Cryptanalysis of Round-Reduced KECCAK using Non-Linear Structures

Mahesh Sreekumar Rajasree

Center for Cybersecurity, Indian Institute of Technology Kanpur, India
mahesr@iitk.ac.in

Abstract. In this paper, we present new preimage attacks on KECCAK-384 and KECCAK-512 for 2, 3 and 4 rounds. The attacks are based on non-linear structures (structures that contain quadratic terms). These structures were studied by Guo et al. [1] and Li et al. [2] [3] to give preimage attacks on round reduced KECCAK. We carefully construct non-linear structures such that the quadratic terms are not spread across the whole state. This allows us to create more linear equations between the variables and hash values, leading to better preimage attacks. As a result, we present the best theoretical preimage attack on KECCAK-384 and KECCAK-512 for 2 and 3-rounds and also KECCAK-384 for 4-rounds.

Keywords: KECCAK, SHA-3, hash function, cryptanalysis, preimage attack

1 Introduction

Cryptographic hash functions are widely used in modern cryptography such as in digital signatures, message integrity and authentication. The U.S. National Institute of Standards and Technology (NIST) announced the “NIST hash function competition” for the Secure Hash Algorithm-3 (SHA-3) in 2006. They received 64 proposals from around the world. Among these, KECCAK designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche [4] became one of the candidates for SHA-3. It won the competition in October 2012 and was standardized as a “Secure Hash Algorithm 3” [5].

The KECCAK hash family is based on the sponge construction [6]. Its design was made public in 2008 and since then, it has received intense security analysis [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]. In 2016, Guo et al. [1] formalised the idea of linear structures and gave practical preimage attacks for 2 rounds KECCAK-224/256. They also gave better preimage attacks for KECCAK-384/512, all variants of 3-rounds KECCAK as well as preimage attacks for 4-rounds KECCAK-224/256. Li et al. [2] improved the complexity of preimage attack for 3-rounds KECCAK-256 by using a new type of structure called cross-linear structure. The best-known attacks for 3 and 4 rounds KECCAK-224/256 is given by Li et al. [3] using a new technique called allocating approach, which consists of two phases - Precomputation phase and Online phase. They gave the first practical preimage attack for 3-rounds KECCAK-224. Apart from the attacks mentioned above, there are several other attacks against KECCAK.

Rounds	**Instances**	**Complexity**	**References**
1	224 256 384 512	Constant	[15]
2	224 256 384 512	Constant Constant 2^{129} 2^{384}	[1]
2	384	Time 2^{89} Space 2^{87}	[16]
2	384 512	$\mathbf{2^{113}}$ $\mathbf{2^{321}}$	Subsection 3.2 Subsection 3.1
3	224 256	2^{38} 2^{81}	[3]
3	384 512	2^{322} 2^{482}	[1]
3	384 512	$\mathbf{2^{321}}$ $\mathbf{2^{475}}$	Subsection 3.4 Subsection 3.5
4	224 256	2^{207} 2^{239}	[3]
4	384 512	2^{378} 2^{506}	[12]
4	384	$\mathbf{2^{371}}$	Subsection 3.6

Table 1. Summary of preimage attacks

Our contributions: In this paper, we give the best theoretical preimage attacks for KECCAK-384 for 2,3,4 rounds and KECCAK-512 for 2,3 rounds. This is achieved by carefully constructing non-linear structures such that the quadratic terms are not spread throughout the whole state and the number of variables in the system of equations is more. Table 1 summaries the best theoretical preimage attacks up to four rounds and our contributions. The space complexity is most of the attacks is constant unless it is explicitly mentioned.

Organization: The rest of the paper contains the following sections. In Section 2, we will give a brief description about KECCAK, some preliminaries and notations that are used throughout the paper and useful observations about KECCAK. Section 3 contains detailed description of all our preimage attacks.

2 Structure of KECCAK

KECCAK hash function is based on sponge construction[6] which uses a padding function *pad*, a bitrate parameter r and a permutation function f as shown in Figure 1.

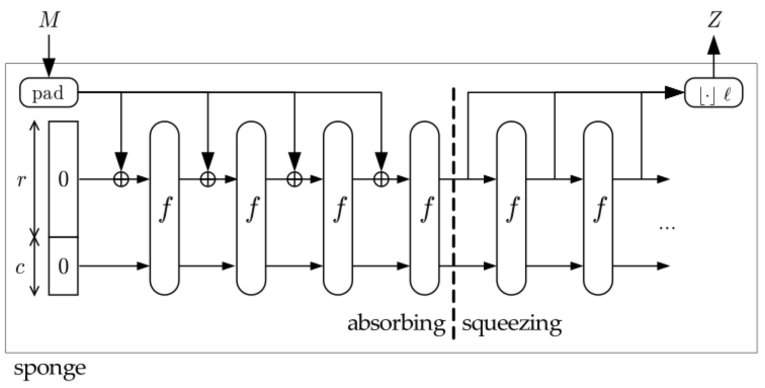

Fig. 1. Sponge function [6]

2.1 Sponge Construction

As shown in Figure 1, the sponge construction consists of two phases - absorbing and squeezing. It first applies the padding function *pad* on the input string M which produces M' whose length is a multiple of r. In the absorbing phase, M' is split into blocks of r bits namely $m_1, m_2, ...m_k$. The initial state (IV) is a b bit string containing all 0. Here $b = r + c$ where c is called the capacity. The first r bits of IV is XORed with first block m_1 and is given as input to f. The output is XORed with the next message block m_2 and then is given as input to f again. This process is continued till all the message blocks have been absorbed.
The squeezing phase extracts the required output, which can be of any length. Let l be the required output length. If $l \leq r$, then the first l bits of the output

of absorbing phase is the output of the sponge construction. Whereas, if $l > r$, then more blocks of r bits are extracted by repeatedly applying f on the output of the absorbing phase. This process is repeated enough number of times until we have extracted at least l bits. The final output of the sponge construction is the first l bits that have been extracted.
In the KECCAK hash family, the permutation function f is a KECCAK-$f[b]$ permutation, and the *pad* function appends 10^*1 to input M. KECCAK-f is a specialization of KECCAK-p permutation.

$$\text{KECCAK-}f[b] = \text{KECCAK-p}[b, 12 + 2\gamma]$$

where $\gamma = log_2(b/25)$.
The official version of KECCAK have $r = 1600 - c$ and $c = 2l$ where $l \in \{224, 256, 384, 512\}$ called KECCAK-224, KECCAK-256, KECCAK-384 and KECCAK-512.

