New Collision Attacks on Round-Reduced Keccak

Kexin Qiao[1,3,4,⋆], Ling Song[1,2,3(✉)], Meicheng Liu[1,⋆], and Jian Guo[2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China
{qiaokexin,songling,liumeicheng}@iie.ac.cn

[2] Nanyang Technological University, Singapore
guojian@ntu.edu.sg

[3] Data Assurance and Communication Research Center, Chinese Academy of Sciences, China

[4] University of Chinese Academy of Sciences, China

Abstract. In this paper, we focus on collision attacks against KECCAK hash function family and some of its variants. Following the framework developed by Dinur *et al.* at FSE 2012 where 4-round collisions were found by combining 3-round differential trails and 1-round connectors, we extend the connectors one round further hence achieve collision attacks for up to 5 rounds. The extension is possible thanks to the large degree of freedom of the wide internal state. By linearization of all S-boxes of the first round, the problem of finding solutions of 2-round connectors are converted to that of solving a system of linear equations. However, due to the quick freedom reduction from the linearization, the system has solution only when the 3-round differential trails satisfy some additional conditions. We develop a dedicated differential trail search strategy and find such special differentials indeed exist. As a result, the first practical collision attack against 5-round SHAKE128 and two 5-round instances of the KECCAK collision challenges are found with real examples. We also give the first results against 5-round KECCAK-224 and 6-round KECCAK collision challenges. It is remarked that the work here is still far from threatening the security of the full 24-round KECCAK family.

Keywords: KECCAK, SHA-3, hash function, linearization, differential.

1 Introduction

The KECCAK [3,5] family of hash functions has attracted intensive cryptanalysis since its submission to the SHA-3 competition in 2008 [1,9,10,11,13,14,16,17,19]. In 2012, the National Institute of Standards and Technology of the U.S. selected KECCAK as the winner of the SHA-3 competition. The SHA-3 family consists of four cryptographic hash functions of fixed digest sizes and two eXtendable-Output

⋆ This work was done while the authors were visiting Nanyang Technological University.

Functions (XOFs) named SHAKE128 and SHAKE256, each of which is based on an instance of the KECCAK algorithms [18]. KECCAK$[r, c, d]$ applies sponge construction with bitrate r and capacity c to generate d bit digests from arbitrary length messages where $d = 224, 256, 384, 512$ in the official SHA-3 versions and $d = 160, 80$ in the KECCAK Crunchy Crypto Collision and Pre-image Contest [2]. Depending on the size of the internal state in $r + c$ bits from the set $\{200, 400, 800, 1600\}$, each of the challenge versions contains 4 variants. SHAKE128 and SHAKE256 generate digests that can be extended to any desired length. The suffixes "128" and "256" indicate the security strengths against generic attacks that these two functions support.

In this paper, we focus on collision attacks against the KECCAK family, *i.e.*, to find two different messages such that their hash digests are the same. The best previous practical collision attacks on KECCAK family are on KECCAK-224 and KECCAK-256 reduced to 4 rounds found by Dinur *et al.* [10] in 2012 and later furnished in the journal version [12]. After this, theoretical results improved to 5-round KECCAK-256 [11]. However, the number of practically attacked rounds remains at 4. To promote cryptanalysis against KECCAK, the KECCAK design team proposed smaller variants in the KECCAK challenge [2] with 160 digest size for collision attack and 80 digest size for preimage attack with each of the 4 sizes of internal states reduced to from 1 to 12 rounds. The ideal security levels of both are set to be 2^{80} unit computations for collision and preimages, respectively. This is a level much lower than that of the main 4 instances of SHA-3, but still beyond the reach of current computation resource one may have. The current best solutions of collision challenges are instances reduced up to 4 rounds by Dinur *et al.* [10] and Mendel *et al.* [17]. Theoretical results were found by Dinur *et al.* [11] against KECCAK-256 with complexities 2^{115} using generalized internal differentials. To the best of our knowledge, this remains as the only results on collision attack against KECCAK reduced to 5 or more rounds up to date.

Our Contribution. We develop an algebraic and differential hybrid method to launch collision attacks on KECCAK family and practically find collisions of 5-round SHAKE128 and two 5-round instances of the KECCAK collision challenges. Theoretical results, with complexities below the birthday bound, against 5-round KECCAK-224 and 6-round KECCAK collision challenges are also achieved.

These results follow a crucial observation that, the KECCAK S-box can be re-expressed as linear transformations, when the input is restricted to some affine subspaces. It was already noted by Daemen *et al.* [3, 8] and Dinur *et al.* [10] that when the input and output differences are fixed, the solution set of the KECCAK S-box contains affine subspaces of dimension up to 3. In this paper, we show the maximum subspaces allowing linearization of S-box is of dimension 2. Furthermore, all affine subspaces of dimension up to 2 allow S-box linearization, and for those of dimension 3, six 2-dimensional affine subspaces out of it could allow the linearization. With this property in mind, we enforce linearization of all S-boxes in the first round, under which the first round function of the

KECCAK permutation is transformed into a linear one. Combining with an invertion method of the S-box layer of the second round, we convert the problem of finding two-round connectors into that of solving a system of linear equations. Solving the equation once will produce sufficiently many solutions so that at least one of them will follow the differential trails in the following 3 rounds or more.

A side effect of linearization of all S-boxes is quick reduction of freedom degrees, which in turn decides the existence of such two-round connectors. To solve this problem, we aim to find differential trails, which impose least possible conditions to the two-round connectors. We design a dedicated search strategy to find suitable differential trails of up to 4 rounds. Implementation confirmed the correctness of this idea, and found real examples of collisions for 5-round SHAKE128 and two instances of challenge versions.

We list our results together with the related previous work in Table 1. Note, the algorithm for building 2-round connectors is heuristic and there is no theoretical bound for the solving time. However, it applies to our attacked instances within practical time, so we indicate the real time cost instead of complexities here. Experiments are run on a server with 32 cores of AMD processors.

Table 1: Collision attack results and comparison

Target $[r, c, d]$	n_r	Searching Complexity	Searching Time	Solving Time[2]	Reference
SHAKE128	5	2^{39}	30m	25m	Sect. 6.2
KECCAK[1440,160,160]	5	2^{40}	2.48h	9.6s	Sect. 6.1
	6	$2^{70.24}$	N.A. [1]	1h	Sect. 6.4
KECCAK[640,160,160]	5	2^{35}	2.67h	30m	Sect. 6.3
KECCAK-224	4	2^{24}	2∼3m		[10]
		2^{12}	0.3s	2m15s	Sect. 6.6
	5	2^{101}	N.A.	N.A.	Sect. 6.5
KECCAK-256	4	2^{24}	15∼30m		[10]
		2^{12}	0.28s	7m	Sect. 6.6

[1] N.A.: Not Available.
[2] There is no theoretical estimate for the solving time of the heuristic algorithms used here.

Organization. The rest of the paper is organized as follows. In Section 2 and Section 3, notations and a brief description of KECCAK family are given. In Section 4, we give a detailed description of the algebraic methods to achieve 2-round connectors. In Section 5, we give the dedicated search strategies for differential trails. Then the experimental results are presented in Section 6. We conclude the paper in Section 7.

2 Notations

We summarize the majority of notations to be used in this paper here.

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 of a hash function
n_r	Number of rounds
$\theta, \rho, \pi, \chi, \iota$	The five step mappings that comprise a round, a subscript i denotes the operation at i-th round, *e.g.*, θ_i denotes the θ layer at i-th round for $i = 0, 1, 2, \cdots$.
L	composition of θ, ρ, π
L^{-1}	Inverse of L
RC	Round constant for a round of KECCAK-f permutation
$\mathtt{S}(\cdot)$	5-bit S-box operating on each row of KECCAK state
$\mathtt{R}^i(\cdot)$	KECCAK permutation reduced to the first i rounds
δ_{in}	5-bit input difference of an S-box,
δ_{out}	5-bit output difference of an S-box
DDT	Differential distribution table
α_i	Input difference of the i-th round function, $i = 0, 1, 2, \cdots$
β_i	Input difference of χ in the i-th round, $i = 0, 1, 2, \cdots$
w_i	Weight of the i-th round, $i = 0, 1, 2, \cdots$
$m_1 \| \| m_2$	Concatenation of strings m_1 and m_2
x	Bit value vector before χ in the first round
y	Bit value vector after the first round
z	Bit value vector before χ in the second round

3 Description of KECCAK

In this section, we give a brief description of KECCAK family of hash functions. KECCAK family applies sponge construction which processes messages 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. We denote the original message by M and the padded message by $\overline{M} = M||10^*1$. The b-bit internal state is initialized to be all 0's. In absorbing phase, the padded message is split into blocks of r-bits and each message block is XORed into the first r bits of current internal state, followed by the application of the fixed permutation to the entire b-bit state. This is repeated until all message blocks are processed. In the squeezing phase, the first r bits of the state are returned as output, then the permutation is applied and another r bits are outputted until all d output bits are produced. When restricted to the case of $b = 1600$ and $c = 2d$, the four official instances of KECCAK family are denoted by KECCAK-224/256/384/512 respectively for $d = 224, 256, 384, 512$. SHAKE128 and SHAKE256 are defined from two instances

of KECCAK with the capacity c being 256 and 512, respectively, and the additional appending of a four-bit suffix 1111 to the original message M before applying the KECCAK padding. Without further specifications, we presume the digest sizes are 256 and 512 for SHAKE128 and SHAKE256, respectively. We use $\overline{M}$ to denote the padded message for SHAKE as well. Instances of KECCAK challenges will be denoted as KECCAK$[r, c, n_r, d]$, where the parameters are explicitly indicated for the rate, capacity, number of rounds the permutation is reduced to, and bit size of the digest, respectively.

