Cryptanalysis of 1-Round KECCAK

Rajendra Kumar[1] ,Mahesh Sreekumar Rajasree[1] and Hoda AlKhzaimi[2]

[1] Center for Cybersecurity, Indian Institute of Technology Kanpur, India
`rjndr@iitk.ac.in` , `mahesr@iitk.ac.in`
[2] Center for Cyber Security, New York University, Abu Dhabi
`hoda.alkhzaimi@nyu.edu`

Abstract. In this paper, we give the first pre-image attack against 1-round KECCAK-512 hash function, which works for all variants of 1-round KECCAK. The attack gives a preimage of length less than 1024 bits by solving a system of 384 linear equations. We also give a collision attack against 1-round KECCAK using similar analysis.

Keywords: Cryptanalysis, KECCAK, SHA-3, Preimage, Collision

1 Introduction

Hash functions are used in digital signatures, message integrity and authentication. In 2006, NIST announced the "NIST hash function competition" which received 64 proposals from around the world. In October 2012, KECCAK designed by Guido Bertoni, Joan Daemen, Michal Peeters, and Gilles Van Assche [3], was selected as the winner of the competition and in 2015, it was standardized as a "Secure Hash Algorithm 3" [9].

The KECCAK hash family is based on the sponge construction[4]. Sponge construction has the property to generate an output of any length and because of this property, SHA3 standards include two extendable output functions which are SHAKE128 and SHAKE256. These can also be used as a pseudo-random generator. Due to its vast applications, a lot of security analysis is being performed on the KECCAK hash family.

In 2010, D. J. Bernstein [1] gave an idea for second preimage of KECCAK variants and in 2014, Chang et al. [5] gave a 1st and 2nd preimage attack. Both have an improvement in time complexity over the brute-force. Morawiecki et al. [11] gave a theoretical preimage attack up to 4 rounds of KECCAK by using a technique called as rotational cryptanalysis. Morawiecki et al. [12] performed a preimage analysis of round reduced KECCAK by using toolkit CryptLogVer and SAT solver PrecoSAT. Mara Naya-Plasencia et al. [13] gave a preimage attack on 2-round for KECCAK-224 and KECCAK-256 by using the meet in middle approach. Dinur et al. [6] [8] gave a collision attack upto 4 rounds using differential and algebraic techniques, and later improved upto 5 rounds using generalized internal differential [7]. In 2016 Guo et al. [10] gave preimage attack for 2 round for KECCAK-224, 256, 384 and 512. The complexity of attack [10] for KECCAK-384 is 2^{129} and for KECCAK-512 is 2^{384}. They extended this upto 4 round for small hash length. Apart from above-mentioned attacks, there are several other attacks against KECCAK.

Our Contribution: In this paper, we give the first preimage attack against 1 round KECCAK-512. The only computation required in this attack is solving 384 linear equations. It is based on exploiting the degree of freedom in the equations between hash values and message bits, and convert these equations to simple assignments of values to message variables. Using this method, we can find a message of length less than 1024 bits corresponding to every hash value. Also, the time complexity of this attack is constant.

Organization: The rest of the paper contains the following sections. In Section 2, we briefly describe the structure of KECCAK. In Section 3, we show the cryptanalysis of 1 round KECCAK-512. Section 4 contains the implementation results and describes how to extend the preimage attack to a collision attack. Section 5 contains conclusion and future works.

2 Structure of KECCAK

KECCAK hash function has 3 parameters: r is the bitrate, c is the capacity and n is the output length. It is based on sponge construction[4] which uses a padding function *pad*, a bitrate parameter r and a permutation function f as shown in figure 1.

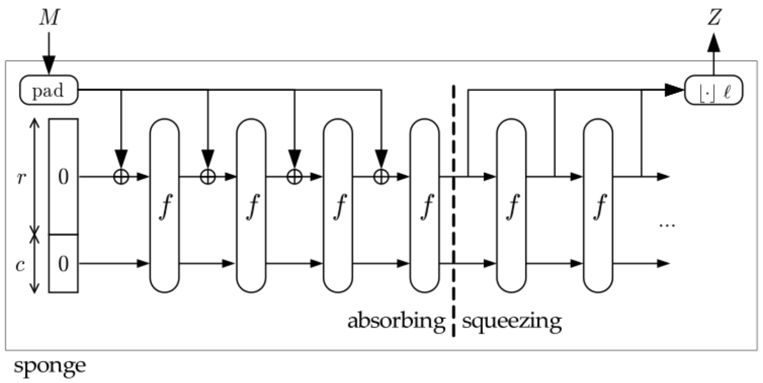


Fig. 1. Sponge function [4]

2.1 Sponge Construction

The sponge construction begins by applying the padding function *pad* on the input string M which produces M' whose length is a multiple of r. M' under goes the absorbing phase as follows.

1. M' is split into blocks of r bits namely $m_1, m_2, ...m_k$.
2. There is an initial string(o_0) which is a b bit string initialized to zero.
3. The initial r bits of o_0 is XORed with first block m_1 and is given as input to f. The output produced by f is denoted by o_1.
4. Similarly, the initial r bits of o_i is XORed with the m_{i+1} and given to f.
5. Finally, the output of the absorbing phase is o_k.

