
# Practical Collision Attacks on Reduced-Round Xoodyak Hash Mode

**Huina Li**[1] (remote) Le He[2] Weidong Qiu[1]

August 14, 2025

[1]School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
lihuina@sjtu.edu.cn, qiuwd@sjtu.edu.cn

[2]School of Cyber Engineering, Xidian University, Xi'an, China
hele@xidian.edu.cn

# Outline

1. Background and Motivation
2. Our Multi-block Differential-based Collision Attack Framework
3. Differential Trails Search for Collision Attack
4. SAT-based Collision Search
5. New Collision Attacks on Xoodyak Hash Mode

# Background and Motivation

# Xoodyak

- One of the ten finalists of the NIST Lightweight Crypto Standardization process, offering both keyed and hash modes.
- The hash mode is based on the sponge construction, operating under a mode of operation known as Cyclist, including the hash function **Xoodyak-Hash** and an eXtendable Output Function **Xoodyak-Xof**.
- Internally, it uses the Xoodoo[12] permutation denoted by $f$.

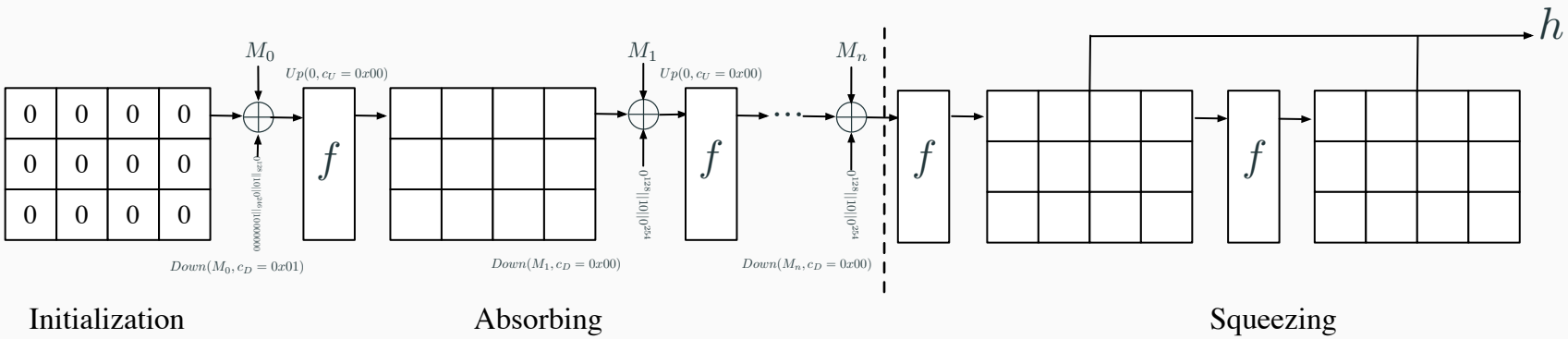
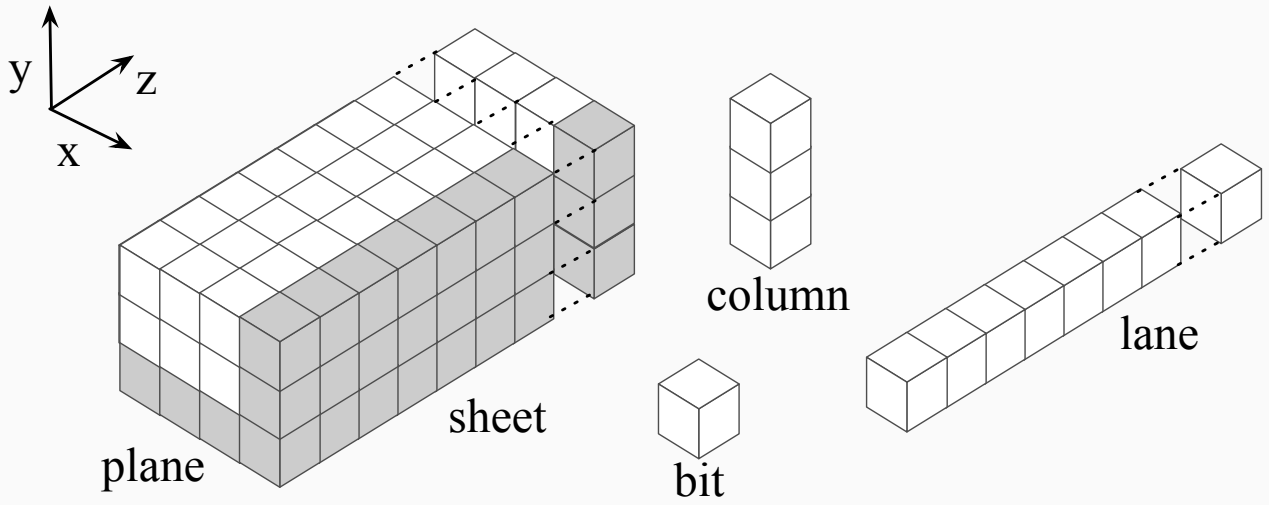



Figure 1: The Xoodyak Hash Mode

# Xoodoo Permutation

- Xoodoo permutation is the underlying permutaion of Xoodyak-Hash and Xoodyak-Xof
- It operates on a 3-dimensional array of size 384 bits. The bits of the state $S$ are indexed by the $(x, y, z)$ coordinates where $0 \leq x < 4$, $0 \leq y < 3$, and $0 \leq z < 32$.
- Each bit is indexed by $S[32 \times (x + 4 \times y) + z]$.