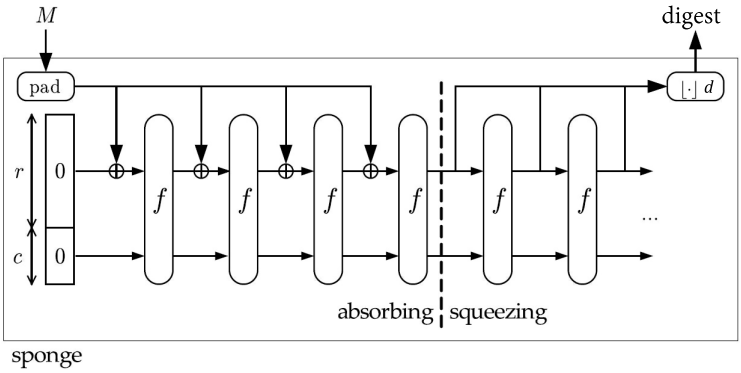

Figure 1: Sponge Construction [4].

The KECCAK permutation function in SHA-3 consists of 24 rounds of five layers operating on the 1600-bit state that can be represented as 5×5 64-bit lanes. In general 2^l is used to denote the bit length of lanes. If A denotes a 5-by-5-by-2^l array of bits that represents the state, then its indices are the integer triples (i, j, k) for which $0 \leq i < 5, 0 \leq j < 5$, and $0 \leq k < 2^l$. The bit that corresponds to (i, j, k) is denoted by $A[i, j, k]$. Names for single-dimensional sub-arrays and two-dimensional ones are defined by the KECCAK designers: $A[\cdot][j][k]$ is called a row, $A[i][\cdot][k]$ is a column, and $A[i][j][\cdot]$ is a lane; $A[i][\cdot][\cdot]$ is called a sheet, $A[\cdot][j][\cdot]$ is a plane, and $A[\cdot][\cdot][k]$ is a slice.

The five layers in each round of the permutation are given below:

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

$$\rho : A[i][j][k] \leftarrow A[i][j][k + T(i,j)], \text{where } T(i,j) \text{ is a predefined constant}$$

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

$$\chi : A[i][j][k] \leftarrow A[i][j][k] + ((\neg A[i+1][j][k]) \wedge A[i+2][j][k]),$$

$$\iota : A \leftarrow A + RC[i_r], \text{where } RC[i_r] \text{ is the round constants.}$$

It is interesting to note that the size of permutation can be reduced to one of $\{25, 50, 100, 200, 400, 800\}$ by choosing $2^l = 1, 2, 4, 8, 16, 32$, respectively for the size of the lanes. In such cases, the round functions are defined exactly in the same way except the rotation constants of the ρ operation are now in modulo the respective 2^l instead of 64 in the original 1600-bit full permutation. These size-reduced permutations are not used in the SHA-3 instances, but in the KECCAK challenges.

The first three layers are linear mappings and we denote their composition by $L \triangleq \theta \circ \rho \circ \pi$. The only non-linear layer of the permutation is χ, which can be seen as a S-box layer that applies 5-bit substitution to 320 rows of the state. We use $\mathtt{S}(x)$ to denote the substitution of a 5-bit input value x. The difference distribution table of the S-box is denoted by DDT, where $\mathtt{DDT}(\delta_{in}, \delta_{out})$ represents the size of the set $\{x : \mathtt{S}(x) + \mathtt{S}(x + \delta_{in}) = \delta_{out}\}$. We denote the KECCAK permutation reduced to the first i rounds as $\mathtt{R}^i$ (note the round functions are identical up to a difference of constant addition in ι and we will omit ι as it has little impact on our differential collision attack), *i.e.*, $\mathtt{R}^i(\overline{M})$ is the state after i rounds processing of the padded message $\overline{M}$.

4 Overview of Our Collision Attack

In this section, we give an overview of our collision attacks, followed by the details of the algebraic methods to achieve two-round connectors. Without further specification, we assume in this paper the length of the messages used are of one block after padding. To fulfil the KECCAK padding rule, one needs to fix the last bit of the padded message to be "1", hence the first $r - 1$ bits of the state are under the full control of the attacker through the message bits, and the last c bits of the state are fixed to zeros as in the IV specified by KECCAK. When applied to SHAKE, there are $r - 6$ free bits under control, by setting the last 6 bits of the padded message to be all 1's so it is compatible with the specific SHAKE padding rule.

Following the framework by Dinur *et al.* [10], as well as many other collision attacks utilizing differential trails, our collision attacks consist of two parts, *i.e.*, a high probability differential trail and a connector linking the differential trail with the initial value, as depicted in Figure 2. Let ΔS_I and ΔS_O denote the input and output differences of the differential trail, respectively. Dinur *et al.* explored a method, which they call "target difference algorithm", to find message pairs (M, M') such that the output difference after one round permutation is ΔS_I, formally $\mathtt{R}^1(\overline{M}||0^c) + \mathtt{R}^1(\overline{M'}||0^c) = \Delta S_I$. In what follows, we show an algebraic method to extend this connector to two rounds, *i.e.*, a new target difference algorithm to find (M, M') such that $\mathtt{R}^2(\overline{M}||0^c) + \mathtt{R}^2(\overline{M'}||0^c) = \Delta S_I$. The differential trail is then fulfilled probabilistically with many such message pairs, collision can be produced if the first d bits of ΔS_O are 0. As we are aiming at low complexity attacks, finding solutions of connectors should be practical so that this part will not dominate the overall complexities of collision finding. Details of the differential trail search will be discussed in Section 5.

Overall, the constraints of the two-round connectors are that the last $c+1$ (or $c+6$) bits of the initial state are fixed, and the output difference after two rounds is given and fixed (this is determined by the differential trail to be used), we are to utilize the degree of freedom from the first $r-1$ (or $r-6$) bits of the initial state to find solutions efficiently. We will start with some observations on the KECCAK S-box, then move to the details of solution finding algorithm.

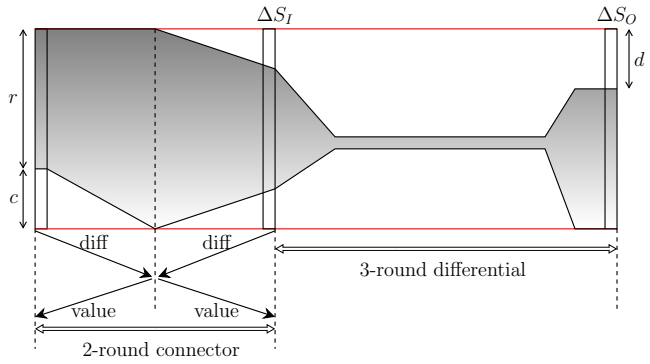

Figure 2: Overview of 5-round collision attack

4.1 S-box linearization and affine subspaces

The key observation is that internal state is much larger than the digest size, providing large number of freedom degrees to attackers. One can choose some subsets of the available spaces with special properties to achieve fast enumerations. In case of KECCAK, we are to choose the subsets which are *linear* with respective to the S-box, *i.e.*, the expression of S-box can be re-written as linear transformation when the input is restricted to such subsets. It is obvious to note the S-box is non-linear when the entire 2^5 input space is considered. However, affine subspaces of size up to 4, as to be shown below, could be found so that the S-box can be linearized. Note that the S-box is the only nonlinear part of the KECCAK round function. Hence, the entire round function becomes linear when restricted to such subspaces. Formally, we define

Definition 1 (Linearizable affine subspace). *Linearizable affine subspaces are affine input subspaces on which S-box substitution is equivalent to a linear transformation. If V is a linearizable affine subspace of an S-box operation $\mathtt{S}(\cdot)$, $\forall x \in V, \mathtt{S}(x) = A \cdot x + b$, where A is a matrix and b is a constant vector.*

For example, when input is restricted to the subset $\{\mathtt{00000}, \mathtt{00001}, \mathtt{00100}, \mathtt{00101}\}$ ($\{\mathtt{00}, \mathtt{01}, \mathtt{04}, \mathtt{05}\}$ in hex), the corresponding output set of the KECCAK S-box is $\{\mathtt{00000}, \mathtt{01001}, \mathtt{00101}, \mathtt{01100}\}$($\{\mathtt{00}, \mathtt{09}, \mathtt{05}, \mathtt{0C}\}$ in hex), and the expression of the

S-box can be re-written as linear transformation:

$$y = \begin{pmatrix} 1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \cdot x \tag{1}$$

where x and y are bit vector representation of input and output values of the KECCAK S-box with the last bit on top. By rotation symmetry, four more linearizable affine subspaces can be deduced from one.

Exhaustive search for the linearizable affine subspaces of the KECCAK S-box shows:

Observation 1 *Out of the entire 5-dimensional input space,*

a. there are totally 80 2-dimensional linearizable affine subspaces, as listed in Table 5 in Appendix A.
b. there does not exist any linearizable affine subspace with dimension 3 or more.

For completeness, any 1-dimensional subspace is automatically linearizable affine subspace.