2.2 KECCAK-p permutation

KECCAK-p permutation is denoted by KECCAK-p$[b, n_r]$, where $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ is the length of the input string and n_r is the number of rounds of the internal transformation. The parameter b is also called the width of the permutation. The b bit input string can be represented as a $5 \times 5 \times w$ 3-dimensional array known as state as shown in Figure 2. A lane in a state S is denoted by $S[x, y]$ which is the substring $S[x, y, 0]|S[x, y, 1]| \ldots |S[x, y, w - 1]$ where w is equal to $b/25$ and "|" is the concatenation function.

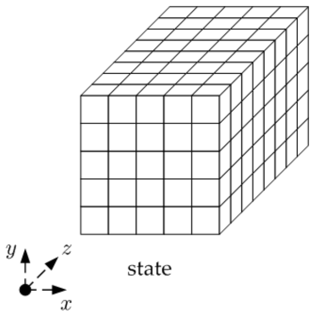

Fig. 2. KECCAK state [17]

In each round, the state S goes through 5 step mappings θ,ρ,π,χ and ι, i.e. $Round(S, i_r) = \iota(\chi(\pi(\rho(\theta(S)))), i_r)$ where i_r is the round index. Except for χ, rest of the step mappings are linear. In the following, S' is the state after applying the corresponding step mapping to S, "$\oplus$" denotes bitwise XOR and "$\cdot$" denotes bitwise AND.

1. θ: The θ step XOR's $S[x, y, z]$ with parities of its neighbouring columns in the following manner.

$$S'[x, y, z] = S[x, y, z] \oplus P[(x + 1) \ mod\ 5][(z - 1) \ mod\ 64] \\ \oplus P[(x - 1) \ mod\ 5][z]$$

where $P[x][z]$ is the parity of a column, i.e,

$$P[x][z] = \bigoplus_{i=0}^{4} S[x, i, z]$$

2. ρ: The ρ step simply rotates each lane by a predefined value given in the table below, i.e

$$S'[x, y] = S[x, y] << r[x][y]$$

where $<<$ means bitwise rotation towards MSB of the 64-bit word.

4	18	2	61	56	14
3	41	45	15	21	8
2	3	10	43	25	39
1	36	44	6	55	20
0	0	1	62	28	27
$y \backslash x$	0	1	2	3	4

3. π: The π step interchanges the lanes of the state S.

$$S'[y, 2x + 3y] = S[x, y]$$

4. χ: The χ step is the only non-linear operation among the 5 step mappings due to the quadratic term.

$$S'[x, y, z] = S[x, y, z] \oplus ((S[(x + 1) \ mod\ 5, y, z] \oplus 1) \cdot \\ S[(x + 2) \ mod\ 5, y, z])$$

5. ι: The ι step is the only step that depends on the round number.

$$S'[0, 0] = S[0, 0] \oplus RC_i$$

where RC_i is a constant which depends on i where i is the round number.

2.3 Preliminaries and Notations

In this paper, we will be using the following observation made by Guo et al. [1]. The χ step mapping is a row dependent operation. Let a_0, a_1, a_2, a_3, a_4 be the 5 input bits to the χ operation and b_0, b_1, b_2, b_3, b_4 be the 5 output bits.

Observation 1 *Let d_0, d_1, d_2, d_3, d_4 be the elements of a column. Then, the parity of column can be fixed to a constant c by choosing for any $i \in \{0,1,2,3,4\}$*

$$d_i = c \oplus \left(\bigoplus_{j=1}^{j=4} d_{i+j} \right)$$

Observation 2 *If the output of χ for an entire row is known, i.e $\chi([a_0, a_1, a_2, a_3, a_4]) = [b_0, b_1, b_2, b_3, b_4]$, then we have*

$$a_i = b_i \oplus (b_{i+1} \oplus 1) \cdot (b_{i+2} \oplus (b_{i+3} \oplus 1) \cdot b_{i+4})$$

In the rest of the paper, all the message variables and hash values are represented in the form of lanes (array) of length 64, and we will use $+$ symbol in place of $\oplus$. For a state A, $A[x, y]$ denotes a lane where $0 \leq x, y \leq 4$. In all the equations, the value inside the brackets '()' indicates the offset by which the lane is shifted. For example, $A[x, y](k)$ denotes lane $A[x, y]$ rotated by an offset of k. Every operation between two lanes is bitwise.

3 Our preimage attacks

In this section, we present the preimage attacks for round reduced KECCAK. In [1], the authors try to set up linear equations between message bits (variables) and hash bits by controlling the diffusion due to θ and χ from producing any non-linear terms. Observation 1 is used to manage the diffusion due to θ. Lanes are fixed to constant to prevent χ from creating any non-linear terms. Furthermore, for KECCAK-384/512, the first row of the hash digest can be inverted due to Observation 2.
In most cases, the number of linear equations between the variables and hash values is strictly less than the hash length. Therefore, they repeat the whole procedure enough number of times by appropriately changing the constants in the system of linear equations. This gives a successful preimage attack. In [2] [3], similar techniques are used to restrict χ from producing many non-linear terms. Here, we allow χ to produce non-linear terms, but at the same time, we control the number of non-linear terms in the state.

3.1 Preimage attack on 2 rounds KECCAK-512

In this subsection, we describe our preimage attack for 2-rounds KECCAK-512. The best-known attack for this variant of KECCAK is by Guo et al.[1] with a complexity of 2^{384}. Their preimage attack is based on a linear structure by keeping four lanes as variables. We give two preimage attacks using six lanes as variables. In the first preimage attack, we keep the lanes in column 1, 3, 4 as variables and get an attack of complexity 2^{337}. However, the second preimage attack chooses a different set of lanes as variables and has complexity of 2^{321}.

Preimage attack with complexity 2^{337}: In figure 3, we set the lanes in column 1, 3 and 4 as variables, and the rest of the lanes are set to some constant. Therefore, we have $6 \times 64 = 384$ variables.

