Collision Attacks on Round-Reduced SHA-3 Using Conditional Internal Differentials*

Zhongyi Zhang[1,2][0009−0008−0491−3006], Chengan Hou[1,2][0009−0009−5618−6979], and Meicheng Liu[1,2] ✉[0000−0002−1825−0097]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, P. R. China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, P. R. China
{zhangzhongyi0714,houchengan,liumeicheng}@iie.ac.cn

Abstract. The Keccak hash function was selected by NIST as the winner of the SHA-3 competition in 2012 and became the SHA-3 hash standard of NIST in 2015. On account of SHA-3's importance in theory and applications, the analysis of its security has attracted increasing attention. In the SHA-3 family, SHA3-512 shows the strongest resistance against collision attacks: the theoretical attacks of SHA3-512 only extend to four rounds by solving polynomial systems with 64 times faster than the birthday attack. Yet for the SHA-3 instance SHAKE256 there are no results on collision attacks that we are aware of in the literatures.

In this paper, we study the collision attacks against round-reduced SHA-3. Inspired by the work of Dinur, Dunkelman and Shamir in 2013, we propose a variant of birthday attack and improve the internal differential cryptanalysis by abstracting new concepts such as *differential transition conditions* and *difference conditions table*. With the help of these techniques, we develop new collision attacks on round-reduced SHA-3 using conditional internal differentials. More exactly, the initial messages constrained by linear conditions pass through the first two rounds of internal differential, and their corresponding inputs entering the last two rounds are divided into different subsets for collision search according to the values of linear conditions. Together with an improved target internal difference algorithm (TIDA), collision attacks on up to 5 rounds of all the six SHA-3 functions are obtained. In particular, collision attacks on 4-round SHA3-512 and 5-round SHAKE256 are achieved with complexity of 2^{237} and 2^{185} respectively. As far as we know, this is the best collision attack on reduced SHA3-512, and it is the first collision attack on reduced SHAKE256.

Keywords: SHA-3 · Hash function · Keccak · Internal differentials · Collision attack · TIDA.

* Supported by the National Natural Science Foundation of China (Grant No. 62122085 and 12231015) and the Youth Innovation Promotion Association of Chinese Academy of Sciences.

1 Introduction

The KECCAK hash function [4], designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche [5], was selected as the winner of the `SHA-3` competition by the National Institute of Standards and Technology of the USA. In 2015, it was published as the new `SHA-3` standard by NIST [11]. The `SHA-3` family has four instances with fixed digest sizes, namely `SHA3-224`, `SHA3-256`, `SHA3-384` and `SHA3-512`, which correspond to KECCAK$[c] \triangleq$ KECCAK$[r = 1600 - c, c]$, where $c \in \{448, 512, 768, 1024\}$. There are two eXtendable-Output Functions (XOFs) named `SHAKE128` and `SHAKE256` of the `SHA-3` family, which can generate digests with any expected length. In Post-Quantum Cryptography competition (PQC), the two XOFs are applied to all candidate algorithms (CRYSTALS-KYBER, CRYSTALS-Dilithium, FALCON, SPHINCS$^+$) identified by NIST for standardization and all the fourth round candidate KEM algorithms (BIKE, Classic McEliece, HQC, SIKE) [1]. Among them, SPHINCS$^+$ [3] is a stateless hash-based signature scheme including three different versions. One of them is obtained by instantiating the SPHINCS$^+$ construction with `SHAKE256`.

KECCAK uses a sponge construction, which ensures that messages of any length can be taken as inputs of the hash function. The message is padded and divided into some message blocks with the same length. The size of message block depends on the expected number of output bits. The 1600-bit initial state of KECCAK is XORed the first message block. Then, the state is updated by applying 24-round permutation KECCAK-f to it and XORing another message block, until all blocks are absorbed. In the end, the state is updated again by using 24-round KECCAK-f, and some bits of the state are output as the digest.

Since its publication in 2008, KECCAK has become one of the most important hash functions and received extensive security analysis [2,9,10,6,14,20,16,12,13]. There are two important security criteria for cryptographic hash functions namely, preimage resistance and collision resistance.

The main focus of this paper is on the security of SHA-3 family against collision attacks. The purpose of a collision attack is to find a pair of different messages such that their digests are the same. In the matter of collision attacks on round-reduced `SHA-3` (KECCAK), Dinur, Dunkelman and Shamir [8] presented practical attacks on 4-round KECCAK[448]/KECCAK[512] in 2012, where the authors developed the target difference algorithm to link a 1-round connector to a 3-round high probability difference characteristic. Following the basic framework of [8], Qiao *et al.* completed the connection of 2-round connectors and 3-round difference characteristics using the linearizaion technique and obtained actual collisions for 5-round `SHAKE128` [19]. In [20,12], the connectors were improved to 3-round connectors by non-full linearization technique. As a result, the practical collision attacks on 5-round `SHA3-224/SHA3-256` were implemented respectively. Almost at the same time as this paper, Huang *et al.* [15] developed new techniques to try to solve the problem of insufficient degrees of freedom, and proposed a collision attack on `SHA3-384` with time complexity of $2^{59.64}$. In [13], with the SAT-based automatic search tool and improved connector construction

algorithms, Guo *et al.* presented the first quantum collision attacks on SHA-3 instances. More specifically, they extended the classical attacks on SHAKE128 to 6-round and proposed 6-round quantum attacks on SHA3-224 and SHA3-256. For internal differentials, Dinur, Dunkelman and Shamir [9] completed practical collision attacks on 3-round KECCAK[768]/KECCAK[1024] and proposed theoretical attacks on 4-round KECCAK[768] and 5-round KECCAK[512] by using generalized internal differentials. In addition to differential and internal differential, Dinur [7] devised a polynomial method-based algorithm for solving multivariate equation systems and formulated the problem of finding a collision as a non-linear equation system. With the help of this technique, the author obtain a theoretical collision attack on 4-round SHA3-512, where the complexity (2^{263}) is 64 times faster than the birthday attack (2^{269}) considering the bit operations.

Our Contribution. Following the framework of Dinur, Dunkelman and Shamir [9], we improve the generalized internal differentials and present theoretical attacks on all the six SHA-3 variants up to 5 rounds. In detail, the attacks on 4-round SHA3-512 and 5-round SHAKE256 are the best attack results at present as far as we know. Our results and comparison with the related previous work are listed in Table 1. The main contributions with respect to techniques are summarized as follows.

1. A variant of birthday attack Since an internal difference produces distinct output internal differences after non-linear operation, collision search is actually carried out in several disjoint subsets. We abstract it as a variant of birthday attack. On one hand, it is more convenient for parallel computation. On the other hand, the size of each subset is much smaller than the number of messages, which greatly saves the space of the hash table using in the attack.

2. Improved generalized internal differentials We introduce the transition condition number to estimate the transition probability of internal differential more accurately. We can construct conditional internal differential characteristics for collision attacks on up to 5 rounds of SHA-3 by adding differential transition conditions to the initial message spaces and their corresponding internal states. This further reduces the time complexity of internal differential cryptanalysis.

3. Improved target internal difference algorithm We link an internal differential characteristic starting from the second round to the initial state of SHA-3 by solving a linear equation system. With the use of 2-block messages, we change the value of the first block instead of changing an affine subspaces of input internal differences to make the system consistent. And since any affine subspace of the input difference can be selected, in the improved TIDA, we can select a specific set of affine subspaces to obtain internal difference characteristics with high probability.

Conditional Internal Differential Attacks. The technique of *internal differential cryptanalysis* was developed by Peyrin [18] in the cryptanalysis of the Grøstl hash function and generalized by Dinur *et al.* [9] in collision attacks on

Target	Rounds	Complexity	Attack method	Reference
SHA3-224	**5**	$\mathbf{2^{105}}$	**Internal differential**	**Sec 6.3**
	5	Practical	Differential	[20]
SHA3-256	5	2^{115}	Internal differential	[9]
	5	$\mathbf{2^{105}}$	**Internal differential**	**Sec 6.3**
	5	Practical	Differential	[12]
SHA3-384	3	Practical	Internal differential	[9]
	4	2^{147}	Internal differential	[9]
	4	$\mathbf{2^{76}}$	**Internal differential**	**Sec 6.1**
	4	$2^{59.64}$	Differential	[15]
SHA3-512	3	Practical	Internal differential	[9]
	4	$\mathbf{2^{237}}$	**Internal differential**	**Sec 6.2**
	4	$2^{263\dagger}$	Solving polynomial systems	[7]
SHAKE128	**5**	$\mathbf{2^{105}}$	**Internal differential**	**Sec 6.3**
	5	Practical	Differential	[19]
	6	$2^{123.5}$	Differential	[13]
SHAKE256	**4**	$\mathbf{2^{76}}$	**Internal differential**	**Sec 6.1**
	5	$\mathbf{2^{185}}$	**Internal differential**	**Sec 6.3**

† The complexity is calculated by bit operations.

Table 1. Comparison of the best collision attacks against the SHA-3 family

SHA-3. This technique resembles standard differential attacks but it uses internal differentials, which consider differences between different parts of a state and follow their statistical evolution, rather than a difference between two states. In [9], Dinur *et al.* proposed the definitions of the weight of internal differences, which can be used to estimate the transition probability of the internal differences with low weight, and obtained internal differential characteristic with probability 1 for the first round by using algebraic methods.

In this paper, we develop an improved variant of internal differential cryptanalysis to launch collision attacks on SHA-3. We introduce several new techniques such as differential transition conditions of the Keccak Sbox, which allow us to estimate the transition probability of internal differences more accurately and use conditional internal differentials to reduce more complexity. And since the non-zero internal difference input to χ produces several output internal differences, we can launch a variant of birthday attack. Namely, one or multiple output internal differences will result in a collision subset, so that we can search for collision in each subset.

Improved Target Internal Difference Algorithm. The target internal difference algorithm [9] is a generalization of the target difference algorithm [8], which enables internal differentials to be used to launch collision attacks on 5-round SHA-3. In [9], the output of TIDA is a subspace of the initial messages whose dimension is not enough to produce a collision. Therefore, in the attack on SHA3-256, the TIDA is run multiple times to output enough messages.

The TIDA has two phases, where in the first phase (called the difference phase) it solves a system E composed of linear equations about capacity and all

Sboxes to fix the initial internal difference, and in the second phase (called the value phase) it outputs the affine subspace of the initial message by solving a linear system. If 1-block messages are used as the initial messages, in the difference phase, the algorithm will change affine subspaces of the input differences until E is consistent. In our attacks on 5-round `SHAKE256`, we input 2-block messages into the sponge function as the initial messages. Thus, we just need to change the value of the first block to make E consistent. Since the affine subspaces corresponding to the input differences of Sboxes will lead to distinct transition probabilities of internal differences, after introducing 2-block messages, we select a specific set of affine subspaces which maximize the expected transition probability so that the number of characteristics to launch collision attacks could be reduced. We combine the two phases in the improved TIDA, and reduce the number of iterations for the second phase by using the partial solutions of E. By using the internal differentials with our techniques, we can launch collision attacks on all variants of `SHA-3`. Our attacks on each variant can reach the most number of rounds at present, except for `SHAKE128`, though in some cases other attacks reaching the same number of rounds are faster. Specially, the complexity of our attack on 4-round `SHA3-512` is 2^{237}, and it is the best result at present. For 4-round `SHA3-384`, the complexity of our attack is 2^{76}, which is much lower than the result of 2^{147} using the same type of cryptanalysis in [9]. The collision attack on `SHAKE256` reaches to 5 rounds for the first time.