The squeezing phase consists of obtaining the output which can be of any length. Let n be the required output length such that $n = pr + q$ where $q < r$.

1. Apply the f function p more times such that $o_{k+i} = f(o_{k+i-1})$.
2. Let O be the concatenation of the first r bits of each o_{k+i} where $0 \leq i \leq p$.
3. The output of the sponge construction is the first n bits of O.

In case of KECCAK hash function, f is a $KECCAK - f[1600]$ permutation and the *pad* function appends 10^*1 to input M. $KECCAK - f$ is a specialization of KECCAK-p permutation.

$$KECCAK - f[b] = KECCAK - p[b, 12 + 2l]$$

where $l = log_2(b/25)$.

2.2 KECCAK-p permutation

KECCAK-p permutation is denoted by KECCAK-p$[b, n_r]$, where b is length of input string which is called the width of the permutation, n_r is number of rounds of internal transformation where $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ and n_r being any positive integer. We can define two more quantities $w = b/25$ and $l = log_2(b/25)$. For KECCAK-512, the number of rounds of internal transformation n_r is 24 and $b = 1600$. 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 $|$ is the concatenation function.

The internal transformation consists of 5 step mappings θ,ρ,π,χ and ι which acts on a state. We give a brief description of each of these step mappings with A and A' being the state before and after applying a step mappings.

1. θ:

$$\begin{aligned} A'[x][y][z] = A[x][y][z] &\oplus CP[(x+1) \ mod \ 5][(z-1) \ mod \ 64] \\ &\oplus CP[(x-1) \ mod \ 5][z] \end{aligned}$$

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

$$CP[x][z] = A[x][0][z] \oplus A[x][1][z] \oplus A[x][2][z] \oplus A[x][3][z] \oplus A[x][4][z]$$

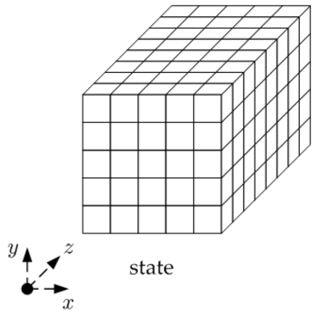


Fig. 2. KECCAK state [2]

2. ρ:

$$A'[x][y] = A[x][y] << r[x][y]$$

where $<<$ means bitwise rotation towards MSB of the 64-bit word. The values of $r[x][y]$ are given in the table below.

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:

$$A'[y][2x+3y] = A[x][y]$$

π interchanges the lanes of the state A.

4. χ:

$$A'[x][y][z] = A[x][y][z] \oplus ((A[(x+1)\ mod\ 5][y][z] \oplus 1) \\ .A[(x+2)\ mod\ 5][y][z])$$

χ is the only non-linear operation among the 5 step mappings.

5. ι:

$$A'[0][0] = A[0][0] \oplus RC_i$$

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

3 Cryptanalysis Of One-Round KECCAK

In this section, we will cover the analysis of 1 round KECCAK for the different preimage message size.

3.1 Preliminaries and Notations

In our analysis, we are going to use following observations of the χ and θ operation [10]. Considering χ as a row 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: In χ operation, if all output bits b_0, b_1, b_2, b_3, b_4 are known, then we can exactly determine the input bits a_0, a_1, a_2, a_3, a_4 using

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

Observation 2: In χ operation, if any 3 non-consecutive input bits are known, then all the output bits can be written as linear combinations of input bits.
Observation 3: In χ operation, given b_0, b_1, b_2 and $a_3 = 1$, we can exactly determine the values of a_0, a_1 and a_2.

$$a_2 = b_2$$

$$a_1 = b_1 \oplus (b_2 \oplus 1)$$

$$a_0 = b_0 \oplus (b_1 \oplus b_2).b_2$$

Observation 4: 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 $i \in \{0, 1, 2, 3, 4\}$

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

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 $\oplus$. In all the equations and figures, the value inside the brackets '$()'$ indicates the offset by which the lane is shifted. Every operation between two lanes are bitwise.

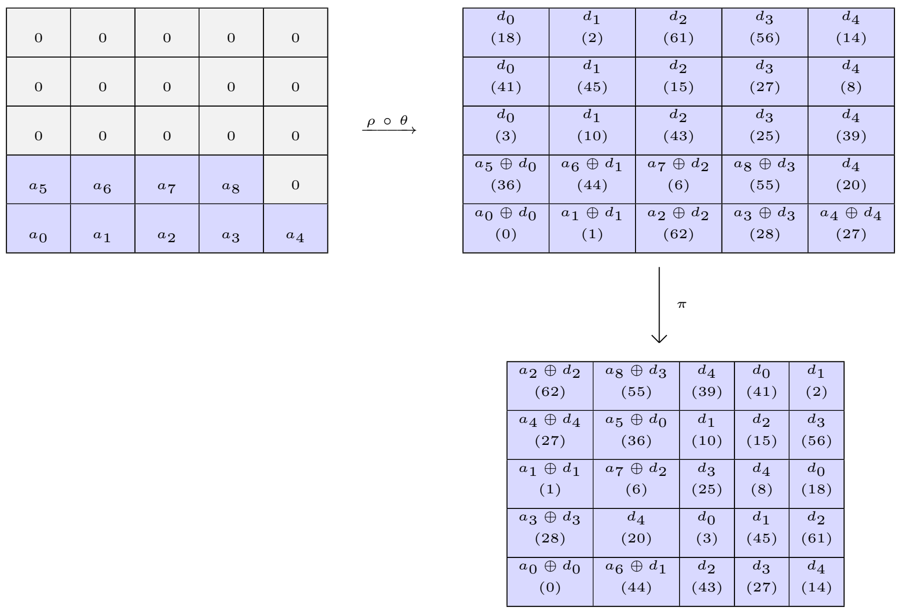
3.2 General description of the attack

We now give the generic description of the attack.

