 1. Prepare n_{r_2} -round high-probability colliding differential trails where the first d bits of the output difference is zero. The input and output differences for the colliding trail are denoted by ΔS_I and ΔS_O , respectively.

Phase 2. Construct n_{r_1} -round connectors that produce a subspace of message pairs together with a connecting differential trail $\Delta \overline{M} || 0^c \rightarrow \Delta S_I$ such that

$$\mathbf{R}^{n_{r_1}}(\overline{M} || 0^c) + \mathbf{R}^{n_{r_1}}((\overline{M} \oplus \Delta \overline{M}) || 0^c) = \Delta S_I.$$

It means the c continuous “0” bits in capacity and padding rule are fulfilled and the message pairs would always reach ΔS_I after the first n_{r_1} rounds.

Phase 3. Exhaustively search a colliding pair among the messages pairs returned by the connectors

Figure 2 illustrates the case where $n_{r_1} = 2$ and $n_{r_2} = 4$.

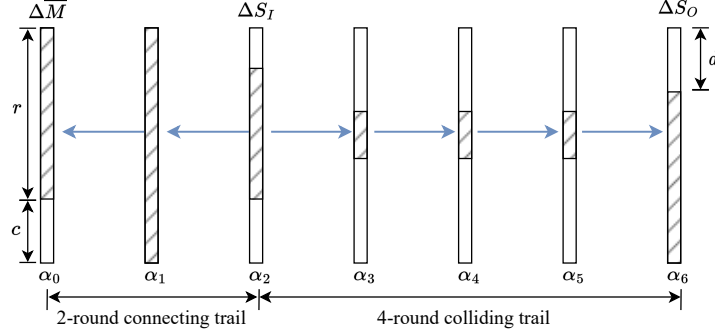


Figure 2: Overview of $(n_{r_1} + n_{r_2})$ -round collision attack on SHA-3

Properties of χ Exploited in Connector Construction. The most sophisticated step of the attack is to construct connectors. The core idea of the techniques for constructing connectors is to convert the problem to building a system of linear equations, which heavily depends on the following properties of the nonlinear operation χ . The statements match the supplementary write-up of Guo *et al.* [GLST22] and ultimately trace back to Dinur *et al.* [DDS12]; we repeat them here so the paper is self-contained.

Property 1 ([DDS12]). **The inputs of χ operation that follow a compatible differential (β, α) form a subspace.** Assume that $(x_4, x_3, x_2, x_1, x_0)$ are the 5 input bits of the χ operation which is also called S-box in general, and $\beta = (b_4, b_3, b_2, b_1, b_0)$ (resp. $\alpha = (a_4, a_3, a_2, a_1, a_0)$) is the 5-bit input (resp. output) difference. Based on the definition of χ , the relationship between the differential bits and the input bits can be deduced. As demonstrated in the following expressions, when the input difference β and the compatible output difference α of χ are given, equations on the five input bits x_i where $0 \leq i < 5$ are of at most degree 1, and the solutions form a subspace.

$$\begin{aligned}
 a_0 &= (x_1 \cdot b_2 \oplus b_1 \cdot x_2 \oplus b_0 \oplus b_2 \oplus b_1 \cdot b_2) \\
 a_1 &= (x_2 \cdot b_3 \oplus b_2 \cdot x_3 \oplus b_1 \oplus b_3 \oplus b_2 \cdot b_3) \\
 a_2 &= (x_3 \cdot b_4 \oplus b_3 \cdot x_4 \oplus b_2 \oplus b_4 \oplus b_3 \cdot b_4) \\
 a_3 &= (x_4 \cdot b_0 \oplus b_4 \cdot x_0 \oplus b_3 \oplus b_0 \oplus b_4 \cdot b_0) \\
 a_4 &= (x_0 \cdot b_1 \oplus b_0 \cdot x_1 \oplus b_4 \oplus b_1 \oplus b_0 \cdot b_1)
 \end{aligned}$$

According to Property 1, when the differential (β, α) of χ is determined, the linear equations on input bits x_i can be directly listed.

Property 2 (S-box linearization [QSLG17, SLG17]). When the inputs $x = (x_4, x_3, x_2, x_1, x_0)$ of S-box are restricted to a small subspace (e.g., a subset of 4 inputs), the outputs $y = (y_4, y_3, y_2, y_1, y_0)$ can be expressed by linear combinations of input bits, i.e., the so-called S-box linearization, while by definition $y_i :=$

$x_i \oplus (x_{i+1} \oplus 1) \cdot x_{i+2}$, y_i bits are computed from nonlinear equations on x_i bits. In detail, the linearization of the S-box is discussed in two classifications.

- For active S-boxes, when the inputs are restricted to the solution set of the differential, some active S-boxes are already fully linearized while the other active S-boxes still need to linearize one bit. More specifically,
 1. for active S-boxes of $\text{DDT} = 2$ or $\text{DDT} = 4$ entry (*i.e.*, for a certain (a, b) , there exist 2 or 4 input pairs that follow the differential), any of the five output bits y_i can be expressed by a linear expression of input bits x_i .
 2. for active S-boxes of $\text{DDT} = 8$ entry, 4 out of the 5 output bits of y are naturally linearized and the only nonlinear y_i bit can be linearized by adding an extra equation on x_i .
- For non-active S-boxes, according to distinct attack scenarios, either the whole S-box or some output bits y_i are linearized. In either case, the linearizations can be achieved by adding equations on input bits.

SAT-based Collision Attacks. The colliding trails and the connecting trails are crucial for collision attacks on **SHA-3**. In order to find good differential trails, various tools and techniques were used in previous works. Particularly, a SAT-based tool was developed in [GLST22] and became the most efficient tool for searching differential trails of **SHA-3** so far, which outperforms other techniques including GPU implementations of searching differential trail of **KECCAK** to some degree. Unlike dedicated tools whose core part is the searching algorithm, the SAT-based tool only converts the problem of searching differential trails into an SAT problem, which can be solved by an off-the-shelf solver. Thus, the core part of a SAT-based tool is the conversion.

Based on the SAT-based automatic search tool, Guo *et al.* [GLST22] generated 4-round colliding differential trails with the collision in exactly 256-bit digests and applied it to mount 6-round collision attacks. Afterward, a connector is prepended before the colliding trails, given the input difference of the colliding trails is exactly the same as the output difference of the connector. Benefiting from the connecting trails found by the tool, 2-round connectors for the first two rounds were obtained for **SHAKE128** and provided many degrees of freedom. The extra degrees of freedom were then used to satisfy a fraction of the constraints in the third round while the remaining constraints are left to be met in the brute force stage, resulting in a *theoretical* 6-round collision attacks on **SHAKE128** with complexity $2^{123.5}$.

Using the same colliding trail, 6-round quantum collision attacks were proposed for **SHA3-224** and **SHA3-256** with partial 2-round connector, where a fraction of constraints in the second round were chosen greedily and satisfied. Constrained by the great amount of DFs consumed and the low probability of the 4-round colliding trail, mounting 6-round collision attacks on **SHA3-224** and **SHA3-256** is not feasible in the classical setting. The results in [GLST22] show that even with SAT-based automatic search toolkit, it is still very hard to efficiently find 4-round colliding differential trails of enough high probability which support collisions in 256-bit digest.

2.4 Rethinking of Previous Collision Attacks

A sequence of works has explored the collision attacks against SHA-3 [DDS12, DDS14, QSLG17, SLG17, GLL⁺20, GLST22]. The latter works follow the same attack framework initially proposed by Dinur *et al.* in [DDS12]. Collisions on *state-reduced* Keccak instances were studied in [SSJ23]. However, there has been no significant advancement in this field since the work by Song *et al.* [SLG17], which extensively investigated the first two stages of the attack framework nine years ago. Song *et al.* developed a series of 5-round practical collision attacks on SHA-3 during that time. Recently, in [GLST22], a SAT-based automatic tool was developed to assist in the collision attack on SHA-3, leading to the first theoretical 6-round collision attack on SHAKE128.

There were no suitable colliding trails for a practical 6-round collision attack. The first step of the collision attack framework is to search differential trails that yield no difference at the digest bits. It is important to emphasize that a differential trail with a probability just slightly exceeding $2^{-\frac{n}{2}}$ is inadequate for our purposes. To achieve practical collision attacks, the probability of the differential trail must be significantly higher. Furthermore, a connector will be prefixed to the differential trail, so more strict constraints need to be taken into consideration, *e.g.*, the number of active S-boxes in the round extended backward should be as small as possible. Due to the huge internal state size, searching differential trails of high probability for *Keccak-f* is tough, let alone finding a high-probability colliding trail where the difference of the first d output bits is restricted to zero. Even with the assistance of SAT-based automatic tools, the discovered colliding trails are still far from meeting the requirements for conducting a practical 6-round collision attack, as observed in [GLST22]. Consequently, without further improving the efficiency of searching strategies, better colliding trails of high probability would be unachievable. In other words, more efficient searching algorithms are essential to search better differential trails. Moreover, for a specific instance of SHA-3, the greater the specialization of the searching algorithm, the more favorable the outcomes are likely to be.

A high-probability connecting trail does not necessarily lead to connectors with a large degree of freedom (DF). A connecting trail, which links the colliding trail with the initial state, forms the foundation of connectors. Let us review the construction of connectors, which entails two key steps:

- Generate a connecting differential trail with an output difference denoted as ΔS_I and an input difference represented as $\Delta \overline{M}$. This trail must adhere to the constraints of capacity and padding rules.
- Add linear equations on the input value into the system according to the connecting trail using Property 1. Furthermore, apply S-box linearization techniques to unify equations of χ using Property 2. Subsequently, amalgamate various segments of equations. If the system of equations is consistent, an output is derived: a subspace of message pairs that will follow the connecting differential trail.

It is noted that the DF of the connector is not solely contingent on the weight of the underlying connecting trail. It also hinges on the linearization process applied to it and the interdependencies among different sets of linear equations. Thus, it is impossible to evaluate the exact DF of the resulting connector during the selection of the connecting trail.

In previous works, various heuristic methods were employed to select an appropriate connecting trail. In works such as [DDS12, DDS14, QSLG17], the process involved successively choosing differences for the S-boxes to maximize the probability of the resulting trail. This approach aimed to facilitate the construction of the connector concurrently. In contrast, some works like [SLG17, GLL⁺20] favored connecting trails with fewer active S-boxes in the initial round. Lastly, in [GLST22], a connecting trail was meticulously chosen with the aid of a SAT-based tool in advance to ensure that its weight was optimized.