Organization. The rest of the paper is organized as follows. In Section 2, we describe the `SHA-3` hash function. In Section 3, some notations used in this paper are given, followed by the overview of our collision attacks and a variant of birthday attack. In Section 4, we give the basic concepts of internal differentials and some new concepts. Section 5 presents the framework of attacks and detailed explanations over our techniques. In Section 6, the details and results of our attack are given. We conclude the paper in Section 7. The internal difference characteristics are postponed to Appendix.

2 Description of SHA-3

In this section, we give a brief description of the sponge construction and the `SHA-3` hash function, *i.e.*, the Keccak hash function. The sponge construction proceeds in two phases: absorbing phase and squeezing phase, as shown in Figure 1. The message is firstly padded by appending a bit string of 10*1, where 0* represents a shortest string of 0's so that the length of padded message is multiple of r, and cut into r-bit blocks. The b-bit internal state is initialized to be all zeros. In absorbing phase, each message block is XORed into the first r bits of the current state, and then it is applied a fixed permutation to the entire b-bit state. The sponge construction switches to the squeezing phase after all message blocks are processed. In this phase, the first r bits of the state are returned as output and the permutation is applied in each iteration. This process is repeated until all d digest bits are produced. The four instances of `SHA-3` family named

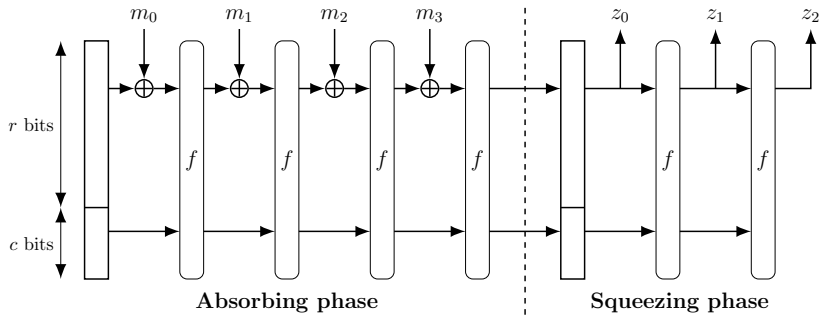


Fig. 1. The sponge construction

`SHA3-d` are defined from Keccak[c] by appending a two-bit suffix '01' to the message, where $b = 1600$, $c = 2d$ and $d \in \{224, 256, 384, 512\}$. After that, the padding of Keccak is applied. `SHAKE128` and `SHAKE256` are two instances with the capacity $c = 256$ or 512 and any output length d, and the original message M is appended with an additional 4-bit suffix '1111' before applying the padding rule, for any output length. The suffixes "128" and "256" indicate the security strengths that these two functions can generally support. We summarize security strengths of the `SHA-3` functions in Table 2.

Function	**Output Size**	**Security Strengths in Bits**		
		Collision	**Preimage**	**2nd Preimage**
`SHA3-224`	224	112	224	224
`SHA3-256`	256	128	256	256
`SHA3-384`	384	192	384	384
`SHA3-512`	512	256	512	512
`SHAKE128`	d	$\min(d/2, 128)$	$\geq \min(d, 128)$	$\min(d, 128)$
`SHAKE256`	d	$\min(d/2, 256)$	$\geq \min(d, 256)$	$\min(d, 256)$

Table 2. Security strengths of `SHA-3` functions

The Keccak permutation has 24 rounds, which operates on the 1600-bit state s that can be viewed as a 3-dimensional array of bits. One bit of the state at position (x, y, z) is noted as $A[x][y][z]$, where $0 \leq x, y < 5$ and $0 \leq z < 64$. The mapping between the bits of s and those of A is $s[64(5y+x)+z] = A[x][y][z]$. Defined by the designers, $A[\cdot][y][z]$ is a row, $A[x][\cdot][z]$ is a column, and $A[x][y][\cdot]$ is a lane; $A[x][\cdot][\cdot]$ is a sheet, $A[\cdot][y][\cdot]$ is a plane, and $A[\cdot][\cdot][z]$ is a slice.

There are five mappings in each round of the permutation:

$$\theta : A[x][y][z] \leftarrow A[x][y][z] + \sum_{y'=0}^{4} A[x-1][y'][z] + \sum_{y'=0}^{4} A[x+1][y'][z-1].$$

$$\rho : A[x][y][z] \leftarrow A[x][y][z + T(x, y)], \text{where } T(x, y) \text{ is a predefined constant.}$$

$$\pi : A[x][y][z] \leftarrow A[x'][y'][z], \text{where } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

χ :$A[x][y][z] \leftarrow A[x][y][z] + (\neg(A[x+1][y][z])) \wedge A[x+2][y][z]$.

ι :$A \leftarrow A + RC[n_r]$, where $RC[n_r]$ is the round constants.

The addition and multiplication are in $GF(2)$. Since we analyse round-reduced variant with at most 5 rounds, we only give the first five round constants: 0000000000000001, 0000000000008082, 800000000000808a, 8000000080008000, 000000000000808b (given in hexadecimal using the little-endian format).

3 Overview of the Attack

3.1 Notations

We summarize the major notations to be used in this paper in Table 3. In this paper, the addition operation of Keccak's state is performed on $GF(2)$ or the linear space over $GF(2)$.

Notation	Description
c	Capacity of a sponge function
r	Rate of a sponge function
b	Width of a Keccak permutation in bits,$b = r + c$
d	Length of the digest in bits
p	Number of fixed bits in the initial state due to padding
i	Period of a symmetric state or an internal difference
$\theta, \rho, \pi, \chi, \iota$	The five mappings that comprise a round.
L	Composition of θ, ρ, π and its inverse denoted by L^{-1}
$\mathtt{R}^j(\cdot)$	Keccak permutation reduced to the first j rounds
$\mathtt{S}(\cdot)$	5-bit Sbox operating on each row of Keccak state
$\delta_{in}, \delta_{out}$	5-bit input and output differences of an Sbox
$\overline{M}$	Padded message of M. Note that $\overline{M}$ is the last block in our attack
$M_0 \| \| M_1$	Concatenation of strings M_0 and M_1
$\alpha_i^{(j-1)}$	Input internal difference of the j-th round function with period i
$\beta_i^{(j-1)}$	Input internal difference of χ in the j-th round with period i
$\mathcal{A}_i^{(j-1)}$	Bit value vector before θ in the j-th round with period i
$\mathcal{B}_i^{(j-1)}$	Bit value vector before χ in the j-th round with period i
$\Delta(\cdot)$	Internal difference of one state
$v^{(j-1)}$	Canonical representative state of the internal difference in the j-th round

Table 3. The Major Notations in Our Attack

3.2 Overview of the Attack

In this section, we give an overview of our collision attacks. Based on the framework of Dinur *et al.* [9] and a variant of birthday attack, our collision attack consists of two parts, *i.e.*, a high probability internal differential characteristic and several collision subsets generated by the characteristic for finding collisions.

Given an $(n_r - 1.5)$-round internal differential characteristic, there are three stages in our n_r-round collision attacks:

- *Stage 1 — Selecting messages stage:* Obtain linear conditions from the 2-round internal differential characteristic and get several subspaces of messages passing the first 2 rounds.
- *Stage 2 — Collecting messages stage:* Compute the outputs after $(n_r - 1.5)$ rounds functions from the subspaces found in Stage 1, and store these outputs into different sets.
- *Stage 3 — Brust-force searching stage:* By brust force, find a collision from the outputs after target round of each set in Stage 2.

The collecting messages stage and the brute-force searching stage are simple, although they take up the main time complexity. Therefore, the core step of our attack is selecting messages to reduce the complexity of the collision searching stage. In [9], Dinur *et al.* use an algebraic method to reduce the workload of finding messages conforming to the first χ transition. In Section 5, we show a new method to select messages passing the first two rounds functions, which saves even more time complexity.

3.3 A Variant of Birthday Attack

When searching for collisions among the outputs of a hash function H, the birthday attack is the simple technique of selecting distinct inputs x_j for $j = 1, 2, \ldots$ randomly and checking for a collision among the $H(x_j)$ values. The probability that no collision is found after 2^t inputs is

$$(1 - 1/2^n)(1 - 2/2^n) \cdots (1 - (2^t - 1)/2^n) \approx e^{-2^t(2^t-1)/2^{n+1}} \tag{1}$$

where 2^n is the cardinality of the range of H [17]. A collision can be found with high probability for $t = n/2$. If t is much smaller than $n/2$, the probability of a collision being found is close to zero. But by repeating the process of selecting 2^t inputs randomly many times, we can also find a collision with high probability. This is a variant of birthday attack, which is reformulated as follows. Assume that the hash function H maps 2^k input subsets $S_1, \ldots, S_{2^k}$ into output subsets $D_1, \ldots, D_{2^k}$ (called collision subsets) and is a random function when it is confined to any set S_j, where S_j $(j = 1, \ldots, 2^k)$ and D_j $(j = 1, \ldots, 2^k)$ are both pairwise disjoint respectively, $|S_j| = 2^l, |D_j| = 2^m$ $(m > 2l)$. If a collision is found with a probability P for $t = n/2$ in Eq. (1), the expected number of the collision subsets to be searched is 2^w $(w \leq k)$ with the same success probability P. The relationship between l, m and w is shown as follows:

$$1 - P \approx (e^{-2^l(2^l-1)/2^{m+1}})^{2^w} \tag{2}$$

$$= e^{-2^{l+w}(2^l-1)/2^{m+1}} \tag{3}$$

then

$$2l + w = m. \tag{4}$$

For the randomly selected input x in the union of all S_j (denoted as S'), assume that we can determine which output subset $H(x)$ belongs to, but cannot determine the input subset corresponding to x. So the probability of $H(x)$ in subset

D_j is 2^{-k} for any $j \in \{1, \ldots, 2^k\}$. In order to ensure that each collision subset has at least two values on average, the number of randomly selected inputs is at least 2^{k+1}. Therefore, the total number N of inputs we need can be expressed as

$$N = \begin{cases} 2^{(m+w)/2} & k < m, \\ 2^{k+1} & k \geq m. \end{cases} \tag{5}$$

In the attack on 3-round `SHA3-512`, we use the internal differential characteristic ($k = 20, m = 40$) given in [4] for finding collisions. With 2^{33} random input messages, we can calculate $l = 33 - 20 = 13, w = 40 - 2 \cdot 13 = 14$. This means that we need to search 2^{14} collision subsets to find a collision with probability $(1 - P) \approx 0.4$. When $w = 15$, we can find a collision with a probability close to 1 according to Eq. (2). In this experiment, about 2^5 collision subsets produced collisions, which means that we will get one collision for every 2^{15} collision subsets searched. It can be seen that the experimental value of the collision subset number (w) is very close to the theoretical value.

Assume that H maps a set S of possible inputs into a set D of possible outputs and $S' = \bigcup_{j=1}^{2^k} S_j$, $D' = \bigcup_{j=1}^{2^k} D_j$. In our attack, take 4-round `SHA3-512` as an example (Figure 2). $|S| = 2^{1600}, |D| = 2^{512}, |S'| = 2^{252}, |D_j| = 2^{320}$. There are 2^{156} output subsets D_j, and the size of their union D' is $2^{156+320} = 2^{476}$. By using conditional internal differentials, the probability of transition from S' to D' is 1. We construct the hash table based on the size of D_j instead of D for collision search, and the search can be performed simultaneously in multiple collision subsets. The expected number of inputs to find a collision is 2^{238}.