1. The given hash value uses the first 8 lanes of a state and we ignore the values in the rest of the 17 lanes.
2. Invert the ι operation by XORing the $(0,0)$ lane with the Round constant RC.
3. Invert the χ operation by using the above given observations. Let's call this state as I.
4. Apply the necessary operations on the message block to reach state I. For making the operations linear, use the above observations.
5. Check for the dependencies among the linear equations. If they are independent, then solve the system of linear equations.

3.3 Analysis of Preimage attack by using 1 message block

In this section, we are going to show that by using only 1 message block it is not possible to find the preimage for all the hash values of KECCAK-512. In the subsection 3.4, we will also characterize the hash values whose preimage can be found by using 1 message block. In KECCAK-512, we have $n = 512$, $c = 1024$ and $r = 576$. So, the hash value occupies 8 lanes and message block is of 9 lanes. Let $A = a_0a_1a_2a_3a_4a_5a_6a_7a_8$ be the message block of 576 bit length where each a_i is an array of 64 bits. Figure 3 shows the state after applying θ,ρ and π on A where $d_i[k] = CP[i-1][k] + CP[i+1][k-1]$ and CP represents the column parity.



0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
a_5 a_6 a_7 a_8 0
a_0 a_1 a_2 a_3 a_4
$\rho \circ \theta$
d_0 (18) d_1 (2) d_2 (61) d_3 (56) d_4 (14)
d_0 (41) d_1 (45) d_2 (15) d_3 (27) d_4 (8)
d_0 (3) d_1 (10) d_2 (43) d_3 (25) d_4 (39)
$a_5 \oplus d_0$ (36) $a_6 \oplus d_1$ (44) $a_7 \oplus d_2$ (6) $a_8 \oplus d_3$ (55) d_4 (20)
$a_0 \oplus d_0$ (0) $a_1 \oplus d_1$ (1) $a_2 \oplus d_2$ (62) $a_3 \oplus d_3$ (28) $a_4 \oplus d_4$ (27)
π
$a_2 \oplus d_2$ (62) $a_8 \oplus d_3$ (55) d_4 (39) d_0 (41) d_1 (2)
$a_4 \oplus d_4$ (27) $a_5 \oplus d_0$ (36) d_1 (10) d_2 (15) d_3 (56)
$a_1 \oplus d_1$ (1) $a_7 \oplus d_2$ (6) d_3 (25) d_4 (8) d_0 (18)
$a_3 \oplus d_3$ (28) d_4 (20) d_0 (3) d_1 (45) d_2 (61)
$a_0 \oplus d_0$ (0) $a_6 \oplus d_1$ (44) d_2 (43) d_3 (27) d_4 (14)

Fig. 3. State after applying θ, ρ, π

Suppose $H = h_0h_1h_2h_3h_4h_5h_6h_7$ is the 512-bit hash value where each h_i is of 64 bits. We know that we can invert the last row of H and obtain the exact values of $h_0^{'}, h_1^{'}, h_2^{'}, h_3^{'}$ and $h_4^{'}$(shown in figure 4) using the formula given in 3.1. The same cannot be done for the second last row.

By equating the 3rd state of figure 3 and 2nd state of figure 4, we get the exact values of d_2, d_3, d_4 and two linear equations

$$a_0 + d_0 = h_0^{'}$$

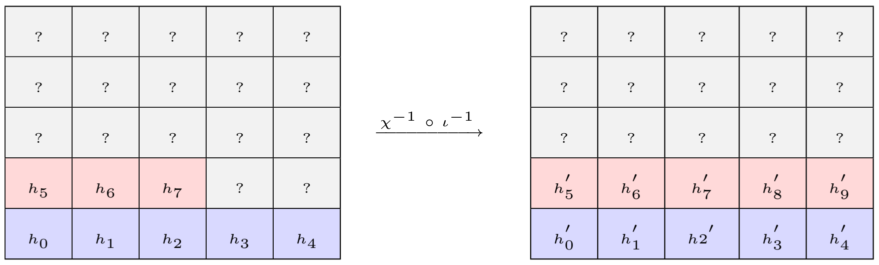

?
h_5 h_6 h_7
h_0 h_1 h_2 h_3 h_4
$\chi^{-1} \circ \iota^{-1}$
h_5' h_6' h_7' h_8' h_9'
h_0' h_1' $h2'$ h_3' h_4'