The greedy algorithm for building partial connectors handled the constraints S-box by S-box. In [GLST22], the construction of partial 3-round connectors was executed on top of a 2-round connector for SHAKE128. These partial 3-round connectors selectively considered constraints from the third round, rather than all of them. It is desirable to consider as many constraints as possible at the lowest cost. Then a greedy algorithm was used. It sorted the active S-boxes of the third round by the number of nonlinear x_2 bits. Each time, it selected the active S-box with the least number of nonlinear x_2 bits and incorporated the constraints of the S-box. However, it is worth noting that if the selection of the greedy algorithm is more fine-grained, better partial connectors are possible.

3 New Techniques on Constructing Connectors and Trail Searching Based on SAT

This section elaborates the SAT-based techniques behind our practical six-round collision on SHAKE128 (Figure 3). We build on the Dinur-framework pipeline and the SAT tooling of Guo *et al.* [GLST22], but we change *how* trails are searched and encoded so that the same high-level pipeline yields a classical feasible attack at $d=160$. In contrast to [GLST22], we highlight three main upgrades: (i) *Joint colliding-trail objective (digest vs. weight)*. The colliding-phase SAT model no longer forces every one of the first d digest bits to be collision-free while ignoring the cost of the outer differential; instead we *jointly* negotiate the digest length d , the allowed pattern of digest differences (nonzeros may appear, but their number and placement are bounded), and the propagation-weight budget of the trailing rounds. This coupling keeps instances solvable and aligns the found trail with the Phase 3 workload at the chosen d . (ii) *Leaner encodings (fewer variables and clauses)*. We drop unnecessary intermediate state variables, for example, wherever constraints are stated on β_2 , we pull them back with L^{-1} to affine constraints on α_2 , instead of allocating fresh variables for β_2 and CNF for $\alpha_2 \rightarrow \beta_2$ differential propagation; this removes redundant state and clauses compared with [GLST22]. (details in Section 3.4). (iii) *Connecting-trail objective aligned*

with DF. In the connecting-trail search, β_1 and β_2 consume degrees of freedom at different rates when equations are merged for linearization. We add the weighted term $w(\beta_1) + k \cdot w(\beta_2)$ directly into the SAT objective, as worked out in the subsections below.

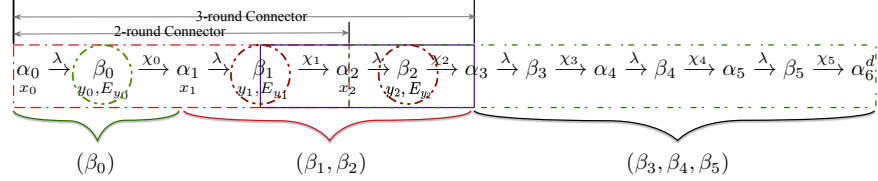


Figure 3: The 6-round collision attack model

We first search for good differential trails $(\alpha_3, \beta_3, \beta_4, \beta_5)$ so that the first d bits of α_6 can be zero with high probability and the number of active S-boxes in α_3 is less than a threshold. Then the connecting trail is searched for a given α_3 . Although the connecting trail covers the first three rounds, a connector can be built only for less than three rounds, depending on the available degrees of freedom. The constraints that are not considered by the connector will be satisfied probabilistically together with the colliding trail.

Joint optimization of digest shape and trail weight. Forcing more digest bits to collide and keeping the later-round differential cheap are optimized together in the SAT formulation, but they pull in opposite directions: requiring more colliding digest bits leaves fewer trails that also keep the remaining rounds light enough for Phase 3. In practice we therefore engineer a single weighted/layered objective (prioritize digest-pattern constraints up to feasibility, then minimize the propagation weight of the colliding segment subject to solver timeouts), instead of expecting both minima to coincide.

3.1 A New Heuristic Method for Searching Connecting Trails

On Choosing (β_0, β_1) . We choose (β_0, β_1) based on an analysis of the consumption of degrees of freedom [GLL⁺20]. For practical collision attacks, the degrees of freedom DF of the connector should be sufficiently large so that a colliding pair can be found from the solutions. In other words, DF of the connector should not be too small, compared to the weight of the colliding trail. As χ is nonlinear, in order to merge equations for χ_0 and χ_1 derived from Property 1, χ_0 should be linearized.

After linearizing the first round, the degree of freedom is $\sum_{i=0}^{b/5-1} \text{DF}_i^{(1)}$, where $\text{DF}_i^{(1)}$ is the degree of freedom of 5-bit input space of the i -th S-box in the first round. Specifically

$$\mathbf{DF}_i^{(1)} = \begin{cases} 1, & \text{DDT}(\delta_{in}, \delta_{out}) = 2, \\ 2, & \text{DDT}(\delta_{in}, \delta_{out}) = 4, \\ 2 \text{ or } 3, & \text{DDT}(\delta_{in}, \delta_{out}) = 8, \\ 2 \sim 5, & \text{DDT}(\delta_{in}, \delta_{out}) = 32, \end{cases} \quad (1)$$

where δ_{in} and δ_{out} are the input and output differences respectively of the i -th S-box in the first round. The exact value of $\mathbf{DF}_i^{(1)}$ is based on whether the output bits of the i -th S-box influence ΔS_I or not.

The constraints of the pre-fixed $(c + p)$ -bit on the initial state cost $(c + p)$ degrees of freedom. Besides, the constraints on the input values of the S-box layer in the second round also consume degrees of freedom. The definition of $\mathbf{DF}_i^{(2)}$, the degree of freedom of 5-bit input values to S-boxes in the second round, is

$$\mathbf{DF}_i^{(2)} = \begin{cases} 1, & \text{DDT}(\delta_{in}, \delta_{out}) = 2, \\ 2, & \text{DDT}(\delta_{in}, \delta_{out}) = 4, \\ 3, & \text{DDT}(\delta_{in}, \delta_{out}) = 8, \\ 5, & \text{DDT}(\delta_{in}, \delta_{out}) = 32, \end{cases} \quad (2)$$

where δ_{in} and δ_{out} are the input and output differences of the i -th S-box in the second round. After fixing the differential transition through an S-box in round 2, only $\mathbf{DF}_i^{(2)}$ bits in that 5-bit slice remain undetermined when values must satisfy the affine equations imposed by Property 1; identifying the missing bits therefore consumes $(5 - \mathbf{DF}_i^{(2)})$ degrees of freedom (cf. Guo *et al.* [GLL⁺20]). Intuitively, larger $\mathbf{DF}_i^{(2)}$ means fewer bits must be tied down and the subtracted term is smaller.

Therefore, the DF of the final system of the 2-round connector is estimated as follows without considering the dependencies.

$$\mathbf{DF} = \sum_{i=0}^{b/5-1} \mathbf{DF}_i^{(1)} - (c + p) - \sum_{i=0}^{b/5-1} (5 - \mathbf{DF}_i^{(2)}). \quad (3)$$

Thus, the DF is affected by the differential transition from β_0 to α_2 , as well as the S-box linearizations. Moreover, 3-round connectors are possible only when DF in Equation 3 is large.

Core ideas. According to the rules in Equation 1, the S-box with $\text{DDT}(\delta_{in}, \delta_{out}) = 2$ in the first round provides the smallest degrees of freedom even after linearization; according to Equation 2, the S-box with $\text{DDT}(\delta_{in}, \delta_{out}) = 2$ in the second round cost the most degrees of freedom. As a consequence, we propose a new method for searching for good connecting trails. The basic idea is to set the number of S-boxes with $\text{DDT}(\delta_{in}, \delta_{out}) = 2$ in the second round to zero and minimize the number of S-boxes with $\text{DDT}(\delta_{in}, \delta_{out}) = 2$ in the first round. This new method not only considers the weight of the connecting trail but also the linearization. Thus, it potentially leads to connectors with larger DF.

On Choosing β_2 . When a 2-round connector can be constructed with a large number of DF, we consider constructing 3-round connectors by satisfying partial constraints of the third round. In this case, there is a new subset of linear equations for χ_2 which should be merged with the system of equations for the obtained 2-round connector. However, the merge is possible only when certain bits of the output of χ_1 are linearized. In order to add one constraint of the third round, it usually consumes some extra degrees of freedom for linearizing the relevant bits of the output of χ_1 . Note, the number of constraints derived from the differential (according to Property 1) in each round is determined by the weight. Satisfying one more χ -compatible condition on β_2 typically forces additional linearizations on the output of χ_1 , which costs several degrees of freedom in expectation; tightening β_1 , by contrast, mainly affects already-localized equations from round 1–2. Therefore the weights $w(\beta_1)$ and $w(\beta_2)$ influence the achievable connector dimension unevenly, which is why we combine them as $w(\beta_1) + k w(\beta_2)$ with $k > 1$ instead of minimizing total Hamming weight alone.

Core ideas. Given α_3 , (β_1, β_2) are to be determined based on the SAT model. In order to balance the effect of $w(\beta_1)$ and $w(\beta_2)$, we introduce a factor k for $w(\beta_2)$. Constraints on minimal weighted summation $w(\beta_1) + k \times w(\beta_2)$ of propagation weight of β_1 and β_2 are also added into the SAT model.

Description of the New Heuristic Method. Given α_3 , the connecting trail over the first three rounds is generated in the following method using SAT.

1. Generate (α_2, β_2) such that $w^{rev}(\alpha_2) + k \times w(\beta_2 \rightarrow \alpha_3)$ is minimal, where $k = 3$ for SHAKE128 and $k = 1$ for SHA3-224 and SHA3-256, respectively.
2. For α_2 obtained in the previous step, generate (β_0, β_1) such that $(\beta_1 \rightarrow \alpha_2)$ is of the lowest weight (no active S-boxes has DDT = 2) and the number of active S-boxes in the first round with DDT = 2 is minimal.

Choice of $k = 3$ for SHAKE128. Satisfying a compatible transition into α_3 at (β_2, α_3) typically consumes more degrees of freedom than securing (β_1, α_2) , so the reverse-weight term on $\beta_2 \rightarrow \alpha_3$ should be weighted more heavily than that on $\beta_1 \rightarrow \alpha_2$. We therefore minimize $w^{rev}(\alpha_2) + k \cdot w(\beta_2 \rightarrow \alpha_3)$ with an integer k . In principle k could range up to about 11 by a crude “one constraint $\approx$ one S-box” count, but in practice we evaluated $k \in \{2, 3, \dots\}$ in the SAT model: $k = 2$ and $k = 3$ both yield sensible trails, with $k = 3$ strictly dominating $k = 2$ in connector quality, whereas increasing k beyond 3 quickly degrades solver performance because the number of variables and clauses grows. This choice of $k = 3$ is empirical, not theoretically optimal; we adopt it as the default for SHAKE128. Using this heuristic, we find more suitable connecting trails and better connectors.