- 12 rounds
- Each round consists of 5 operations, *i.e.*, $R = \rho_{east} \circ \chi \circ \iota \circ \rho_{west} \circ \theta$.

# Round Function of Xoodoo

$\theta$:

$$P[x][z] \leftarrow \sum_{y=0}^{2} S[x][y][z]$$

$$E[x][z] \leftarrow P[x+1][z+5] \oplus P[x+1][z+14] \tag{1}$$

$$S[x][y][z] \leftarrow S[x][y][z] \oplus E[x][z]$$

$\rho_{west}$:

$$S[x][1][z] \leftarrow S[x+1][1][z] \tag{2}$$

$$S[x][2][z] \leftarrow S[x][2][z+11]$$

$\iota$:

$$S[0][0][z] \leftarrow S[0][0][z] \oplus C_i$$

$\chi$:

$$S[x][0][z] \leftarrow S[x][0][z] \oplus \neg S[x][1][z] \wedge S[x][2][z]$$

$$S[x][1][z] \leftarrow S[x][1][z] \oplus \neg S[x][2][z] \wedge S[x][0][z] \tag{3}$$

$$S[x][2][z] \leftarrow S[x][2][z] \oplus \neg S[x][0][z] \wedge S[x][1][z]$$

$\rho_{east}$:

$$S[x][1][z] \leftarrow S[x][1][z+1] \tag{4}$$

$$S[x][2][z] \leftarrow S[x+2][2][z+8]$$

# General Collision Attack on Xoodyak Hash Mode

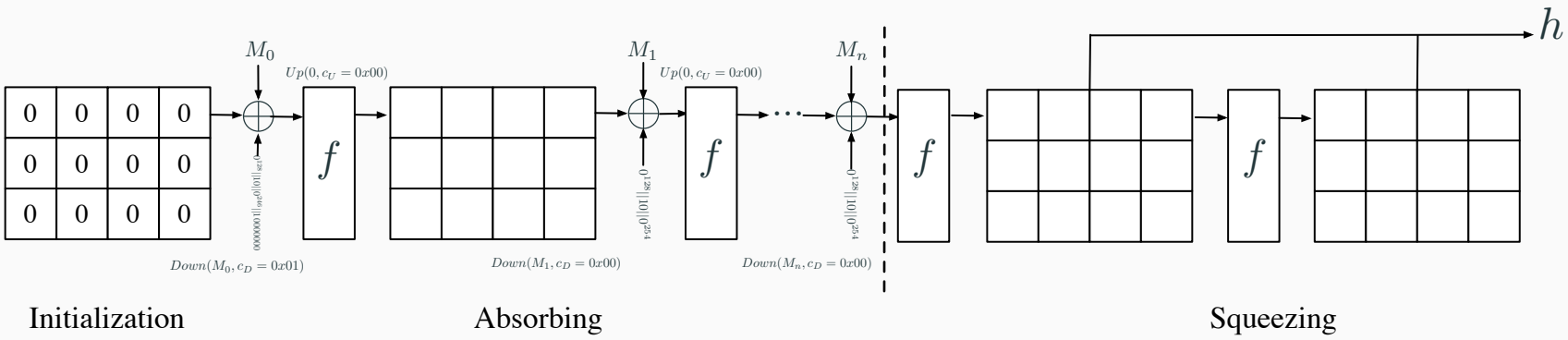



Figure 2: The Xoodyak Hash Mode

- In the squeezing phase, it produces the $h$-bit digest. To find a collision,

$$H(IV, M) = H(IV, M') \tag{5}$$

  requires at least Time complexity $\approx \min\{2^{c/2}, 2^{h/2}\}$
- A successful collision attack on a sponge-based hash function $H$ is to find a pair of distinct messages $M$ and $M'$ such that $H_{IV}(M) = H_{IV}(M')$ with a time complexity of less than $2^{min(h/2, c/2)}$.