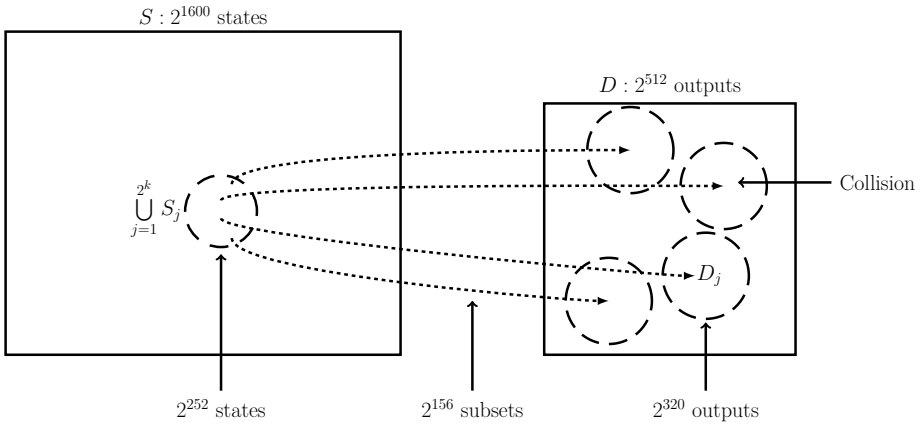


Fig. 2. A variant of birthday attack

4 Description of Internal Difference

In this section we first review the concepts proposed by Dinur *et al.* [9] in their generalization of internal differential, and then define a new metric that can be used to calculate the transition probability of the internal differential more accurately.

4.1 Internal Difference Sets and Representatives

An important property of Keccak is that if one state has period i in the z-axis (*i.e.*, satisfies $A[x][y][z] = A[x][y][(z+i) \bmod 64]$, for all (x, y, z)), then applying to it any of the θ, ρ, π, χ operations, still maintains the period. This state is called a *symmetric state*. Since the fundamental period corresponding to i is $\gcd(i, 64)$, we can redefine $i \in \{1, 2, 4, 8, 16, 32\}$. For $i = 16$, a symmetric state $A[x][y][z]$ is composed of four repetitions of slices 0–15. Each such sequence of slices (0–15, 16–31, 32–47, 48–63) is called a *consecutive slice set* or *CSS* in short. This definition can extend naturally to any $i \in \{1, 2, 4, 8, 16, 32\}$.

Note that the ι operation disturbs the symmetry because all round constants are not periodic. Namely, the round constants are not the same among the CSS.

In an internal differential characteristic, unlike standard differential analysis, the round constant affects the characteristic by introducing a difference between all CSS's. This difference then propagates through the other operations, and its development has to be further studied and controlled. To characterize the difference between general states and symmetric states, the internal difference is defined as follows. Given a period i, the set $\{v + u | u \text{ is symmetric}\}$ obtained by adding all symmetric states with period i to a single state v is called the *internal difference*, recorded as $[i, v]$. The *zero internal difference* $[i, \mathbf{0}]$ is exactly the set of all symmetric states, and other internal differences $[i, v]$ are cosets of $[i, \mathbf{0}]$ (satisfies $[i, v] = [i, \mathbf{0}] + v$). The state v is called the *representative state*, and all of $[i, v]$ can be regarded as the representative state. In this paper, we choose v satisfying $v[x][y][z] = 0$ $(z \in \{64 - i, \ldots, 63\})$ as the *canonical representative state* (exists uniquely in each internal difference). Since the internal difference is an affine space on $GF(2)$, the action of linear mappings on $[i, v]$ is determined by their action on the representative state. Namely, $\theta([i, v]) = [i, \theta(v)], \rho([i, v]) = [i, \rho(v)], \pi([i, v]) = [i, \pi(v)], \iota([i, v]) = [i, \iota(v)]$.

4.2 Transition Probability of Internal Difference

As in standard differential cryptanalysis, the output difference of the internal difference applying χ depends on the actual input, only if the input difference is zero internal difference, the output difference is unique and also a zero internal difference. In other words, we randomly select several states from the same internal difference as the inputs of χ, and the outputs may belong to different internal differences. As in standard differential analysis, when an internal difference is identified as an input to χ, we need to calculate its all possible output internal differences and the transition probability to a possible output internal

difference. For this purpose, we propose the concept of differential transition conditions in combination with the properties of the KECCAK Sbox.

Property 1. Given the input difference $\delta_{in} = (\delta_0, \ldots, \delta_4)^T$ of the 5-bit KECCAK Sbox, the output difference δ_{out} is determined by q $(2 \leq q \leq 4)$ linear conditions with respect to the actual input $x = (x_0, \ldots, x_4)^T$. The q linear conditions $\{l_t(x)\}_{t=0}^{q-1}$ (without constant terms) are called *differential transition conditions*. Equivalently, $\delta_{out} = \mathtt{S}(x) \oplus \mathtt{S}(x \oplus \delta_{in}) = C \cdot x \oplus \eta$, where $C \in \mathbb{F}_2^{5\times 5}$ is a matrix $(rank(C) \in \{2, 3, 4\})$ and $\eta \in \mathbb{F}_2^5$ is a constant vector. It can be easily verified that C and η can be represented by δ_{in} as

$$C = \begin{pmatrix} \delta_2 & \delta_1 & & & \\ & \delta_3 & \delta_2 & & \\ & & \delta_4 & \delta_3 & \\ \delta_4 & & & \delta_0 & \\ \delta_1 & \delta_0 & & & \end{pmatrix}, \eta = \mathtt{S}(\delta_{in}) = \begin{pmatrix} \delta_0 \oplus (\delta_1 \oplus 1)\delta_2 \\ \delta_1 \oplus (\delta_2 \oplus 1)\delta_3 \\ \delta_2 \oplus (\delta_3 \oplus 1)\delta_4 \\ \delta_3 \oplus (\delta_4 \oplus 1)\delta_0 \\ \delta_4 \oplus (\delta_0 \oplus 1)\delta_1 \end{pmatrix}.$$

Remark 1. Property 1 holds for the KECCAK Sbox due to its algebraic degree of 2, and a similar property applies to any Sbox with algebraic degree of 2.

Take $\delta_{in} = 0\text{x}3$ as an example, the output difference is

$$\delta_{out} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

The differential transition conditions are $\{l_0 = x_4, l_1 = x_2, l_2 = x_0 + x_1\}$, and their corresponding output differences are recorded in Table 4. We call the table containing differential transition conditions *difference conditions table* (DCT) of the Sbox. The new table constructed by assigning the differential transition conditions in DCT and recording their resulting output differentials is called *values of difference conditions table* (VDCT).

δ_{out}	0x0b	0x1b	0x0a	0x1a	0x03	0x13	0x02	0x12
(l_0, l_1, l_2)	$(0,0,0)$	$(0,0,1)$	$(0,1,0)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$	$(1,1,1)$

Table 4. The differential transition conditions of $\delta_{in} = 0\text{x}03$

In internal differential cryptanalysis, we call the set E composed of all differential transition conditions obtained from the canonical representative state of a internal difference as the differential transition conditions of the internal difference. The rank of E is called the *transition condition number*, and the transition condition number of the i-th round is denoted as k_i. If the transition condition

number of $[i, v]$ is k, there are 2^k possible output internal differences and the upper bound of transition probability is 2^{-k}. In procedure IDTC , we show the details of calculating transition condition number.

Procedure IDTC$(A, A', i, DCT, VDCT, E)$

Input: The input internal difference A before χ, the output internal difference A' of A, period i, `DCT`, `VDCT`, a linear equation system E.
Output: The updated linear equation system E.

for *each integer* $j \in [0, 320)$ **do**
  Obtain the input difference δ_{in} of the j-th Sbox from A.
  Obtain the output difference δ_{out} of the j-th Sbox from A'.
  if $\delta_{in} \neq 0$ **then**
    Obtain differential transition conditions $E_0 = \{l_j(W)\}_{j=0}^{q-1}$ from `DCT`$[\delta_{in}]$, where W is a $25i$-bit variable vector.
    Obtain value of conditions $(\varepsilon_0, \ldots, \varepsilon_{q-1})$ from `VDCT`$[\delta_{in}][\delta_{out}]$.
    Update $E_0 = \{l_j(W) = \varepsilon_j\}_{j=0}^{q-1}$.
    $E = E \cup E_0$.
  end
end
return E

5 The Framework and Basic Techniques

In this section, we first present the basic framework of collision attacks on the `SHA-3` hash functions with reduced rounds. Then we show the details of techniques and some optimizations for improving our attack.

5.1 The Framework of the Attack

Following the variant of birthday attack, we adopt the strategy of first collecting messages and storing them in different sets, and then performing collision search in each set in turn. Figures 3 and 4 show the basic framework of our attack.

In Stage 1, the probability of the first two rounds in Figure 3 can be increased to 1 by algebraic methods (as described in next section), while losing exactly $(k_1 + k_2)$ degrees of freedom. In the collecting messages stage, the messages after two round functions and linear operations of the third round (*i.e.*, the first 2.5 rounds) are stored in different sets according to certain rules for the third stage. In Stage 3, the internal difference after 2.5 rounds results in 2^{k_3} different internal differences after the χ operation, and the set generated by the j-th internal difference after another round function is denoted as $D^{(j)}$. Since the probability of collision in the same $D^{(j)}$ is much higher than between different sets, we search for collisions in each $D^{(j)}$ in sequence. But in most cases, we conduct collision

search on subsets one by one before χ operation of the last round. In this case, the subsets can be regarded as collision subsets and are pairwise disjoint. Note that all the messages selected in Stage 1 enter a certain collision subset in Stage 3, so we establish an internal differential with probability 1.

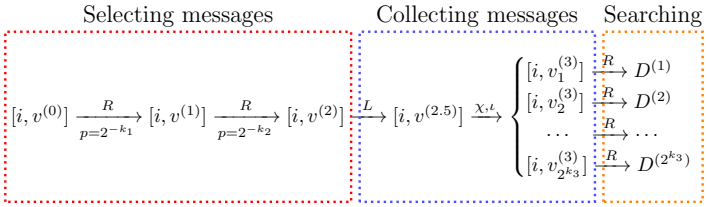


Fig. 3. The framework of 4-round collision attack

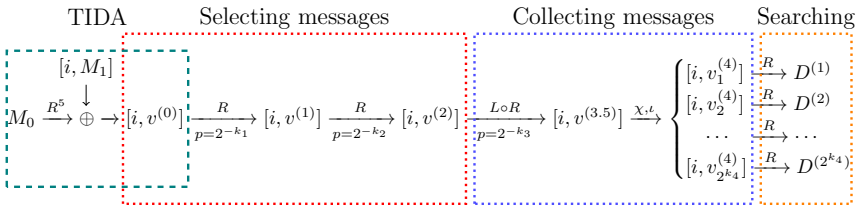


Fig. 4. The framework of 5-round collision attack

In the framework of collision attack on 5-round `SHA-3`, we select 2-block messages as inputs and use improved target internal difference algorithm (TIDA). The internal differential characteristic used in the attack is given in Characteristic 3 in Appendix A, which covers 2.5 rounds starting from the second round. TIDA is used to find the initial internal difference and the first messages to construct a complete internal differential characteristic with high probability. The full details and analysis of the attack are given in Section 5.5.