3.2 New Techniques for Constructing Connectors on the First Two Rounds

Regarding constructing 2-round connectors, in [GLL⁺20], Guo *et al.* gave a technical analysis on choosing β_1 and pointed out that the choice of β_1 has a strong effect on the resulted DFs. In [GLST22], the authors attempted to search connecting trails of highest probability which may give more DFs based on a 2-round SAT model. Now we improve the algebraic-aided method adapted from previous works [DDS12, DDS14, QSLG17, SLG17, GLL⁺20] to construct connectors. Besides, we present another way to use SAT techniques to get a valid connector more efficiently along with a better connecting trail which is obtained following the new heuristic method.

Let $(\beta_0, \alpha_1, \beta_1, \alpha_2)$ denote the differential trails of the two rounds. Variables that represent a pair of messages (x_0^1, x_0^2) which are input values of χ_0 in the first round, and the input difference β_0 are introduced to describe the half-round propagation. Constraints implied by the capacity and padding are set by fixing the $c + p$ corresponding variables to be 0 or some settled value, but they are not directly added to the SAT model. Instead, these constraints on the initial state are transformed into the constraints on the input value of χ_0 in the first round. Constraints on the weight $w(\beta_0)$ of β_0 , namely the objective function are also added to make sure that $w(\beta_0)$ is low enough. The constraints that need to be covered in χ_1 in the second round are also taken into consideration. Rather than adding those constraints on the input value to χ_1 in the second round into the SAT model directly, they are transformed into constraints on the output value of χ_0 in the first round.

As a result, fewer variables are used and the connecting trails under a target weight are collected. Among all the collected β_0 , the trails with the smallest number of $\text{DDT}(\delta_{in}, \delta_{out}) = 2$ are chosen as a candidate to construct 2-round connectors. Afterward, the trail that yields the largest DFs returned by the 2-round connectors is picked up to play the final role of connecting trail. Thus, with the new SAT model, we obtain a good connecting trail in a shorter time, which could be used to construct a connector with a larger message space compared with [GLST22]. In other words, our new simpler model achieves higher DFs with higher efficiency. In Section 4 we will illustrate the construction of the 2-round connector at length.

3.3 A More Fine-grained Greedy Algorithm for Constructing Partial Connectors

In the attack on 6-round SHAKE128, 3-round connectors are constructed on top of a 2-round connector. However, full 3-round connectors can be hardly achieved or their solution set is too small. We then propose to construct partial 3-round connectors, where partial constraints of the third round are satisfied and their solution set is still large. The principle is to satisfy a subset of constraints that requires the least number of degrees of freedom. As there are too many possible cases, finding an optimal set is difficult. In this subsection, we propose a new

greedy algorithm that deals with constraints in a more fine-grained way. Note that this algorithm is also used to construct partial 2-round connectors for SHA3-224 and SHA3-256, but our following description is dedicated to partial 3-round connectors, for simplicity.

In the case of partial 3-round connectors, when a 2-round connector over the first two rounds succeeds, another 2-round connector starting from the second round to the third round is constructed where χ_1 in the second round is partially linearized. We then try to fulfill some of the equations for input bits of χ_2 corresponding to differential (β_2, α_3) (derived according to Property 1) greedily. First, the equations are converted to equations E_{x_2} on x_2 (the output of χ_1). We then pick up the equation which needs to linearize the smallest number of bits in α_2 iteratively. Since the 3-round connector is based on the first 2-round connector, we could maintain the equation system of the first 2-round connector, and just change the corresponding equations for linearizing χ_1 in the second round. As a result, its solutions will bypass the first two rounds and meet some difference of α_3 .

The new greedy algorithm helps obtain a larger dimension of message space in a faster way. As can be seen from Table 2, the new techniques for constructing partial 3-round connectors show great advantages over previous strategies. Concretely, after listing all χ_2 -constraints implied by (β_2, α_3) , we convert them to equations on x_2 , sort them by how many fresh nonlinear bits of χ_1 must be linearized, and insert them one-by-one while checking consistency—instead of committing to whole active S-boxes at once as in [GLST22]. This reduces wasted degrees of freedom when an S-box only partially overlaps prior equations. The new method could obtain a valid 3-round connector in an average time of 2^{-8} s while it took 0.8s on average in [GLST22]. This speed-up is *not* an apples-to-apples benchmark on identical connecting trails: on the same SHAKE128 instance we deliberately combine a different (better) connecting trail with the refined greedy routine, so both ingredients contribute. We nevertheless view the orders-of-magnitude gap as evidence that the new trail choice plus greedy updates dominate wall-clock, consistent with Table 2. More details that demonstrate the efficiency of our new techniques can be found in Section 4.

3.4 SAT Encoding of the *Keccak-f* Round Function

This subsection records how we encode the standard building blocks (round differential propagation, active-S-box counts, propagation weights, digest-bit patterns) into CryptoMiniSAT. The low-level CNF for Keccak-f is instance-specific, but this translation could transfer to other sponge/XOF settings with analogous output slicing. The CNF templates follow common Keccak cryptanalysis practice and tools such as Logic Friday for truth-table conversion; our *technical novelty* lies in how these blocks are combined with the search strategies in Section 3, not in the low-level clauses themselves. We give a concise introduction to automatic trail search tools based on SAT method, which describes the *Keccak-f* permutation and its related objective functions. We select CryptoMiniSAT, a solver introduced in [SNC09]. This choice was motivated by the solver’s exceptional performance

and its straightforward XOR expression interface. This feature allows us to focus on the challenges encountered in collision attacks on SHA-3. By applying different constraints to this tool, we harness its capabilities for searching colliding differential trails as well as connecting differential trails. Our tools are available via the [link](#)¹ here.

When implementing the SAT-based automatic trail search model, we introduce three distinct types of constraints into CryptoMiniSAT. Firstly, we incorporate conditions imposed by the differential propagation over round functions, which essentially describes the *Keccak-f* round functions using the conjunctive normal form (CNF). Secondly, we utilize objective functions, such as the number of active S-boxes and the propagation probability. The simple interface of CryptoMiniSAT facilitates the overall straightforward implementation, as we can easily specify and incorporate these constraints. At last, as for searching colliding trails, constraints are added to make sure the number of 1s in the output difference of the last round is as few as possible. For connecting trails, constraints are added to control the difference in capacity.

Round Function. The round function can be represented by the following model:

$$\alpha_r \xrightarrow{\theta} c_r \xrightarrow{\pi \circ \rho} \beta_r \xrightarrow{\chi} \alpha_{r+1}$$

In the SAT implementation for a single round, two state differences are introduced: α_r , which represents the input difference of the r -th round, and β_r , which represents the input difference of the χ operation in the r -th round. The 1600-bit difference α_r is represented by 1600 variables, with each bit indexed as $\alpha_r[i][j][k]$, where $0 \leq i, j < 5$ and $0 \leq k < 64$. This establishes a mapping relationship between the variables and the corresponding state difference. The three linear operations (θ , ρ , and π) are treated as a whole in SAT implementation.

The differential propagations over the linear operations ρ and π are described through mappings of variables index without additional SAT computations. When an active bit $c_r[i][j][k]$ is transformed to $\beta_r[i'][j'][k']$ through $\pi \circ \rho$. The θ operation updates each bit by XORed with two columns. This operation is described using XOR clauses that can be easily understood by CryptoMiniSAT. The XOR sums of 320 columns, denoted as $\alpha[i][k]$ are introduced as auxiliary variables. The nonlinear operation χ , generally considered as 5-bit Sboxes, is interpreted by using a truth table to describe the difference distribution table (DDT), which relates the input difference β_r to the output difference α_{r+1} , is depicted by writing CNFs for each Sbox. This truth table consists of 11 variables, including 10 variables representing input and output differences and 1 variable marking compatibility of DDT entries. The DDT is represented by using 46 CNFs generated by Logic Friday.

¹ These codes enable us to search for various types of differential trails, ranging from generic ones to more advanced ones, such as colliding differential trails, connecting differential trails, and differential trails suitable for near-collisions. Moreover, the search can be customized by specifying the desired weight for each round, allowing for a flexible and targeted approach to cryptanalysis.

Multiple rounds, each described in a similar fashion, are interconnected by linking the variables as follows: (1) the input variables of each round are the output variables of the previous round, and (2) the output variables of each round are the input variables of the next round.

Objective Function. In the context of collision attacks on SHA-3, two main objectives are considered: the number of active S-boxes and the propagation weight (referred to as *weight*). The sequential encoding method [Sin05] is utilized to handle transforming constraints on the number of active S-boxes and weights, into Conjunctive Normal Form (CNF). To be detailed, the constraint $\sum_{i=0}^{n-1} x_i \leq w$ can be represented by CNF, where $w \geq 1$. This encoding method introduces $(n \times (w + 1) - w)$ auxiliary variables.

Take the constraint on the propagation weight as an example. Entries in the DDT can take four possible values (2, 4, 8, and 32), corresponding to propagation weights within the set $\{0, 2, 3, 4\}$. Four auxiliary variables denoted as (p_0, p_1, p_2, p_3) are introduced to represent the weight of each S-box. This means that $(320 \times 4 \times (w + 1) - w)$ additional variables are added to describe constraints on the weight of the entire state. The weight constraint, obtained by summing up these variables, is then transformed into CNF using the sequential encoding method. The mapping of DDT entries to weight variables is illustrated in Equation 4.

$$(p_0, p_1, p_2, p_3) = \begin{cases} (1, 1, 1, 1), & \text{DDT}(\delta_{in}, \delta_{out}) = 2; \\ (0, 1, 1, 1), & \text{DDT}(\delta_{in}, \delta_{out}) = 4; \\ (0, 0, 1, 1), & \text{DDT}(\delta_{in}, \delta_{out}) = 8; \\ (0, 0, 0, 0), & \text{DDT}(\delta_{in}, \delta_{out}) = 32. \end{cases} \quad (4)$$

Constraints on digest bits. We parameterize a family of digest-difference patterns, which makes truncated collisions searchable at feasible complexity.