# Previous Work: Dong *et al.* Collision Attack Framework[2]

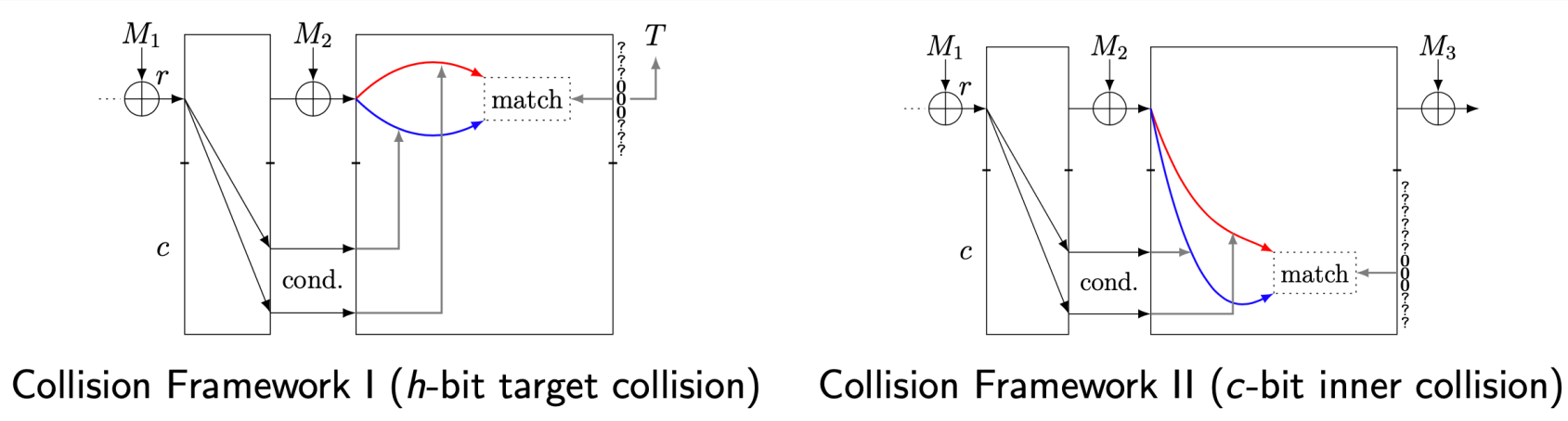



**Figure 3:** Collision Framework

- At CRYPTO 2024, Dong *et al.* proposed a generic MitM collision attack framework for sponge constructions for the first time.
- They presented the first collision attack on 3-round Xoodyak-Hash with the time complexity of $2^{125.23}$, the memory complexity of $2^{124}$.

Limitation: relatively huge memory complexity.

# Previous Work: Differential-based Collision Attack Framework

- Differential-based collision attack framework is one of the most powerful attack framework on sponge construction proposed by Dinur *et al.* at FSE 2012[1] and subsequently improved by[5, 6, 3, 4] with novel linearization techniques and new tools.
- The full attack primarily consists of two steps:
  1. **Colliding trail search phase**: Finding a relatively high-probability $n_2$ differential trail that ensures $h$-bit digest collision.
  2. **Connector construct phase**: Constructing a $n_1$-round connector with some linearization techniques in order to obtain a sufficiently large set of message pairs that simultaneously satisfy the colliding trail, as well as the constraints imposed by the padding rule and the initial value (IV) of the hash function

Limitation: CICO problem arises, resulting in difficulty in generating relatively high-probability differential trails over more rounds for collision attacks;

# Our Multi-block Differential-based Collision Attack Framework

# Overview of Collision Attack Framework i

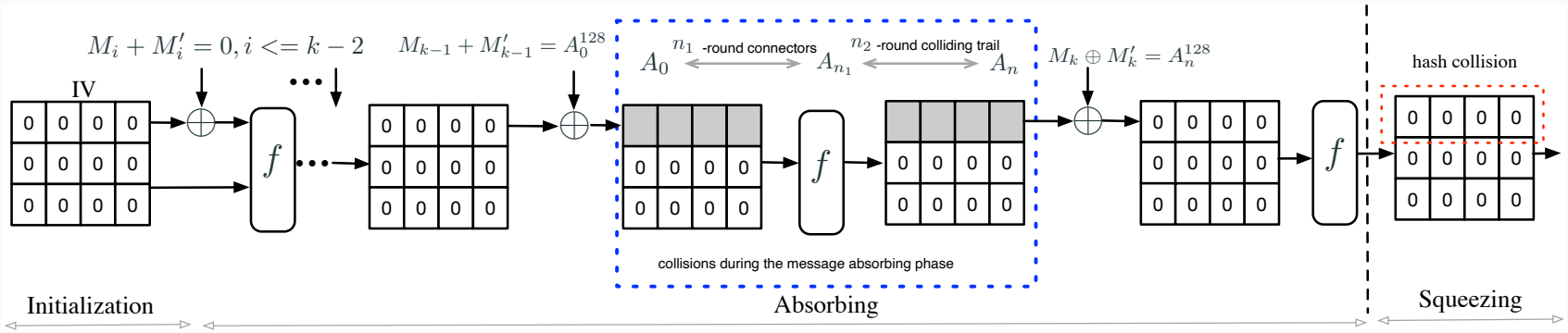



**Figure 4:** Our Collision Attack Framework

In our attack framework, we require an $n$-round differential trail, which includes a $n_2$-round colliding trail with weight of $w_2$ and a $n_1$-round connector.

1. Prepare $n_2$-round colliding trails. $A_{n_1}$ and $A_n$ stand for the input and output difference of $n_2$-round colliding trails.

# Overview of Collision Attack Framework ii

2. Construct $n_1$-round connectors that promise a subspace of message pairs $(M_0, \ldots, M_{k-1})$ and $(M_0, \ldots, M'_{k-1})$ which meet both the message difference imposed by the Sponge construction and the input difference $A_{n_1}$ of the colliding trails.

$$A_{n_1} = f^n(S_{k-1} \oplus (M_{k-1}||0^{256})) \oplus f^n(S_{k-1} \oplus (M'_{k-1}||0^{256})) \quad (6)$$

3. Once we construct such a $n$-round differential trail and find a message pair $(M_0, \ldots, M_{k-1})$ and $(M'_0, \ldots, M'_{k-1})$ that follows this trail where $A_0^{128} = M_{k-1} \oplus M'_{k-1}$, we introduce one more pair of message blocks $(M_k, M'_k)$ at the last absorbing phase, *i.e.*, $A_n^{128} = M_k \oplus M'_k$, the matching difference is then canceled out, resulting in zero difference during the squeezing phase, thus successfully converting it into a real collision.