Since the affine subspaces are to be used together with differential trails, we are interested in those linearizable affine subspaces with fixed input and output differences, which is more relevant with the differential distribution table (DDT) of S-boxes. Referring to the DDT of KECCAK S-box postponed to Appendix B, we observe:

Observation 2 *Given a 5-bit input difference δ_{in} and a 5-bit output difference δ_{out} such that $\mathtt{DDT}(\delta_{in}, \delta_{out}) \neq 0$, denote the value solution set $V = \{x : \mathtt{S}(x) + \mathtt{S}(x + \delta_{in}) = \delta_{out}\}$ and $\mathtt{S}(V) = \{\mathtt{S}(x) : x \in V\}$, we have*

a. if $\mathtt{DDT}(\delta_{in}, \delta_{out}) = 2$ or 4, then V is a linearizable affine subspace.
b. if $\mathtt{DDT}(\delta_{in}, \delta_{out}) = 8$, then there are six *2-dimensional subsets $W_i \subset V, i = 0, 1, \cdots, 5$ such that $W_i (i = 0, 1, \cdots, 5)$ are linearizable affine subspaces.*

It is interesting to note the 2-dimensional linearizable affine subspaces obtained from analysis of DDT cover all the 80 cases in observation 1. It is already noted in [15] there is one-to-one correspondence between linearizable affine subspaces and entries with value 2 or 4 in DDT. As for the DDT entries of value 8, we will leave the 6 choices of 2-dimensional linearizable affine subspaces for later usage. As an example, the 3-dimensional affine subspace corresponding to $\mathtt{DDT}(\mathtt{01}, \mathtt{01})$, *i.e.*, with both input and output differences being 01, is $\{\mathtt{10}, \mathtt{11}, \mathtt{14}, \mathtt{15}, \mathtt{18}, \mathtt{19}, \mathtt{1C}, \mathtt{1D}\}$ and the six 2-dimensional linearizable affine subspaces from it are

$$\begin{aligned} &\{\mathtt{10}, \mathtt{11}, \mathtt{14}, \mathtt{15}\}, \\ &\{\mathtt{10}, \mathtt{11}, \mathtt{18}, \mathtt{19}\}, \\ &\{\mathtt{10}, \mathtt{11}, \mathtt{1C}, \mathtt{1D}\}, \\ &\{\mathtt{14}, \mathtt{15}, \mathtt{18}, \mathtt{19}\}, \\ &\{\mathtt{14}, \mathtt{15}, \mathtt{1C}, \mathtt{1D}\}, \\ &\{\mathtt{18}, \mathtt{19}, \mathtt{1C}, \mathtt{1D}\}. \end{aligned} \tag{2}$$

When projected to the whole KECCAK state, direct product of affine subspaces of each individual S-box form affine subspaces of the entire state with larger dimensions. In other words, when all the S-boxes in the round function are linearized, the entire round function becomes linear. This will be the way we are to handle the S-box layer of the first round of the 2-round connector.

4.2 A connector covering two rounds

The core idea of our two-round connector is to convert the problem to solving a system of linear equations. Two rounds of KECCAK permutation can be expressed as $\chi_1 \circ L_1 \circ \chi_0 \circ L_0$ (omitting the ι). With the χ_0 layer linearized by the techniques discussed above, *i.e.*, given the input and output differences of χ, the first three operations $L_1 \circ \chi_0 \circ L_0$ become linear. We will give details of the method how input and output differences of χ_0 are selected later. Now, we show how the χ_1 can be inverted by adding more constraints of linear equations. In our attack setting, the output difference of χ_1 is given as ΔS_I—input difference of the 3-round differential trail. It is not necessary that all S-boxes of the χ_1 layer are active, *i.e.*, with a non-zero difference. Here only active S-boxes are concerned, and each of them is inverted by randomly choosing an input difference with non-zero number of solutions, we call it *compatible* input difference. Formally, given the output difference δ_{out}, compatible input differences are $\{\delta_{in} : \mathtt{DDT}(\delta_{in}, \delta_{out}) \neq 0\}$. As noted previously [3,8,10], for any pair of $(\delta_{in}, \delta_{out})$, the solution set $V = \{x : \mathtt{S}(x) + \mathtt{S}(x + \delta_{in}) = \delta_{out}\}$ forms an affine subspace. In other words, V can be deduced from the set $\{0,1\}^5$ by setting up i constraints that turn to be binary linear equations, when the size of solution set V is 2^{5-i}. For example, corresponding to $\mathtt{DDT}(\mathtt{03},\mathtt{02})$ is the 2-dimensional affine subspace $\{\mathtt{14},\mathtt{17},\mathtt{1C},\mathtt{1F}\}$ which can be formulated by the following three linear equations:

$$\begin{pmatrix} 0\,0\,1\,0\,0 \\ 1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \cdot x = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}. \tag{3}$$

It is important to note, under the i linear constraints or set V, there is a bijective relation between δ_{in} and δ_{out}, *i.e.*, given one the other can be deduced deterministically. Hence, each active S-box in χ_1 layer is inverted by a choice of compatible input difference together with the corresponding i linear constraints on the input values. Once input difference and linear constraints for all active S-boxes of χ_1 are enforced and fulfilled, solutions of 2-round connector are found. Note a compatible input difference of χ_1 is a choice of β_1, and α_1 can be uniquely determined by the relation $\alpha_1 = L^{-1}(\beta_1)$. In the remaining part of this subsection, more details on implementation of this idea are given.

As depicted in Figure 3, the variables of our equation system are the bit values before the first χ layer denoted by vector x. y and z are bit vectors of intermediate values for further interpretation where y represents the output after the first χ layer and z the bits before the second χ layer. The main task is to derive all constraints on differences and affine subspaces to that on the variables

x. Now, suppose β_1 and β_0 (details will be given in Section 4.4) are fixed, and ΔS_I (aka. α_2) is given, we show how the system of equations could be set up. With the input difference β_1 and output difference ΔS_I of χ_1, all the linear constraints on the input affine subspaces of the active S-boxes can be derived and stored as

$$G \cdot z = m,$$

where G is a block-diagonal matrix in which each diagonal block together with corresponding constants in m formulates the constraints of one active S-box. Similar procedure is done for input affine subspaces of the first round, except that the input is restricted to linearizable affine subspaces for all S-boxes regardless whether or not the S-box is active so that χ_0 layer can be replaced by a linear transformation χ_L. We denote the constraints by

$$A \cdot x = t. \tag{4}$$

Then x and y can be linked by

$$\chi_L \cdot x + \chi_C = y,$$

where χ_C denotes the constant offsets for the affine subspaces. Furthermore, the two equation systems can be linked by

$$L \cdot (y + RC[0]) = z,$$

where $RC[0]$ denotes the round constant of the first round. Note, only active S-boxes of the second round are concerned, *i.e.*, only part of bits of z are known, hence the same applies to y, and we use y' to denote the known bits of y for later. Overall, the constraints on z can be derived to that on x as

$$G \cdot L \cdot (\chi_L \cdot x + \chi_C + RC[0]) = m. \tag{5}$$

Note an additional constraint x needs to fulfil is that the last $c+1$ (or $c+6$) bits of initial state are pre-fixed, which can be derived as

$$L^{-1}(x) = CA, \tag{6}$$

where CA denotes the preset values for bits of the inner state and padding bits. We use E_M to denote the equation systems (4), (5) and (6), solutions fulfilling E_M will be solutions of 2-round connectors.

Algorithm for building two-round connectors. We use *the basic linearization procedure* to generate the equations for confining x to a smaller subspace suitable for linearization of the first χ layer and use *the main linearization procedure* to generate the final equations to bypass the second χ layer. One of the inputs of the basic procedure is the equation system E_M on x values, other inputs include the input and output differences of the first S-box layer β_0, α_1 and y'.

The Basic Linearization Procedure.
Inputs: E_M, β_0, α_1, y'.
Outputs: updated E_M, χ_L, χ_C.

1. Initialize a matrix χ_L and a vector χ_C.
2. Iterate on each bit of y', calculate the index of the bit in S-box level, say the j-th bit of the i-th S-box in the first round. Then for the i-th S-box in the first round:
   (a) If the i-th S-box has not been processed in this procedure before, then:
      (i) If it is non-active, randomly choose a linearizable 2-dimensional subspace, check whether the 3 equations specifying this 2-dimensional affine subspace is consistent with the current E_M.
      If so, add them to E_M and update χ_L and χ_C with the j-th line of the matrix which specifies the affine linear transformation. Continue to next bit of y' in step 2.
      Otherwise, try another linearizable 2-dimensional subspace. If all linearizable 2-dimensional subspaces have been tried and no consistent equations exist, output "No Solution in basic procedure".
      (ii) Otherwise it is active: find its input and output differences from β_0 and α_1, *i.e.*, $\delta_{in}, \delta_{out}$.
      Case 1. When $\texttt{DDT}(\delta_{in}, \delta_{out}) = 8$, randomly choose one of the six linearizable 2-dimensional subspaces and the corresponding equation to specialize this 2-dimensional subspace (the other two of the three equations to formulate the 2-dimentional subspace have already been indicated in E_M after choosing β_0 procedure).
      If current E_M is consistent with this linear equation, add it to E_M and update χ_L and χ_C with the j-th line of the matrix which specifies the linear map from the 2-dimensional subspace to the output 2-dimensional subspace of S-box. Continue to next bit of y' in step 2.
      Otherwise, try another randomly chosen 2-dimensional linearizable subspace. If all six 2-dimensional linearizable subspaces have been chosen and no consistent equation exist, output "No Solution in basic procedure".
      Case 2. When $\texttt{DDT}(\delta_{in}, \delta_{out}) = 2$ or 4, update χ_L and χ_C with the j-th line of the matrix which specifies the affine linear transformation of the input 1 or 2-dimensional subspace to the output 1 or 2-dimensional subspace of S-box. Continue to next bit of y' in step 2.
   (b) Otherwise, if the i-th S-box has already been processed in this procedure: update χ_L and χ_C with the j-th line of the matrix which specifies the affine linear transformation of the predefined linearizable subspace to the output subspace of S-box.
3. Output the current equations system E_M as well as χ_L and χ_C such that $\chi_L \cdot x + \chi_C = y'$.

The inputs to the Main procedure are $\beta_0, \alpha_1, \beta_1, \alpha_2(\Delta S_I)$ and E_M we get after choosing β_0.

The Main Linearization Procedure.
Input: $E_M, \beta_0, \alpha_1, \beta_1, \alpha_2$.

Output: Updated E_M.

1. Using β_1 and α_2, initialize a coefficient matrix G and a constant vector m that specify the linear equations to constrain the input bits of the second S-box layer for deriving the equation $G \cdot z = m$.
2. Derive the L into the matrix format for $L \cdot (y + RC[0]) = z$.
3. Initialize a counter to 0.
4. Execute the basic linear procedure with indexes of know bits y' in y and E_M, β_0 and α_1. If the procedure succeeds, it will return the matrix specifying the linearization of the first S-box layer such that $\chi_L \cdot x + \chi_C = y'$, then continue to Step 6. Otherwise, go to step 5.
5. Increment the counter. If the counter's value is equal to a preset threshold $T1$, output "Failed". Otherwise, go to step 4.
6. Test whether the equation system (5) is consistent with E_M. If so, add the new system to E_M and output final E_M. Otherwise, go to step 5.