For the output difference of any S-box, it can take on values from the set $\{00000, 00001, 0001x, 001xxx, 01xxx, 1xxxx\}$. The space of the input differences of χ can be represented in CNF. Thereby full expressions of χ in this round could be omitted to enhance performance in some cases. Consider the case of 00001, the input difference must be one of the following values: 00001, 00101, 10101, 00011, 01011, 00111, 10111, 01111, 11111. By aggregating these cases, it becomes possible to represent any input difference state which could lead to a specific collision state, by transforming these values into CNFs.

For some other cases, we do not omit the expression of last χ . It is common to set a multiple of 32 as the constraint for the number of collision bits in the digest. In our near-collision attack scenario, we aim for a higher number of collision bits in the first plane. We can impose constraints like $\sum_{i=0}^{255} x_i \leq w$ to restrict the number of 1s in the first 4 lanes of the output difference state. The flexible combination settings of this tool enable it to perform various types of trail searching tasks.

4 Collision Attacks against SHA-3 Instances

In this section, we elaborate on each phase of the attack. As shown in Figure 2, following the framework introduced by Dinur *et al.* in [DDS12], our collision attack also consists of three phases, *i.e.*, colliding trail searching phase, connector construction phase, and brute-force searching phase.

4.1 Colliding Trail Searching Phase

In this phase, the target is to search n_{r_2} -round colliding trails with collision in digest bits. According to our observations, the number of active S-boxes of the input difference ΔS_I has a strong impact on the construction of connectors, so the number of active S-boxes in ΔS_I is expected to be as small as possible. Regarding the efficiency of SAT techniques, we model the 3-round differential trail of KECCAK, *i.e.*, $(\alpha_3, \beta_3, \beta_4, \beta_5)$, using SAT, where the objective is switched to find a high-probability differential trail satisfying two conditions: 1) contain as many 0s as possible in the first d -bit of ΔS_O , 2) the number of active S-boxes in ΔS_I is small.

The colliding differential trails in [GLST22] which are applied to conduct collision attacks contain exactly 256-bit 0s in the digest difference. Compared with that, we do not require the digest difference of the colliding differential trails to be all zero. Instead, we just focus on the colliding differential trails with 0s as many as possible in the digest difference, which will lead to near collisions or collision on a shorter digest. The relaxed restrictions make the adapted SAT-based tool perform more efficiently and better colliding trails which could be applied to mount collision attacks can be generated. Details of the differential trail used in our attacks are shown in Table 4, from which we could see the number of active S-boxes in the output difference of the second round is 9, indicating that we likely construct 3-round connectors based on this trail. In the meantime, the weight of this 3-round trail $(\beta_3, \beta_4, \beta_5)$ is around 50, such that mounting practical collision might be possible.

4.2 Connector Construction Phase

In this phase, n_{r_1} -round connecting trails are generated and connectors that generate a message space with sufficient degrees of freedom are constructed. First, the new techniques of SAT-based connector construction described in Section 3.2 are applied to determine (β_1, β_2) and β_0 ; then construct a valid 2-round connector followed by a 3-round connector, where a fraction of the conditions for differential (β_2, α_3) are fulfilled. We note that due to the lack of enough degrees of freedom, all the constraints of differential (β_2, α_3) can not be considered at the same time. For different SHA-3 instances, the specific steps would vary. Let us take the procedure on SHAKE128 as an illustration. In the case of SHAKE128, 2-round connectors are constructed for the first two rounds with full linearizations enforced on the first nonlinear layer χ_0 .

Algorithm 1: Constructing 2-round connector.

Input: α_2
Output: A system of linear equations on y_1

- 1 Build a SAT model to find (α_1, β_1) such that $\beta_1 \rightarrow \alpha_2$ is of minimal weight and α_1 has minimal number of active S-boxes;
- 2 List and denote linear equations on y_0 for $(c + p)$ -bit conditions of α_0 by E_{y_0} ;
- 3 List and denote equations on y_1 for conditions of differential (β_1, α_2) by E_{y_1} and convert them to equations on x_1 denoted by E_{x_1} ;
- 4 Build an SAT model that incorporates E_{x_1} and E_{y_0} and determines β_0 of high probability and of the smallest number of S-boxes with $\text{DDT}(\delta_{in}, \delta_{out}) = 2$;
- 5 List and add equations on y_0 for conditions of differential (β_0, α_1) to E_{y_0} ;
- 6 **do**
- 7 Fully linearize χ_0 and record the linearized mapping of χ_0 ;
- 8 Merge E_{y_0} with equations for linearizing χ_0 on y_0 and get E'_{y_0} ;
- 9 Convert E'_{y_0} to E'_{y_1} ; // According to the linearized mapping of χ_0
- 10 Add E_{y_1} to E'_{y_1} ;
- 11 **while** E'_{y_1} is not consistent;
- 12 **return** E'_{y_1}

2-round connectors. Given α_3 , we determine the connecting trail based on the SAT model using the techniques described in Section 3.2. Suppose α_2 is obtained. Algorithm 1 spells out the loop that merges sponge constraints, SAT-chosen (α_1, β_1) , and β_0 until χ_0 can be fully linearized and the resulting system on y_1 is consistent. At a high level we (i) collect E_{y_0} for capacity/padding and (β_0, α_1) , (ii) encode (β_1, α_2) as E_{y_1} / E_{x_1} , (iii) solve for a low-cost β_0 , and (iv) iterate linearizations of χ_0 until E_{y_0} lifts to compatible E'_{y_1} merged with E_{y_1} .

Partial 3-round connectors. Our partial 3-round connectors are constructed upon a 2-round connector where full linearizations are enforced on the first nonlinear layer χ_0 . After a 2-round connector for the first two rounds is obtained, we construct another 2-round connector for the second and the third round where the second round is partially linearized. Due to a great consumption of degree of freedom, the degrees of freedom returned by the first 2-round connector can not meet all the constraints imposed by the difference in the third round. Thus, only part of the conditions is selected to construct the 3-round connector according to a greedy algorithm. Algorithm 2 combines the same outer SAT steps as above with an inner loop that greedily picks an equation subset $E_{x_2}^p \subseteq E_{x_2}$ (Section 3.3), partially linearizes χ_1 only on the involved bits, and merges until E'_{y_1} stays consistent.

Until now, the system of equations E_{y_1} on y_1 is prepared, which means the 3-round connector is successfully constructed. The subspace of message pairs returned by E_{y_1} satisfies (1) the conditions imposed by sponge constructions, (2) the difference of the first two rounds, and (3) part of the output difference of the third round.

Algorithm 2: Constructing partial 3-round connector.

Input: α_3
Output: A system of linear equations on y_1

- 1 Build a SAT model to find (α_2, β_2) such that $w^{rev}(\alpha_2) + k \times w(\beta_2 \rightarrow \alpha_3)$ is minimal, where $k = 3$ for SHAKE128;
- 2 Build a SAT model to find (α_1, β_1) such that $\beta_1 \rightarrow \alpha_2$ is of minimal weight and α_1 has minimal number of active S-boxes;
- 3 List and denote linear equations on y_0 for $(c + p)$ -bit conditions of α_0 by E_{y_0} ;
- 4 List and denote equations on y_1 for conditions of differential (β_1, α_2) by E_{y_1} and convert them to equations on x_1 denoted by E_{x_1} ;
- 5 Build an SAT model that incorporates E_{x_1} and E_{y_0} and finds β_0 of high probability and of the smallest number of S-boxes with $DDT(\delta_{in}, \delta_{out}) = 2$;
- 6 List and add equations on y_0 for conditions of differential (β_0, α_1) to E_{y_0} ;
- 7 **do**
 - 8 Fully linearize χ_0 and record the linearized mapping of χ_0 ;
 - 9 Merge E_{y_0} with equations for linearizing χ_0 on y_0 and get E'_{y_0} ;
 - 10 Convert E'_{y_0} to E'_{y_1} ; // According to the linearized mapping of χ_0
 - 11 Add E_{y_1} to E'_{y_1} ;
 - 12 **if** E'_{y_1} is consistent **then**
 - 13 Pick equations $E_{x_2}^p$ out of E_{x_2} with a greedy algorithm;
 - 14 **do**
 - 15 List equations on y_1 to linearize χ_1 for bits at x_2 involved in $E_{x_2}^p$ and merge them with E'_{y_1} to get E''_{y_1} ;
 - 16 Record the partially linearized expression of χ_1 ;
 - 17 Convert $E_{x_2}^p$ to equations on y_1 and add to E''_{y_1} ;
 - 18 // According to the partially linearized expression of χ_1
 - 18 **while** E''_{y_1} is not consistent or more solutions are needed;
 - 19 **end**
 - 20 **while** E'_{y_1} is not consistent;
 - 21 **return** E''_{y_1}

In [SLG17], *adaptive* 3-round connectors were proposed, where the first 2-round connectors remained static but the second 2-round connectors were constructed repeatedly. This method is applied to generate multiple solution spaces in this paper due to that the dimension of the solution space returned by only one 3-round connector may be smaller than the weight of the following differential. It could help to find a real collision within the generated multiple solution spaces. We adopt this technique, as shown in line 18 of Algorithm 2, considering that the degrees of freedom returned by our connectors are usually smaller than the weight of the followed differential trails in our collision attacks against 6-round SHA-3 instances.

4.3 Brute-force Searching Phase

When the number of freedom degrees corresponding to the solution spaces returned by the connectors is relatively large, the connector construction phase is claimed successful, then we move on to the brute-force searching phase to search for a colliding message pair.

Early-abort technique. We follow the work in [GLST22] and use the early-abort technique. With the early-abort technique incorporated, the brute-force complexity 2^w could be reduced further by iteratively filtering out half of the message pairs at each step in the *brute-force search phase*.

Take a 6-round SHA-3 instance as an example. Suppose we could construct a full 2-round connector and a partial 3-round connector with only a fraction of constraints fulfilled on β_2 . Fixing one additional bit condition on β_2 during brute-force amounts to evaluating the six-round trail roughly once per surviving pair *and* halves the candidates on average; empirically calibrating these effects against one another yields an amortized cost of about $2^{-9.8}$ six-round evaluations per extra bit check under our GPU counting model (11 bits of α_2 participate in that layer). When checking all the 2^w message pairs with one-bit constraint, only half of the pairs satisfy the restriction while the other half is discarded, *i.e.*, the so-called *early-abort*. For the remaining message pairs, another bit constraint will be checked and half of those message pairs will be filtered out. This iterative process continues on the surviving message pairs until all the bit constraints on β_2 are checked. $1/2$ of the messages stop by first bit constraint, $1/4$ by the second bit constraint, $1/8$ by the third bit etc. Hence the time complexity would be $2^w \cdot 2^{-9.8} \cdot (1 \cdot 1/2 + 2 \cdot 1/4 + 3 \cdot 1/8 + \dots) \approx 2^{w-8.8}$ 6-round SHA-3 computations.