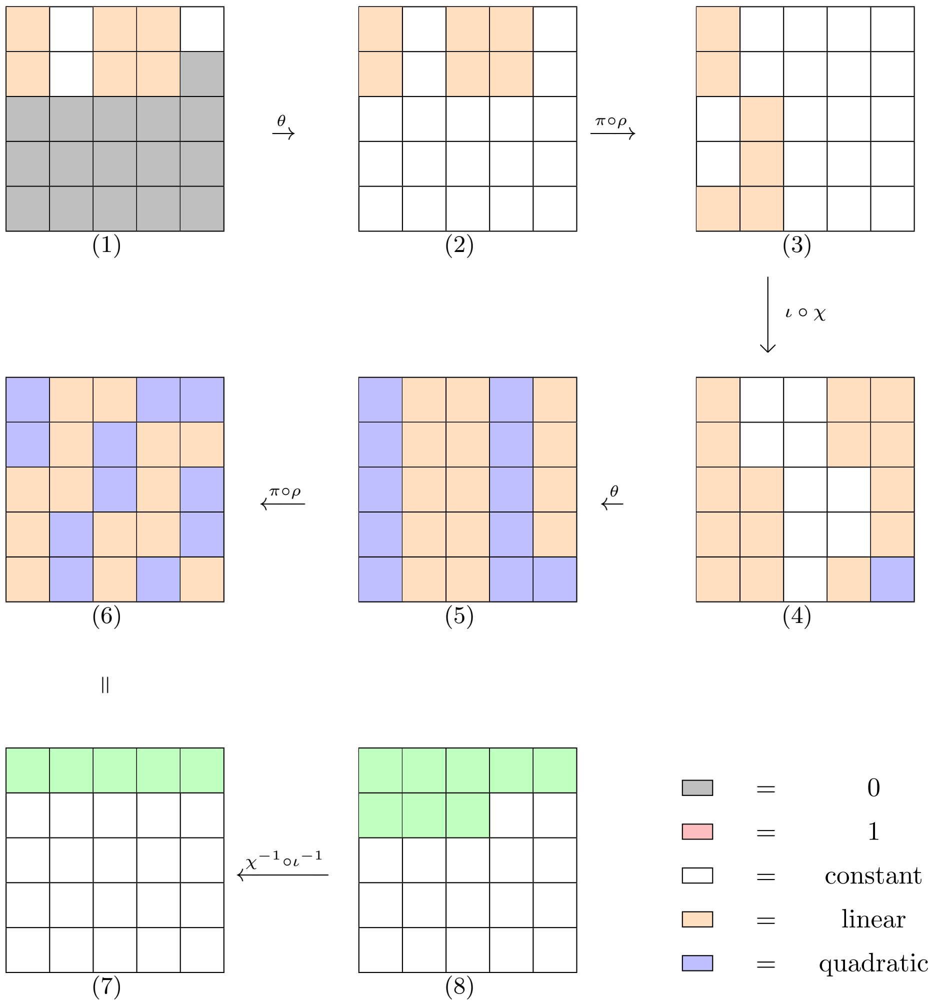

Fig. 3. Preimage attack on 2-round KECCAK-512

To avoid the propagation by θ in the first round, we use Observation 1, i.e., $\bigoplus_{j=0}^{4} A[i,j] = \alpha_i, \forall i \in [0,2,3]$ where α_i is some constant and hence include $3 \times 64 = 192$ linear constraints to the system. Also, since the hash length is 512, we can invert the first row of the hash value due to Observation 2.

Observe that after the application of the χ operation in the first round, state (4) contains a lane with quadratic terms. Due to the θ of the second round, these will get propagated only to the neighbouring columns. Hence, majority of the lanes in the state (5) contains only linear terms. But, while equating state (6) and state (7), we are only able to obtain $2 \times 64 = 128$ linear equations between the hash values and the variables. Observe that we have set up only 320 linear equations but have 384 variables.

Applying the techniques used in [1], we can linearize the quadratic term and use them to create more linear equations between hash value and the variables. Notice that in state (5), there is atmost one quadratic term in each polynomial. This is because the state before the application of θ in the second round has only one lane containing polynomials with only one quadratic term. More precisely, $A[4,4]$ of state (4) contains a polynomial of the form $p_1 + \overline{p_2}.p_3$ where p_i's are linear polynomials. This non-linear polynomial can be linearlized by adding one more linear equation to the system, say $p_3 = \beta$ where β is a constant. Therefore, if we linearize one quadratic term in state (4), we will be able to linearize 11 quadratic terms in state (5). But, only 3 out of the 11 linearized terms can be equated to the values in state (7). Therefore, we can set up an additional 64 linear equations of which $3\lfloor 64/4 \rfloor = 48$ equations are between message bits and hash values. But, we need to include one more linear equation for the last message bit to be 1 to satisfy the padding condition of KECCAK. Therefore, we have a system of linear equation in 384 variables and 384 equations. Since, we have $128 + 48 - 1 = 175$ linear equations between hash values and variables, we get a valid preimage with probability $1/2^{337}$.

To get a successful preimage attack, we must repeat the above procedure for at least 2^{337} times where the system of linear equations are different each time. Observe that there is enough degrees of freedom to perform this, i.e. 192 bits from $A[1,0], A[1,1]$ and $A[4,0]$ and 192 bits from α_i for $i \in [0,2,3]$ which sums up to 384 bits. Therefore, we have a preimage attack for 2-rounds KECCAK-512 with complexity of 2^{337}.

Preimage attack with complexity 2^{321}: By choosing a different set of lanes as variables, we achieve a better preimage attack. In figure 4, columns 1,2 and 4 are set as variables and the rest are set to constant. We also set $\bigoplus_{j=0}^{4} A[i,j] = \alpha_i, \forall i \in [0,1,3]$ where α_i is some constant, thus adding 192 linear equations to the system. Observe that in this case, we can set up $3 \times 64 - 1$ linear equations between the hash values and the variables. We must also include one more linear constraint for the last bit of message to be 1 to satisfy the padding condition for KECCAK. Therefore, we have a system of linear equation in 384 variables and 384 equations.