5.2 Finding Messages Conforming 2-round Internal Differential Characteristic

In this section, we present an algorithm for finding messages conforming to the first two χ transitions as depicted in Algorithm 1. For a known internal differential characteristic, as shown in Figure 3, let the states with period i in the internal difference be expressed as follows:

$$u^{(0)} = v^{(0)} + \mathcal{A}_i^{(0)}, u^{(0.5)} = v^{(0.5)} + \mathcal{B}_i^{(0)}, u^{(1)} = v^{(1)} + \mathcal{A}_i^{(1)}, u^{(1.5)} = v^{(1.5)} + \mathcal{B}_i^{(1)},$$

Algorithm 1: Finding Messages Passing 2-round Internal Differential

Input: An internal differential characteristic, and a period i.
Output: Initial messages conforming 2-round internal differential

$E_j = \emptyset, j \in \{0,1,2,3,4,5\}$
Set $\mathcal{A}_i^{(0)}, \mathcal{B}_i^{(0)}, \mathcal{A}_i^{(1)}, \mathcal{B}_i^{(1)}$ to being $25i$-variable vectors.
Add constraints on $\mathcal{A}_i^{(0)}$ such that the symmetric state generated by $\mathcal{A}_i^{(0)}$ satisfies the padding rule and equal to 0 in the capacity part.
Obtain the input internal difference $v^{(1.5)}$ and the output internal difference $v^{(2)} + \Delta(RC[2])$ of the second χ from the given characteristic.
IDTC$(v^{(1.5)}, v^{(2)} + \Delta(RC[2]), i, \texttt{DCT}, \texttt{VDCT}, E_1)$.
Transform $E_1(b_0^{(1)}, \ldots, b_{25\cdot i-1}^{(1)})$ to $E_1(a_0^{(1)}, \ldots, a_{25\cdot i-1}^{(1)})$.
for *each* $a_j^{(1)}$ *appearing in* $E_1(\mathcal{A}^{(1)})$ **do**
    $E_2 = E_2 \cup \{b^{(0)}_{\lfloor j/5i \rfloor \cdot 5i + [(j+i) \bmod 5i]} = x_t\}$.
    $t = t + 1$.
end
Obtain the input internal difference $v^{(0.5)}$ and the output internal difference $v^{(1)} + \Delta(RC[1])$ of the first χ from the given characteristic.
IDTC$(v^{(0.5)}, v^{(1)} + \Delta(RC[1]), i, \texttt{DCT}, \texttt{VDCT}, E_0)$.
$E_4 = E_0 \cup E_2$.
Transform $E_4(b_0^{(0)}, \ldots, b_{25\cdot i-1}^{(0)})$ to $E_4(a_0^{(0)}, \ldots, a_{25\cdot i-1}^{(0)})$.
Reduce E_4.
do
    Randomly assign values to the free variables in $\{x_j\}_{j=0}^{t-1}$.
    Obtain E_3 by subsituting E_2 into E_1.
    Transform $E_3(b_0^{(0)}, \ldots, b_{25\cdot i-1}^{(0)})$ to $E_3(a_0^{(0)}, \ldots, a_{25\cdot i-1}^{(0)})$.
    $E_5 = E_3 \cup E_4$.
while E_5 *is not consistent*;
Solve E_5 and obtain the initial messages.
return *initial messages*

where $u^{(j+0.5)}$ and $v^{(j+0.5)}$ are the state and the internal difference before the $(j+1)$-th χ transition respectively, $\mathcal{A}_i^{(0)}, \mathcal{B}_i^{(0)}, \mathcal{A}_i^{(1)}, \mathcal{B}_i^{(1)}$ are all symmetric state with period i and they have the following vector form (determined by their CSS):

$$\mathcal{A}_i^{(t)} = (a_0^{(t)}, \ldots, a_{25\cdot i-1}^{(t)}), \mathcal{B}_i^{(t)} = (b_0^{(t)}, \ldots, b_{25\cdot i-1}^{(t)}),$$

where $(a_{j\cdot i}^{(t)}, \ldots, a_{j\cdot i+i-1}^{(t)})$ are the first i bits of the j-th lane of $\mathcal{A}^{(t)}$, and $\mathcal{A}_i^{(0)}$ satisfies the padding rule and equals to 0 in the capacity part. In order to pass the χ operations of the first two rounds with probability 1, we should find $\mathcal{A}_i^{(0)}$ such that $\mathcal{B}_i^{(0)}$ and $\mathcal{B}_i^{(1)}$ satisfy the respective differential transition conditions (denoted as $E_0(\mathcal{B}_i^{(0)})$ and $E_1(\mathcal{B}_i^{(1)})$ appearing in Algorithm 1). $E_0(\mathcal{B}_i^{(0)})$ and $E_1(\mathcal{B}_i^{(1)})$ can be transformed into $E_0(\mathcal{A}_i^{(0)})$ and $E_1(\mathcal{A}_i^{(1)})$ by the linear operation $(L(\mathcal{A}_i^{(0)}) = \mathcal{B}_i^{(0)}, L(\mathcal{A}_i^{(1)}) = \mathcal{B}_i^{(1)})$. Since $\mathcal{A}_i^{(1)} = \chi(\mathcal{B}_i^{(0)})$, $a_j^{(1)} = b_j^{(0)} \oplus (b_{j+i}^{(0)} \oplus 1) \cdot$

$b^{(0)}_{j+2i}$, we regard $b^{(0)}_{j+i}$ as a variable x, then

$$a^{(1)}_j = \begin{cases} b^{(0)}_j \oplus b^{(0)}_{j+2i} & , x = 0, \\ b^{(0)}_j & , x = 1. \end{cases}$$

In general, all bits $b^{(0)}_{j+i}$ corresponding to the bits $a^{(1)}_j$ appearing in $E_1(\mathcal{A}^{(1)}_i)$ are set as intermediate variables $\{x_t\}_{t\in I}$ (I is index set), and the system composed of $b^{(0)}_{j+i} = x_t$ is E_2. Noting that each x_t is the value of a bit $b^{(0)}_j$, it is actually a linear equation about $\mathcal{A}^{(0)}_i$. So $\{x_t\}_{t\in I}$ may be linearly independent. We call the variables in the maximal linearly independent system of $\{x_t\}_{t\in I}$ are the free variables (short for free intermediate variables). After assigning a value to $\{x_t\}_{t\in I}$, E_1 can be expressed as a linear system of $\mathcal{B}^{(0)}_i$, which is also a linear system of $\mathcal{A}^{(0)}_i$, denoted as E_3. For the convenience of calculation, we combine the transition conditions of the first round and assignment conditions ($E_0 \cup E_2$). Then, by solving the linear system $E_0 \cup E_2 \cup E_3$ and XORing each solution with $v^{(0)}$, the message M conforming 2-round internal differential characteristic is obtained. In fact, all messages M satisfying the first two χ transitions can be found in this way, because we can traverse all possible values of $\{x\}_{t\in I}$. Therefore, we lose $(k_1 + k_2)$ degrees of freedom in total. We need to remove linear related conditions in the linear system and $\{x_t\}_{t\in I}$, the details of algorithm are shown in Algorithm 1. As a simple example, in Figure 5, we set $k_1 = 3, k_2 = 2$, and the number of free variables is $t = 3$. If the total number of initial messages is 2^n, the differential transition conditions of the first round divides the message spaces into eight subspaces, one of which (named S_1) conforms the first χ transition. Each assignment of the free variables divides the space S into $2^{k_3} = 8$ subspaces, and also divides S_1 into $2^t = 8$ parts (named S_2). The second round differential transition conditions divide each S_2 into $2^{k_2} = 4$ subspaces, one of which (named S_3, as shown by the shadow) conforming the second χ transition. The expected size of S_3 is 2^{n-8}. After all possible values of the free variables are retrieved, about 2^{n-5} messages will conform the first two χ transitions.

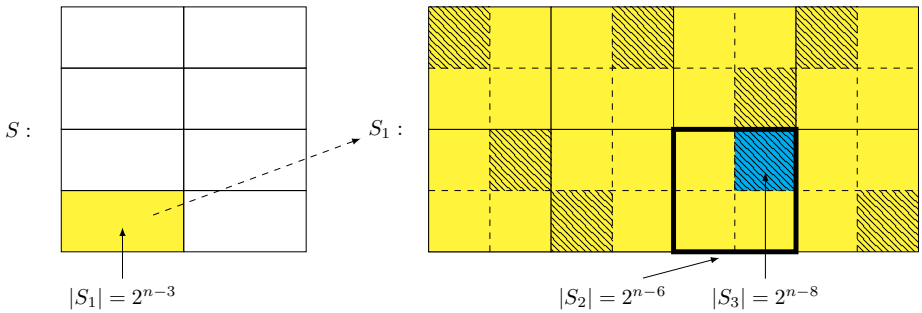


Fig. 5. Message subspaces S and S_1

5.3 Collecting Messages Belonging to Different Internal Difference

We give the details of Stage 2, which collect and store messages into different sets to determine which internal difference the state belongs to after the last linear operation.

For 4-round collision attacks, the values of differential transition conditions of the third round are denoted as $E(\mathcal{B}_i^{(2)}) = \{l_t(\mathcal{B}_i^{(2)})\}_{t=0}^{k_3-1}$, where $\mathcal{B}_i^{(2)} = \mathtt{R}^{(2.5)}(M) \oplus v^{(2.5)}$ (M is the message output by Algorithm 1). Then M is stored in $D^{(index)}$ with $index = \sum_{t=0}^{k_3-1} l_t \cdot 2^t$, where $D^{(j)}$ corresponds to the j-th output internal difference. In 5-round collision attacks, we calculate the corresponding data for the 4-th round instead. The details of this step for 4-round collision attacks are shown in Algorithm 2. In order to apply the variant of birthday attack and reduce complexity, in most cases, we will search collisions before the last χ operation, where $D^{(j)}$ is the internal difference of $(n_r - 0.5)$ rounds for n_r-round collision attacks.

Algorithm 2: Store Messages into Set $D^{(i)}$

Input: Message M output from Algorithm 1, subset family $\mathcal{D} = \{D^{(0)}, \ldots, D^{(2^{k_3}-1)}\}$, 2.5-round internal differential characteristic.

Output: Subset family $\mathcal{D} = \{D^{(0)}, \ldots, D^{(2^{k_3}-1)}\}$.

```
Compute B_i^(2) = R^(2.5)(M) ⊕ Δ(R^(2.5)(M)) and set E = ∅.
for each integer j ∈ [0, 320) do
    Get the input difference δ_in of the j-th Sbox from Δ(R^(3.5)(M)).
    if δ_in ≠ 0 then
        Get differential ransition conditions E_1 from DCT[δ_in].
        E = E ∪ E_1.
    end
end
Reduce E = {l_0, ..., l_{k_3-1}}.
Compute (η_0, ..., η_{k_3-1}) = (l_0(B_i^(2)), ..., l_{k_3-1}(B_i^(2))).
index = Σ_{j=0}^{k_3-1} η_j · 2^j.
D^(index) = D^(index) ∪ {M}.
return D
```

5.4 Bounding the Size of Collision Subset

In the variant of birthday attack, the size of the collision subset determines the time complexity. We use the method in [9] to bound the size of the collision subset.

The collision subset is essentially the output of $\iota \circ \chi$, and the subset of internal difference is the input of χ. Obviously, operation ι can be ignored since it does

not affect the size of the collision subset. A property of χ is that it is applied independently on each plane of the state and in particular, maps each plane to itself. When the output is the first d bits of the final state, we bound the number of its possible values by computing the size of which the internal difference is projected onto its first $320\lceil n/320 \rceil$ bits. For $d = 384$ and $d = 448$, the size of collision subset can be bounded more accurately: each output bit $A[x][y][z]$ of χ depends on the 3 input bits $A[x][y][z], A[x+1][y][z]$ and $A[x+2][y][z]$. Therefore, each output lane $A[x][y][\cdot]$ of χ only depends on the 3 input lanes $A[x][y][\cdot], A[x+1][y][\cdot]$ and $A[x+2][y][\cdot]$. In the case of $d = 384$, the first 5 lanes are mapped to themselves by χ, and the remaining lane depends on only 3 consecutive lanes. For $d = 448$, the 7 output lanes depend on 9 input lanes.