Fig. 4. Inverse operation on hash values

$$a_6 + d_1 = h_1^{'}$$

By applying χ on the 3rd state of figure 3, we have the following equations

$$a_3(28) + d_3(28) + (d_4(20) + 1)d_0(3) = h_5$$

$$d_4(20) + (d_0(3) + 1)d_1(45) = h_6$$

$$d_0(3) + (d_1(45) + 1)d_2(61) = h_7$$

The 2nd equation is quadratic while the other two are linear. It can be easily seen that for many cases, there isn't a solution to this system of equations. For example, take the case where $d_4 = h_4^{'} = 0$ and $d_2 = h_2^{'} = 0$. Then,

$$(d_0(3) + 1)d_1(45) = h_6$$

$$d_0(3) = h_7$$

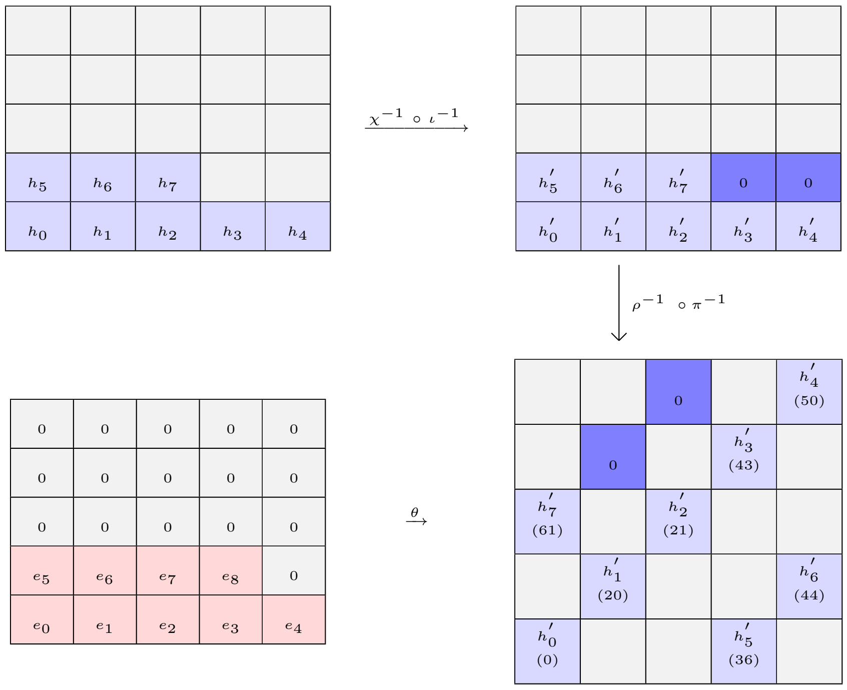
The above equations cannot be solved simultaneously if $h_7 = 1$ and $h_6 = 1$. So, by using only 1 message block we can't get all possible hash values of KECCAK-512

3.4 Preimage Attack

In this section, we give the preimage attack to one round KECCAK-512. This preimage attack gives a message of length less than 1024 bits, i.e two message blocks. In each of the following subsection, we describe a preimage attack for one round KECCAK-512 by considering different settings for the attack and give an analysis on it. We will be considering $h_0, ..., h_7$ as the hash value where each h_i is of length 64 bits.

Using one message block and keeping θ as identity: We first invert the hash values by applying $\chi^{-1} \circ \iota^{-1}$. To do this, we make the lanes at $(3, 1)$ and $(4, 1)$ of the inverted state as 0(refer figure 5). We further invert this state through π and ρ. $e_0, ..., e_8$ is the message block represented in 9 lanes. Since we are keeping θ as identity, we have only four free lane variables. Also, we must keep the last bit of the message block as 1 inorder to satisfy the padding rule, so the last bit of e_8(which is equal to e_3) should be 1. Therefore, we can assign

$$e_0 = h'_0(0), e_3 = h'_5(36), e_6 = h'_1(20)$$



h_5 h_6 h_7
h_0 h_1 h_2 h_3 h_4
$\chi^{-1} \circ \iota^{-1}$
h'_5 h'_6 h'_7 0 0
h'_0 h'_1 h'_2 h'_3 h'_4
$\rho^{-1} \circ \pi^{-1}$
0 h'_4 (50)
0 h'_3 (43)
h'_7 (61) h'_2 (21)
h'_1 (20) h'_6 (44)
h'_0 (0) h'_5 (36)
θ
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
e_5 e_6 e_7 e_8 0
e_0 e_1 e_2 e_3 e_4

Fig. 5. Using one message block and keeping θ as identity

Therefore, we can successfully find the preimage for hash values of the form $h_2 = h_6 = h_7 = 0$ and both h_0 and h_1 can take any arbitary values whereas h_5 can have arbitary values except for a single bit which must be 1 because $h_5 = h'_5 = e_3(28)$. Also, h_4 must be equal to $\overline{h_0}h_1$ and $h_3 = h_0$. Hence for 2^{191} hash values we can find preimage by using 1 message block and keeping θ as identity.