Note that the algorithms do not succeed all the time. To overcome this problem, from the input difference ΔS_I of a 3-round differential trail, we repeat random picks of compatible input differences β_1 until the main procedure succeeds. As the number of active S-boxes in α_2 is large enough (range from tens to hundreds in our experiments), there are enough different cases for β_1 resulting in high final success probability. An interesting point is that the invertion from α_2 to β_1 does not need to maintain high probability because this transition is covered in our two-round connector. Besides, the unconstrained number of active S-boxes of an input difference allows more freedom in searching of the most suitable three round differential trails . We will describe the the searching strategies in Section 5. Finally, exhaustive search of solution for the following 3-round differential trails can be performed from the solution space of E_M.

4.3 Analysis of degree of freedom

The degree of freedom of solution space of final E_M is a key factor on success of our method. A solution space with degree of freedom larger than the weight of the 3-round differential trail is possible to suggest a message pair with collision digest. After the linearization of the first round, the degree of freedom is $\sum_{i=0}^{\frac{b}{5}-1} \mathtt{DF}_i^{(1)}$ in which $\mathtt{DF}_i^{(1)}$ is the degree of freedom of 5-bit input space of the i-th S-box in the first round. The value is assigned for $\mathtt{DF}_i^{(1)}$ according to rules in Table 2.

The constraints on the initial state reduce $(c + p)$ degree of freedom where c is the capacity and p is due to the padding rule. We have $p = 1$ for KECCAK and $p = 6$ for SHAKE. Another decrease on degree of freedom is due to the constraints on the input values of the S-box layer in the second round. The definition of $\mathtt{DF}_i^{(2)}$, the degree of freedom of 5-bit input values to S-boxes in the second round, is

$$\mathtt{DF}_i^{(2)} = \begin{cases} 1, & \mathtt{DDT}(\delta_{in}, \delta_{out}) = 2, \\ 2, & \mathtt{DDT}(\delta_{in}, \delta_{out}) = 4, \\ 3, & \mathtt{DDT}(\delta_{in}, \delta_{out}) = 8, \\ 5, & \mathtt{DDT}(\delta_{in}, \delta_{out}) = 0, \end{cases} \tag{7}$$

Table 2: Rules for value assignment for $\mathtt{DF}_i^1$.

$\mathtt{DF}_i^{(1)*}$	Non-active	$\mathtt{DDT}(\delta_{in},\delta_{out})$=2	$\mathtt{DDT}(\delta_{in},\delta_{out})$=4	$\mathtt{DDT}(\delta_{in},\delta_{out})$=8
Involved in y'	2	1	2	2
Not involved in y'	5	1	2	3

* The value of $\mathtt{DF}_i^{(1)}$ is based on whether the i-th S-box is involved in y' and the value $\mathtt{DDT}(\delta_{in},\delta_{out})$ where δ_{in} and δ_{out} are the input and output differences of the i-th S-box in the first round.

where δ_{in} and δ_{out} are the input and output differences of the i-th S-box in the second round. For the i-th S-box in the second round, we add $(5-\mathtt{DF}_i^{(2)})$ equations to E_M and suppose to deduce the degree of freedom by this amount.

The degree of freedom of the final E_M is estimated as

$$\mathtt{DF}=\sum_{i=0}^{\frac{b}{5}-1}\mathtt{DF}_i^{(1)}-(c+p)-\sum_{i=0}^{\frac{b}{5}-1}(5-\mathtt{DF}_i^{(2)}). \tag{8}$$

Large DF benefits our search for collisions in rounds beyond the second round.

4.4 How to choose β_0

So far we have not given details on how β_0 can be selected. We follow Dinur *et al.*'s work [10] in a more general way to uniquely determine β_0, the difference before χ layer in the first round. The algorithm is called "target difference algorithm" and consists of difference phase and value phase.

Given ΔS_I, we have randomly chosen a compatible input difference β_1. We then build two equation systems E_Δ and E_M accordingly. E_Δ is on differences of the message pairs and E_M is on values of one message. The initialization of E_Δ should abide by 1) the constraints implied by padding rules that the last $c+1$ difference bits of initial state equal to 0, and 2) the input difference bits of nonactive S-boxes in the first round equal to 0. The initialization of E_M should abide by the padding rules that the last $c+p$ value bits equal to $1^p||0^c$. We set $p=1$ for KECCAK and $p=6$ for SHAKE. These rules are easy to be implemented as the variable vector x is an invertible linear mapping of the initial vector. Therefore, in the initialization period, we equate the corresponding bits to their enforced values in E_Δ and E_M.

For E_Δ, we add additional equations to enforce that α_1 is possibly deduced from β_0. Though the obvious way is to equate the 5 input difference bits to a specific value for each active S-box in β_0, this will restricts the solution space significantly. As suggested in [10], we chose one of the 2-dimensional affine subsets of input differences instead of a specific value for each active S-box. This is based on the fact that given any nonzero 5-bit output difference to a KECCAK S-box, the set of possible input differences contains at least five 2-dimensional affine subspaces. After a consistent E_Δ system has been constructed, the solution space is an affine subspace of candidates for β_0. Then we continue to maintain E_Δ by

iteratively add the additional 2 equations to uniquely specify each 5-bit input difference for the active S-boxes. For all active S-boxes, once the specific input differences is determined, we add equations to E_M system to enforce every active 5-bit of x (input bits to active S-box) to an affine subspace corresponding to the uniquely determined δ_{in} and δ_{out}. In this way, we always find a compatible β_0 from α_1 fulfilling the constraints from the $c + p$ bits of padding and pre-set bits of capacities.

5 Search for Differential Trails

In this section, we elaborate on our searching algorithms for finding differential trails of KECCAK. Our ideas greatly benefit from previous works of searching differential trails for KECCAK [9,14,19]. We start by recalling several properties of the operations in the round function, followed by our considerations in finding differential trails. Then, we describe our searching algorithms which provide differential trails for practical collision attacks against KECCAK$[1440, 160, 5, 160]$, 5-round SHAKE128 and KECCAK$[640, 160, 5, 160]$ respectively, and trails for theoretical collision attack against 5-round KECCAK-224 and KECCAK$[1440, 160, 6, 160]$.

5.1 Properties of θ, ρ, π, ι and χ

θ, ρ, π, ι are linear operations while χ acts as the parallel application of 5-bit nonlinear S-boxes on the rows of the state. Since ι adds a round constant and has no essential effect on difference, we ignore it in this section. Additionally, ρ and π do not change the number of active bits in a differential trail, but only positions. Therefore, θ and χ are the crucial parts for differential analysis.

To describe the properties of θ, we take definitions from [3]. The *column parity* (or parity for short) $P(A)$ of a value (or difference) A is defined as the parity of the columns of A, i.e. $P(A)[i][k] = \Sigma_j A[i][j][k]$. A column is even, if its parity is 0, otherwise it is odd. A state is in CP-kernel if all its columns are even.

θ adds a pattern to the state, and this pattern is called the θ-effect. The θ-effect of a state A is $E(A)[i][k] = P(A)[i-1][k] + P(A)[i+1][k-1]$. So θ depends only on column parities. The θ-gap is defined as the Hamming weight of the θ-effect divided by two. Note that if the θ-gap is g, after applying θ there are $10g$ bits flipped. Given a state A in CP-kernel, the θ-gap is zero and hence the Hamming weight of A remains after θ. Another interesting property is that θ^{-1} diffuses much faster than θ. More exactly, a single bit difference can be propagated to about half state bits through θ^{-1}.

Given an input difference to χ, all possible output differences occur with the same probability. On the contrary, given an output difference to χ, it is not the same case, but the highest probability of all possible input differences is determined. Moreover, for one-bit differences, each S-box of χ acts as identity with probability 2^{-2}.

5.2 Representation of trails and their weights

As in previous sections, we denote the differences before and after i-th round by α_i and α_{i+1}, respectively. Let $\beta_i = L(\alpha_i)$. Therefore an n-round differential trail starting from 0-th round is of the following form

$$\alpha_0 \xrightarrow{L} \beta_0 \xrightarrow{\chi} \alpha_1 \xrightarrow{L} \cdots \alpha_{n-1} \xrightarrow{L} \beta_{n-1} \xrightarrow{\chi} \alpha_n.$$

For the sake of simplicity, a trail can also be represented with only β_i's or α_i's.

The weight of a differential $\beta \to \alpha$ over a function f with domain $\{0,1\}^b$ is defined as

$$w(\beta \to \alpha) = b - \log_2|\{x : f(x) \oplus f(x \oplus \beta) = \alpha\}|.$$

In other words, the weight of a differential $\beta \to \alpha$ is equal to $-\log_2 \Pr(\beta \to \alpha)$. If $\Pr(\beta \to \alpha) > 0$, we say α and β are compatible, otherwise the weight of $\beta \to \alpha$ is undefined.

We denote the weight of i-th round differential by w_i where i starts from 0, and thus the weight of a trail is the sum of the weights of round differentials that constitute the trail. In addition, we use $\#\mathrm{AS}(\alpha)$ to represent the number of active S-boxes in a state difference α. According to the properties of χ, given β_i the weight of $(\beta_i \to \alpha_{i+1})$ is determined; also, given β_i the minimum reverse weight of $(\beta_{i-1} \to L^{-1}(\beta_i))$ is fixed.

As in [3], $n-1$ consecutive β_i's, say $(\beta_1, \cdots, \beta_{n-1})$ is called an n-round **trail core** which defines a set of n-round trails $\alpha_0 \xrightarrow{L} \beta_0 \xrightarrow{\chi} \alpha_1 \xrightarrow{L} \beta_1 \cdots \xrightarrow{L} \beta_{n-1} \xrightarrow{\chi} \alpha_n$ where the first round is of the minimal weight determined by $\alpha_1 = L^{-1}(\beta_1)$, and α_n is compatible with β_{n-1}. The first step of mount collision attacks against n-round KECCAK is to find good $(n-1)$-round trail cores.