For d-bit collision subset, given that it depends only on the first d' bits before the χ mapping. As the period is i, each lane can assume at most 2^i values. Thus, for $256 \leq d \leq 320$ ($d' = 320$) we obtain a basic bound of 2^{5i} and for $512 \leq d \leq 640$ ($d' = 640$) we obtain a basic bound of 2^{10i}. For $d = 384$ ($d' = 512$) and $d = 448$ ($d' = 576$), the computation can be divided into two parts: the first 5 lanes of the output can assume at most 2^{5i} values, and the remaining bits can assume at most $\min(2^{64}, 2^{3i})$ and $\min(2^{128}, 2^{4i})$ values. From the previous analysis, we obtain a bound of $2^{5i} \cdot \min(2^{64}, 2^{3i})$ for $d = 384$ and a bound of 2^{9i} for $d = 448$. Clearly, the bound only depends on i and d' (which determined by d).

5.5 The Target Internal Difference Algorithm

Dinur *et al.* [8] explored the *target difference algorithm* (TDA) to link a differential characteristic to the initial state of the Keccak permutation. In [9], they generalized this method as a variant for internal differential cryptanalysis, which is called *target internal difference algorithm* (TIDA). Analogously, the TIDA is used to link an internal differential characteristic to the initial state, using one permutation round. The initial internal difference of the internal differential characteristic is called the *target internal difference*, denoted by Δ_{T^*}. The outputs of the algorithm are single-block messages whose internal difference after one permutation round is Δ_{T^*}.

In this section, we focus on $\Delta_T = \Delta_{T^*} \oplus \Delta(RC[1])$ and set it as the target internal difference. We modify the output of the algorithm to 2-block messages and apply it to our 5-round collision attack. Namely, given a target internal difference Δ_T, we use TIDA (Algorithm 3) to find 2-block messages $(M_0||M_1)$ such that

$$\Delta[\chi \circ L(\mathtt{R}^5(M_0||0^c) \oplus (\overline{M_1}||0^c))] = \Delta_T. \tag{6}$$

The first step in constructing the algorithm is to choose period $i = 32$ (and all operations of internal difference are performed on the state that the length of each lane is 32) so that the equations in the algorithm will have enough degrees of freedom. Then, according to a property of χ provided in [8] (as shown in Property 2), the input internal differences corresponding to Δ_T span several affine subspaces.

Property 2 ([8]). For a non-zero 5-bit output difference δ_{out} to a KECCAK Sbox, the set of possible input differences, $\{\delta_{in}|DDT(\delta_{in}, \delta_{out}) > 0\}$, contains at least 5 (and up to 17) 2-dimensional affine subspaces.

Algorithm 3: Target Internal Difference Algorithm (TIDA)

Input: Target internal difference Δ_T, target transition condition number k_T, and target number of rounds n_r.
Output: 2-block message (M_0, M_1), initial internal difference Δ_I, k_1 of $L(\Delta_I)$

Set $E_\Delta = \emptyset$ and $\Delta_I^\star = L^{-1}(W)$ with a variable vector $W = (w_0, \ldots, w_{799})$.
for *each non-active Sbox of* Δ_T **do**
  $E_\Delta = E_\Delta \cup \{w_{5j} = 0, \ldots, w_{5j+4} = 0\}$, for the j-th Sbox.
end
for *each active Sbox of* Δ_T **do**
  Select one 2-dim affine subspace from δ_{in} according to Δ_T.
  Add 3 affine equations to E_Δ according to the 2-dim affine subspace.
end
do
  Randomly select M_0 and compute $\Delta(\mathtt{R}^{n_r}(M_0))$.
  $E_c = \{\Delta_I^\star[j] = \Delta(\mathtt{R}^{n_r}(M_0))[j]\}_{j=800-p-c/2}^{799}$
  // $\Delta_T^\star[j]$ is the j-th bit of $\Delta_T^\star$, the same in $\Delta(\mathtt{R}^{n_r}(M_0))$
while $E_\Delta \cup E_c$ *is not consistent*;
Randomly select a solution Δ_I of $E_\Delta \cup E_c$.
Calculate k_1 corresponding to Δ_I.
if $k_1 < k_T$ **then**
  Set $E_0 = \emptyset$.
  IDTC$(L(\Delta_I), \Delta_T, 32, \mathtt{DCT}, \mathtt{VDCT}, E_0)$.
  if E_0 *is consistent* **then**
    Break.
  end
end
$M_1 = \mathtt{R}^{n_r}(M_0) \oplus \Delta_I \oplus sym$, where sym is a random symmetric state satisfying padding rule and equals to 0 in the capacity part.
return $(M_0, M_1), \Delta_I, k_1$

We select an 800-dimension subspace (named W) and map it to the initial internal difference $\Delta_I^\star = L^{-1}(W)$. The first block M_0 is randomly selected until it satisfies that there is a internal difference Δ_I in $L^{-1}(W)$ equal to the internal difference of $\mathtt{R}^5(M_0||0^c)$ on the padding and capacity bits. Note that the internal difference Δ_I is obtained after the first block M_0 is determined, we can calculate the transition condition number k_1 of $L(\Delta_I)$. In order to obtain smaller k_1, we select more M_1 to get several Δ_I and choose the best input internal difference. The remaining bits of Δ_I can be satisfied by modifying the value of the second block M_1. However, due to the insufficient degrees of freedom of the second block, Eq.(1) may not have a solution. In other words, if the transition condition number of $L(\Delta_I)$ is k_1, it results in 2^{k_1} different output internal differences. But

the degree of freedom of M_1 is less than 1600, not all Sboxes have enough inputs to generate all possible outputs. The number of actual output internal differences may be less than 2^{k_1}, and Δ_T may not be among them. When this occurs, we need to redefine the first block M_0. The details are given in Algorithm 3, and the above process is equivalent to solving equation systems.

6 Results and Complexity Analysis

In this section, we present the details and experimental results of our collision attack on different versions of `SHA-3`. Given an internal differential characteristic spanning $(n_r - 1.5)$ rounds of the Keccak permutation, a collision attack on n_r-round `SHA-3` consists of the following steps:

1. Construct linear equation systems according to the differential transition conditions of the first two rounds and solve them to get enough initial messages.
2. Pick an arbitrary message obtained in Step 1 and calculate its internal difference after $(n_r - 1.5)$ rounds. If the internal differential characteristic is satisfied, store the message into the corresponding subset. Otherwise, discard the message and go back to Step 2 until collect enough states.
3. Choose an unselected subset.
   (a) Pick a state and store its output after the n_r-th round in a hash table (along with its initial message) and check for a collision.
   (b) If a collision is found, stop and output it. Else if all states are chosen and there is no collision, go back to Step 3. Otherwise, go back to (a).

6.1 Collision Attacks on 4-round SHA3-384 and SHAKE256

For 4-round `SHA3-384` and `SHAKE256`, we use the same 2.5-round characteristic (Characteristic 1 in Appendix A). We choose $i = 8$ and use the techniques of Section 5.1-5.3. The transition condition numbers of the characteristic are $(k_1, k_2, k_3) = (11, 8, 78)$. The size of $\{x_t\}_{t\in I}$ is 80. There are 59 free variables in `SHA3-384` and 72 free variables in `SHAKE256`. Therefore, in Stage 1, the number of assignments to $\{x_t\}_{t\in I}$ will not exceed 2^{72}.

For `SHA3-384` and `SHAKE256` ($d \leq 448$), the size of each collision subset is less than k_3. In order to launch collision attack, we need to ensure that there are at least two messages in each collision subset on average. So in the first stage we select 2^{79} messages conforming to the first two χ transitions. For $d = 448$, we can find a collision with good probability by searching 2^{71} collision subsets, and we need fewer collision subsets for smaller d. So the complexity is mainly caused by the first two stages, and it can be reduced by placing eight messages in each state. More specifically, if the initial state can be expressed as $u^{(0)} = \mathcal{A}^{(0)} + v^{(0)}$, where $\mathcal{A}^{(0)}$ is a fully symmetric state with period $i = 8$ and $v^{(0)}$ is the initial internal difference in Characteristic 1. Rewrite $\mathcal{A}^{(0)}$ as follows:

$$\mathcal{A}^{(0)} = (\mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}, \mathcal{A}_8^{(0)}),$$

where $\mathcal{A}_8^{(0)}$ is a CSS of $\mathcal{A}^{(0)}$, and $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_j)$ means each lane of the state $\mathcal{A}$ is the concatenation of the lane corresponding to $\mathcal{A}_k$ $(k = 1, \ldots, j)$. The linear operation L acting on the $(5 \times 5 \times 8)$-bit state is denoted by $L_{[8]}$. Due to the property of Keccak mentioned in Section 4.1, $L(\mathcal{A}^{(0)})$ has the following expression:

$$L(\mathcal{A}^{(0)}) = (L_{[8]}(\mathcal{A}_8^{(0)}), \ldots, L_{[8]}(\mathcal{A}_8^{(0)})).$$

In Characteristic 1, the canonical representative states of the first two rounds are fixed-points of θ. Namely, the states are in the CP-kernel. As a result, The first 2.5 rounds of operation of the messages $u^{(0)}$ selected in Stage 1 can be simplified:

$$\begin{aligned}
L(u^{(0)}) = \quad & L(\mathcal{A}^{(0)}) + L(v^{(0)}) && = (L_{[8]}(\mathcal{A}_8^{(0)}), \ldots, L_{[8]}(\mathcal{A}_8^{(0)})) + L(v^{(0)}) \\
u^{(0.5)} = \quad & \mathcal{B}^{(0)} + v^{(0.5)} && = (\mathcal{B}_8^{(0)}, \ldots, \mathcal{B}_8^{(0)}) + v^{(0.5)} \\
\iota \circ \chi(u^{(0.5)}) = \chi(\mathcal{B}^{(0)} + v^{(0.5)}) + RC'[1] &= (\mathcal{A}_8^{(1)}, \ldots, \mathcal{A}_8^{(1)}) + v^{(1)}.
\end{aligned}$$

Since the last 8 bits in each lane of the canonical representative states are zero, the last CSS of $\mathcal{B}^{(0)} + v^{(0.5)}$ is $\mathcal{B}_8^{(0)}$ which means $\mathcal{A}_8^{(1)} = \chi(\mathcal{B}_8^{(0)})$. In the second round:

$$\begin{aligned}
L(u^{(1)}) = \quad & L(\mathcal{A}^{(1)}) + L(v^{(1)}) && = (L_{[8]}(\mathcal{A}_8^{(1)}), \ldots, L_{[8]}(\mathcal{A}_8^{(1)})) + L(v^{(1)}) \\
u^{(1.5)} = \quad & \mathcal{B}^{(1)} + v^{(1.5)} && = (\mathcal{B}_8^{(1)}, \ldots, \mathcal{B}_8^{(1)}) + v^{(1.5)} \\
\iota \circ \chi(u^{(1.5)}) = \chi(\mathcal{B}^{(1)} + v^{(1.5)}) + RC'[2] &= (\mathcal{A}_8^{(2)}, \ldots, \mathcal{A}_8^{(2)}) + v^{(2)},
\end{aligned}$$

where $\mathcal{A}_8^{(2)} = \chi(\mathcal{B}_8^{(1)}), RC'[j] = \Delta(RC[j]), j \in \{1, 2\}$. And in the third round, $\mathcal{B}_8^{(2)} = L_{[8]}(\mathcal{A}_8^{(2)})$. Therefore, in the collecting messages stage, we can simultaneously calculate eight messages $u_0^{(0)}, \ldots, u_7^{(0)}$ and calculate their transition conditions of the third round in the following way:

$$\begin{aligned}
& L \circ \chi \circ L \circ \chi \circ L(\mathcal{A}_{8,0}^{(0)}, \ldots, \mathcal{A}_{8,7}^{(0)}) \\
= & (L_{[8]} \circ \chi \circ L_{[8]} \circ \chi \circ L_{[8]}(\mathcal{A}_{8,0}^{(0)}), \ldots, L_{[8]} \circ \chi \circ L_{[8]} \circ \chi \circ L_{[8]}(\mathcal{A}_{8,7}^{(0)})) \\
= & (\mathcal{B}_{8,0}^{(2)}, \mathcal{B}_{8,1}^{(2)}, \mathcal{B}_{8,2}^{(2)}, \mathcal{B}_{8,3}^{(2)}, \mathcal{B}_{8,4}^{(2)}, \mathcal{B}_{8,5}^{(2)}, \mathcal{B}_{8,6}^{(2)}, \mathcal{B}_{8,7}^{(2)}),
\end{aligned}$$

where $\mathcal{A}_{8,j}^{(0)}$ is a CSS of the symmetric state of $u_j^{(0)}$. From each $\mathcal{B}_{8,j}^{(2)}$ we can obtain the transition conditions and calculate the index w_j of its internal difference, then store it into the set $D^{(w_j)}$ (the w_j-th output internal difference after the third χ).