Since we are able to set up only 191 linear equations between the hash values and the variables, we get a valid preimage with probability $1/2^{321}$. Observe that there is enough degrees of freedom to repeat this procedure for 2^{321} due 192 bits from $A[2,0], A[2,1]$ and $A[4,0]$ and 192 bits from α_i for $i \in [0,1,3]$ which sums up to 384 bits. Therefore, we have a preimage attack for 2-rounds KECCAK-512 with complexity of 2^{321}.

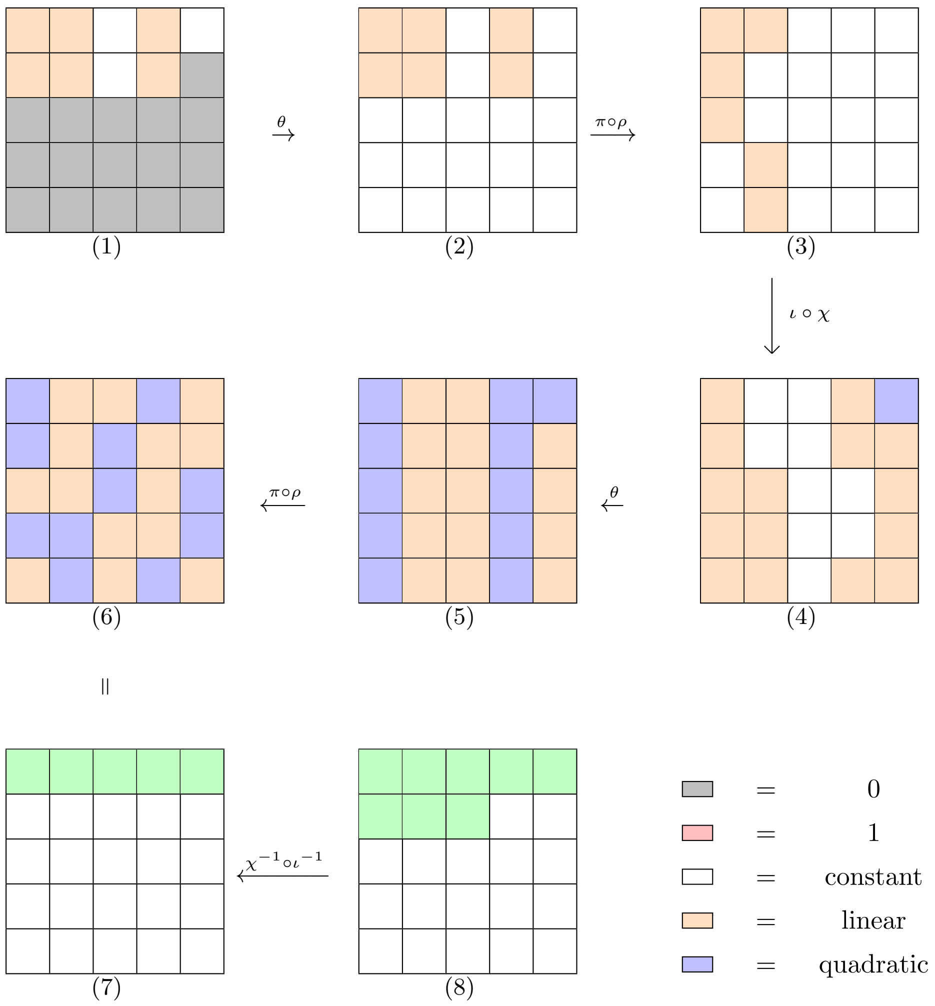

Fig. 4. Better preimage attack on 2-round KECCAK-512

3.2 Preimage attack on 2 rounds KECCAK-384

The preimage attack given by Guo. et al [1] for 2 rounds KECCAK-384 has a complexity of 2^{129} by constructing a linear structure with 6×64 variables. In our attack, we use 8×64 variables as shown in Figure 5. In-order to avoid propagation by θ in first round, we add the following 3×64 linear constrains into the system, $\bigoplus_{j=0}^{4} A[i,j] = \alpha_i, \forall i \in [0,2,3]$ where α_i is some constant.

By equating state (5) and state (6), we get $2 \times 64 = 128$ linear equations between variables and hash values. Observe that we have only set up 320 linear equations but have $8 \times 64 = 512$ variables. Applying the techniques used in subsection 3.1, we can set up an additional 3×64 linear equations of which

$3\lfloor(3 \times 64)/4)\rfloor = 144$ equations are between message bits and hash values. After satisfying the padding rule, we have a complexity gain over brute force of $2^{128+144-1} = 2^{171}$ and hence a preimage attack of complexity $2^{384-171} = 2^{113}$. Note that this result cannot be compared with the preimage attack given by Kumar et al. [16] because their attack has a space complexity of 2^{87}.