# Complexity Analysis: Connector Construction Complexity

- The core idea of constructing $n_1$-round connector is to convert the problem to solving an algebraic system.
- Assume that an algebraic system of $n_1$-round connector is constructed:
  $n_l$: the number of the linear equations;
  $n_q$: the number of non-linear equations;
  $n'_l$: the number of non-linear equations that can be linearized by guessing $d_f$ extra equations (we call them guess equations which consume $d_f$ DF).
- Thus, the final linear system includes $n_l + n'_l + d_f$ linear boolean equations in 128 Boolean variables.

# Complexity Analysis: Connector Construction Complexity

- Assume $n_l + n'_l + d_f \leq 128$, then the system has a non-trivial solution. The time complexity of Gaussian elimination $T_g$ for a system of linear equations is approximately

$$T_g \approx \frac{1}{3}(n_l + n'_l + d_f)^2 \times 128.$$

- The complexity of solving a linear system denoted by $T_S = \frac{T_g}{1824n}$.
- During each iteration of Gaussian elimination, it is expected to obtain $2^{128-n_l-n'_l-d_f}$ solutions, so that the verification time complexity is estimated as $T_v = 2^{128-n_l-n'_l-d_f}$.
- As each guess equation can be assigned a constant value of either 0 or 1. In this way, for one connector, we can construct $2^{d_f}$ linear systems.
- Thus, The total time complexity of constructing one connector is equivalent to,

$$2^{d_f} \times (T_S + T_v) \tag{7}$$

# Complexity Analysis: Connector Construction Complexity

- To satisfy the remaining $n_q - n'_l$ non-linear equations and the $w_2$ conditions of the difference-value of the colliding trail, we need to prepare at least $2^{w_2+n_q-n'_l}$ solutions.
- Thus, the time complexity corresponds to the time required to construct $2^{(w_2+n_q-n'_l)-(128-n_l-n'_l-d_f)-d_f} = 2^{w_2+n_q+n_l-128}$ connectors.
- The total time complexity in connector construction is equivalent to,

$$T_1 = 2^{w_2+n_q+n_l-128+d_f} \times \left(\frac{\frac{1}{3}(n_l + n'_l + d_f)^2 \times 128}{1824n}\right) + 2^{w_2+n_q-n'_l} \quad (8)$$

# Complexity Analysis: Exhaustive Search Complexity

- The remaining $n_q - n'_l$ equations and the $w_2$ conditions on the difference values of the colliding trail are satisfied via exhaustive search.
- In the exhaustive search phase of the attack, we try $2^{w_2+n_q-n'_l}$ different message pairs in order to find a pair whose difference evolves according to the specified $n_2$-round colliding trail. The time complexity is

$$T_2 = 2^{w_2+n_q-n'_l} \tag{9}$$

# Complexity Analysis

In total, the time complexity denoted by $T_{total}$ of the $n$-round collision attack is

$$\begin{aligned} T_{total} &= T_1 + T_2 \\ &= 2^{w_2+n_q+n_l-128+d_f} \times \left(\frac{\frac{1}{3}(n_l + n_l' + d_f)^2 \times 128}{1824n}\right) + 2^{w_2+n_q-n_l'+1} \end{aligned} \tag{10}$$

# Differential Trails Search for Collision Attack

# $n$-round Differential Trail Search

For the study of the differential trail propagation of Xoodoo, we take a $n$-round differential trail as an example.

$$A_0 \xrightarrow{\theta} B_0 \xrightarrow{\rho_{west}} C_0 \xrightarrow{\chi} D_0 \xrightarrow{\lambda} C_1 \xrightarrow{\chi} \ldots \xrightarrow{\lambda} C_{n-1} \xrightarrow{\chi} D_{n-1} \xrightarrow{\rho_{east}} A_n \qquad (11)$$

- let $C_i$ denote the input difference and $D_i$ denote the output difference of the $i$-th $\chi$ mapping. The last 256 bits of both the input and output differences (*i.e.*, $A_0, A_n$) are zero.
- We denote the linear layer as $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$.

# Loop Differential Trail

## Definition 1 (loop)

A parity *loop* (or *loop* for short) is a state value with 32 active bits in a sheet, each in a distinct column.

## Definition 2 (Loop Differential Trail)

For an $n$-round differential trail, if all internal differences consist solely of *loop* elements, we refer to it as a *loop differential trail*.

