Cryptanalysis of 2 round KECCAK-384

Rajendra Kumar ,Nikhil Mittal, Shashank Singh

Center for Cybersecurity, Indian Institute of Technology Kanpur,
Indian Institute of Science Education and Research Bhopal

Indocrypt 2018, Delhi

Table of contents

Outline

1. Introduction
2. Known Attacks
3. Our Contribution
4. Conclusion

Cryptographic Hash Function

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$ where n is a fixed value.

Cryptographic Hash Function

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$ where n is a fixed value.
- Given m, easy to compute $H(m)$.

Cryptographic Hash Function

- $H : \{0,1\}^* \to \{0,1\}^n$ where n is a fixed value.
- Given m, easy to compute $H(m)$.
- Computationally hard problems:

Cryptographic Hash Function

- $H : \{0,1\}^* \to \{0,1\}^n$ where n is a fixed value.
- Given m, easy to compute $H(m)$.
- Computationally hard problems:
  1. Preimage: Given $H(m)$, Find message m.

Cryptographic Hash Function

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$ where n is a fixed value.

- Given m, easy to compute $H(m)$.

- Computationally hard problems:
  1. Preimage: Given $H(m)$, Find message m.
  2. Second-Preimage: Given m, Find m' such that $H(m) = H(m')$.

Cryptographic Hash Function

- $H : \{0,1\}^* \to \{0,1\}^n$ where n is a fixed value.
- Given m, easy to compute $H(m)$.
- Computationally hard problems:
  1. Preimage: Given $H(m)$, Find message m.
  2. Second-Preimage: Given m, Find m' such that $H(m) = H(m')$.
  3. Collision: Find m and m', such that $H(m) = H(m')$.

Cryptographic Hash Function

- $H : \{0,1\}^* \to \{0,1\}^n$ where n is a fixed value.
- Given m, easy to compute $H(m)$.
- Computationally hard problems:
  1. Preimage: Given $H(m)$, Find message m.
  2. Second-Preimage: Given m, Find m' such that $H(m) = H(m')$.
  3. Collision: Find m and m', such that $H(m) = H(m')$.
- Hash functions are used for Authentication, Non-repudiation and Integrity.

Sponge Construction

- Components of the Construction.

Sponge Construction

- Components of the Construction.
  1. f: underlying function on fixed length strings.

Sponge Construction

- Components of the Construction.
  1. f: underlying function on fixed length strings.
  2. r: rate (block size)

Sponge Construction

- Components of the Construction.
  1. f: underlying function on fixed length strings.
  2. r: rate (block size)
  3. pad: padding rule (10^*1)

Sponge Construction

- Components of the Construction.
  1. f: underlying function on fixed length strings.
  2. r: rate (block size)
  3. pad: padding rule (10^*1)

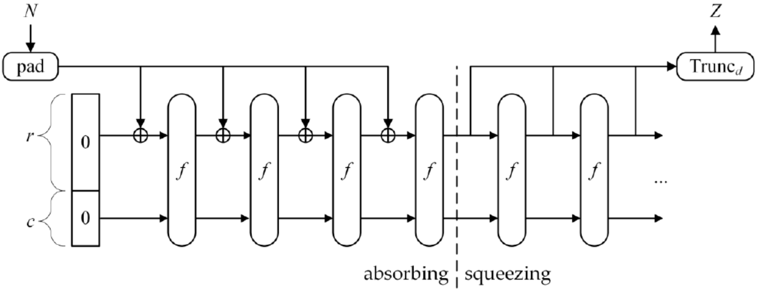

Figure: Sponge Construction $Z = Sponge[f, pad, r](N, d)$

Credit: *http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf*

State

- State is represented by 5-by-5-by-w bits.

State

- State is represented by 5-by-5-by-w bits.
- Converting Strings to State Arrays; $A[x, y, z] = S[w(5y + x) + z]$.

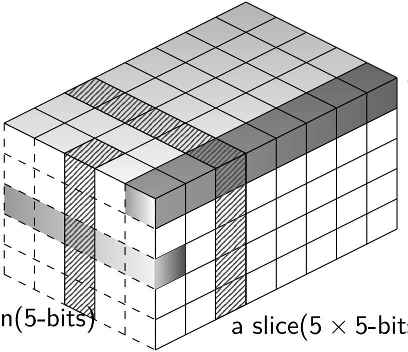

Figure: A state in KECCAK

KECCAK-*p* Permutation

- `Round`- A round of KECCAK-*p* permutation. Consist of five transformations, called Step Mappings ($\theta, \rho, \pi, \chi, \iota$)

Specification of θ

- XOR each bit in the state with the parities of two columns.

Specification of θ

- XOR each bit in the state with the parities of two columns.

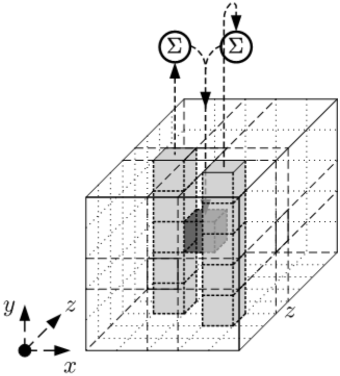

Figure: Credit:https://keccak.team/figures.html

Specification of θ

- For all pairs (x,z)

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

Specification of θ

- For all pairs (x,z)

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

For all triples (x,y,z)

$$A'[x,y,z] = A[x,y,z] \oplus C[x-1,z] \oplus C[x+1,z-1]$$

Specification of ρ

- ρ : Rotate the bits of each lane by predefind constants.

Specification of ρ

- ρ : Rotate the bits of each lane by predefind constants.