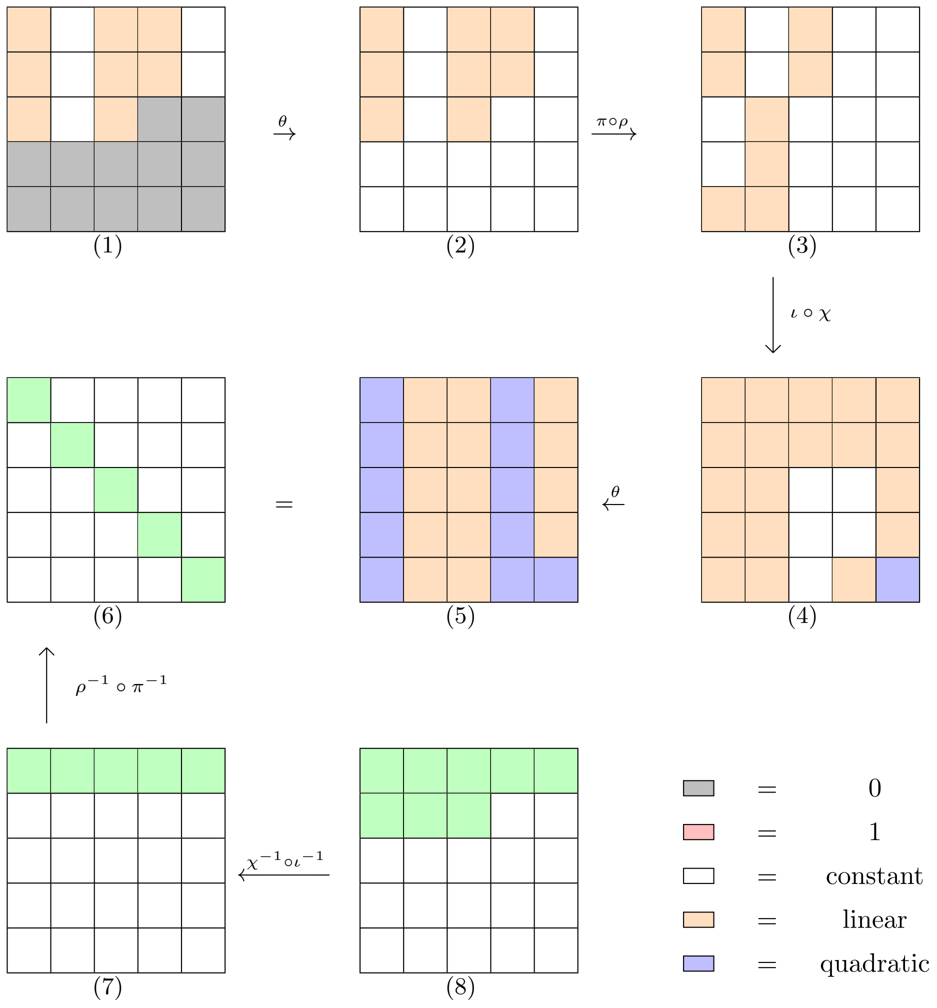

Fig. 5. Preimage attack on 2-round KECCAK-384

3.3 Preimage attack for higher rounds

In the previous subsections, we were able to get better preimage attack due to the fact the states are not filled with quadratic terms. If we were to find a similar attack for 3-rounds, we need to keep the following guidelines in mind.

1. The state after the application of second θ must be sparse of lanes with linear terms and comprised mostly of lanes with constant terms. This is because it would lead to a state with lesser quadratic terms after the application of χ of the second round.
2. Even if the propagation due to the θ in the third round cannot be restricted, the state before the application of the third θ must contain all its quadratic terms either in a single column or in two columns adjacent to each other. This would lead to a state with at least one column containing linear terms only after the application of θ.

3.4 Preimage attack on 3 rounds KECCAK-384

The following is our attack on KECCAK-384 for 3-rounds which uses two message blocks as shown in Figure 6. The first message block is chosen in such a way that after the application of 3 round KECCAK on this block, we get state such that $A[1,3] = A[3,3] = 0$ and $A[1,4] = A[4,4] = 1$ where A is state (2) as shown in the figure given above. The first message block can be found by randomly choosing $2^{4\times 64}$ message block and expecting one of them to give the required output. This works because the output of a hash function is random and therefore the complexity for brute force preimage attack is $1/2^l$ where l is the number of bits in the hash digest. The same technique has been used in [3] subsection 4.3.

The second message block contains $6 \times 64 = 384$ variables. We want to keep the columns 2, 4 and 5 unchanged after the application of first θ. For this, we first set $\bigoplus_{j=0}^{4} A[i,j] = \alpha_i$, for $i \in [0,2]$ and then set up equation between column 1 and column 3 so that column 2 does not get affected after the application of first θ. This means that the α_i's are dependent. Similarly, c_2 and c_3 can be set according to α_i's such that column 4 and 5 do not get affected after the first θ. Therefore, we have 2×64 linear equations in our system. c_1 can be fixed to some randomly chosen value.

To avoid propagation after second θ, we set up 3×64 linear equations to make the column parties equal to some constant β_i. Observe that after the application of the second χ, there are two lanes with quadratic terms in state (8). But after the application of the third θ, the fourth column will contain only linear terms. By equating state (9) and state (10), we can set up 63 linear equations between message bits and hash values. Also, we have one more equation to keep the last message bit equal to 1. Therefore, we have a preimage attack with a time of complexity $2^{384-63} = 2^{321}$ because computing the first message block has a complexity of 2^{256}.

Note that there are enough degrees of freedom due to the 256 bits from α_i's and the β_i's, 64 bits from c_1 and enough bits from the first message block.