After collecting messages stage, we compute the outputs of each set after $L \circ \chi$ (the state also can be divided into eight parts to simultaneously calculate eight symmetric states) and find a collision. According to the analysis in Section 5.4, in order to produce the final collision, the location of the collision needs to be determined. For $256 \leq d \leq 320$, we have to find collisions on the first 5 lanes before the last χ. For $d = 384$, the collision location is the first 8 lanes. And for $d = 448$ and $512 \leq d \leq 640$, the number is 9 and 10. Taking $d = 320$

as an example, if there is a collision $(\mathcal{B}_{8,p}^{(3)}, \mathcal{B}_{8,q}^{(3)})$ on the first 5 lanes, then for any internal difference $\Delta = (\Delta_0, \ldots, \Delta_7)$, $\iota \circ \chi(\mathcal{B}_{8,p}^{(3)} + \Delta_0, \ldots, \mathcal{B}_{8,p}^{(3)} + \Delta_7)$ and $\iota \circ \chi(\mathcal{B}_{8,q}^{(3)} + \Delta_0, \ldots, \mathcal{B}_{8,q}^{(3)} + \Delta_7)$ will be equivalent on the first 320 bits. Therefore, we can calculate their initial symmetric states $\mathcal{A}_{8,p}^{(1)}$ and $\mathcal{A}_{8,q}^{(1)}$, and get the initial messages M_p and M_q by XORing with the initial internal difference. After the previous analysis, we can find a collision $\mathtt{R}^4(M_p)$ and $\mathtt{R}^4(M_q)$ on the first 320 bits. Note that this technique can be extended naturally to any $i \in \{1, 2, 4, 8, 16, 32\}$. And for $d > 448$, we also need 2^{79} messages to launch attack from the Eq. (1). In all cases, the expected complexity is $2^{79-3} = 2^{76}$. This is 2^{71} times faster than the internal differential attack by Dinur *et al.* [9] for 4-round `SHA3-384`.

6.2 A Collision Attack on 4-round SHA3-512

For 4-round `SHA3-512`, we choose $i = 32$ and used the 2.5-round internal difference characteristic given in Characteristic 2 in Appendix A. The transition condition numbers of the characteristic are $(k_1, k_2, k_3) = (16, 16, 170)$. For $k_3 = 170$, there are 2^{170} different internal differences after the χ mapping of the third round. These internal differences actually compose a 170-dimensional affine space (denoted as U) over $GF(2)$. The projection of the affine space $L \circ \iota(U)$ on the first 10 lanes consists of internal differences which are projected to the first 10 lanes, and its dimension is 156. For $d = 512$, we collect the messages with the same internal difference in the first 10 lanes into a set. Then we need fewer messages for generating collisions.

In selecting messages stage, the size of the variable set $\{x_t\}_{t \in I}$ is 172, and there are 138 free variables. Therefore, we obtain $2^{284-(138+16+16)} = 2^{114}$ expected messages after the variables of $\{x_t\}_{t \in I}$ are assigned each time. Since the size of collision subset is $2^{10 \cdot 32} = 2^{320}$, we need to choose $2^{(320+156)/2} = 2^{238}$ messages, which conforming the first two rounds internal differential characteristic, in order to find a collision with high probability. In order to obtain enough messages, we make $2^{238-114} = 2^{124}$ different assignments to $\{x_t\}_{t \in I}$.

In the next stage, we compute the outputs after 3.5 rounds functions from the subspaces found in Stage 1. Then collect the outputs with the same internal difference in the first 10 lanes into a set.

In searching stage, the sets obtained from the previous step are considered as collision subsets, and we search for collisions on the first 10 lanes in each collision subset in turn. If there is a collision $(\mathcal{B}_{32,0}^{(3)}, \mathcal{B}_{32,1}^{(3)})$ in a collision subset with Δ (projection of the canonical representative state on the first 10 lanes), it will result in a collision $\chi(\widetilde{\mathcal{B}}_{32,0}^{(3)}, \widetilde{\mathcal{B}}_{32,0}^{(3)} \oplus \Delta) = \chi(\widetilde{\mathcal{B}}_{32,1}^{(3)}, \widetilde{\mathcal{B}}_{32,1}^{(3)} \oplus \Delta)$ after 4-round, where $\widetilde{\mathcal{B}}_{32,j}^{(3)}$ is the projection of $\mathcal{B}_{32,j}^{(3)}$ on the first 10 lanes for $j = 0, 1$. In fact, the value of this internal difference does not need to be calculated. We can use the inverse operation to obtain the initial messages corresponding to $\mathcal{B}_{32,0}^{(3)}$ and $\mathcal{B}_{32,1}^{(3)}$, and then calculate the outputs after 4-round function, following Section 6.1.

Since period $i = 32$ is half the lane size, we can put the first CSS of two completely symmetric states in each state to represent the corresponding message

for calculation (as in Section 6.1). The expected time complexity of calculating the output of all messages after 4-round permutation is bounded by $2^{238-1} = 2^{237}$, and the complexity caused by assignment to $\{x_t\}_{t\in I}$ can be ignored. This is 2^{19} times faster than the general birthday attack.

6.3 A Collision Attack on 5-round SHAKE256

In this section, we present a collision attack on 5-round `SHAKE256`. Our attack uses internal differential characteristic given in Characteristic 3 in Appendix A, which covers 2.5 rounds satrting from the second round. The transition condition numbers of the characteristic are $(k_2, k_3, k_4) = (21, 18, 16)$. For $[32, \iota^{-1}(v^{(1)})]$, there 129 active Sboxes and 31 non-active Sboxes. Each active Sbox provides 3 linear equations, each non-active Sbox provides 5 linear equations, and the first message block provides 262 linear equations. The size of the linear system used to solve the input internal difference $L(\Delta_I)$ is 804, and the rank is 779. Therefore, we can obtain a consistent linear system $E_\Delta \cup E_c$ by randomly select 2^{25} the first blocks on average. Since the number of variables in the linear system is 800 and the rank is 779, each consistent linear system $E_\Delta \cup E_c$ has 2^{21} solutions (Δ_I). For $i = 32$, the internal differential transition conditions number is the sum of the transition condition numbers of all Sboxes of the canonical representative state. Let $k(S_j)$ be the transition condition number of the j-th Sbox for Δ_T, then we can calculate the expectation of k_1:

$$\mathtt{E}(k_1) = \sum_{j=0}^{159} \mathtt{E}(k(S_j)). \tag{7}$$

Assume that $\delta_{out}^{(j)}$ is the output difference of S_j and the 2-dimension affine subspace of its possible input differences we choose is $\{\delta_0, \delta_1, \delta_2, \delta_3\}$. For input difference $\delta_{in}^{(j)}$, its transition condition number is the rank of the subspace formed by all possible output differences. So the expectation of $k(S_j)$ is expressed as:

$$\mathtt{E}(k(S_j)) = \frac{1}{4}\sum_{j=0}^{3}(5 - \log_2 \mathtt{DDT}(\delta_{in}^{(j)}, \delta_{out}^{(j)})). \tag{8}$$

The expected transition condition number of Δ_I is 410.5, so we can easily obtain massive characteristics with $k_1 < 400$ to get enough messages. The smallest k_1 we have found is 375. We also use the technique in Section 6.1 to put two messages in the same state when performing the attack, which reduces the complexity on average by half. In selecting messages stage, the size of $\{x_t\}_{t\in I}$ is 224 and there are 93 free variables for the characteristic. In collecting messages stage, we store the states after 3.5 rounds in different subsets, which are the output internal differences of states after the fourth χ operation. In searching stage, for $d = 384$, the collision subsets are the outputs after 5-th round of subsets in Stage 2. In this case, the size of each collision subset is bounded by $2^{32\cdot 5+64} = 2^{224}$ and the collision subsets are pairwise disjoint. Actually, due to χ acting on the first 5 lanes

is bijection, the projection of any collision subset and its internal difference before the last χ is one to one on the first 320 bits. It can be seen from Characteristic 3 that there are $2^{k_3} = 2^{16}$ different internal differences, and their projections on the first 5 lanes can be verified to be different from each other. When conducting collision search, we need to calculate the complete state. So the time complexity in total is $2^{18+(224+16)/2} = 2^{138}$. If we take the internal difference of 4.5-round as the collision subset, in order to produce a collision after 5-round, we need to find a collision of the first 8 lanes before the last χ. The size of each collision subset is $2^{8 \cdot 32} = 2^{256}$, which will lead to more time complexity $2^{18+(256+16)/2} = 2^{154}$. For other d, we still search collisions before the last χ and use the techniques in Section 6.1 to reduce time complexity. In addition, since $p = 6$ and $c = 512$ in `SHAKE256`, we can obtain full-bit (1600-bit) collisions by searching for the last $(p + c)$-bit collisions. Assume that the outputs N and N' of 2-block messages $(M_0||M_1)$ and $(M'_0||M'_1)$ are equal in the last 518 bits. We can introduce the third block messages M_2 and M'_2 satisfying $N \oplus (\overline{M_2}||0^c) = N' \oplus (\overline{M'_2}||0^c)$ to obtain the full-bit collision $\mathtt{R}^5(N \oplus (\overline{M_2}||0^c)) = \mathtt{R}^5(N' \oplus (\overline{M'_2}||0^c))$. And the complexity of searching for full-bit collisions is the same as the case of $d \leq 640$.

The number of messages for collision attack and attack complexity are listed in Table 5 for different d. Note that the same internal differential characteristic is also applicable to the collision attacks on `SHA3-224`, `SHA3-256` and `SHAKE128`, where attack complexities are both the complexity corresponding to $d = 256$.

d	Number of characteristics	Complexity ($\log_2$)
$256 \sim 320$	1	$106 - 1 = 105$
384	1	138
448	2^{63}	$170 - 1 = 169$
≥ 512	2^{79}	$186 - 1 = 185$

Table 5. The parameters of characteristics and complexity

6.4 Summary of Collision Attacks

We summarize different versions of collision attacks in Table 6. For 4-round `SHA-3`, the first two canonical representative states of Characteristics we used is in CP-kernel. Since ι brings about extra internal differences, we do not find the characteristics where the canonical representative states of the first three rounds are all in CP-kernel. For 5-round `SHA-3`, we first determine the internal difference of the third round and then search for the appropriate target internal difference. Too few active Sboxes of the target internal difference will make it difficult to find the first block message M_0, and too many will make the differential transition probability of the first round too small, which will consume many degrees of freedom. Therefore, the number of active Sboxes for target internal difference is preferably between 128 and 135.