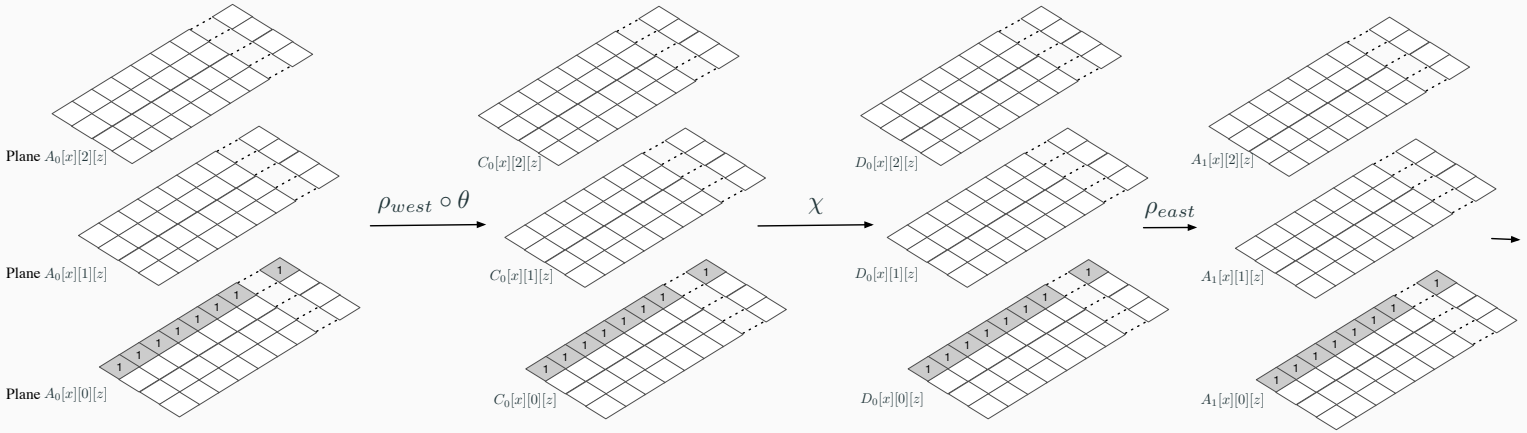
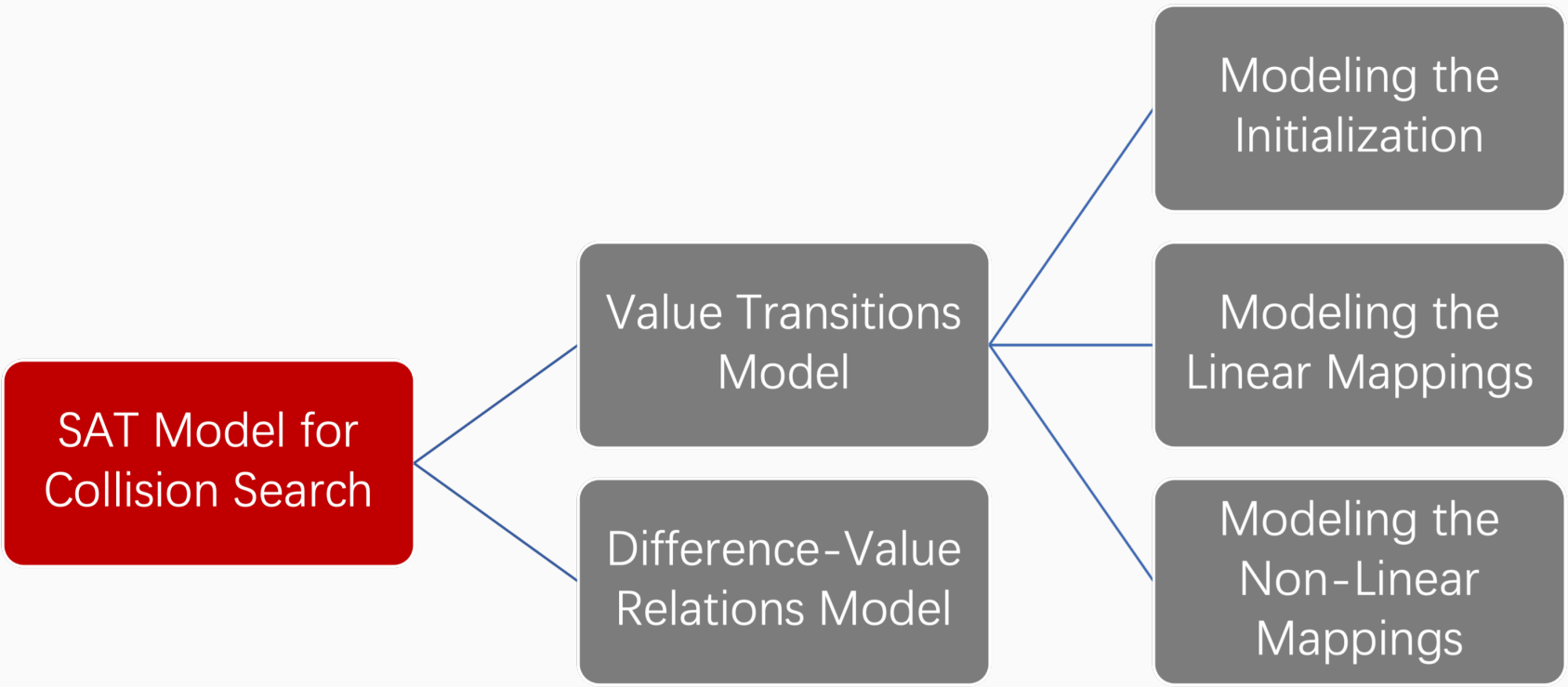



Figure 5: 1-round Loop Differential Trail

# SAT-based Collision Search

# Overview of Our Collision Search Model



# Difference-Value Relations

**Property**

For a Boolean function $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ representing a certain output bit of a primitive, the output difference $\beta_i \in \mathbb{F}_2$ is the derivative of $f$ with respect to input difference $\alpha$ at point $x \in \mathbb{F}_2^n$,

$$\beta_i = \mathcal{D}_\alpha f_i(x) = f_i(x) \oplus f_i(x \oplus \alpha) \tag{12}$$