GPU Implementation. GPU (Graphics Processing Unit) is used to improve our computing capacity. We refine the basic GPU implementation of KECCAK from [SLG17, GLL⁺20] to speed up the brute-force search since the GPU implementation outperforms the existing CPU implementations. We adopt the strategy of one thread for one KECCAK and implement the brute-force search in CUDA. Moreover, we incorporate this CUDA implementation with the early-abort technique to benefit us in finding real collisions faster. Eventually, we do the search on an NVIDIA PCIe A100 graphic card. While on a single CPU core, it is expected

to run 2^{21} KECCAK- f evaluations per second [SLG17], this number could be $2^{30.5}$ per second on NVIDIA PCIe A100 graphic card. It can be seen that NVIDIA PCIe A100 can be roughly $2^{9.5}$ times faster than a CPU core.

Complexity. Searching for colliding trails takes several hours. It is negligible when compared to the time complexity of the latter two phases. In other words, the overall complexity of our collision attacks is mainly comprised of two parts, the complexity of the connector construction phase and the complexity of the brute-force search phase, while the latter one is more dominant in our attacks.

5 Experimental Results and Summary

Our collision attacks on the six-round targets use the same colliding trail core² as shown in Table 4 in the appendix, whose brief information is listed in Table 3 below. **Provenance.** This core is exactly the one Song *et al.* [SLG17] and Guo *et al.* [GLL⁺20] adopted for contest-scale 160-bit collisions; our SAT search re-derives it in minutes and confirms that it remains favorable once connector engineering is updated. The connecting trails and connectors (Tables 5–8 in the appendix), by contrast, are newly found in this work. Notably, the weight of the last round within the colliding trail differs depending on the digest length.

5.1 Collision Attack against 6-Round SHAKE128

Following the procedures in Section 4, a practical collision attack on 6-round SHAKE128 is mounted. This attack is based on partial 3-round connectors and a 3-round colliding trail. As the digest length d of SHAKE128 is variable, where the collision resistance level is $\min(d/2, 128)$ correspondingly, we have the flexibility to select an appropriate value for d such that a practical attack is possible within our computing capacity. As in the KECCAK contest [BDPV], we typically choose $d = 160$, which results in a weight of 75 for the last four rounds.

After obtaining a 2-round connector in 2^{-8} seconds based on the connecting trail in Table 5, the resulting solution space has a degree of freedom DF of 104, which is sufficiently larger than 75, the weight of the followed 4-round differential trail. This is already a collision attack against SHAKE128 with time complexity 2^{75} if we do not take any further actions.

Since the degree of freedom DFs returned by the 2-round connector is relatively sufficient, we try to cover as many constraints as possible in the third round to achieve partial 3-round connectors with the techniques of non-full S-box linearization in the second round.

² A trail core [BDPV11] is a sequence of intermediate differences. It denotes a set of differential trails whose input/output differences are restricted to a subset.

Complexity. After the 2-round connector is acquired, we satisfy 18 constraints on β_2 out of the 25 in total imposed by (β_2, α_3) to construct partial 3-round connectors. The partial 3-round connectors output a subspace of messages with DFs 30 averagely. The remaining 7 constraints are left to the brute-force searching phase. Therefore, about 2^{57} messages are needed from partial 3-round connectors. This number drops to $2^{54.81}$, where $2^{54.81} = 2^7 \cdot 2^{47.81}$, if we consider multiple trails leading to collisions on a 160-bit digest. In order to find an actual collision, it is expected to repeat the adaptive 3-round connector for $2^{24.81}$ times. In this case, a valid 3-round connector could be obtained in an average of 2^{-8} seconds, which is equivalent to 2^{15} 6-round KECCAK evaluations. Thus, repeating the adaptive 3-round connector takes $2^{39.81}$ 6-round KECCAK evaluations.

In the brute-force search phase, the time complexity is dominated by the time for testing $2^{54.81}$ messages, which is equivalent to $2^{54.81}$ 6-round SHAKE128 computations. After applying the early-abort technique, it would be reduced by a factor. However, this factor is smaller than $2^{8.8}$ according to our experiments. The possible reason might be that the estimation does not include the inherent relative loss in efficiency when using GPU, as also encountered in the practical attack on SHA-1 [SBK⁺17], or the bit constraints are satisfied with probability biased from 1/2. Therefore, applying the theoretic reduction factor $2^{8.8}$ in estimation may underestimate the actual complexity. In our analysis of the complexity, we adopt the conservative factor 2^7 stated below.

It is expected that the time complexity of attacking 6-round SHAKE128 is somewhere below $2^{54.81}$, which is practical. We run our CUDA implementation on NVIDIA A100 GPUs twice and obtain two instances of collisions in 41.5 and 9.5 core hours, which are equivalent to testing about $2^{50.11}$ and $2^{47.61}$ message pairs or $2^{49.69}$ and $2^{47.57}$ ³ 6-round evaluations, respectively. The second collision is found much earlier than the first one. Relative to the $2^{54.81}$ pairs that would be scanned without early abort, the two runs give reduction factors $2^{5.12}$ and $2^{7.24}$ when wall-clock is converted to six-round evaluations, or $2^{4.70}$ and $2^{7.20}$ when converted to message pairs tested. All four figures lie below the theoretical $2^{8.8}$; the geometric means are $2^{6.18}$ and $2^{5.95}$, respectively. For the seven leftover β_2 constraints of this attack the finite-sum model yields $2^{8.78}$, essentially the same as the infinite series. We therefore adopt the conservative factor 2^7 in the complexity tables (Table 13 and the SHA3-224/256 counterparts). We give one example of collisions in Table 9 (appendix). The first 160 bits of the two digests listed there agree; under the standard $d=256$ SHAKE128 output, the same pair differs in eight digest bits (lanes beyond the collision slice), as noted in Table 1.

The optimal trade-off between brute-force search and connector construction for our SHAKE128 attack (18 third-round β_2 constraints absorbed, total complexity $2^{47.82}$) is given in Supplementary Material, Table 13. With more constraints satisfied, connectors provide smaller DFs, thus the brute force complexity gets lower and the connector complexity gets higher.

³ 9.5 hours are equal to $2^{15.07}$ seconds. It is expected to run $2^{32.5}$ 6-round KECCAK evaluations per second. Thus, 9.5 hours mean $2^{47.57}$ 6-round KECCAK evaluations.

When $d = 256$. Through the application of our new methods on the same path as presented in [GLST22], we managed to reduce the complexity from $2^{123.5}$ to 2^{121} . The 2^{121} figure is obtained by re-running the accounting of Table 13 (with the same early-abort model) for the $d=256$ colliding trail: the colliding segment still needs to cover the full 256-bit digest, so Phase 3 remains near the 2^{123} – 2^{125} range; our connector-side savings buy roughly 2.5 bits in the total because we can now satisfy about three more third-round β_2 constraints than in [GLST22] before the connector dimension collapses, and each such constraint was estimated there to cost > 10 degrees of freedom. This improvement is therefore modest for *theoretical* attacks at full width $d=256$. For the practical $d=160$ attack, however, the same extra third-round constraints are essential: without them the first practical collision on six-round SHA-3 would be out of reach.

5.2 Practical Collision Attacks against 4- and 5-Round SHAKE256

The same pipeline yields practical collision attacks on four- and five-round SHAKE256 with the full 512-bit digest. Both attacks exploit a 2-round partial connector for the first two rounds, followed by a short colliding trail—two rounds for the four-round attack and three rounds for the five-round attack—ending in a full 512-bit digest collision. The explicit colliding pairs are listed in Table 11 and Table 12 (appendix).

2-round connectors. For the four-round attack, all 22 χ -compatible equations of (β_1, α_2) —the full weight of that transition—are absorbed, and six independent constructions return message spaces with $\text{DF} \in [150, 152]$ at about 0.1 s per linearization. For the five-round attack, the (β_1, α_2) transition is much heavier: 195 χ -compatible equations (the weight of that transition) are greedily absorbed, and logs of this setting (seed 4120575793) give connector dimensions $\text{DF} \in [9, 16]$ with average 12 at about 0.14 s per linearization.

Complexity. In the four-round attack, the remaining colliding-trail weight is 18: 15 from the third-round transition (β_2, α_3) and 3 from the digest condition of the last round, so a single trail collides with probability 2^{-18} . As in Section 5.1, multiple trails lead to collisions on the 512-bit digest, and the measured hit rate is 2^{-16} per pair, four times the single-trail probability. Since $\text{DF} \approx 150 \gg 16$, one connector already contains enough candidates and is never rebuilt: the one-time connector construction dominates the wall-clock, while the GPU search itself succeeds within seconds. We repeated the four-round attack six times with independent connectors; the first collision appears after $2^{13.70}$ – $2^{16.31}$ pairs (geometric mean $2^{15.5}$), and each batch of 2^{22} pairs yields 59–74 collisions, consistent with the 2^{-16} rate. This improves on the previous best 4-round attack of complexity 2^{76} [ZHL23]. In the five-round attack, the remaining colliding-trail weight is 36 ($w(\beta_2) + w(\beta_3) = 21 + 15$), so about 2^{36} pairs must be tested; since $\text{DF} \approx 12 < 36$, the 2-round connector is rebuilt about 2^{24} times and Phase 3 dominates. This improves on the previous best 5-round attack of complexity $2^{163.28}$ [ZHL24] and is practical on a single GPU.

5.3 Near Collision Attack against 6-Round SHA-3-224

The capacity of SHA3-224 is much larger than that of SHAKE128, which means about 190 more freedom degrees are needed. As a result, our collision attack against 6-round SHA3-224 can only exploit a 2-round connector for the first two rounds which is followed by a 4-round colliding trail. The 4-round trail core is the same as in the attack on SHAKE128 which can be found in Table 4. However, we translate this trail core by 30 bits to the MBS, *i.e.*, rotate each lane of the trail core to the left by 30 bits, so that it will lead to a 224-bit collision with 4 bits difference with probability 2^{-75} . The 2-round connecting trail is presented in Table 6.