5.3 Requirements for differential trails

Good trail cores are those satisfying all the requirements which we will explain as follows. The first requirement is that the difference of the output is zero, i.e. $\alpha^d_{n_r} = 0$ (we denote output digest difference after n_r rounds with $\alpha^d_{n_r}$). The second requirement relates to the freedom degree budget.

With the definition of weight, Equations (8) can be represented in an alternative way

$$\mathtt{DF} = \sum_{i=0}^{b/5-1} \mathtt{DF}_i^{(1)} - (c+p) - w_1. \tag{9}$$

The first term of the formula depends on the number of S-boxes that need to be linearized and its corresponding DDT entry as depicted in Table 2. Empirically, when all S-boxes are active and linearized in the first round it is more possible to get a consistent equation system. Therefore, we heuristically set $\frac{b}{5} \times 2$ as a threshold for the first term in (9), and denote a threshold of the first two terms in (9) for further search conditions by

$$\mathtt{TDF} = \frac{b}{5} \times 2 - (c+p).$$

To mount collision attacks against KECCAK$[r, c, n_r, d]$ with methods described in Section 4, it is necessary that

$$\texttt{TDF} > w_1 + \cdots + w_{n_r-2} + w^d_{n_r-1} \tag{10}$$

where $w^d_{n_r-1}$ is the part of w_{n_r-1} that relates to the digest[5]. The trail searching phase is performed to provide ΔS_I for the connector building algorithm. However, the sufficient conditions for a good trail core is restrained by solving results of the connector, i.e. the number of freedom degrees of the solution space of E_M. So we take (10) as a heuristic condition for searching good trail cores which are promising for collision attacks.

Thirdly, the collision attack should be practical. Note that after we obtain a subspace of message pairs making it sure to bypass the first two rounds, the complexity for searching a collision is $2^{w_2+\cdots+w^d_{n_r-1}}$. To make our attacks practical, we restrict $w_2 + \cdots + w^d_{n_r-1}$ to be small enough, say 48.

We summarize the requirements for differential trails as follows and list TDFs for different versions of KECCAK$[r, c, n_r, d]$ in Table 3.

(1) $\alpha^d_{n_r} = 0$, i.e. the difference of output must be zero.
(2) $\texttt{TDF} > w_1 + \cdots + w^d_{n_r-1}$, i.e. the degree of freedom must be sufficient;
(3) $w_2 + \cdots + w^d_{n_r-1} \leq 48$, the complexity for finding a collision should be low.

Table 3: TDFs of different versions of KECCAK$[r, c, n_r, d]$.

KECCAK$[r, c, n_r, d]$	TDF	Remarks
KECCAK$[1440, 160, 5, 160]$	479	Challenge
KECCAK$[1344, 256, 5, 256]$	378	SHAKE128
KECCAK$[\ 640, 160, 5, 160]$	159	Challenge
KECCAK$[1440, 160, 6, 160]$	479	Challenge
KECCAK$[1152, 448, 5, 224]$	191	KECCAK-224

* 'Challenge' means that that version is included in Keccak Crunchy Crypto Collision and Pre-image Contest [2].

5.4 Searching strategies

Searching from light β_3's. Our initial goal is to find collisions for 5-round KECCAK. To facilitate a 5-round collision of KECCAK, we need to find 4-round differential trails satisfying the three requirements mentioned previously. However it is difficult to meet all of them simultaneously even though each of them can be fulfilled easily.

[5] Suppose all the equations are independent. In later parts we will show these equations are not necessarily independent.

We explain as follows. Since we aim for practical attacks, $w_2 + w_3 + w_4^d$ must be small enough, say 48. That is to say, the last three rounds of the trail must be light and sparse. When we restrict a 3-round trail to be lightweight and extend it backwards for one round, we almost always unfortunately get a heavy state α_2 (usually $\#AS(\alpha_2) > 120$) whose weight may exceeds the TDF. We take KECCAK-224 as an example. The TDF of KECCAK-224 is 191, which indicates $\#\mathrm{AS}(\alpha_2) < 92$ as the least weight for an S-box is 2. For a lightweight 3-round trail, it satisfies Requirement (1) occasionally. The greater d is, the less trails satisfy Requirement (1).

With these requirements in mind, we search for 4-round differential trail cores from light middle state differences β_3's. From light β_3's we search forwards and backwards, and check whether Requirement(1) and (2) are satisfied respectively; once these two requirements are satisfied, we compute the weight $w_2 + w_3 + w_4^d$ for brute force, hoping it is small enough for practical attacks.

α_3, α_4 in CP-kernel. The designers of KECCAK show in [3] that it is not possible to construct 3-round low weight differential trails which stay in CP-kernel. However, 2-round differential trails in CP-kernel are possible, as studied in [9, 14, 19].

We restrict α_3 in CP-kernel. If $\rho^{-1} \circ \pi^{-1}(\beta_3)$ is outside the CP-kernel and sparse, say 8 active bits, the active bits of $\alpha_3 = L^{-1}(\beta_3)$ will increase due to the strong diffusion of θ^{-1} and the sparseness of β_3. When $\#AS(\alpha_3) > 10$, the complexity for searching backwards for one β_3 is greater than $2^{31.7}$ which is too time-consuming. We had better also confine α_4 to the CP-kernel. If not, the requirement $\alpha_{n_r}^d = 0$ may not be satisfied. As can be seen from the lightest 3-round trail for KECCAK-f[1600] [14], even though the θ-gap is only one, after θ the difference bits are diffused among the state making a 224-bit collision impossible (a 160-bit collision is still possible). So our starting point is special β_3's which makes sure $\alpha_3 = L^{-1}(\beta_3)$ lies in CP-kernel, and for which there exists a compatible α_4 in CP-kernel. Fortunately, such kind of β_3's can be obtained with KeccakTools [6].

Steps for searching 4-round differential trails. We sketch below our steps for finding 4-round differential trail cores for KECCAK and provide a description in more detail in Appendix C. To mount collision attacks on 6-round KECCAK, 5-round differential trail cores are needed. In this case, we just extend our forward extension for one more round.

1. Using KeccakTools, find special β_3's with a low Hamming weight, say 8.
2. For every β_3 obtained, traverse all possible α_4 using a tree structure, compute $\beta_4 = L(\alpha_4)$ and test whether there exists a compatible α_5 where $\alpha_5^d = 0$. If so, keep this β_3 and record its forward extension, otherwise discard it.
3. For remaining β_3's, also using a tree structure traverse all possible β_2 which is compatible with $L^{-1}(\beta_3)$'s, compute $\#AS(\alpha_2)$ from β_2. If $\#AS(\alpha_2)$ is small enough, say below 110, check whether this trail core $(\beta_2, \beta_3, \beta_4)$ under consideration is sufficient for collision attacks.

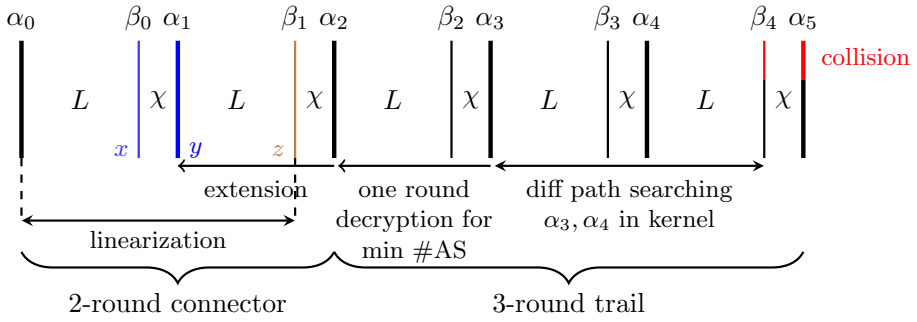

Figure 3: Collision attacks on 5-round KECCAK.

5.5 Searching results

Some of the best differential trail cores we obtained are listed in Table 4. As can be seen that Trail cores No. 1∼3 are all suitable for collision attacks against KECCAK[1440,160,5,160], and Trail cores No. 1 and 2 for SHAKE128. Trail core No. 4 is sufficiently good for collision attacks against KECCAK$[640, 160, 5, 160]$. However, to mount collision attacks on KECCAK-224, all the first three trail cores are not good enough. Fortunately, a doubled version of Trail core No. 4 can make our two-round attack possible because $85 \times 2 = 170 < 191$. For KECCAK$[1440, 160, 6, 160]$, we also find a trail core ripe for collision attacks except that Requirement (3) is not satisfied. Details of these differential trail cores are provided in Appendix D.

Table 4: Differential trail cores for KECCAK$[r, c, n_r, d]$.

No.	$r+c$	$\#AS(\alpha_2\text{-}\beta_2\text{-}\beta_3\text{-}\beta_4^d)$	$w_1\text{-}w_2\text{-}w_3\text{-}w_4^d$	d
1	1600	102-8-8-2	240-19-16-4	256
2	1600	88-8-7-0	195-21-15-0	256
3	1600	85-9-10-2	190-25-20-3	224
4	800	38-8-8-0	85-20-16-0	160
No.	$r+c$	$\#AS(\alpha_2\text{-}\beta_2\text{-}\beta_3\text{-}\beta_4\text{-}\beta_5^d)$	$w_1\text{-}w_2\text{-}w_3\text{-}w_4\text{-}w_5^d$	d
5	1600	145-6-6-10-14	340-15-12-22-23	160

6 Experiments and Results

In this section, we employ 4-round (5-round) trail cores to mount collision attacks against 5-round (6-round) KECCAK$[r, c, n_r, d]$. Our attack consists of two main stages:

– *Connecting stage.* Find a subspace of messages bypassing the first two rounds.
– *Brute-force searching stage.* Find a colliding pair from this subspace by brute force.

In the first stage, with α_2 fixed by the trail core, we choose compatible β_1 where $\alpha_1 = L^{-1}(\beta_1)$ and all the S-boxes in α_1 are active. In order to save freedom degrees, we also restrict that $\beta_1 \to \alpha_2$ should be of least weight. When β_1 is chosen, we run the two-round connector. If a certain number of failures is reached, we select another β_1 until a solution is found, i.e. a subspace of message pairs definitely reaching to α_2 is obtained. If the number freedom degrees of this subspace is large enough, the first stage succeeds. Once the first stage succeeds, we move on to the second stage for finding a colliding message pair.