This is equivalently expressed as the partial derivative of $f_i(x \oplus u \cdot \alpha)$ with respect to the boolean variable $u \in \mathbb{F}_2$. Formally, this relationship is defined as:

$$\beta_i = \mathcal{D}_\alpha f_i(x) = \mathcal{D}_u f_i(x \oplus u \cdot \alpha) \tag{13}$$

# Example

Let us consider an active S-box represented by $\chi(x_0, x_1, x_2)$, which follows a valid difference pattern ($\alpha = (1, 0, 0), \beta = (1, 0, 0)$). The input values $x = (x_0, x_1, x_2)$ of the S-box correspond to the solution space of the following linear equations,

$$\begin{cases} \beta_0 = \mathcal{D}_u \chi_0(x_0 \oplus u \cdot \alpha_0, x_1 \oplus u \cdot \alpha_1, x_2 \oplus u \cdot \alpha_2) \\ \beta_1 = \mathcal{D}_u \chi_1(x_0 \oplus u \cdot \alpha_0, x_1 \oplus u \cdot \alpha_1, x_2 \oplus u \cdot \alpha_2) \\ \beta_2 = \mathcal{D}_u \chi_2(x_0 \oplus u \cdot \alpha_0, x_1 \oplus u \cdot \alpha_1, x_2 \oplus u \cdot \alpha_2) \end{cases} \tag{14}$$

One Solution can be derived from Equation 14:

$$\begin{cases} 1 = 1 \\ 0 = x_2 \oplus 1 \\ 0 = x_1 \end{cases}$$

# Constructing Difference-Value Relations Model

**Algorithm 2** Difference-Value Relations SAT Model

**Require:** $c_i, 0 \leq i < n$, the non-linear operations $\chi^{n-1} \circ \cdots \circ \chi^0$, the number of rounds $n$, a given differential trail $(A_0, B_0, C_0, D_0, A_1, \ldots, A_n)$(please refer to trail model 12), and an auxiliary binary variable $u$.

**Ensure:** The value of $a_0$ or "Invalid".

1: Initialize a set $Q = \emptyset$;
2: **for** $i$ from 0 to $n$ **do**
3: $f^i = c_i \oplus uC_i$
4: Compute the output of the nonlinear operation: $f^{i+1} \leftarrow \chi^i(f^i)$;
5: Add $\mathcal{D}_u f^{i+1} \oplus D_i$ to the set $Q$;
6: **end for**
7: Convert the set $Q$ into CNF clauses in DIMACS format using the *sage.sat* module.);

Figure 6: Difference-Value Relations SAT Model

# New Collision Attacks on Xoodyak Hash Mode

# Linearization Strategy of 2-round Connector i

Two rounds of Xoodoo permutation are expressed as

$$a_0 \xrightarrow{\iota\circ\rho_{west}\circ\theta} c_0 \xrightarrow{\chi} d_0 \xrightarrow{\iota\circ\rho_{west}\circ\theta\circ\rho_{east}} c_1 \xrightarrow{\chi} d_1 \xrightarrow{\rho_{east}} a_1 \tag{15}$$

- In the first round, we have $n_l = 64$ linear equations derived from the difference-value relations $D_0 = \mathcal{D}_u\chi(c_0 \oplus uC_0)$, *i.e.*, $c_0[0][1][z] = 0$ and $c_0[0][2][z] = 1$, where $0 \leq z < 32$.
- In the second round, the difference-value relations $D_1 = \mathcal{D}_u\chi(c_1 \oplus uC_1)$ give rise to $n_q = 64$ quadratic equations, *i.e.*, $c_1[0][1][z] = 0$ and $c_1[0][2][z] = 1$.

# Linearization Strategy of 2-round Connector ii

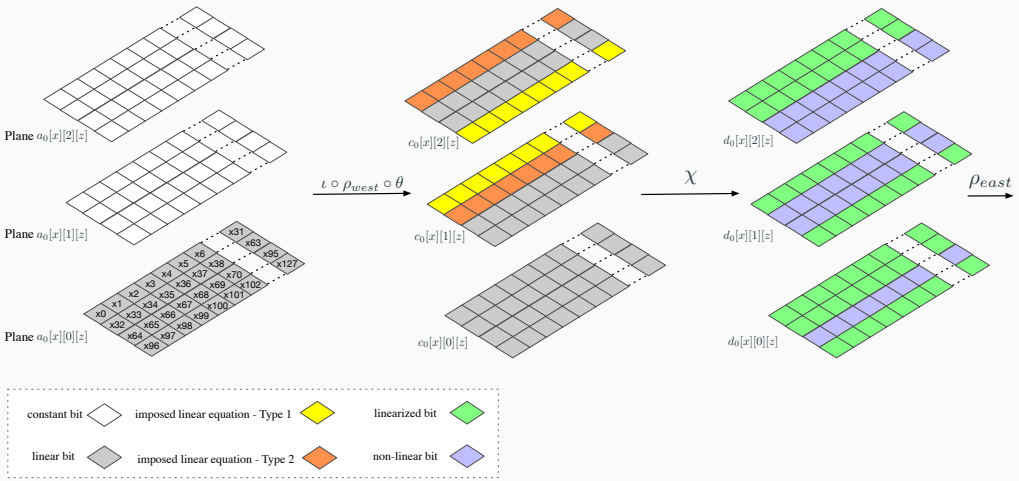
$$T_{total} = 2^{w_2+n_q+n_l-128+d_f} \times \left(\frac{\frac{1}{3}(n_l + n_l' + d_f)^2 \times 128}{1824n}\right) + 2^{w_2+n_q-n_l'+1}$$

The linearization of a 3-bit S-box can be categorized into two classes.