Target	n_r	i	DF†	k_1	k_2	k_3	k_4	Complexity ($\log_2$)
SHA3-384	4	8	$104-4=100$	11	8	78	-	$79-3=76$
SHA3-512	4	32	$288-4=284$	16	16	170	-	$238-1=237$
SHAKE256	4	8	$136-6=130$	16	8	78	-	$79-3=76$
SHA3-224/SHA3-256/SHAKE128	5	32	≥ 540	-	21	18	16	$106-1=105$
SHAKE256	5	32	$544-6=538$	-	21	18	16	≤ 185

† Degree of freedom of the initial message space.

Table 6. The parameters of characteristics and complexities

7 Conclusions

In this paper, we presented collision attacks on up to 5 rounds of all the six SHA-3 functions by developing conditional internal differential cryptanalysis. We introduced the differential transition conditions to describe the evolution of internal differences and estimate the transition probability more accurately. By solving the linear systems constructed with the difference transition conditions of two rounds, we obtained the messages that conform 2-round internal differential characteristic. According to the linear conditions on their middle states, these messages were divided into different subsets. We described a variant of birthday attack and applied it to these subsets for getting the collisions.

Compared with differential cryptanalysis, searching for internal differential characteristics in CP-kernel might be more difficult because ι cannot be ignored in internal differential cryptanalysis, while the length of internal differential characteristic used in the collision attack is shorter. It seems that standard differential cryptanalysis is more effective for reduced versions of SHA-3 with a low security strength and a large rate, while internal differential has advantages for higher security strengths since the collision is easier produced for a longer digest. In spite of this, our collision attack on each variant of SHA-3 expect SHAKE128 can reach the most rounds at present. For 4-round SHA3-512, our collision attack outperforms the best known attacks, and the collision attack on 5-round SHAKE256 is presented for the first time.

We stress that our attack does not threaten the security of the full SHA-3.

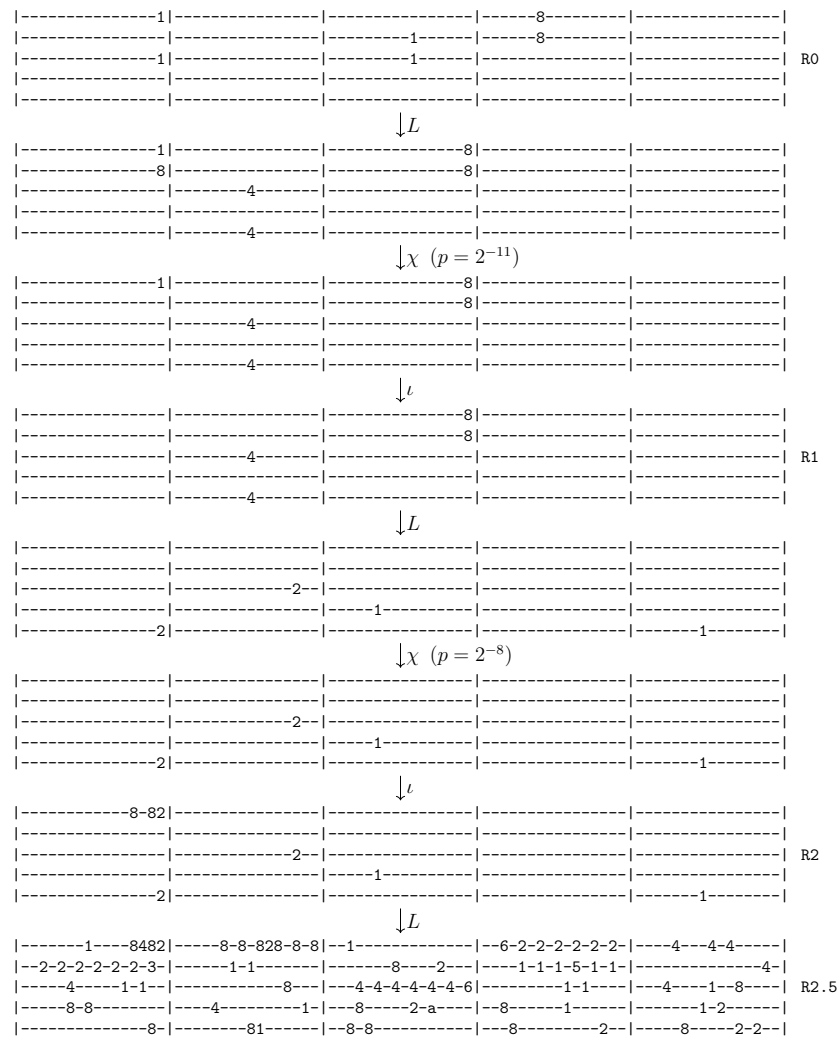
References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. National Institute of Standards and Technology, Gaithersburg (2022)
2. Bernstein, D.J.: Second preimages for 6 (7?(8??)) rounds of keccak. NIST mailing list (2010)
3. Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., et al.: SPHINCS (2017)
4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 313–314. Springer (2013)

5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document. Submission to NIST (Round 2) **3**(30), 320–337 (2009)
6. Chang, D., Kumar, A., Morawiecki, P., Sanadhya, S.K.: 1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512. In: SHA-3 workshop (2014)
7. Dinur, I.: Improved Algorithms for Solving Polynomial Systems over GF(2) by Multiple Parity-Counting. In: Marx, D. (ed.) Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021. pp. 2550–2564. SIAM (2021). https://doi.org/10.1137/1.9781611976465.151
8. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. In: International Workshop on Fast Software Encryption. pp. 442–461. Springer (2012)
9. Dinur, I., Dunkelman, O., Shamir, A.: Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In: International Workshop on Fast Software Encryption. pp. 219–240. Springer (2013)
10. Dinur, I., Dunkelman, O., Shamir, A.: Improved practical attacks on round-reduced Keccak. Journal of cryptology **27**(2), 183–209 (2014)
11. Dworkin, M.J.: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015). https://doi.org/10.6028/nist.fips.202
12. Guo, J., Liao, G., Liu, G., Liu, M., Qiao, K., Song, L.: Practical Collision Attacks against Round-Reduced SHA-3. J. Cryptol. **33**(1), 228–270 (2020). https://doi.org/10.1007/s00145-019-09313-3
13. Guo, J., Liu, G., Song, L., Tu, Y.: Exploring SAT for Cryptanalysis: (Quantum) Collision Attacks against 6-Round SHA-3. In: 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. pp. 645–674. Springer (2023)
14. Guo, J., Liu, M., Song, L.: Linear structures: applications to cryptanalysis of round-reduced Keccak. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 249–274. Springer (2016)
15. Huang, S., Ben-Yehuda, O.A., Dunkelman, O., Maximov, A.: Finding Collisions against 4-round SHA3-384 in Practical Time. IACR Transactions on Symmetric Cryptology pp. 239–270 (2022)
16. Li, T., Sun, Y.: Preimage attacks on round-reduced Keccak-224/256 via an allocating approach. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 556–584. Springer (2019)
17. Nishimura, K., Sibuya, M.: Probability To Meet in the Middle. Journal of Cryptology **2**(1), 13–22 (1990)
18. Peyrin, T.: Improved Differential Attacks for ECHO and Grostl. IACR Cryptol. ePrint Arch **2010**, 223 (2010), http://eprint.iacr.org/2010/223
19. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced keccak. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 216–243. Springer (2017)
20. Song, L., Liao, G., Guo, J.: Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 428–451. Springer (2017). https://doi.org/10.1007/978-3-319-63715-0_{1}{5}

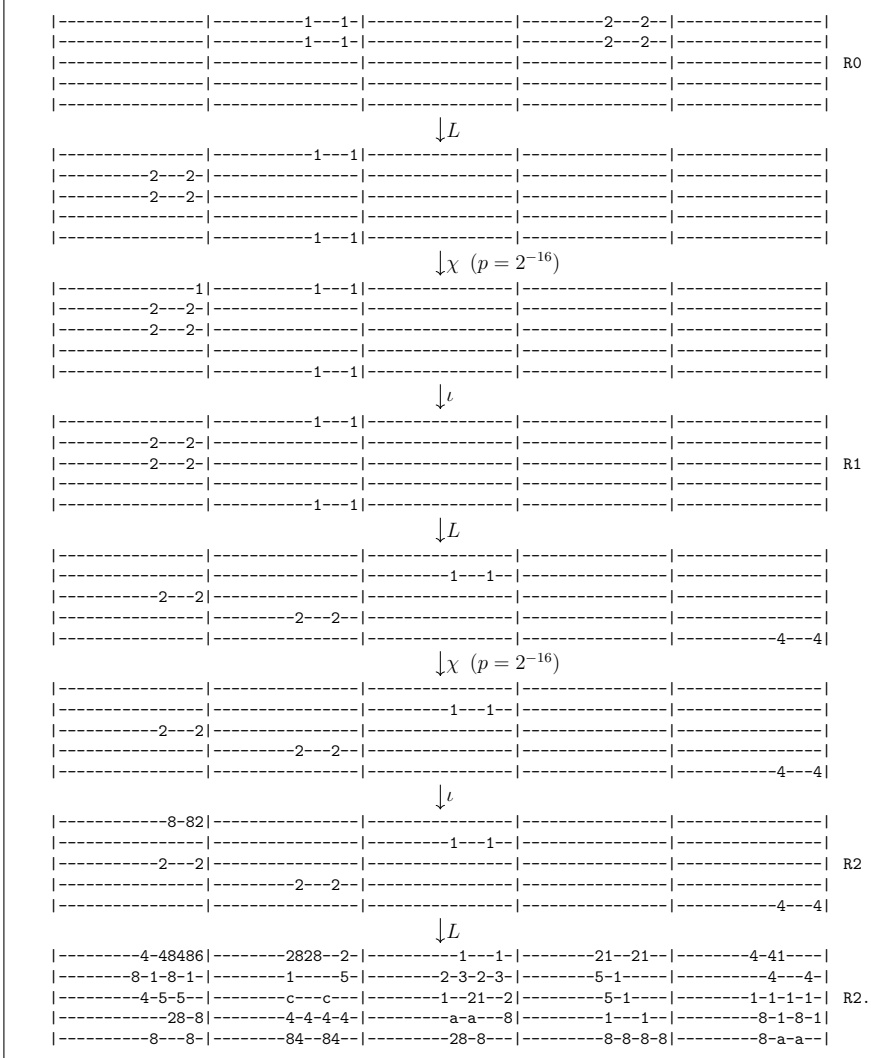
A Internal Differential Characteristics for the Attacks

The internal difference $[i, v]$ is represented by its canonical representative state defined in Section 4.1. Each state is given as a matrix of 5×5 lanes of 64 bits, order from left to right, where each lane is given in hexadecimal using the little-endian format. The symbol '-' is used in order to denote a zero 4-bit value.