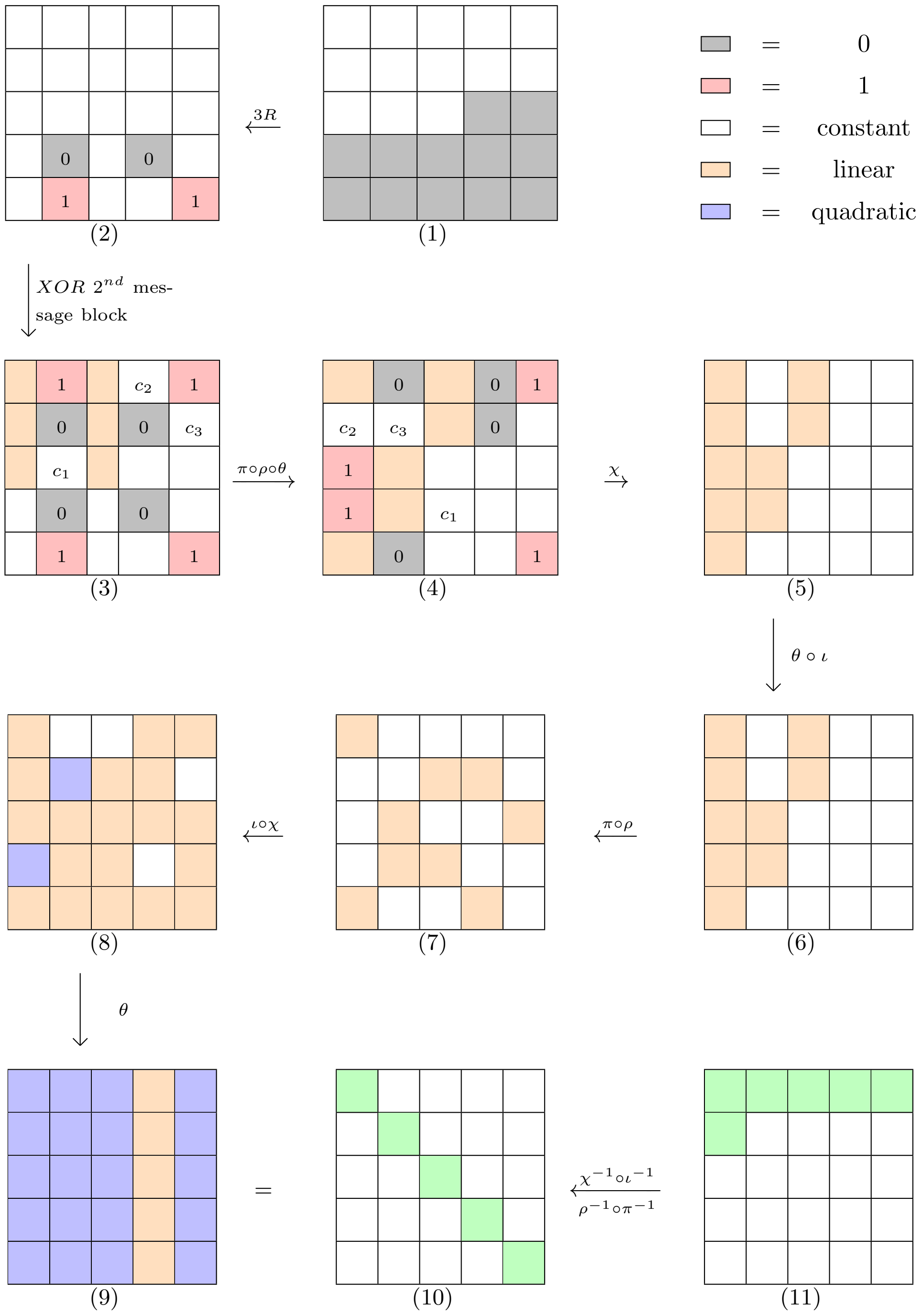

Fig. 6. Preimage attack on 3-round KECCAK-384

3.5 Preimage attack on 3 rounds KEECAK-512

We use two message block and $4 \times 64 = 256$ variables for this attack as shown in Figure 7. The first message block is used so that we get enough degree of freedom to launch a preimage attack. Observe that after the application of θ in first round, we require certain lane to be 1/0 in state (4). To achieve this, we first set $A[1,0] \oplus A[1,1] = \alpha_1$ where α_1 is some constant. Then, we set up 64 linear equations of the form $\bigoplus_{i=0}^{4} (A[1,i] \oplus A[3,i](1) = e_2 + 1)$. Observe that due to this constrain, after the application of first θ, we will get $A[2,0] = A[2,4] = 1$ and $A[2,1] = 0$ where A is state (4). Similarly, by fixing x_6 and x_2 appropriately, we can get the required state (4).

To avoid propagation due to the θ in second round, we add only 64 linear equations to the system to make the parity of the first columns in state (6) as a constant. Observe that after the application of θ of the third round, the lanes in the first two columns will contain only one quadratic term. So, if we linearize one quadratic term in $A[2,4]$ of state (9), then we have linearized five polynomials in column 2 of state (10). Similarly, if we linearize one quadratic term in $A[4,2]$ of state (9), then we have linearized five polynomial in column 1 of state (10).

But, out of these 6 linearized polynomials, only one can be used to create a linear equation between message bits and hash value by equating state (10) and state (11). Therefore, we have $\lfloor 64/2 \rfloor = 32$ linear equations between message bits and hash value and hence obtained a preimage of complexity $2^{512-32+1} = 2^{481}$. Due to the first message block, we have enough degree of freedom.

Improved analysis Observe that if we carefully linearize one quadratic term from $A[2,4]$ and one from $A[4,2]$ of state (9), we also linearize one more polynomial in column 4 of state (10), i.e. we have also linearized a polynomial in the lane $A[3,3]$. Therefore, now we have $3\lfloor \frac{64}{5} \rfloor + 2 = 3 \times 12 + 2 = 38$. Therefore, we have an improved preimage attack of complexity $2^{512-38+1} = 2^{475}$.

3.6 Preimage attack on 4 rounds KECCAK-384

This attack requires two message blocks and $6 \times 64 = 384$ variables as shown in Figure 8. As done in subsection 3.4, the first message block is found by trying randomly many message blocks so that after the application of 4-rounds and XORing the second message block, we get state (2). Observe that in state (2), there are two lanes with entries c and $\overline{c}$. We also require state (2) to satisfy one more equation.

$$d(-1) + b(-2) + (g(-1) + (\overline{c} + a + b)(-2))(-2) + (a + b)(1) = k \qquad (1)$$

Therefore, we would require a complexity of 2^{128} to find the appropriate first message block. We will use the following strategy to obtain state (3). We include $A[0,0] = A[0,2]$ to the system of linear equations, fix $x_1 = 0$ and randomly assign value to x_7 whereas we fix $x_2 = \overline{c}, x_3 = d, x_5 = g$. Since we require state (3) after the application of θ, we have the following equations.