6.1 Collision Attack of KECCAK[1440, 160, 5, 160]

We apply Trail core No. 2 to the collision attack of 5-round KECCAK[1440, 160, 5, 160]. In this case, we choose compatible β_1s randomly. After solving the two-round problem in 9.6 seconds, the degree of freedom is 162, which is enough for collision search of the remaining 3 rounds with probability 2^{-40}. The searching time for the collision is 2.48 hours. We give one example of collisions in Table 10, with which we solve a challenge of Keccak Crunchy Crypto Collision and Pre-image Contest [2].

6.2 Collision Attack of 5-round SHAKE128

We apply Trail core No. 1 to the collision attack of 5-round SHAKE128[6]. As the capacity of SHAKE128 is much larger than that of KECCAK [1440, 160, 5, 160], which means about 100 more freedom degrees are needed, we just choose compatible β_1s where $\beta_1 \to \alpha_2$ is of least weight. We also follow this rule in later collision attacks. After solving the two-round problem with 25 minutes, the degree of freedom is 94 and the search for 3-round collision with probability 2^{-39} costs half an hour. We give an instance of collision in Table 11.

6.3 Collision Attack of KECCAK[640, 160, 5, 160]

We apply Trail core No. 5 to the collision attack of KECCAK[640, 160, 5, 160]. The methods used in this case are similar to those of 5-round SHAKE128. The first stage succeeds in 30 minutes. The second stage takes 2 hours 40 minutes to find a collision which happens with probability 2^{-35}. An example of collision is provide in Table 12, with which we solve another challenge of Keccak Crunchy Crypto Collision and Pre-image Contest [2].

[6] We also utilized Trail core No. 2, but Trail core No. 1 produces a colliding pair in a relatively shorter time.

6.4 Collision Attack of KECCAK[1440, 160, 6, 160]

We found four trail cores for which there exist zero 160-bit output differences. The one with the best probability is Trail core No. 5 which is displayed in Table 9. From β_4 there are 24 trails to zero α_6^d. Taking all these trails into consideration, we get a complexity of $2^{67.24} \sim 2^{70.24}$ for the second stage. If we let $\#AS(\alpha_2)$ (w_2) be the smallest, the complexity for the second stage is $2^{70.24}$ ($2^{67.24}$). In the experiments, we let $\#AS(\alpha_2)$ be the smallest. In one hour our two-round algorithm returns a subspace of messages with freedom degree 135, and in 20 minutes we get a message pair shown in Table 13 that follows the first four rounds of the differential trail, which demonstrates that in time complexity of $2^{70.24}$ a collision for 6-round KECCAK[1440, 160, 6, 160] will be found with great confidence.

6.5 Collision Attack of 5-round KECCAK-224

For the collision attack of 5-round KECCAK-224, all the 4-round trail cores we found for KECCAK-f[1600] are not good enough, i.e. the weight of the trail cores exceeds TDF too much and even $w_2 >$ TDF. However, our two-round connector is still likely to work. For one hand, from Trail core No. 4 for KECCAK-f[800] we can construct a 4-round trail core for KECCAK-f[1600] with weight pattern (170-40-32-0) which makes our two-round connector possible. From the other, as the capacity increases, it is probable that equations added in connecting phase are not always mutually independent, which means the assumption of freedom degrees of our connector may be less than TDF. The applicability of our connector in this case is verified with experiments. With Trail core No. 4, the two round connector returns a subspace of messages of freedom degree 11 and 2 or 3 for Trail core No. 3. Since the message subspaces derived are too small to mount collision attacks against 5-round KECCAK-224, we turn to two-block messages. Once we get c bits from the first block, we set corresponding c bit constants in E_M to the value we obtained and then solve the system to find a subspace of messages for the second block. Now the attack proceeds in the following way.

- *Connecting stage.*
  - Use the two-round connector to find a message subspace with freedom degree s as large as possible, hoping that $t = (c+p)+\text{rank}(E_M \setminus E_{(c,p)})-\text{rank}(E_M)$ is as small as possible.
- *Brute-force searching stage.*
  - Choose the first message randomly and compute the c-bit value for the second block. Replace the corresponding c bit constants in E_M and check whether it is still consistent. If it is consistent, we obtain another subspace with size 2^s.
  - Search for collision with the subspace.
  - Repeat until we find a two-block collision.

In our experiment, using Trail core No. 3 the connector returns a message subspace with freedom degree $s = 2$, and $t = 55$. Then the complexity for find a two-block collision is $2^{55+(48-2)} = 2^{101}$.

6.6 Re-launch 4-round Collision Attacks of KECCAK-224 and KECCAK-256

Though the 4-round collisions of KECCAK-224 and KECCAK-256 have already been found [10], we use our method to optimize the complexity. We start from the same 2-round differential trail in Dinur *et al.*'s work [10] and build a two-round connector. The time spent on building and solving the two-round connectors is 2 minutes 15 seconds for KECCAK-224 and 7 minutes for KECCAK-256. Then the complexity for brute forth searching is reduced to 2^{12} and cost 0.325 seconds and 0.28 seconds respectively which outperforms 2^{24} on-line complexity in [10]. Besides, it is pointed out in [10] that even though they got subsets with more than 2^{30} message pairs from their target difference algorithm, they were not able to find collisions within some of these subsets. The reason was suspected to be the incomplete diffusion within the first two rounds and the closely related message pairs within a subset. While in our algorithm, we did not encounter such a problem. In other words, we always find collisions from the subsets deduced from the two-round connector. Thus once we succeed in the 2-round connector building phase with a large enough subset, we never need to repeat it.

7 Conclusion

In conclusion, we observed that the KECCAK S-box can be re-expressed as linear transformations under some restricted input subspaces. With this property, we linearized all S-boxes of the first round, and extended the existing connector by one round. Implementations confirmed our idea, and found us real examples of 5-round SHAKE128, and two instances of KECCAK challenges. Theoretical results on 5-round KECCAK-224 and a 6-round KECCAK challenge version are projected.

It is noted that the algorithm for solving the two-round connectors are heuristic, further work includes finding the theoretical bounds of this algorithm and factors deciding the complexities for possible improvements. Note, any relaxation on the restrictions of ΔS_I might lead us to better differential trails in the searching phase.

Acknowledgement. The authors would like to thank anonymous reviewers and Joan Daemen for their helpful comments and suggestions. The work of this paper was supported by the National Key Basic Research Program of China (2013CB834203) and the National Natural Science Foundation of China (Grants 61472417, 61472415, 61402469, and 61672516).

References

1. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. rump session of Cryptographic Hardware and Embedded Systems-CHES **2009** (2009) 67

2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak Crunchy Crypto Collision and Pre-image Contest. http://keccak.noekeon.org/crunchy_contest.html
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference, version 3.0 (2011). http://keccak.noekeon.org/Keccak-reference-3.0.pdf
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic Sponge functions. Submission to NIST (Round 3) (2011)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak SHA-3 submission. Submission to NIST (Round 3) **6**(7) (2011) 16
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccaktools. http://keccak.noekeon.org/ (2015)
7. Canteaut, A., ed.: FSE 2012. In Canteaut, A., ed.: FSE 2012. Volume 7549 of LNCS., Washington, DC, USA, Springer, Heidelberg, Germany (March 19–21, 2012)
8. Daemen, J.: Cipher and hash function design strategies based on linear and differential cryptanalysis. PhD thesis, Doctoral Dissertation, March 1995, KU Leuven (1995)
9. Daemen, J., Van Assche, G.: Differential Propagation Analysis of Keccak. [7] 422–441
10. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. [7] 442–461
11. Dinur, I., Dunkelman, O., Shamir, A.: Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials. In Moriai, S., ed.: FSE 2013. Volume 8424 of LNCS., Singapore, Springer, Heidelberg, Germany (March 11–13, 2014) 219–240
12. Dinur, I., Dunkelman, O., Shamir, A.: Improved Practical Attacks on Round-Reduced Keccak. Journal of Cryptology **27**(2) (April 2014) 183–209
13. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2015) 733–761
14. Duc, A., Guo, J., Peyrin, T., Wei, L.: Unaligned Rebound Attack: Application to Keccak. [7] 402–421
15. Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., Sim, S.M.: Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs. IACR Transactions on Symmetric Cryptology **1**(1) (2017) To appear.
16. Jean, J., Nikolic, I.: Internal Differential Boomerangs: Practical Analysis of the Round-Reduced Keccak-f Permutation. In Leander, G., ed.: Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Volume 9054 of LNCS., Springer (2015) 537–556
17. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In Lee, D.H., Wang, X., eds.: ASIACRYPT 2011. Volume 7073 of LNCS., Seoul, South Korea, Springer, Heidelberg, Germany (December 4–8, 2011) 288–307
18. National Institute of Standards and Technology: SHA-3 STANDARD: PERMUTATION-BASED HASH AND EXTENDABLE-OUTPUT FUNCTIONS. Federal Information Processing Standards (FIPS) Publication Series (2015)
19. Naya-Plasencia, M., Röck, A., Meier, W.: Practical Analysis of Reduced-Round Keccak. In Bernstein, D.J., Chatterjee, S., eds.: INDOCRYPT 2011. Volume 7107 of LNCS., Chennai, India, Springer, Heidelberg, Germany (December 11–14, 2011) 236–254

A 2-dimentional Linearizable Affine Subspaces of Keccak S-box

There are totally 80 2-dimensional linearizable affine subspaces for Keccak S-box as listed in Table 5.