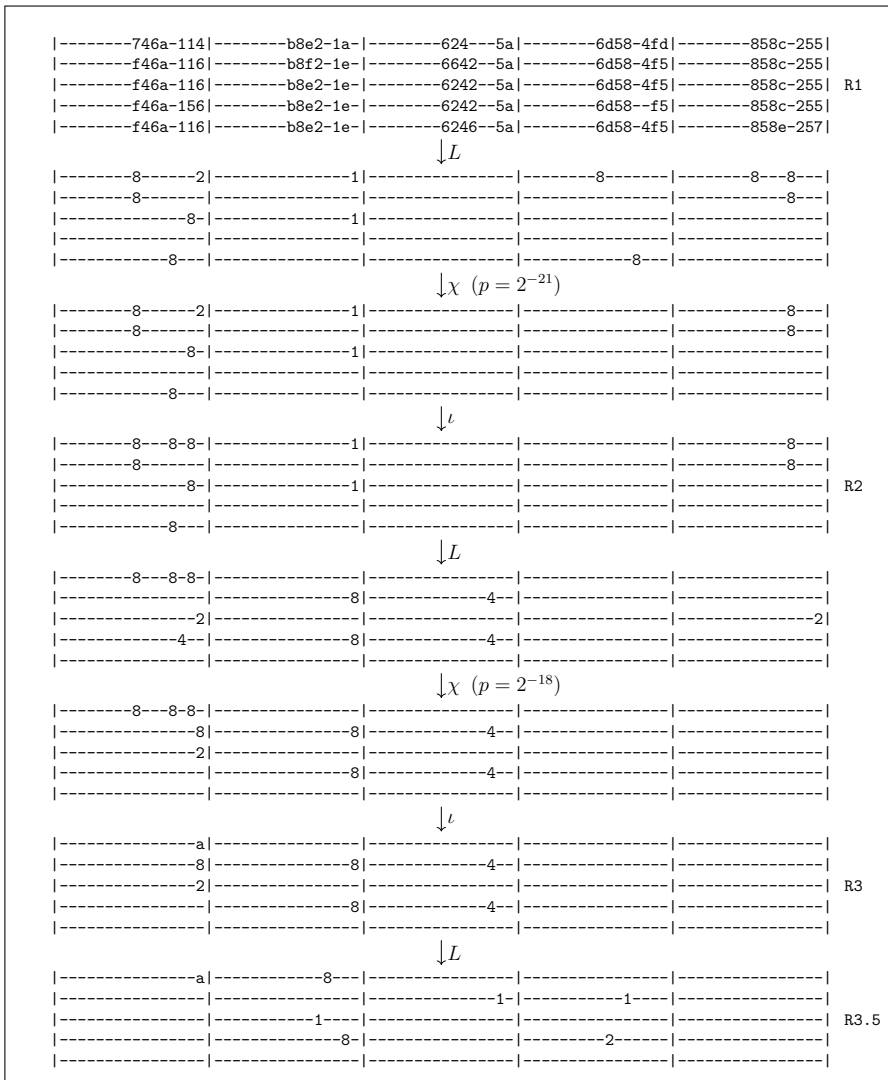
The characteristic has a period of $i = 8$ for the 4-round attack on `SHA3-384` and `SHAKE256`, as described in Section 6.1.

Characteristic 1: The 2.5-round internal differential characteristic with probability 2^{-19} and $(k_1, k_2, k_3) = (11, 8, 78)$.



The characteristic has a period of $i = 32$ for the 4-round attack on `SHA3-512`, as described in Section 6.2.

Characteristic 2: The 2.5-round internal differential characteristic with probability 2^{-32} and $(k_1, k_2, k_3) = (16, 16, 170)$.



The characteristic has a period of $i = 32$ for the 5-round attack on `SHA3-224`,`SHA3-256`,`SHAKE128` and `SHAKE256`, as described in Section 6.3.

Characteristic 3: The 1-3.5 round internal differential characteristic with probability 2^{-39} and $(k_2, k_3, k_4) = (21, 18, 16)$.

B Appendix: Difference Conditions Table of Keccak Sbox

Here we list the differential transition conditions of non-zero input differences.

δ_{out}	Differential transition conditions
01	$l_0 = x_4, l_1 = x_1$
02	$l_0 = x_0, l_1 = x_2$
03	$l_0 = x_4, l_1 = x_2, l_2 = x_0 + x_1$
04	$l_0 = x_1, l_1 = x_3$
05	$l_0 = x_4, l_1 = x_3, l_2 = x_1$
06	$l_0 = x_0, l_1 = x_3, l_2 = x_1 + x_2$
07	$l_0 = x_4, l_1 = x_3, l_2 = x_1 + x_2, l_3 = x_0 + x_1$
08	$l_0 = x_2, l_1 = x_4$
09	$l_0 = x_2, l_1 = x_1, l_2 = x_4$
0a	$l_0 = x_0, l_1 = x_4, l_2 = x_2$
0b	$l_0 = x_4, l_1 = x_2, l_2 = x_0 + x_1$
0c	$l_0 = x_1, l_1 = x_4, l_2 = x_2 + x_3$
0d	$l_0 = x_1, l_1 = x_4, l_2 = x_2 + x_3$
0e	$l_0 = x_0, l_1 = x_4, l_2 = x_2 + x_3, l_3 = x_1 + x_2$
0f	$l_0 = x_4, l_1 = x_2 + x_3, l_2 = x_1 + x_2, l_3 = x_0 + x_1$
10	$l_0 = x_3, l_1 = x_0$
11	$l_0 = x_3, l_1 = x_1, l_2 = x_4 + x_0$
12	$l_0 = x_3, l_1 = x_2, l_2 = x_0$
13	$l_0 = x_3, l_1 = x_2, l_2 = x_0 + x_1, l_3 = x_4 + x_0$
14	$l_0 = x_1, l_1 = x_0, l_2 = x_3$
15	$l_0 = x_3, l_1 = x_1, l_2 = x_4 + x_0$
16	$l_0 = x_0, l_1 = x_3, l_2 = x_1 + x_2$
17	$l_0 = x_3, l_1 = x_1 + x_2, l_2 = x_0 + x_1, l_3 = x_4 + x_0$
18	$l_0 = x_2, l_1 = x_0, l_2 = x_3 + x_4$
19	$l_0 = x_2, l_1 = x_1, l_2 = x_4 + x_0, l_3 = x_3 + x_4$
1a	$l_0 = x_2, l_1 = x_0, l_2 = x_3 + x_4$
1b	$l_0 = x_2, l_1 = x_0 + x_1, l_2 = x_4 + x_0, l_3 = x_3 + x_4$
1c	$l_0 = x_1, l_1 = x_0, l_2 = x_3 + x_4, l_3 = x_2 + x_3$
1d	$l_0 = x_1, l_1 = x_4 + x_0, l_2 = x_3 + x_4, l_3 = x_2 + x_3$
1e	$l_0 = x_0, l_1 = x_3 + x_4, l_2 = x_2 + x_3, l_3 = x_1 + x_2$
1f	$l_0 = x_3 + x_4, l_1 = x_2 + x_3, l_2 = x_1 + x_2, l_3 = x_0 + x_1$

Table 7. Difference Conditions Table of Keccak Sbox

C Appendix: Values of Difference Conditions Table of Keccak Sbox

Here we list the values of differential transition conditions of some input differences, and the other input differences and their differential transition conditions' values can be obtained through cyclic shifting of existing input differences and conditions.

01($<<< j$)	09				19				01				11			
l_0	0				0				1				1			
l_1	0				1				0				1			
03($<<< j$)	0b		1b		0a		1a		03		13		02		12	
l_0	0		0		0		0		1		1		1		1	
l_1	0		0		1		1		0		0		1		1	
l_2	0		1		0		1		0		1		0		1	
05($<<< j$)	0c		1d		0e		1f		04		15		06		17	
l_0	0		0		0		0		1		1		1		1	
l_1	0		0		1		1		0		0		1		1	
l_2	0		1		0		1		0		1		0		1	
0b($<<< j$)	01		11		02		12		0d		1d		0e		1e	
l_0	0		0		0		0		1		1		1		1	
l_1	0		0		1		1		0		0		1		1	
l_2	0		1		0		1		0		1		0		1	
07($<<< j$)	0f	1f	0e	1e	0d	1d	0c	1c	07	17	06	16	05	15	04	14
l_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
l_1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
l_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
l_3	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0f($<<< j$)	07	17	06	16	05	15	04	14	0b	1b	0a	1a	09	19	08	18
l_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
l_1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
l_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
l_3	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1f	1f	07	16	0e	15	0d	1c	04	13	0b	1a	02	19	01	10	08
l_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
l_1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
l_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
l_3	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Table 8. Values of Difference Conditions Table of Keccak Sbox

D Appendix: 2D Affine Subspaces of Keccak Sbox

Here we give the 2-dimensional affine subspaces and affine equations to the output differences of Sbox using in TIDA.

δ_{out}	2D affine subspaces	Corresponding linear equations
01	{01,11,09,19}	$x_0 = 1, x_2 = 0, x_3 = 0$
02	{02,12,03,13}	$x_2 = 1, x_3 = 0, x_4 = 0$
03	{02,12,09,19}	$x_2 = 0, x_0 + x_3 = 0, x_1 + x_3 = 1$
04	{04,06,05,07}	$x_2 = 1, x_3 = 0, x_4 = 0$
05	{04,06,19,1b}	$x_0 + x_2 = 1, x_0 + x_3 = 0, x_0 + x_4 = 0$
06	{04,12,05,13}	$x_3 = 0, x_1 + x_2 = 1, x_1 + x_4 = 0$
07	{04,12,19,0f}	$x_0 + x_3 = 0, x_2 + x_4 = 0, x_0 + x_1 + x_2 = 1$
08	{08,0c,0a,0e}	$x_0 = 0, x_3 = 1, x_4 = 0$
09	{01,11,0e,1e}	$x_0 + x_1 = 1, x_0 + x_2 = 1, x_0 + x_3 = 1$
0a	{08,0c,13,17}	$x_0 + x_1 = 0, x_0 + x_3 = 1, x_0 + x_4 = 0$
0b	{0c,0a,19,1f}	$x_3 = 1, x_0 + x_4 = 0, x_0 + x_1 + x_2 = 1$
0c	{08,0c,0a,0e}	$x_0 = 0, x_3 = 1, x_4 = 0$
0d	{0c,09,0b,0e}	$x_3 = 1, x_4 = 0, x_0 + x_2 = 1$
0e	{08,0c,1b,1f}	$x_3 = 1, x_0 + x_1 = 0, x_0 + x_4 = 0$
0f	{0c,0a,15,13}	$x_0 + x_3 = 1, x_0 + x_4 = 0, x_1 + x_2 = 1$
10	{10,18,14,1c}	$x_0 = 0, x_1 = 0, x_4 = 1$
11	{01,14,09,1c}	$x_1 = 0, x_0 + x_2 = 1, x_0 + x_4 = 1$
12	{02,03,1c,1d}	$x_1 + x_2 = 1, x_1 + x_3 = 1, x_1 + x_4 = 1$
13	{02,09,1c,17}	$x_1 + x_3 = 1, x_2 + x_4 = 0, x_0 + x_1 + x_2 = 1$
14	{10,18,07,0f}	$x_0 + x_1 = 0, x_0 + x_2 = 0, x_0 + x_4 = 1$
15	{06,05,1c,1f}	$x_2 = 1, x_3 + x_4 = 0, x_0 + x_1 + x_3 = 1$
16	{18,14,13,1f}	$x_4 = 1, x_0 + x_1 = 0, x_0 + x_2 + x_3 = 1$
17	{14,05,0d,1c}	$x_1 = 0, x_2 = 1, x_0 + x_4 = 1$
18	{10,0a,15,0f}	$x_0 + x_2 = 0, x_1 + x_3 = 0, x_1 + x_4 = 1$
19	{01,14,0e,1b}	$x_0 + x_2 = 1, x_1 + x_3 = 0, x_0 + x_1 + x_4 = 1$
1a	{18,12,16,1c}	$x_0 = 0, x_4 = 1, x_1 + x_3 = 1$
1b	{12,0a,16,0e}	$x_0 = 0, x_1 = 1, x_3 + x_4 = 1$
1c	{10,18,17,1f}	$x_4 = 1, x_0 + x_1 = 0, x_0 + x_2 = 0$
1d	{16,11,1b,1c}	$x_4 = 1, x_0 + x_2 = 1, x_0 + x_1 + x_3 = 1$
1e	{18,0b,0e,1d}	$x_3 = 1, x_1 + x_4 = 1, x_0 + x_1 + x_2 = 0$
1f	{14,05,0e,1f}	$x_2 = 1, x_1 + x_3 = 0, x_0 + x_1 + x_4 = 1$

Table 9. 2D Affine Subspaces of Keccak Sbox