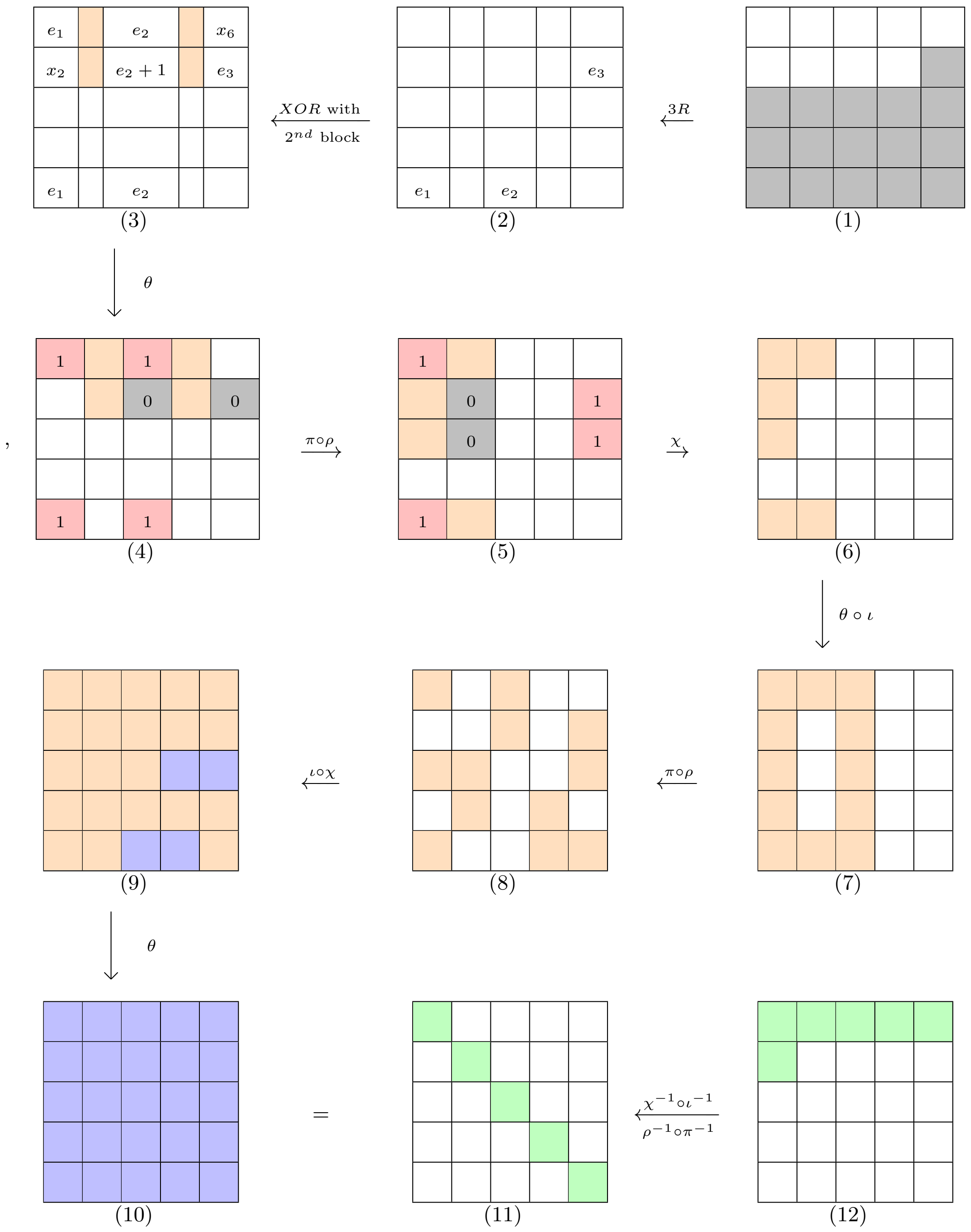

Fig. 7. Preimage attack on 3-round KECCAK-512

$$(a+b)+(A[2,0]+A[2,2]+e)(1)=\bar{c} \tag{2}$$

$$(A[2,0]+A[2,2]+e)+(x_6+x_7+i+j+k)(1)=g \tag{3}$$

$$(x_6+x_7+i+j+k)+(A[1,0]+A[1,2]+c)(1)=b \tag{4}$$

$$(A[1,0]+A[1,2]+c)+(x_4+f+h)(1)=d \tag{5}$$

$$(x_4+f+h)+(a+b)(1)=k \tag{6}$$

Therefore, we add equation (7) and (9) to the system of equations and fix x_6 and x_4 according to equation (8) and (10). Observe that due to the following equations, all equations from (2)-(6) are satisfied, particularly, equation (6) is satisfied due to equation (1).

$$A[2,0]+A[2,2]=(\bar{c}+a+b)(-1) \tag{7}$$

$$x_6=g(-1)+(\bar{c}+a+b)(-2)+x_7+i+j+k \tag{8}$$

$$A[1,0]+A[1,2]=b(-1)+(g(-1)+(\bar{c}+a+b)(-2))(-1)+c \tag{9}$$

$$\begin{aligned} x_4 &= d(-1)+f+h+(b+x_6+x_7+i+j+k)(-2) \\ &= d(-1)+f+h+b(-2)+(g(-1)+(\bar{c}+a+b)(-2))(-2) \end{aligned} \tag{10}$$

Also, we include 2×64 linear equations for restricting the propagation due to θ in the second round. Observe that each polynomial in the state (9) has 11 quadratic terms. In [1] subsection 6.3, Guo et al. gave a technique that carefully linearizes the quadratic terms such that if the number of free variables is t, we can construct $2\lfloor(t-5)/8\rfloor$ linear equations between hash values and the variables. For more details, refer [1]. In our case, we have $t=64$ and therefore, we have 14 linear equations between hash value. Observe that we have enough degree of freedom due to x_7, the parity of the two columns of the second θ and and rest from the first message block. Therefore, the complexity of our attack is 2^{371}.

4 Conclusion

In this paper, we give the best theoretical preimage attacks on 2,3 rounds KECCAK-512 and 2,3,4 rounds KECCAK 384 by studying non-linear structures carefully. It would be interesting to see whether non-linear structures along with other techniques can be used to find better preimage attacks for higher rounds.

Acknowledgement: We would like to thank Rajendra Kumar for valuable discussions.