- For active S-boxes, when the input difference and the compatible output difference are given, the outputs $y$ can be expressed as linear combinations of the input bits. From this algebraic perspective, these S-boxes are already fully linearized.
- For non-active S-boxes, linearization can be achieved by introducing two additional equations involving the input bits.

# The first round Connector

- The first 32 active S-boxes are already fully linearized. Thus, the output bits $d_0[0][*][z]$ of the 32 active S-boxes $c_0[0][*][z]$ after the first $\chi$ mapping must be linear bits *i.e.*, $(d_0[0][0][z], d_0[0][1][z], d_0[0][2][z], 0 \leq z < 32)$.
- If the bit $c_0[0][1][z]$ is imposed a linear equation, then the bit $c_0[3][2][z]$ is also imposed a linear equation equivalent to bit $c_0[0][1][z+11]$, where $0 \leq z < 32$.
- If the bit $c_0[0][2][z]$ is imposed a linear equation, then the bit $c_0[1][1][z]$ is also imposed a linear equation equivalent to bit $c_0[0][2][z-11]$, where $0 \leq z < 32$.



# The Second round Connector

To better represent the 64 quadratic conditions of the second round, we introduce 384 Boolean variables $x_{384}, \ldots, x_{767}$ to represent each bit of $d_0$.

Table 8: Statistics of 64 Quadratic Equations. $x(1,2,3) \triangleq x_1 \oplus x_2 \oplus x_3$; Orange bit is non-liner variable in $d_0$; Black bit is linear variable in $d_0$.

| No. | Quadratic Equations | No. | Quadratic Equations |
|---|---|---|---|
| 1 | $x(466, 475, 593, 602, 639, 650, 659)$ | 33 | $x(487, 496, 614, 623, 680, 703, 717)$ |
| 2 | $x(467, 476, 594, 603, 608, 651, 660)$ | 34 | $x(488, 497, 615, 624, 672, 681, 718)$ |
| 3 | $x(468, 477, 595, 604, 609, 652, 661)$ | 35 | $x(489, 498, 616, 625, 673, 682, 719)$ |
| 4 | $x(469, 478, 596, 605, 610, 653, 662)$ | 36 | $x(490, 499, 617, 626, 674, 683, 720)$ |
| 5 | $x(470, 479, 597, 606, 611, 654, 663)$ | 37 | $x(491, 500, 618, 627, 675, 684, 721)$ |
| 6 | $x(448, 471, 598, 607, 612, 655, 664)$ | 38 | $x(492, 501, 619, 628, 676, 685, 722)$ |
| 7 | $x(449, 472, 576, 599, 613, 656, 665)$ | 39 | $x(493, 502, 620, 629, 677, 686, 723)$ |
| 8 | $x(450, 473, 577, 600, 614, 657, 666)$ | 40 | $x(494, 503, 621, 630, 678, 687, 724)$ |
| 9 | $x(451, 474, 578, 601, 615, 658, 667)$ | 41 | $x(495, 504, 622, 631, 679, 688, 725)$ |
| 10 | $x(452, 475, 579, 602, 616, 659, 668)$ | 42 | $x(496, 505, 623, 632, 680, 689, 726)$ |
| 11 | $x(453, 476, 580, 603, 617, 660, 669)$ | 43 | $x(497, 506, 624, 633, 681, 690, 727)$ |
| 12 | $x(454, 477, 581, 604, 618, 661, 670)$ | 44 | $x(498, 507, 625, 634, 682, 691, 728)$ |
| 13 | $x(455, 478, 582, 605, 619, 662, 671)$ | 45 | $x(499, 508, 626, 635, 683, 692, 729)$ |
| 14 | $x(456, 479, 583, 606, 620, 640, 663)$ | 46 | $x(500, 509, 627, 636, 684, 693, 730)$ |
| 15 | $x(448, 457, 584, 607, 621, 641, 664)$ | 47 | $x(501, 510, 628, 637, 685, 694, 731)$ |
| 16 | $x(449, 458, 576, 585, 622, 642, 665)$ | 48 | $x(502, 511, 629, 638, 686, 695, 732)$ |
| 17 | $x(450, 459, 577, 586, 623, 643, 666)$ | 49 | $x(480, 503, 630, 639, 687, 696, 733)$ |
| 18 | $x(451, 460, 578, 587, 624, 644, 667)$ | 50 | $x(481, 504, 608, 631, 688, 697, 734)$ |
| 19 | $x(452, 461, 579, 588, 625, 645, 668)$ | 51 | $x(482, 505, 609, 632, 689, 698, 735)$ |
| 20 | $x(453, 462, 580, 589, 626, 646, 669)$ | 52 | $x(483, 506, 610, 633, 690, 699, 704)$ |
| 21 | $x(454, 463, 581, 590, 627, 647, 670)$ | 53 | $x(484, 507, 611, 634, 691, 700, 705)$ |
| 22 | $x(455, 464, 582, 591, 628, 648, 671)$ | 54 | $x(485, 508, 612, 635, 692, 701, 706)$ |
| 23 | $x(456, 465, 583, 592, 629, 640, 649)$ | 55 | $x(486, 509, 613, 636, 693, 702, 707)$ |
| 24 | $x(457, 466, 584, 593, 630, 641, 650)$ | 56 | $x(487, 510, 614, 637, 694, 703, 708)$ |
| 25 | $x(458, 467, 585, 594, 631, 642, 651$ | 57 | $x(488, 511, 615, 638, 672, 695, 709)$ |
| 26 | $x(459, 468, 586, 595, 632, 643, 652)$ | 58 | $x(480, 489, 616, 639, 673, 696, 710)$ |
| 27 | $x(460, 469, 587, 596, 633, 644, 653)$ | 59 | $x(481, 490, 608, 617, 674, 697, 711)$ |
| 28 | $x(461, 470, 588, 597, 634, 645, 654)$ | 60 | $x(482, 491, 609, 618, 675, 698, 712)$ |
| 29 | $x(462, 471, 589, 598, 635, 646, 655)$ | 61 | $x(483, 492, 610, 619, 676, 699, 713)$ |
| 30 | $x(463, 472, 590, 599, 636, 647, 656)$ | 62 | $x(484, 493, 611, 620, 677, 700, 714)$ |
| 31 | $x(464, 473, 591, 600, 637, 648, 657)$ | 63 | $x(485, 494, 612, 621, 678, 701, 715)$ |
| 32 | $x(465, 474, 592, 601, 638, 649, 658)$ | 64 | $x(486, 495, 613, 622, 679, 702, 716)$ |