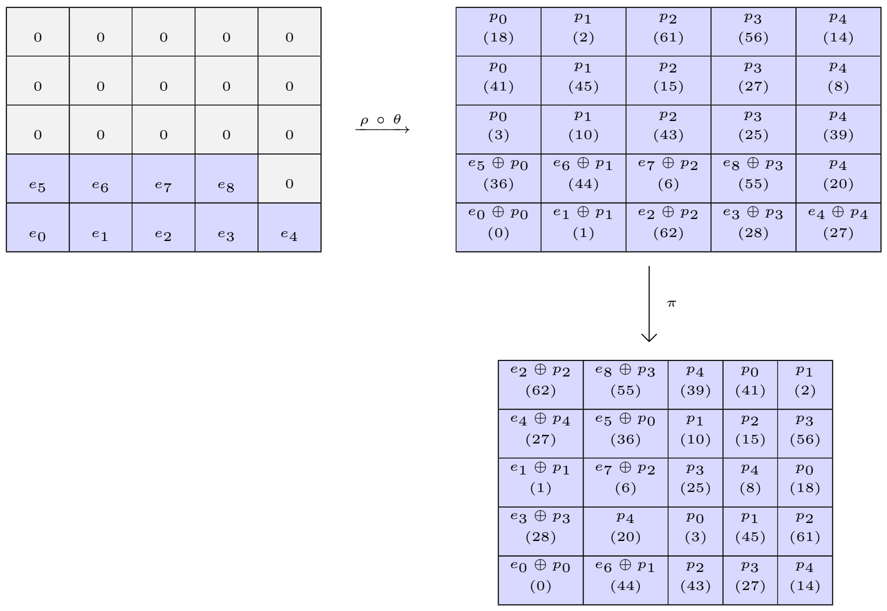
Using one message block without making θ as identity: We first invert the hash values by applying $\chi^{-1} \circ \iota^{-1}$, but this time keeping the lane at $(3,1)$ as 1. So, now we have $h_7 = h'_7$. In figure 6, p_i denotes the value added to an element in i^{th} column by the θ function. By comparing the 3^{rd} state in figure 6 and the state obtained by inverting the hash values, we use the following assignments.

$$e_0(0) = h'_0(0) + h'_7(61) \text{ because } p_0(3) = h'_7(0)$$

$$e_6(44) = h'_1(0) + 1 \text{ because } p_1(45) = 1$$

$$e_3(28) = h'_5(0) + h'_3(1) \text{ because } p_3(27) = h'_3(0)$$

Lane (4,0) and (1,1) are rotations of each other. From 320 linear equations in 320 variables, there are only 319 linearly independent equations. Therefore, for at-most 2^{383} hash values, we can find the preimage by solving these equations.



0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
e_5	e_6	e_7	e_8	0
e_0	e_1	e_2	e_3	e_4

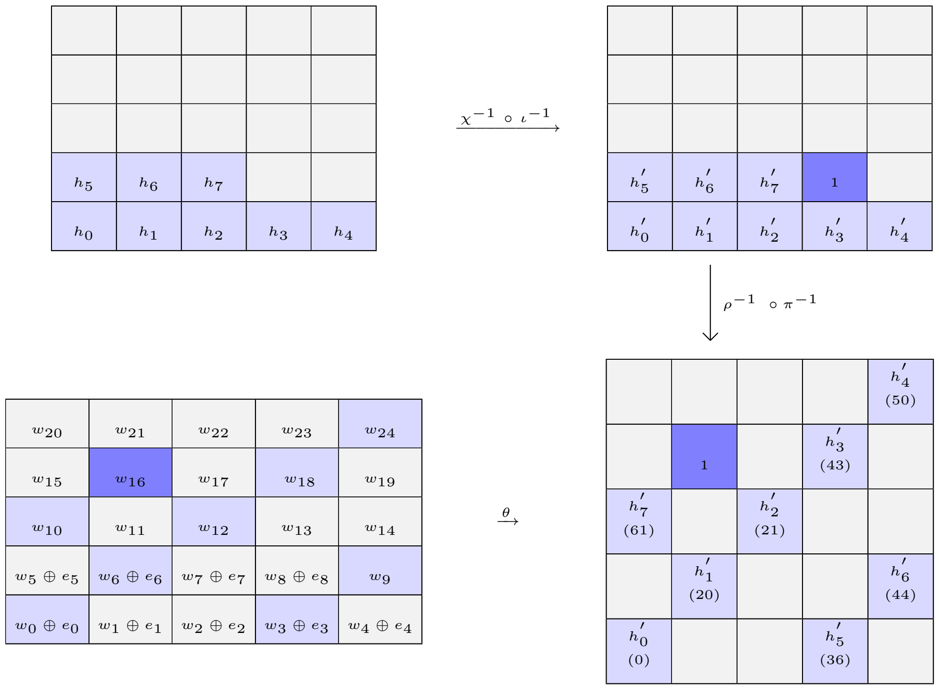
p_0 (18)	p_1 (2)	p_2 (61)	p_3 (56)	p_4 (14)
p_0 (41)	p_1 (45)	p_2 (15)	p_3 (27)	p_4 (8)
p_0 (3)	p_1 (10)	p_2 (43)	p_3 (25)	p_4 (39)
$e_5 \oplus p_0$ (36)	$e_6 \oplus p_1$ (44)	$e_7 \oplus p_2$ (6)	$e_8 \oplus p_3$ (55)	p_4 (20)
$e_0 \oplus p_0$ (0)	$e_1 \oplus p_1$ (1)	$e_2 \oplus p_2$ (62)	$e_3 \oplus p_3$ (28)	$e_4 \oplus p_4$ (27)