Table 5: Linearizable affine subspaces of Keccak S-box

{0, 1, 4, 5}	{2, 3, 6, 7}	{0, 1, 8, 9}	{4, 5, 8, 9}	{0, 2, 8, A}
{1, 2, 9, A}	{0, 3, 8, B}	{1, 3, 9, B}	{2, 3, A, B}	{6, 7, A, B}
{0, 1, C, D}	{4, 5, C, D}	{8, 9, C, D}	{4, 6, C, E}	{5, 6, D, E}
{4, 7, C, F}	{5, 7, D, F}	{2, 3, E, F}	{6, 7, E, F}	{A, B, E, F}
{0, 2, 10, 12}	{8, A, 10, 12}	{1, 3, 11, 13}	{9, B, 11, 13}	{0, 4, 10, 14}
{1, 5, 10, 14}	{2, 4, 12, 14}	{0, 4, 11, 15}	{1, 5, 11, 15}	{3, 5, 13, 15}
{10, 11, 14, 15}	{0, 6, 10, 16}	{2, 6, 12, 16}	{3, 7, 12, 16}	{4, 6, 14, 16}
{C, E, 14, 16}	{1, 7, 11, 17}	{2, 6, 13, 17}	{3, 7, 13, 17}	{5, 7, 15, 17}
{D, F, 15, 17}	{12, 13, 16, 17}	{10, 11, 18, 19}	{14, 15, 18, 19}	{0, 2, 18, 1A}
{8, A, 18, 1A}	{10, 12, 18, 1A}	{11, 12, 19, 1A}	{10, 13, 18, 1B}	{1, 3, 19, 1B}
{9, B, 19, 1B}	{11, 13, 19, 1B}	{12, 13, 1A, 1B}	{16, 17, 1A, 1B}	{8, C, 18, 1C}
{9, D, 18, 1C}	{A, C, 1A, 1C}	{8, C, 19, 1D}	{9, D, 19, 1D}	{B, D, 1B, 1D}
{10, 11, 1C, 1D}	{14, 15, 1C, 1D}	{18, 19, 1C, 1D}	{8, E, 18, 1E}	{A, E, 1A, 1E}
{B, F, 1A, 1E}	{4, 6, 1C, 1E}	{C, E, 1C, 1E}	{14, 16, 1C, 1E}	{15, 16, 1D, 1E}
{9, F, 19, 1F}	{A, E, 1B, 1F}	{B, F, 1B, 1F}	{14, 17, 1C, 1F}	{5, 7, 1D, 1F}
{D, F, 1D, 1F}	{15, 17, 1D, 1F}	{12, 13, 1E, 1F}	{16, 17, 1E, 1F}	{1A, 1B, 1E, 1F}

B Differential Distribution Table of Keccak S-box [14]

C Steps for Finding Differential Trials

In this section, we describe more at length about the steps for finding differential trails.

1. Generate β_3s each of which makes sure $\alpha_3 = L^{-1}(\beta_3)$ lies in CP-kernel, and for each β_3 there exists a compatible α_4 in CP-kernel using TrailCoreInKernelAtC of KeccakTools [6] where the parameter *aMaxWeight* is set to be 52. As a result, 503 such β_3s are obtained.
2. For every β_3 obtained, if $\#AS(\beta_3) < 16$ we traverse all possible α_4 using a tree structure, compute $\beta_4 = L(\alpha_4)$ and test whether there exists a compatible α_5 where $\alpha_5^d = 0$. If so, keep this β_3 and record its forward extension, otherwise discard it. For $d = 224$, 351 β_3s are left while 495 β_3s are left for $d = 160$.
3. For remaining β_3s, we also use a tree structure to search backwards. For each β_3 compute $\alpha_3 = L^{-1}(\beta_3)$. If $\#AS(\alpha_3) < 10$, traverse all possible β_2s which is compatible with α_3, compute $\#AS(\alpha_2)$ from β_2. If $\#AS(\alpha_2)$ is small enough, say below 110, check whether this trail core $(\beta_2, \beta_3, \beta_4)$ under consideration is sufficient for collision attacks.

Parameters and conditions in our algorithm can be changed and we just set them as described for the sake of practicality. For example, in the third step if

Table 6: The differential distribution table of the χ when viewed as S-box. The first bit of a row is viewed as the least significant bit. Given input difference Δ_{in} and output difference Δ_{out} the number in the table shows the size of the solution set $\{v \mid \chi(v) + \chi(v + \Delta_{in}) = \Delta_{out}\}$. Differences are in hex number.

Δ_{in} \ Δ_{out}	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00	32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
01	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	-	8	-	-	-	-	-
02	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-
03	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-
04	-	-	-	-	8	8	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
05	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4
06	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-
07	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2
08	-	-	-	-	-	-	-	-	8	-	8	-	8	-	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
09	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4
0A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4
0B	-	4	4	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	4	4	-
0C	-	-	-	-	-	-	-	-	4	4	4	4	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0D	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-
0E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2
0F	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-
11	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-
12	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4
13	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	-	-	4	4	4	4	-	-	-	-	4	4
15	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-
16	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-
17	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-
19	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2
1A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-
1B	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-
1C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1D	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-
1E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	-	-	2	2	-	-	-	-
1F	-	2	2	-	2	-	-	2	2	-	-	2	-	2	2	-	2	-	-	2	-	2	2	-	-	2	2	-	2	-	-	2

$\#AS(\alpha_3) \geqslant 10$, it costs too much time for one β_3 ($\geqslant 2^{31.7}$). For each β_3, the backward search costs more time than the forward search because of the property of χ. Since the corresponding α_3 has the same number of active bits with β_3, the numbers of active S-boxes are the same for both extension. Further, active bits are rather sparse in both α_3 and β_3, and the active S-boxes in them are almost all with one active bit. Note that given a one-bit input difference of an S-box, there are only 4 possible output difference, while there are 9 possible input differences given an one-bit output difference. This is the reason why the backward search is more time-consuming.

To find a 5-round trail core for KECCAK$[1440, 160, 6, 160]$, we adapt the second step as follows.

2. We first extend forwards from β_3 for one round using KeccakFTrailExtension of KeccakTools [6] with weight up to 36. Then for β_4 of each trail core, if $\#AS(\beta_4) < 16$, traverse all possible α_5, compute $\beta_5 = L(\alpha_5)$ and test whether there exists a compatible α_6 where $\alpha_6^d = 0$. If so, we keep the β_3 and record the three-round trail core $(\beta_3, \beta_4, \beta_5)$, otherwise we discard the β_3.

In the end four trail cores remain. In order to reduce the weight, we take multiple trails from β_4 to α_5 into consideration and the trail core in Table 9 is the best among the four trail cores.

D Differential Trails

In this section, we give details of differential trails of KECCAK mentioned in Section 5. Actually we present trail cores. For example, a 4-round tail core $(\beta_2, \beta_3, \beta_4)$ consisting three state differences represents a set of 4-round differential trails

$$\alpha_1 \xrightarrow{L} \beta_1 \xrightarrow{\chi} \alpha_2 \xrightarrow{L} \beta_2 \xrightarrow{\chi} \alpha_3 \xrightarrow{L} \beta_3 \xrightarrow{\chi} \alpha_4 \xrightarrow{L} \beta_4 \xrightarrow{\chi} \alpha_5$$

where α_5 is compatible with β_4 and $\beta_1 \to \alpha_2$ is of the least weight determined by β_2. In our collision attacks of 5-round KECCAK, 4-round trail cores are needed. In this section, we not only present trail cores used in collision attacks, but also two 5-round trail cores we found.

Each state difference is represented with a matrix of 5×5 lanes, ordered from left to right, where '|' is used as a separator between lanes; each lane is given in hexadecimal using the little-endian format and '0' is replaced with '-'.

E Collisions for Keccak$[r, c, n_r, d]$

In this section, we give instances of collisions against KECCAK$[1440, 160, 5, 160]$, 5-round SHAKE128 and KECCAK$[640, 160, 5, 160]$ respectively. Note that we denote two colliding messages with M_1, M_2. For 5-round KECCAK-224 and KECCAK $[1440, 160, 6, 160]$, we are unable to find collisions for them because of the limitation of computation power. However, we can demonstrate the soundness of our method by providing instances of massage pairs that follow first 4 rounds of the trail of KECCAK$[1440, 160, 6, 160]$ and first 2 rounds of the trail of 5-round KECCAK-224.

Table 7: Trail core No.1 $\sim$ 3 used in the collision attacks

Trail core No. 1, used in the collision attack of 5-round SHAKE128	
β_2	----------|----------24|----------2-|----------|---------- ----------|----------4|----------|----------|--------4------ ----------|------4---------|----------2-|----------|---------- ----------|----------|----------|----------|---------- --------4------|------4---------|---------4------|--------4------|--------4------
β_3	----------|----4----------|---1----------|----------|-----1---------- ----------|----4----------|----------|----------|---------- ----------8|----------|----------|----------|---------- ----------|----------|---1----------|----------|---------- ----------8|----------|----------|----------|-----1----------
β_4	----------|----------|----------|----------|2-4--------- ----------|----------8|----------4-|---------8------|---------- ----------|----------|----------|--8---------|---------2----- ----------4-8|----------4----|-1----------|8----------|---------- ----4----------|----------|---------4-----|----------|---1----------

Trail core No. 2, used in the collision attack of KECCAK[1440, 160, 5, 160]	
β_2	----------|----------|----------|----------|---------- -------1------|----------8----|----------1--|-------1----1--|---------- -------1------|----------|----------1-|----------|---------- ----------|----------8----|----------|----------|----------1- ----------|----------|----------1--|----------1--|----------
β_3	----------|8---------|----------|----------|---------- ----------|----------|-------8------|----------1|---------- ----------|----------|----------|----------1|---------- ----------|----------1|----------|----------|----------1 ----------|8---------|-------8------|----------|----------
β_4	----------|----------|----------|----------|---------- ----------|----------|----------|----2-------|--------1------ ----------1|------2---------|---------2------|----------|---------- ----------|----------|----------4--|----------|---------- ----------|--8---------|----------|----------|----------2

Trail core No. 3, used in the collision attack of 5-round KECCAK-224	
β_2	----------|----4----------|----4------1---|----4----------|----4---------- --------4------|----------4-----|----------|--------4------|--------4------ ----------|----------|----------|----------|---------- -----------1---|----------4-----|-----------1---|----4----------|---------- --------4------|----4----------|--------4------|--------4------|--------4------
β_3	----------|----------4|----------|----------8|---------- ----------4--|----------|----------|--------8----8|---------- ----------|----------|----------|----------|---1---------- ----------|----------4|----------|--------8-----|---------- ----------4--|----------|----------|----------|---1----------
β_4	----------|----------|----------|---1----------|4---------- --------8------|----------|----------|----8----------|---------- ----------8|----------|----------|----------|--------1------ ----------|----4----------|----------|----------|---------- ----------|-4--------4----|---------8-----|----------|----------

Table 8: Trail core No. 4, used in the collision attack of KECCAK[640, 160, 5, 160] and its doubled version can be used in the collision attack of KECCAK-224. β_4 has two choices.