**Figure 8:** Statistics of 64 Quadratic Equations.

# The Second round Connector

Specially, for non-linear bits $x_i$, $704 \leq i \leq 735$, each non-linear bit require one extra linear equation to achieve linearization, resulting in the consumption of 1 DF. For example, if we set $x_7 + x_{30} = c$, where $c = 0$ or $c = 1$, $x_{716}$ can be linearized.

$$\begin{cases} x_{i+686} : (x_i + x_{i+9})(x_{i+32} + x_{i+41} + x_{i+46}), 18 \leq i \leq 22 \\ x_{i+709} : (x_i + x_{i+23})(x_{i+32} + x_{i+55} + x_{i+69}), 0 \leq i \leq 8 \\ x_{i+718} : (x_i + x_{i+9})(x_{i+32} + x_{i+41} + x_{i+78}), 0 \leq i \leq 17 \end{cases}$$

In this way, this process enables us to obtain one linearized quadratic equation at the cost of 1 DF, which can be mathematically expressed as $d_f = n'_l$.

# Summary of Our Collision Attacks on Xoodyak Hash Mode

$$T_{total} = 2^{w_2+n_q+n_l-128+d_f} \times \left(\frac{\frac{1}{3}(n_l + n_l' + d_f)^2 \times 128}{1824n}\right) + 2^{w_2+n_q-n_l'+1}$$

- For 2-round collision attack, when $n_l' = d_f = 29$, the time complexity of the 2-round collision attack reaches its minimum value, *i.e.*, $T_{total} = 2^{37.2388}$.
  **We find the first actual 2-round collision in 64.7 seconds based on collision search model.**
- For 3-round collision attacks, when $n_l' = d_f = 29$, the time complexity of the 3-round collision attack reaches its minimum value, *i.e.*, $T_{total} = 2^{100.9311}$.
  **We present the best known 3-round collision attacks with the time complexity of $2^{100.9311}$, and the negligible memory complexity.**
- **We find a practical 3-round semi-free-start (SFS) collision in 3.67 seconds based on collision search model.**

# References i


I. Dinur, O. Dunkelman, and A. Shamir.
**New attacks on Keccak-224 and Keccak-256.**
In *Fast Software Encryption - 19th International Workshop, FSE 2012*, volume 7549, pages 442–461, Berlin, Heidelberg, 2012. Springer.

X. Dong, B. Zhao, L. Qin, Q. Hou, S. Zhang, and X. Wang.
**Generic MitM attack frameworks on sponge constructions.**
In *CRYPTO (4)*, volume 14923 of *Lecture Notes in Computer Science*, pages 3–37. Springer, 2024.

J. Guo, G. Liao, G. Liu, M. Liu, K. Qiao, and L. Song.
**Practical collision attacks against round-reduced SHA-3.**
*Journal of Cryptology*, 33(1):228–270, 2020.

# References ii



S. Huang, O. A. Ben-Yehuda, O. Dunkelman, and A. Maximov.
**Finding collisions against 4-round SHA3-384 in practical time.**
*IACR Cryptol. ePrint Arch.*, page 194, 2022.


K. Qiao, L. Song, M. Liu, and J. Guo.
**New collision attacks on round-reduced Keccak.**
In *EUROCRYPT (3)*, volume 10212 of *Lecture Notes in Computer Science*, pages 216–243, 2017.


L. Song, G. Liao, and J. Guo.
**Non-full Sbox linearization: Applications to collision attacks on round-reduced Keccak.**
In *CRYPTO (2)*, volume 10402 of *Lecture Notes in Computer Science*, pages 428–451. Springer, 2017.