$e_2 \oplus p_2$ (62)	$e_8 \oplus p_3$ (55)	p_4 (39)	p_0 (41)	p_1 (2)
$e_4 \oplus p_4$ (27)	$e_5 \oplus p_0$ (36)	p_1 (10)	p_2 (15)	p_3 (56)
$e_1 \oplus p_1$ (1)	$e_7 \oplus p_2$ (6)	p_3 (25)	p_4 (8)	p_0 (18)
$e_3 \oplus p_3$ (28)	p_4 (20)	p_0 (3)	p_1 (45)	p_2 (61)
$e_0 \oplus p_0$ (0)	$e_6 \oplus p_1$ (44)	p_2 (43)	p_3 (27)	p_4 (14)

Fig. 6. Using one message block without making θ as identity

Therefore, hash values with $h_6 = h_7(0) \oplus h_4(6) \oplus \overline{h_0(6)}.(h_1(6) \oplus \overline{h_2(6)}.h_3(6)) \oplus 1$, we can find the preimage using 1 message block.

Using two message blocks and making both θ operation as identity: By using the same idea used above, we are going to invert the hash values by applying $\rho^{-1} \circ \pi^{-1} \circ \chi^{-1} \circ \iota^{-1}$.We are going to make θ operation as identity by using the 5 lane variables of message block E. So, from second message block we are left with only 4 free lane variables.



h_5 h_6 h_7
h_0 h_1 h_2 h_3 h_4
$\chi^{-1} \circ \iota^{-1}$
h'_5 h'_6 h'_7 1
h'_0 h'_1 h'_2 h'_3 h'_4
$\rho^{-1} \circ \pi^{-1}$
h'_4 (50)
1
h'_3 (43)
h'_7 (61)
h'_2 (21)
h'_1 (20)
h'_6 (44)
h'_0 (0)
h'_5 (36)
θ
w_{20} w_{21} w_{22} w_{23} w_{24}
w_{15} w_{16} w_{17} w_{18} w_{19}
w_{10} w_{11} w_{12} w_{13} w_{14}
$w_5 \oplus e_5$ $w_6 \oplus e_6$ $w_7 \oplus e_7$ $w_8 \oplus e_8$ w_9
$w_0 \oplus e_0$ $w_1 \oplus e_1$ $w_2 \oplus e_2$ $w_3 \oplus e_3$ $w_4 \oplus e_4$

Fig. 7. Using two message blocks and making both θ operation as identity

For the first message block we are using the message block

$$D = d_0, d_1, d_2, d_3, 0, d_0, d_1, d_2, d_3$$

This makes the θ operation identity. Figure 8 represents the state W which is obtained after applying a KECCAK-p permutation on message block D.

Substituting w_i, we get the following equations

$$h'_0 = d_0 + RC_0 + e_0,\ h'_1(20) = e_6$$

$$h'_2(21) = 0,\ h'_3(43) = 0,\ h'_4(50) = \overline{d_2(62)}d_3(55)$$

$$h'_5(36) = e_3 + d_0,\ h'_6(44) = 0$$

$$h'_7(61) = d_1(1),\ 1 = d_0(36)$$

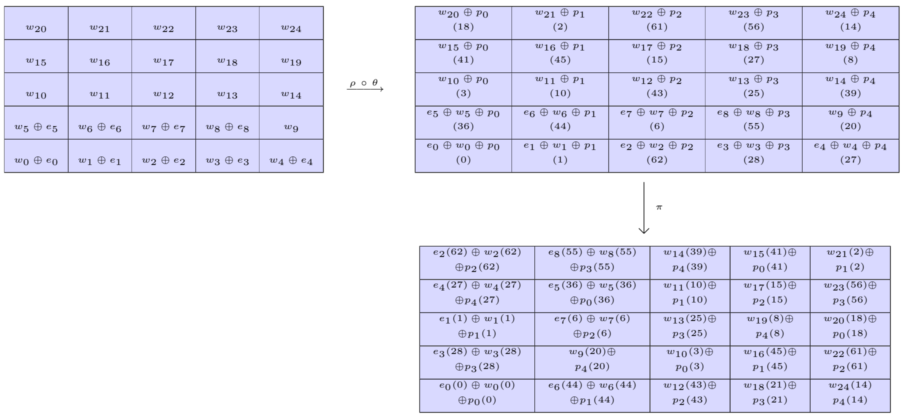
$d_2(62)$	$d_3(55)$	0	$d_2(62)$	$\overline{d_2(62)}d_3(55)$
0	$d_0(36)$	0	0	$d_0(36)$
$d_1(1)$	$d_2(6)$	0	$d_1(1)$	$\overline{d_1(1)}d_2(6)$
$d_3(28)$	0	0	$d_3(28)$	0
$d_0 \oplus RC_0$	$d_1(44)$	0	d_0	$\overline{d_0}d_1(44)$

Fig. 8. State W

From the above equation, we have to assign $d_0 = 1$ and by using the message lanes e_0, e_6, e_3, d_1, d_2 and d_3 we can get any values for $h'_0, h'_1, h'_4, h'_5, h'_7$. Hence, for hash values $h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7$ with constraints $h_3 = \overline{h_2}h_0$, $h_4 = h_2 + \overline{h_0}h_1$, $h_7 = \overline{h_6}$, preimage can be found by using 2 message block and keeping both the θ operation as identity. So, for total 2^{320} hash values we can find preimage by using this analysis.