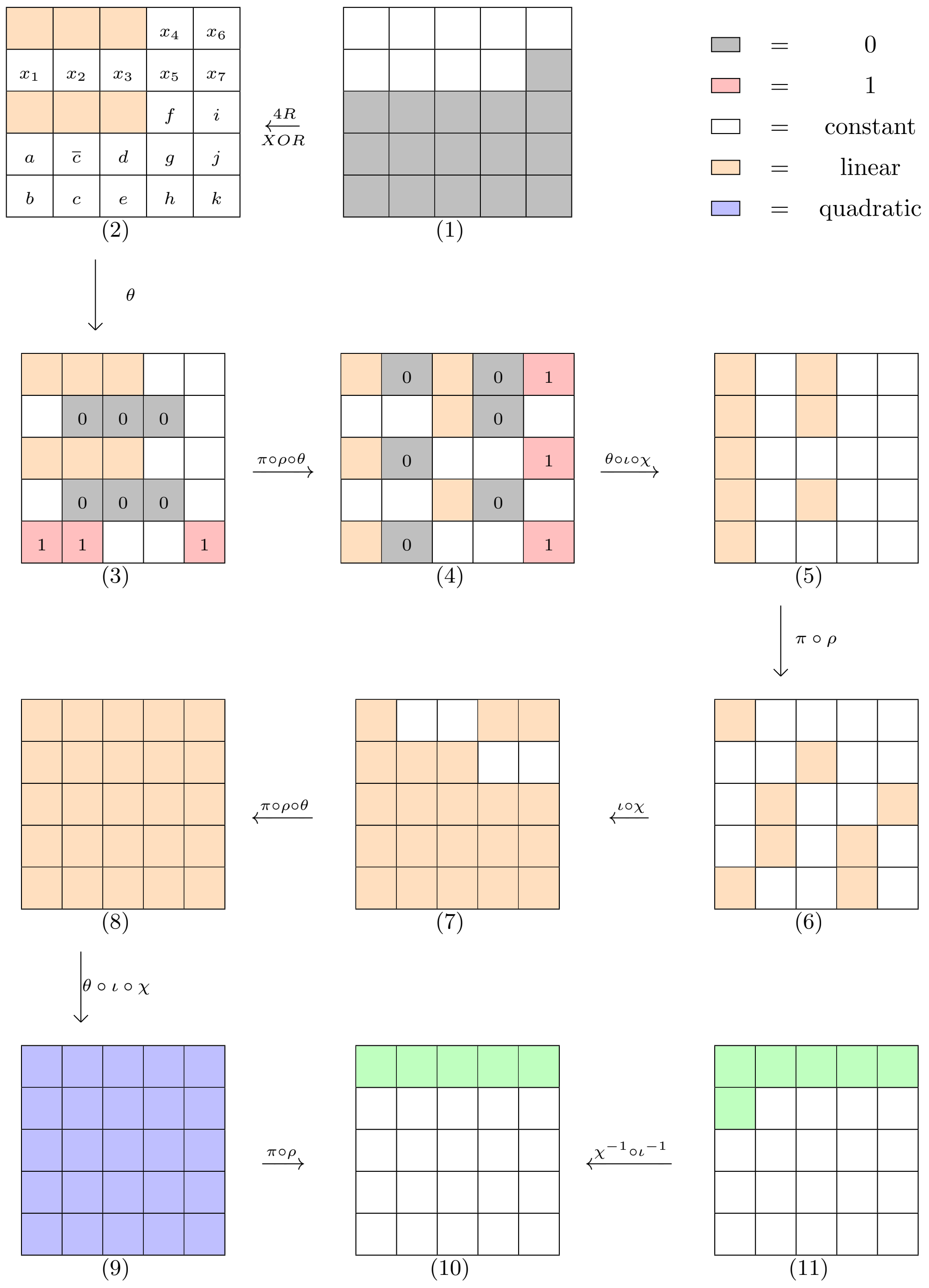
x_4 x_6
x_1 x_2 x_3 x_5 x_7
f i
a $\bar{c}$ d g j
b c e h k
(2)
$4R$
XOR
(1)
= 0
= 1
= constant
= linear
= quadratic
θ
0 0 0
0 0 0
1 1 1
(3)
$\pi \circ \rho \circ \theta$
0 0 1
0
0 1
0
0 1
(4)
$\theta \circ \iota \circ \chi$
(5)
$\pi \circ \rho$
(6)
$\iota \circ \chi$
(7)
$\pi \circ \rho \circ \theta$
(8)
$\theta \circ \iota \circ \chi$
(9)
$\pi \circ \rho$
(10)
$\chi^{-1} \circ \iota^{-1}$
(11)

Fig. 8. Preimage attack on 4-round KECCAK-384

References

1. Jian Guo, Meicheng Liu, and Ling Song. Linear structures: applications to cryptanalysis of round-reduced keccak. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 249–274. Springer, 2016.
2. Ting Li, Yao Sun, Maodong Liao, and Dingkang Wang. Preimage attacks on the round-reduced keccak with cross-linear structures. *IACR Transactions on Symmetric Cryptology*, pages 39–57, 2017.
3. Ting Li and Yao Sun. Preimage attacks on round-reduced keccak-224/256 via an allocating approach. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 556–584. Springer, 2019.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak specifications. *Submission to nist (round 2)*, pages 320–337, 2009.
5. Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.
6. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponges. *online] http://sponge. noekeon. org*, 2011.
7. Daniel J Bernstein. Second preimages for 6 (7?(8??)) rounds of keccak. *NIST mailing list*, 2010.
8. María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical analysis of reduced-round keccak. In *INDOCRYPT*, volume 7107, pages 236–254. Springer, 2011.
9. Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on keccak-224 and keccak-256. In *FSE*, volume 12, pages 442–461. Springer, 2012.
10. Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of sha-3 using generalized internal differentials. In *International Workshop on Fast Software Encryption*, pages 219–240. Springer, 2013.
11. Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *Journal of cryptology*, 27(2):183–209, 2014.
12. Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced keccak. In *International Workshop on Fast Software Encryption*, pages 241–262. Springer, 2013.
13. Paweł Morawiecki and Marian Srebrny. A sat-based preimage analysis of reduced keccak hash functions. *Information Processing Letters*, 113(10-11):392–397, 2013.
14. Donghoon Chang, Arnab Kumar, Pawell Morawiecki, and Somitra Kumar Sanadhya. 1st and 2nd preimage attacks on 7, 8 and 9 rounds of keccak-224,256,384,512. In *SHA-3 workshop (August 2014)*, 2014.
15. Rajendra Kumar, Mahesh Sreekumar Rajasree, and Hoda AlKhzaimi. Cryptanalysis of 1-round keccak. In *International Conference on Cryptology in Africa*, pages 124–137. Springer, 2018.
16. Rajendra Kumar, Nikhil Mittal, and Shashank Singh. Cryptanalysis of 2 round keccak-384. In *International Conference on Cryptology in India*, pages 120–133. Springer, 2018.
17. G Bertoni, J Daemen, M Peeters, and GV Assche. The keccak reference. online at http://keccak. noekeon. org/keccak-reference-3.0. pdf, 2011.