2-round connectors. Due to the relatively large capacity, a large number of degrees of freedom would be consumed in the connector construction phase. In this case, non-full linearization techniques proposed in [SLG17] are expected to be applied on the S-boxes of the first round to generate a larger message subspace when 2-round connectors are constructed. If we select 270 constraints out of the 292 equations on β_1 imposed by (β_1, α_2) , the 2-round connectors of the first two rounds will succeed within a second and return a message space with DFs 33 on average.

Complexity. The remaining 22 out of the equations on β_1 imposed by (β_1, α_2) are left to the brute-force searching phase together with the latter 4-round differential trail whose weight is 75. To find one colliding pair, at least 2^{97} pairs of messages are required, which is larger than the message space returned by the 3-round connector. This could be achieved by repeating constructing the adaptive 3-round connector for 2^{64} times, each of which takes 0.62 seconds on average. After considering the early-abort technique, the brute-force complexity could be reduced to 2^{90} . Thus the total time complexity for this case is $2^{90.02}$. The optimal choice (270 β_1 constraints in the connector) and the full trade-off over β_1 constraints are given in Supplementary Material, Table 14.

5.4 Near Collision Attack against 6-Round SHA3-256

The capacity of SHA3-256 is larger than that of SHA3-224, requiring even more degrees of freedom. Still, our collision attack against 6-round SHA3-256 exploits a 2-round connector for the first two rounds which is followed by a 4-round colliding trail. The 4-round trail core is the same as in the attack on SHAKE128 in Table 4 and leads to a 256-bit collision with 6 difference bits. The probability of the colliding trail is 2^{-79} . The 2-round connecting trail is presented in Table 7. Even though this colliding trail does not have the lowest weight for the second round (300 instead of the minimal 292), it leads to good connectors.

2-round connectors. When we select 255 constraints out of the 300 equations on β_1 imposed by (β_1, α_2) , the 2-round connectors of the first two rounds will succeed within a second and return a message space with DFs 26 on average.

Complexity. The remaining 45 out of the equations on β_1 imposed by (β_1, α_2) are left to the brute-force searching phase together with the latter 4-round differential trail whose weight is 79. In order to find one colliding pair, at least 2^{124} pairs of messages are required, which is larger than the message space returned by the 3-round connector. This could be achieved by repeating constructing the adaptive 3-round connector for 2^{98} times, each of which takes 0.28 seconds on average. After considering the early-abort technique, the brute-force complexity could be reduced to 2^{117} . Thus the total time complexity for this case is $2^{117.85}$. The optimal choice (255 β_1 constraints in the connector) and the full trade-off are given in Supplementary Material, Table 15.

Generic cost of the near-collision targets. For an n -bit ideal hash, a pair at Hamming distance k at *fixed* digest positions is found generically in about $2^{(n-k)/2}$ evaluations, by the birthday argument on the remaining $n-k$ bits. (Allowing the k differences at *arbitrary* positions is a different, easier target, costing roughly $2^{(n-k-\log_2 \binom{n}{k})/2}$.) Our SHA3-224 ($n=224, k=4$) and SHA3-256 ($n=256, k=6$) near-collisions are of the fixed-position kind, whose generic costs are 2^{110} and 2^{125} ; our complexities $2^{90.02}$ and $2^{117.85}$ stay below these bounds.

5.5 Improved Collision Attack against Keccak[1440, 160, 6, 160]

To further demonstrate the strength of our new strategy, we compare the experimental results on KECCAK[1440, 160, 6, 160] with the previous work [GLL⁺20]. The results show that the time complexities are reduced significantly from the previous best results. Both attacks on KECCAK[1440, 160, 6, 160] exploit 3-round connectors to the first three rounds where a 2-round connector for the first two rounds is constructed first. In [GLL⁺20], based on the obtained 2-round connector, adaptive 3-round connectors were constructed for $2^{2.81} \sim 2^{7.81}$ times while our work used only one 3-round connector. The 2-round connector of the first two rounds in [GLL⁺20] succeeded in 4.5 core hours and returned a message space with 174 degrees of freedom, while our new 2-round connector could be obtained in several seconds and DFs of the returned message space is 205. Their 3-round connector could output a subspace of messages with $\text{DF} \in [40, 45]$, while the message space returned by our 3-round connector has a dimension of 68, which is sufficiently larger than the weight of the followed 3-round differential trail. In brief, our techniques not only help to generate connectors faster but also increase the degrees of freedom of the solution space.

In the case of KECCAK[1440, 160, 6, 160], the time complexity is dominated by brute-force searching. Our experiment shows the real time to find a colliding pair for KECCAK[1440, 160, 6, 160] is 0.75 core hour on an NVIDIA A100 GPU.

5.6 Comparison and Discussion

The comparison of our collision attacks with the previous best attacks is presented in Table 2, including digest length d where it differs across rows. Notably, we give

the first practical collision attack on 6-round SHAKE128 at $d=160$. Furthermore, we introduce near-collision attacks on six rounds for both SHA3-224 and SHA3-256. For the first time, practical collision attacks on SHA-3 extend to six rounds. The same methods also yield practical full 512-bit digest collisions on four- and five-round SHAKE256 (Section 5.2).

Fair comparison with Guo et al. [GLST22] at $d=160$. A natural question is what complexity [GLST22] would attain if the digest length were $d=160$ instead of $d=256$. They published a SHAKE128 attack with a fixed *full* 256-bit zero digest difference and reported $2^{123.5}$ six-round evaluations (Table 3 of [GLST22]). We distinguish two counterfactuals; both are estimates, not experiments. (i) *Same differential trail, shorter verification.* If one merely checks collision on the first 160 digest bits while keeping their 256-bit colliding trail (Trail No. 3 in their appendix), the connector accounting is unchanged and the total cost stays at their reported $2^{123.5}$. (ii) *Shorter trail, their connector strategy.* A lighter $d=160$ colliding trail ($(\beta_3, \beta_4, \beta_5)$ weights would be around 44 rather than 52) would lower w_{col} only by a few bits; with their connector model the total cost plausibly remains $\geq 2^{115.5}$ —an extrapolation, not an experimental proof—well above the 2^{80} birthday scale at $d=160$ and far from our $2^{47.82}$. Our practical gain at $d=160$ therefore comes from *joint* improvements to trail search, parameterized digest encodings, and connector heuristics (Sections 3–4). Re-running the [GLST22] or Guo et al. [GLL⁺20] pipelines on our hardware would improve wall-clock times *only* by software and constant-factor gain. The speedups in Table 2 come from *smaller exponents* (better connectors and trails).

6 Conclusion

We advance practical SHA-3 collision cryptanalysis within the Dinur connector framework by raising the degrees of freedom returned by partial connectors. On the colliding side, a parameterized SAT model chooses digest length d and difference templates together with lean β/α encodings. On the connector side, connecting trails are ranked by $w(\beta_1) + k w(\beta_2)$ with costly DDT=2 transitions penalized; partial connectors then absorb third-round (or second-round) χ -compatible equations via a bit-granular greedy rule and, where capacity demands, non-full S-box linearization, rather than locking whole S-boxes at once. With these tools we obtain the first classical practical 6-round collision on NIST-standardized SHAKE128 at $d=160$ (total cost $2^{47.82}$ six-round evaluations), 6-round near-collisions on SHA3-224 and SHA3-256, a markedly faster 6-round attack on contest instance KECCAK[1440, 160, 6, 160], and practical full-digest collisions on four- and five-round SHAKE256. We will release our implementation upon acceptance.

Limitations and outlook. In our attacks, connectors stay within three linearized rounds; extending to seven rounds (e.g. a 3+4 connector/colliding split) appears unrealistic because collision trails carry far more propagation weight than minimal

Table 2: Comparison of our collision attacks with previous best attacks. **SHAKE128** rows state digest length d explicitly. “Comp.” is $\log_2$ six-round evaluation cost unless a wall-clock entry is marked.

Target $[r, c, d]$	Rounds	Type	T_c	DF	Comp.	Ref.
SHAKE128	6	Full ($d=256$)	0.8s	27	$2^{123.5}$	[GLST22]
	6	Near ($d=256$) [§]	$2^{-8}s$	30	$2^{47.82}$	Table 9
	6	Full ($d=160$)	$2^{-8}s$	30	$2^{47.82}$	Table 9
	6	Full ($d=160$)	0.8s	27	$\geq 2^{115.5\dagger}$	est. [GLST22]
SHA3-256	5	Full	428.8h	37	45.6h	[GLL ⁺ 20]
	6	Near	0.28s	26	$2^{117.85}$	Sect. 5.4
SHA3-224	5	Full	11.7h	83	29h	[GLL ⁺ 20]
	6	Near	0.62s	33	$2^{90.02}$	Sect. 5.3
KECCAK[1440,160,160]	6	Full	4.5h	45	112h*	[GLL ⁺ 20]
	6	Full	4s	68	0.75h	Table 10

^{Type} indicates whether it is a near-collision attack or a full-collision attack.

T_c denotes the time consumed by the connector construction phase.

* The actual time using three NVIDIA GeForce GTX970 GPUs.

[§] Same message pair as the $d=160$: exact collision on the first 160 digest bits and an 8-bit near-collision (Sect. 5.1).

[†] Conservative extrapolation of [GLST22]’s accounting if Phase 1 targets $d=160$ but their connector strategy is unchanged.

ones (weight 52 vs. 32 on three rounds; Liu *et al.* [LQT19] bound four rounds at ≥ 56 , with best known 133).

Acknowledgement

We used Gemini to assist in writing: correcting grammar, improving clarity, and refining the flow of sentences. The LLMs do not contribute to research ideation, methodology, experimental design, data analysis, interpretation of results, or the creation of substantive academic content or references.

References

- BDPV. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak Crunchy Crypto Collision and Pre-image Contest. http://keccak.noekeon.org/crunchy_contest.html.
- BDPV11. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak Reference. <http://keccak.noekeon.org>, January 2011. Version 3.0.
- BDPVA07. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.