By using two message block and keeping only first θ operation as identity: We make the first θ operation applied to the message block D as identity. So, we are left with only 4 message lanes d_0, d_1, d_2, d_3. After applying a KECCAK-p permutation on message block D we get the state W which is same as the one shown above.

For the second message, we are not putting any constraints.



w_{20}	w_{21}	w_{22}	w_{23}	w_{24}
w_{15}	w_{16}	w_{17}	w_{18}	w_{19}
w_{10}	w_{11}	w_{12}	w_{13}	w_{14}
$w_5 \oplus e_5$	$w_6 \oplus e_6$	$w_7 \oplus e_7$	$w_8 \oplus e_8$	w_9
$w_0 \oplus e_0$	$w_1 \oplus e_1$	$w_2 \oplus e_2$	$w_3 \oplus e_3$	$w_4 \oplus e_4$

$w_{20} \oplus p_0$ (18)	$w_{21} \oplus p_1$ (2)	$w_{22} \oplus p_2$ (61)	$w_{23} \oplus p_3$ (56)	$w_{24} \oplus p_4$ (14)
$w_{15} \oplus p_0$ (41)	$w_{16} \oplus p_1$ (45)	$w_{17} \oplus p_2$ (15)	$w_{18} \oplus p_3$ (27)	$w_{19} \oplus p_4$ (8)
$w_{10} \oplus p_0$ (3)	$w_{11} \oplus p_1$ (10)	$w_{12} \oplus p_2$ (43)	$w_{13} \oplus p_3$ (25)	$w_{14} \oplus p_4$ (39)
$e_5 \oplus w_5 \oplus p_0$ (36)	$e_6 \oplus w_6 \oplus p_1$ (44)	$e_7 \oplus w_7 \oplus p_2$ (6)	$e_8 \oplus w_8 \oplus p_3$ (55)	$w_9 \oplus p_4$ (20)
$e_0 \oplus w_0 \oplus p_0$ (0)	$e_1 \oplus w_1 \oplus p_1$ (1)	$e_2 \oplus w_2 \oplus p_2$ (62)	$e_3 \oplus w_3 \oplus p_3$ (28)	$e_4 \oplus w_4 \oplus p_4$ (27)

$e_2(62) \oplus w_2(62) \oplus p_2(62)$	$e_8(55) \oplus w_8(55) \oplus p_3(55)$	$w_{14}(39) \oplus p_4(39)$	$w_{15}(41) \oplus p_0(41)$	$w_{21}(2) \oplus p_1(2)$
$e_4(27) \oplus w_4(27) \oplus p_4(27)$	$e_5(36) \oplus w_5(36) \oplus p_0(36)$	$w_{11}(10) \oplus p_1(10)$	$w_{17}(15) \oplus p_2(15)$	$w_{23}(56) \oplus p_3(56)$
$e_1(1) \oplus w_1(1) \oplus p_1(1)$	$e_7(6) \oplus w_7(6) \oplus p_2(6)$	$w_{13}(25) \oplus p_3(25)$	$w_{19}(8) \oplus p_4(8)$	$w_{20}(18) \oplus p_0(18)$
$e_3(28) \oplus w_3(28) \oplus p_3(28)$	$w_9(20) \oplus p_4(20)$	$w_{10}(3) \oplus p_0(3)$	$w_{16}(45) \oplus p_1(45)$	$w_{22}(61) \oplus p_2(61)$
$e_0(0) \oplus w_0(0) \oplus p_0(0)$	$e_6(44) \oplus w_6(44) \oplus p_1(44)$	$w_{12}(43) \oplus p_2(43)$	$w_{18}(21) \oplus p_3(21)$	$w_{24}(14)$ $p_4(14)$

Fig. 9. By using two message block and keeping only first θ operation as identity

After applying $\pi \circ \rho \circ \theta$ over the state $W \oplus E$ and equating this state to the state we got after applying $\chi^{-1} \circ \iota^{-1}$ on the state of the hash value, we get 9 lane equation (576 equation) over the message block D and E.

$$h'_0 = e_0(0) + d_0(0) + RC(0) + p_0(0),\ h'_1 = e_6(44) + p_1(44)$$
$$h'_2 = p_2(43),\ h'_3 = p_3(21),\ h'_4 = \overline{d_2(12)}d_3(5) + p_4(14)$$
$$h'_5 = e_3(28) + d_0(28) + p_3(28),\ h'_6 = p_4(20)$$
$$h'_7 = d_1(4) + p_0(3),\ 1 = d_0(17) + p_1(45)$$