Trail core No. 4			
β_2	--------|-----8--|-----A--|-----8--|-----8--| $\beta_4^{(1)}$	--------|--------|--------|--------|-------- 1------4|-4------|--------|--------|-------- --------|------4-|-2------|--------|-2------ -------2|-----8--|--------|--------|4------- 4-------|--------|--------|--------|--------
β_3	--------|--------|--------|-------1|------4- ------8-|--------|-------1|--------|------4- --------|--------|--------|-------1|-------- --------|--------|--------|--------|-------- ------8-|--------|--------|------4-|--------	$\beta_4^{(2)}$	--------|--------|--------|--------|-------- 1-------|-4------|--------|--------|-------- --------|------4-|-2------|--------|-2------ -------2|-----8--|--------|--------|4------- 4-------|2-------|--------|--------|--------

Table 9: Trail cores No. 5, used in the collision attack of Keccak[1440, 160, 6, 160].

β_2	----------|----------|----------|----------|----------	----------|---2------|----------|---2------|----------1-	----------1-|----------|----------|---2------|----------	----------1|----------1|----------|---2------|----------	----------1-|----------1|----------|----------|----------
β_3	----------|----------|----------|----------|----------	----------|----------|----------|----2-----|----------	----------|----------|----------4--|----------|----------	----------|-----1----|----------4--|----------|----------	----------|-----1----|----------|----2-----|----------
β_4	----------|----------|--2-------|----------|----------	----------|----------|----------2---|---------2-----|-----4----	----------|---8------|----------|----------|----------	----------|----------|----------|---------2-----|------2---------	----------|------1----|----------|---8------|-----4----
β_5	----------|-4-----4----|-------8--28-22|-C--4--4-----4--|-1-8--88------2	-----2-----2---4|42--22-------8-|----------|-8-----8-------|---2---A--88----	--8---------8---|-4--14-11--8----|8-----4-----4---|-8-----22------|----------	--11--------4--1|----------|2--------1-----1|--28-22--------8|----28--4-----2-	---C--14-11-----|----1---2---1-1-|1--------4--1--1|----------|-12---4----1----

Table 10: Collision for Keccak[1440, 160, 5, 160]

M_1	C09C5501A913CC3C|7406D907E6569334|89182C870A0387A0|980A9D8F82C40A90|9306194AEBBC1C17	6D7DFE249ED35BB5|35C1981BFF84755C|37E7FA11AAD390EB|19485675C7530B8E|042893444D9EC364	6D317B9B40DE874C|E2EC2A3613678DDA|3939A7F72AC29BF6|4FABBC80AE5192EA|AB50ABCBCC7E5CC7	0F152006D01F65AC|AEC5B4B7EEC068E1|58E287388571520A|569ED102CB7D2EFA|4AC1C2A0645D5B2C	4C323DADBB2DAFC4|36F6BEEB558F2B22|0000000089F71BE8|0000000000000000|0000000000000000
M_2	D634EAE0EF26F002|90371C35BB5CFABC|7396C3D058D2F577|78CDF403D882B742|22ECA6BCBFC9501F	2352A9667EB05FCD|4CA3FD90EFB8A2D3|8DDFF276C0B60599|4B4CCD54AD6B2646|A490FAFA55BF4E37	234734EA58D9191D|3C580CA9664107ED|29E6AEB01815FB08|8FB33829BABDF8C2|48A21B6E764A7987	D9FA24DCB0331C80|9272D67CEF52F8E3|0C82810B4BE7307A|CF164B325F4DEEBE|BA41517B4D315C3C	99CD68FF39016FC4|AB018238479D9A8D|00000000E3233895|0000000000000000|0000000000000000
digest	A6E173DCDFC3E8EF|8242EAEA1EE736D5|	E33875A0			

Table 11: Collision for 5-round SHAKE128

M_1	0A3E44EBE62104A0|1E8617C352E80FBC|B69A38114369962F|1237F5EEA8045DAB|D4144AC64E22044C	1240A93D79FCCB2E|C8C63A830CBACFFC|B36B34C0E1719824|F94803ECC5586680|ACF133FE29839CAD	CA5F88F260DEFAA7|972FE7E882A4AB03|D11344BE12431A54|814488EBAE68F93B|56D10CF0251FAED3	77A665FCC5F52D9F|D50EF69FAB128ACC|87F3F1816E740894|770D4D55489234B0|737134B1243F3A3D	FE0E2AE7F23D8E40|0000000000000000|0000000000000000|0000000000000000|0000000000000000
M_2	0CDDD5D25A8BD7CA|F71D259EE445A4D8|EB84F177D51C9D45|0A70C1FC50024C24|7108096E3F024F63	3B8D1EEBC5E9150E|EADCC9FB19824E75|B8A97CB74697BAAD|5988E2CD64063AD1|FB55123185E2E4A4	E74FF74033CA1486|915F016B41BDAF6B|145441AAC9EFA342|D9A609CF15E6C626|5609C4F58F5DEE0A	AB4E178C43BA8687|3774B01D78F2ABE4|AA35E3D371664594|A26EAD50F73069A7|DE4E25F8A0F8E928	FE431BE34F8371D8|0000000000000000|0000000000000000|0000000000000000|0000000000000000
digest	9D2E953AD7C6A939|326F59A68A6016EF|A71EAFEE371700D7|3C463D5D098D9B76				

Table 12: Collision for Keccak[640, 160, 5, 160].

M_1	297DB73F|CE5FB46D|63EFD5AB|AB75DBB2|020119E7	06927773|A645A6A4|68E6E3F8|15282462|633AAB83	96C7A5FB|5E4CBEB5|92614C96|DD9647DA|D4B0094F	4C68376F|D3B63751|6286AB56|DE577A52|9003EA0F	00000000|00000000|00000000|00000000|00000000	
M_2	5B150BCB|C0F3F2FC|5907B5A5|22736DC3|914CF0C5	87477D63|A675A649|8BBEA96F|52EB8AE3|19402D41	D9FB4CC3|669FD630|D8C9FC71|57558554|0662F64A	64B4B5C5|7F12BF56|2BEADBF0|F6207B10|F2FD9787	00000000|00000000|00000000|00000000|00000000	
digest	F90B5ABA|7430682D|85668C62|66E1B0AD|B052AC35					

Table 13: Collision attack of Keccak[1440, 160, 6, 160]. From a message space of freedom degree 135 which definitely leads to β_2 of Trail core No. 5, we present a pair of messages following the first 4 rounds. To find a collision of Keccak[1440, 160, 6, 160], it costs a time complexity of $2^{70.24}$

M_1	F33E499A09AD9B73	CA5358DF1D89473D	1B984D8B14D538AB	F7B7ADD4FBE9425A	B9B58D552AB12786
	CEA1C136D6ADE06C	7CCD4A72E45C8222	3337867744F8E6B1	416948B8E8F30A01	E9BA1151837A99C0
	CB2F620C029A0E29	6BDA24629364CE16	C1B6C702D518B1C0	DF3D3F6121C87C5E	1A5154511DCF5069
	97CF66E84A7DF86F	E6D669B526963387	EC88B00FD1D1328A	7DB7FCD1A05744B2	288722B23E653CF2
	051F63BA5C5EC16E	C7F1EF8734BFB4FA	000000008CE14DA4	0000000000000000	0000000000000000
M_2	B3411F19C5C972E6	C6CF4F990A6FAD57	354F4EA8568AB4A4	E4A48E7A516C6A62	92B3E65F1A4114A9
	28D238874AD48D50	2F4B715A451EFF5E	516C7EC96CCF73BB	24638F92E701C38B	D83A34C323CBB335
	6F9D34ADABA26565	48276ACC061BF678	37B2B688A051EDB2	E93C28A0A17F2CEA	5A976AD0DAF3CB8D
	5235F8FF84041376	95F8173A97D0D448	D3D8B045A3008325	28C4FD73A1542DB7	B8AB1796BACD1E17
	C8ECEF0F993328A3	C2F7160C897CD9A4	00000000C6BAF84B	0000000000000000	0000000000000000

Table 14: Collision attack of 5-round Keccak-224. From a message space of freedom degree 2 which definitely leads to β_2 of Trail core No. 3, we present a pair of messages following the first 2 rounds. To find a collision of 5-round Keccak-224, it costs a time complexity of 2^{101}.

M_1	8E85F0BC15BBA27B	776FF9140B9AED24	52C6D4A9251C9886	D74E6FC4FB7EE6CB	40C5BF16312FEA11
	0618F45C4F30B4EA	FEA4F176FFB65180	8B03CFC2E0C168A8	E47CFF2303F924D6	280AC9CC77707399
	790244BCD16F3621	4125A834D1FEB877	5DA576EB0306BE03	5498B00302BECD5C	F13E10DD2A230829
	26AADDAF76496EA0	E3EC0DC10D9FC852	9CDF3DE3421ACD7B	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
M_2	D756F690FEDC7326	248DA8A7D0F3432D	276828C7C3A3728A	CFAC96E3956C5F35	AC2B2D679F6F1745
	933191AEDE3EB500	6562F098D4099896	D24C31AF425CE2E7	ADC9058BB5E4FEFC	06D4880875530CB0
	C2C4BCB373FFAD24	C2E9DCD965CAA725	06B1F37EE7F51056	4D43F63490D82FBC	18FA91952DC4DB40
	14F2289283846D81	73FE0284B87D9815	91590D20B7E9251F	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000