- BDPVA13. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 313–314. Springer, 2013.
- BNR21. Rachelle Heim Boissier, Camille Noûs, and Yann Rotella. Algebraic collision attacks on Keccak. *IACR Trans. Symmetric Cryptol.*, 2021(1):239–268, 2021.
- DDS12. Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on Keccak-224 and Keccak-256. In *International Workshop on Fast Software Encryption*, pages 442–461. Springer, 2012.
- DDS13. Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In *International Workshop on Fast Software Encryption*, pages 219–240. Springer, 2013.
- DDS14. Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced Keccak. *Journal of cryptology*, 27(2):183–209, 2014.
- DVA12. Joan Daemen and Gilles Van Assche. Differential propagation analysis of keccak. In *International Workshop on Fast Software Encryption*, pages 422–441. Springer, 2012.
- Dwo15. Morris J Dworkin. SHA-3 standard: Permutation-based hash and extendable-output functions. 2015.
- GJMG11. Bertoni Guido, Daemen Joan, P Michaël, and VA Gilles. Cryptographic sponge functions, 2011.
- GLL⁺20. Jian Guo, Guohong Liao, Guozhen Liu, Meicheng Liu, Kexin Qiao, and Ling Song. Practical collision attacks against round-reduced SHA-3. *Journal of Cryptology*, 33(1):228–270, 2020.
- GLST22. Jian Guo, Guozhen Liu, Ling Song, and Yi Tu. Exploring SAT for cryptanalysis: (quantum) collision attacks against 6-round SHA-3. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 645–674. Springer, 2022.
- HBDM22. Senyang Huang, Orna Agmon Ben-Yehuda, Orr Dunkelman, and Alexander Maximov. Finding collisions against 4-round SHA-3-384 in practical time. *IACR Trans. Symmetric Cryptol.*, 2022(3):239–270, 2022.
- LQT19. Guozhen Liu, Weidong Qiu, and Yi Tu. New techniques for searching differential trails in keccak. *IACR Transactions on Symmetric Cryptology*, pages 407–437, 2019.
- QSLG17. Kexin Qiao, Ling Song, Meicheng Liu, and Jian Guo. New collision attacks on round-reduced Keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 216–243. Springer, 2017.
- SBK⁺17. Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596. Springer, 2017.
- Sin05. Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

- SLG17. Ling Song, Guohong Liao, and Jian Guo. Non-full Sbox linearization: Applications to collision attacks on round-reduced Keccak. In *Annual International Cryptology Conference*, pages 428–451. Springer, Cham, 2017.
- SNC09. Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT Solvers to Cryptographic Problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009. available via <https://github.com/msoos/cryptominisat>.
- SSJ23. Sahiba Suryawanshi, Dhiman Saha, and Shashwat Jaiswal. TIDAL: Practical collisions on state-reduced Keccak variants. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part I*, volume 13905 of *Lecture Notes in Computer Science*, pages 208–233. Springer, 2023.
- ZHL23. 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.
- ZHL24. Zhongyi Zhang, Chengan Hou, and Meicheng Liu. Probabilistic linearization: Internal differential collisions in up to 6 rounds of SHA-3. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part IV*, *Lecture Notes in Computer Science*, pages 241–272. Springer, 2024.

Supplementary Material

A Trails and Messages

In this section, details of differential trails of *Keccak-f* permutation and the colliding message pairs are provided.

The 1600-bit state is displayed as a 5×5 array, ordered from left to right, where ‘|’ acts as the separator; each lane is denoted in hexadecimal using little-endian format; ‘0’ is replaced with ‘-’ for differential trails.

Table 3: Brief information of the colliding trails sharing the same trail core in Table 4

$\#AS(\alpha_2-\beta_2-\beta_3-\beta_4-\beta_5^d)$	$w_1-w_2-w_3-w_4$	w_5^d	d
127-9-8-8-10	292-25-18-16	16	160
		16	224
		20	256

A.1 Differential Trails

In this section, a differential trail of 6-round collision attacks is listed.

A.2 Instances of Solutions of Connectors

Instances of collisions against 6-round **SHAKE128** and **KECCAK**[1440, 160, 6, 160] are given as below respectively. Colliding messages are denoted as M_1, M_2 .

B Comparison with Guo *et al.* [GLST22]

This section expands the comparison summarized in Section 1 between our work and the SAT-based collision framework of Guo *et al.* [GLST22]. We both adopt the same Dinur-framework pipeline [DDS12], but our SAT tooling targets a broader class of output-collision goals and yields a classical practical 6-round attack at feasible d . Changing d , truncating the digest slice, or allowing structured nonzero differences is outside their search interface; a new target requires instance-specific hand encoding of the β_5 constraints. We also slim the trail and connector encodings (fewer intermediate states and sparser propagation clauses). The following upgrades distinguish our work. The first two are *engineering generalizations* of the SAT interface rather than new cryptographic principles; the last two carry the algorithmic content.

Table 4: The 4-round trail core used in the 6-round collision attack on SHAKE128, SHA3-224, and SHA3-256, respectively

β_2	----- ---1----- ---1-----8--- ----- -----8--- -----4-1- -----4--- -----1- -----4-1- -----4--- ----- ---1----- ----- ----- ---1-----8--- -----4--- ----- ---1----- -----4-1- ----- ----- ----- ----- ----- ----- -----	2^{-25}
β_3	----- ----- ----- -----2--- ----- ----- ----- ----- -----2--- ----- ----- ----- ----- ----- ----- ----- -----4----- ---4----- ---4----- ---8----- ----- -----4----- ---8----- ---4----- ---8----- -----	2^{-18}
β_4	----- ----- ----- -----1--- ----- --2----- ----- ----- ----- ---8----- ----- ----- ----- ----- 1----- ----- ----- ----- ----- ---2--- ---8----- ----- ----- ----- ----- ---2--- ---8----- ----- -----1--- ----- -----8--- -----	2^{-16}
β_5	1----- -----2----- 9--2--2----- -----4-4--- -----24--8--18 --1---2----- -----9--2--68- 8----- -----4----- ---8-----24--8 -4-----2--- 1-----48-1-- -----4----- 68-----9--2--- -----4--- ---48-1---3--- -----1-2----- 8----- -----9--2--2- ---2-----8-- --1-----48-1- --1-1----- 48-1--83----- -----2----- ---8-----4---	

Table 5: Connecting trail for SHAKE128

β_0	1000000B04400022 9AE8041010440000 8420802201050221 51D820420900018B 00A9802C01048020 3144002200410021 0A40002293480200 13880005008002A2 40D0A87180440008 162981140808A384 08B1800001180101 431C801502840100 1A490200009F0223 4000A46310402189 9641002519148080 88A0C00200340141 535808B399810361 0C22A0040AE50042 012008408010A1C1 1B51080408E420C0 0200082002610121 4D6C08151268818A C2E0800001890280 A2982C4202458128 50694024091C804B
β_1	8040000A04800000 4030000002C00000 500000420030000A 0010001401200005 3000008400100000 8040000804300000 1030004202400000 400000000080000A 0010011601900005 3080008400200000 8040000A04100002 5020000002400000 4010004000B00008 1000011201A00005 2090008400000000 9050000C04B00000 0000002202400000 5010004400000002 0020011201300005 60A0008400000000 A040000A00300000 1030000402C00000 2000004400000008 6000011201900005 5090008000800002

Table 6: Connecting trail for SHA3-224

β_0	036A603043943006 048A200891B01704 023520F2A2E0280A 424F400108820001 8102003AAD880000 046B401A90032004 240100408828240C 041028D8E9221801 2446280126920002 C688480140900002 006BA80B26962107 02176881C30E150D 0206A81865920103 02040800EAB62000 068680D90834090F 0204005AF082B400 02F360D92F1A240E 42A82840C982A008 6663080066289A03 4410000A127A0008 00120092F8201507 445120818CA21A04 0461400A8E926A0B 0031601942021506 067560D3401A4306
β_1	0104000030140002 00B0000004080000 8008000290000010 006C000140040005 800000001C000021 0108000024100002 80B00000000C0010 0004000294000000 0060000140000045 800C000008240021 81080000A0100002 00900000000C0000 8024000204000010 0060000150040044 000C00001C200021 812C000024180003 0090000000000008 0000000900C0011 804C000144000044 00000000182C0021 8008000038140002 00B0000004080000 8004000218040010 0060000148000045 0004000094200021

Table 7: Connecting trail for SHA3-256

β_0	94381C1F9180A0C1 8700E43C95058081 D738100907642022 71109015149C0080 FC300C02079820E8 C26C9C230B3D80CB C97C48019A18828E 8B689C2D88290008 60249C3C8800800F 40184832810C8046 48D8743A1307C0CB 6E78A8198923C0C1 48995408103180E9 A3F0C030081E6083 4F41C82B197D802B E6BAE43B829D8003 ADF8BC18932D0202 6FF2F0090B088049 20CA043201190061 A4E00403888480A8 5D086C1B010180E5 1960B81812358023 4D701808023100AB 5F20CC11000400A1 9048880A1B3520AF
β_1	8050000804900000 5020004202C00000 400000440020000A 0010001001300005 3000008600000000 9000000804300000 0030004202C00008 404000040010000A 1050011201800005 2080008400200008 9050000804200002 0020000202600000 0000004000900008 5010011001A00005 6080008600100000 9060000605B00000 0010002002400000 4030004C01000002 3010011600300005 60A0008C00000000 A050000800300000 1020000602400000 2010004400900008 5000011001800005 4080008200000002

Table 8: Connecting trail for KECCAK[1440, 160, 6, 160]

β_0	0808000805880000 1080DC0013A4006B 4548203808250300 0850000D01000068 80008002001480E9 007C801400210045 1CC0102011900002 8008E01400280224 0C00800C80040048 0600001309908127 C600000000240000 1838D02000040240 84000010006B0222 0058840C00B00008 009820101910A141 1140801001940040 1018D02011210261 0442001002950003 C010A80C81108000 0AC0001108042100 1898041401A00021 0409580111980100 8080A03400210362 E841040800100008 028900100A248060
β_1	C050000804900000 0020000202C00000 500000400020000A 0010001401300005 7000008600000000 9040000A04200000 0030004002500000 500000000080000A 0000011601800005 2090008400300000 8040000A04300002 1030000002400000 0000004200900008 4010011001A00005 7080008400000000 D070000E04B00000 0000002002400000 4030004400000002 1000011201300005 60A0008400000000 F050000A00200000 0020000002C00000 4010004400100008 1000011201800005 6080008400900002