These equations are not linear because the column parities p_0 and p_3 contains the terms $\overline{d_0(0)}d_1(44), \overline{d_1(1)}d_2(6)$, and $\overline{d_2(62)}d_3(55)$. To make these equations linear we assigned $d_0 = 0$ and $d_2 = 0$. By this, we also get assignment of e_3, e_6 and d_3.

$$e_3(28) = h'_5(0) + h'_3(7) \text{ because } d_0 = 0 \text{ and } h'_3(0) = p_3(21)$$
$$e_6(44) = h'_1(0) + 1 \text{ because } d_0 = 0 \text{ and } d_0(17) + p_1(45) = 1$$
$$d_3(5) = h'_4(0) + h'_6(6) \text{ because } d_2 = 0 \text{ and } h'_6 = p_4(20)$$

The remaining linear equation are

$$h'_0 = e_0(0) + RC(0) + d_1(44) + d_3(55) + e_4(0) + d_1(43) + d_3(54) + e_1(63) + e_6(63)$$
$$h'_2 = d_1(23) + d_3(34) + e_1(43) + e_6(43) + d_1(43) + d_3(6) + e_3(42) + e_8(42)$$
$$h'_3 = e_2(21) + e_7(21) + d_1(0) + d_3(11) + e_4(20)$$
$$h'_6 = d_3(48) + d_1(21) + e_3(20) + e_8(20) + d_1(20) + d_3(47) + RC(19) + e_0(19) + e_5(19)$$
$$h'_7 = d_1(1) + d_1(47) + d_3(58) + e_4(3) + d_1(46) + d_3(57) + e_1(2) + e_6(2)$$
$$1 = RC(45) + d_3(9) + d_1(46) + e_0(45) + e_5(45) + e_2(44) + e_7(44)$$

We assign e_7 to 0 and assign the last bit of e_8 to 1 to satisfy the padding condition while the rest of the bits of e_8 are assigned to 0. This is done so that the preimage length is minimized. Now we are left with total 384 linear equation in 384 variables. All of these linear equations are linearly independent. Applying Gaussian elimination, we can completely find the message block D and E that gives the required hash value on application of 1 round of KECCAK-512.

4 Results and Extension to Collision attack

The above preimage attack was implemented in C++ [15] using the NTL library[14] from Victor Shoup. The code was executed on a laptop with Intel Core i5-7200 processor and 16 GB RAM giving the preimage in less than 0.005 seconds. In the analysis given in 3.4, if we randomly choose e_7 and e_8 while also keep the last bit of e_8 as 1, we can get 2^{127} preimages for the same hash value, thus giving us a collision attack.

The following tables describe the characterization of the hash values that can be found using the preimage analysis done in section 3.4.

Type of attack	Number of hash values	Characterization of hash values
1 message block, θ as identity	2^{191}	$h_2 = h_6 = h_7 = 0$, $h_5[35] = 1$, $h_4 = \overline{h_0}h_1$, $h_3 = h_0$
1 message block, θ not identity	2^{447}	$\overline{h_6} = h_7(0) \oplus h_4(6) \oplus \overline{h_0(6)}.(h_1(6) \oplus \overline{h_2(6)}.h_3(6)) \oplus 1$
2 message blocks, both θ as identity	2^{320}	$h_3 = \overline{h_2}.h_0$, $h_4 = h_2 \oplus \overline{h_0}.h_1$, $h_7 = \overline{h_6}$
2 message blocks, first θ as identity	2^{512}	All possible hash values

Table 1. Characterization of hash values

5 Conclusion and Future Works

Our approach gives a preimage and collision attack to all the variants of 1 round KECCAK hash functions. These are currently the fastest attacks known. These attacks does not pose a threat to the security of 24-round KECCAK. In future, we need to find whether this idea can be extended to 2 rounds KECCAK-384 and KECCAK-512.

References

1. Daniel J Bernstein. Second preimages for 6 (7?(8??)) rounds of keccak. *NIST mailing list*, 2010.
2. G Bertoni, J Daemen, M Peeters, and GV Assche. The keccak reference. online at http://keccak. noekeon. org/keccak-reference-3.0. pdf, 2011.
3. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak specifications. *Submission to NIST (Round 2)*, 2009.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponges. *online] http://sponge. noekeon. org*, 2011.
5. 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.
6. Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on keccak-224 and keccak-256. In *FSE*, volume 12, pages 442–461. Springer, 2012.
7. 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.
8. Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *Journal of cryptology*, 27(2):183–209, 2014.
9. Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. *Federal Inf. Process. Stds.(NIST FIPS)-202*, 2015.
10. Jian Guo, Meicheng Liu, and Ling Song. Linear structures: Applications to cryptanalysis of round-reduced keccak. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 249–274. Springer, 2016.

11. 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.
12. Paweł Morawiecki and Marian Srebrny. A sat-based preimage analysis of reduced keccak hash functions. *Information Processing Letters*, 113(10-11):392–397, 2013.
13. María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical analysis of reduced-round keccak. In *INDOCRYPT*, volume 7107, pages 236–254. Springer, 2011.
14. Victor Shoup. Ntl: A library for doing number theory. *www. shoup. net/ntl/*, 2001.
15. Mahesh S.R and Rajendra Kumar. Keccak - cryptanalysis of keccak. *https://github.com/Mahe94/KECCAK*.