Table 9: Collision for 6-round SHAKE128

M_1	CB69A458BF6D63F8 B0199B1EBC9B2638 D3C76C1036C5C9B6 FF2A4AB022607107 9F5E389A21C41FAC E609C811344FD1EF 6F0D5C9594DF5245 ECEEA3D2FBD29AC9 84153F78BDD76D30 744A08B4C9B0028B 3D09A009C3279F2D 6993366365845637 B33258995B550F25 F84B82DA7D9E9634 0D5B5EB76A5C1D79 BCC7D13A8A1AB3E0 CF1DFE90A36CBC04 278267036983A7FE F0A261FABE1AED3C A93C18ADFD6846C2 FE272056BC05A4E 0000000000000000 0000000000000000 0000000000000000 0000000000000000
digest	31063331CCBA3AB0 E4BE613F3596EAB8 F7F9F74B4C660429 217301B2574935AF
M_2	F82703F7CE28E804 E05B0B17BE5006AA 6A8B44307F04D112 AA324CABD200B33C 995E120D356C1BFE FCDF7F83547FDABA FA1608DC9691DCD7 5CAED972EF9D96ED 0D0D115488F4B9B1 F4EA0AB6C9022FAD 5C3607AD163214A7 AD0A6EC266C1CF65 063E70995F209715 E6C2975FBD3F562F 0DD84E146B69D86B BE88545E6B4E7428 588BACB9A1699A93 D54C6DE33947BBFF A1A63F7C9B383D6F 20FC1AAF9EC802F1 FD49A2623ECC97D0 0000000000000000 0000000000000000 0000000000000000 0000000000000000
digest	31063331CCBA3AB0 E4BE613F3596EAB8 67FBF7494C660429 217301B2534535BF

Table 10: Collision for the contest instance KECCAK[1440, 160, 6, 160]

M_1	68B66A5DBB705C39 2B72F0CA57AF1C66 6E0BF8F6DBF9EB79 9DA083A74A6B6BEE BD75FD77A83B2828 053D288F73B714EE CEA0AAF9E5BD966F C3F574C9D58978AE 7C2097C000BAD785 7BD5B160EC799F72 C26E40F236A1041D 807FC946F874F438 A175E2A6E24ADA4D 06B62A11B04FD35C 823FD1B215CD4DD4 34FFFFF2A1C2232E8 A4B84653D69F753B 318AA0C2DF0FBAA3 33D109CD55FA3026 887C593436609C5C 6E5D370114FED3DA 89B67550DC9D04B4 00000000C3E6B32F 0000000000000000 0000000000000000
M_2	C6B705E7FDE5C730 913034510450E59F 3C6BE83EE36A3DD5 5FB096FF49826AA4 8C53F57780232A7A A226571B21AB02E5 D7E35422B0E1679B F39597116B9ABE8F AE9084EB3499D787 787390AEED739E39 F4663342F5B99F50 59FC1D5DAF99A881 9674E69EBC51A561 F3372A5BB46EDB56 E99993B6D0C44CC6 96F698985B4B8963 7D9E82E88732ECC6 018B207BB1056EAA F3D25887D1DB5824 83DA01B23A682C0E E00418955FE74E97 5056F1CF8DF9DD55 00000000B1716C3F 0000000000000000 0000000000000000
digest	D9678CCAE4F9BCC4 CA9B404ECACC51E0 25A59EC5

Table 11: Collision for 4-round SHAKE256 with the full 512-bit digest. The two messages agree on lanes 0–7 of the state after four rounds, *i.e.*, on all 512 digest bits.

M_1	42E01CODEE55F48B 0F711FCF0511D8CD D9A9FD99EF5C9226 5CCFCF98E5EE1F4B 1112CAC3F170BDA7 2BE64E635F62AD64 COD3525BA6C5607C A78A28491A3537D8 B70CDBA660353B7A 6E7C3E277448145E 4C493214112873D7 18551CEFC9244EB3 E654DEED299075C2 6E3C616FE25D5D50 7C6F19BE0565FA30 4799147840D416B4 FD7D956926E27DD5
M_2	C2E08B13891F63EC AF715A714889D8C3 D9A151F6585C9313 D6CE7FE8E5E72D4C F952CF5757A6665C EC8CE6293C838FD 371BE43986396070 EE82A90FAEC53618 B7DC09D3603511FA 6E3DAA23D5506832 14499F3B237AE48F 7B85549BFD6F06BA 79F0E2ED5240740B 7E3DD70FE27AE136 FF3C59BABEBB5AF9 95BBC4780A1E8162 FCCC060B05907DDD
digest	4431FFDBE5AACE18 CFB064BA3A1B8F48 8BF8CB8EB48FC443 688EBAD9672F9B7E C7958127E336F0EC D158C686A609995D 9C131CBBD23CD92C 5BCCEDCC3D0D0748

Table 12: Collision for 5-round SHAKE256 with the full 512-bit digest. The two messages agree on lanes 0–7 of the state after five rounds, *i.e.*, on all 512 digest bits.

M_1	EA19DB869D32D722 84464BE751C313D0 E8D99D3C1B5E2822 496654A1429A840C D5091A3E674526C4 DAE61172E6DBE348 4552166F66627C4F C33A5227A665E90 16A940647B8F1059 8759387740A44891 3E109302EFC1A229 5313DF39B03A80FA 5F6632D14C31482B 54A427E100FAA781 42AF4599D6A67137 E38766A168E21B73 FF18F76D7BBB636B
M_2	12B11724F9035B75 6CA09AA5D9CA58B4 E1B33E090A92D4C5 0B14EC01713EFF2B 578991301349403B AD7616F0FAE3FD36 079BE74F54F3E645 DBB580C307AA8E92 554E99BCF8DE67B2 067480464B87EF08 B6958D0E0EC9851E 8C68A23806101ED0 027219A9BF943E7B E5A6104A8B9BE5A0 674F6128E7CB253B 7C2158F5E4B9911F FE6226C9402861B4
digest	4CE6E61C440A7A08 3A3E90A28A81292D FE741EF1A1C5ED30 8D81938239B53B88 0FB2CE07B280CC90 21CA7B71237D6F26 0A254BDF71B70A81 FA90A112AD1A551C

- **Fixed vs. configurable digest constraints (engineering).** In [GLST22], the colliding-trail SAT instance is tied to a *single* digest pattern per SHA-3 variant: typically a full collision with zero difference on every digest bit at the instance’s nominal length (e.g. all 256 bits in their SHAKE128 analysis). Our colliding-phase model is *parameterized at setup*: the analyst chooses an effective digest length d , which output lanes or bit ranges must vanish, and Hamming-weight or mask bounds on the remaining differences, while reusing the same CNF backbone. One implementation therefore searches truncated collisions ($d=160$ in our practical attack), near-collisions on SHA3-224/256, and full- d collisions without rebuilding the solver input for each instance.
- **Digest-side encoding (engineering).** As in [GLST22], every constraint on the collision pattern at α_6 is expressed on β_5 (the input difference to the last χ), rather than by duplicating a full 1600-bit α_6 together with the $\beta_5 \rightarrow \alpha_6$ differential layer. Our implementation materializes only the part of β_5 that the digest template actually touches: in the SHAKE128 attack at $d=160$, just **320** SAT variables (the 64 S-boxes in the digest slice, five bits each) suffice to encode the admissible β_5 patterns, instead of carrying the entire β_5 state. When advantageous, the same digest conditions can instead be pulled back through λ^{-1} to affine constraints on α_5 , in which case an explicit β_5 layer is omitted altogether.

- **Colliding trails.** We jointly optimize d , the digest-difference template, and the propagation-weight budget of the outer rounds, instead of forcing all 256 digest bits to vanish. This yields feasible SAT instances and real collisions on truncated SHAKE128 at $d=160$, beyond what their fixed full-digest encoding achieves classically.
- **Connecting trails and connectors.** When searching connecting trails for $(\beta_0, \beta_1, \beta_2)$, we minimize the weighted propagation cost $w(\beta_1) + k w(\beta_2)$ with $k > 1$ for SHAKE128, and *also* penalize transitions with DDT=2, which are known to consume degrees of freedom heavily during later linearization. After a trail is fixed, the induced χ -compatible conditions along the path are expanded into a pool of candidate linear equations; a greedy routine then selects which equations to enforce so as to maximize the connector dimension (DF), rather than committing to whole S-boxes at once as in [GLST22].

For a complexity-oriented discussion when extrapolating [GLST22] to $d=160$, see Paragraph 5.6 in the main text.

C Connector constraint–complexity trade-offs

This section lists the connector constraint–complexity trade-off tables referenced in the main text (Section 4). Each table varies how many connector-phase equations are absorbed before brute force; the optimal rows used in the attacks are highlighted in bold.

C.1 SHAKE128 at $d=160$

Table 13: Complexity trade-offs for SHAKE128 ($d=160$) as a function of how many third-round β_2 constraints are absorbed inside the connector. “Data complexity” is $\log_2$ message pairs drawn from (repeated) partial connectors; “Connector” and “Brute force” are $\log_2$ cost of those phases in six-round SHAKE128 evaluations (Section 4.3); early-abort uses the experimentally rectified 2^7 factor.

#constraints	DF of Connector	Data Complexity	Connector Complexity	Brute Force Complexity	Total Complexity
15	50	57.81	22.81	50.81	50.81
16	43	56.81	28.81	49.81	49.81
17	36	55.81	35.81	48.81	48.81
18	30	54.81	39.81	47.81	47.82
19	21	53.81	47.81	46.81	48.39
20	12	52.81	55.81	45.81	55.81
21	5	51.81	61.81	44.81	61.81
22	2	50.81	63.81	43.81	63.81

Table 14: Summary of complexity estimations for **SHA3-224** corresponding to the number of constraints solved by taking the reduction factor as 2^7 .

#constraints	DF of Connector	Data Complexity	Connector Complexity	Brute Force Complexity	Total Complexity
240	117	127.00	28.33	120.00	120.00
260	62	107.00	63.96	100.00	100.00
265	46	102.00	75.47	95.00	95.00
267	42	100.00	77.73	93.00	93.00
269	35	98.00	83.19	91.00	91.01
270	33	97.00	83.82	90.00	90.02
271	27	96.00	89.99	89.00	90.58
272	22	95.00	93.79	88.00	93.81
275	16	92.00	96.86	85.00	96.86
277	11	90.00	100.36	83.00	100.36
281	4	86.00	125.88	79.00	125.88

C.2 SHA3-224

C.3 SHA3-256

Table 15: Summary of complexity estimations for **SHA3-256** corresponding to the number of constraints solved by taking the reduction factor as 2^7 .

#constraints	DF of Connector	Data Complexity	Connector Complexity	Brute Force Complexity	Total Complexity
240	60	139.00	94.50	132.00	132.00
250	39	129.00	106.50	122.00	122.00
252	32	127.00	112.18	120.00	120.01
254	27	125.00	115.18	118.00	118.19
255	26	124.00	116.69	117.00	117.85
256	24	123.00	118.24	116.00	118.51
257	23	122.00	119.68	115.00	119.74
258	22	121.00	122.82	114.00	122.83
261	18	118.00	126.50	111.00	126.50