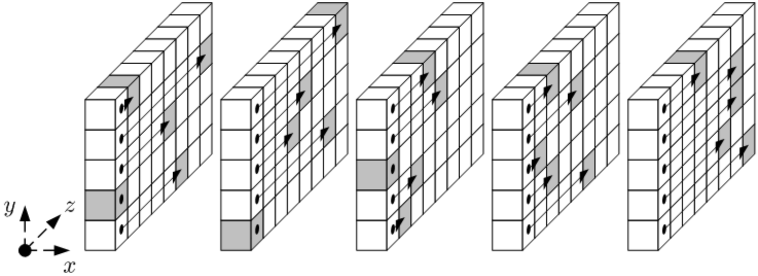

Figure: Credit:https://keccak.team/figures.html

Specification of π

- π: Rearrange the position of the Lanes.

Specification of π

- π: Rearrange the position of the Lanes.

for all triples (x, y, z)

$$A'[x, y, z] = A[(x + 3y) \bmod 5, x, z]$$

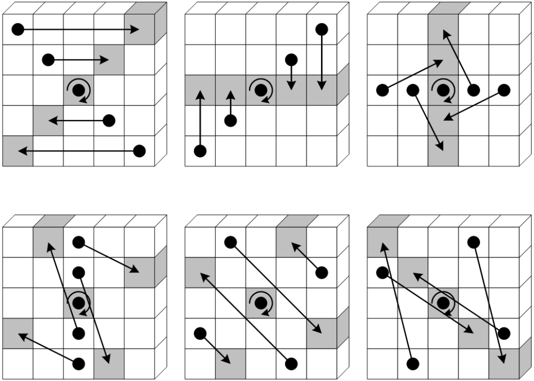

Figure: Credit:https://keccak.team/figures.html

Specification of χ and ι

- χ: XOR each bit with a non linear function of two other bits in its row.

Specification of χ and ι

- χ: XOR each bit with a non linear function of two other bits in its row.
- For all triples (x, y, z)

$$A'[x, y, z] = A[x, y, z] \oplus (\overline{(A[(x+1), y, z])}).A[(x+2), y, z])$$

Specification of χ and ι

- χ: XOR each bit with a non linear function of two other bits in its row.
- For all triples (x, y, z)

$$A'[x, y, z] = A[x, y, z] \oplus (\overline{(A[(x+1), y, z])}).A[(x+2), y, z])$$

- ι: XOR the $\texttt{lane}(0, 0)$ with the round constant and other lanes are unaffected. Round dependent step Mapping.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.
- Consists of n_r rounds of $\texttt{Round}$.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.
- Consists of n_r rounds of $\texttt{Round}$.
- $\textsc{Keccak}\text{-}p[b, n_r](S)$
    1. Convert S into a state Array A.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.
- Consists of n_r rounds of $\texttt{Round}$.
- $\textsc{Keccak}$-$p[b, n_r](S)$
    1. Convert S into a state Array A.
    2. For i_r from 0 to $n_r - 1$, let $A = \texttt{Round}(A, i_r)$.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.
- Consists of n_r rounds of $\texttt{Round}$.
- $\textsc{Keccak}\text{-}p[b, n_r](S)$
  1. Convert S into a state Array A.
  2. For i_r from 0 to $n_r - 1$, let $A = \texttt{Round}(A, i_r)$.
  3. Convert A into String S' of length b.

Specification of KECCAK-p[b,n_r]

- $\texttt{Round}(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r)$.
- Consists of n_r rounds of $\texttt{Round}$.
- $\textsc{Keccak}\text{-}p[b, n_r](S)$
  1. Convert S into a state Array A.
  2. For i_r from 0 to $n_r - 1$, let $A = \texttt{Round}(A, i_r)$.
  3. Convert A into String S' of length b.
  4. Return S'.

KECCAK-384 and 2-round KECCAK-384

- KECCAK-384 $=$ KECCAK-$p[1600, 24][\mathrm{rate} = 1600 - 2 \cdot 384]$.

Keccak-384 and 2-round Keccak-384

- Keccak-384 $=$ Keccak-$p[1600, 24][\text{rate} = 1600 - 2 \cdot 384]$.

- 2-round Keccak-384 $=$ Keccak-$p[1600, 2][\text{rate} = 1600 - 2 \cdot 384]$.

Outline

Pre-image Attacks

Table: Preimage attack results.

No. of rounds	Hash length	Time Complexity	Reference
1	KECCAK- 224/256/384/512	1	[KRA18]

Pre-image Attacks

Table: Preimage attack results.

No. of rounds	Hash length	Time Complexity	Reference
1	KECCAK- 224/256/384/512	1	[KRA18]
2	KECCAK- 224/256	2^{33}	[NPRM11]

Pre-image Attacks

Table: Preimage attack results.

No. of rounds	Hash length	Time Complexity	Reference
1	KECCAK- 224/256/384/512	1	[KRA18]
2	KECCAK- 224/256	2^{33}	[NPRM11]
2	KECCAK- 224/256	1	[GLS16]
2	KECCAK- 384/512	$2^{129}/2^{384}$	[GLS16]
3	KECCAK- 224/256/384/512	$2^{97}/2^{192}/2^{322}/2^{484}$	[GLS16]
4	KECCAK- 224/256	$2^{213}/2^{251}$	[GLS16]

Pre-image Attacks

Table: Preimage attack results.

No. of rounds	Hash length	Time Complexity	Reference
1	KECCAK- 224/256/384/512	1	[KRA18]
2	KECCAK- 224/256	2^{33}	[NPRM11]
2	KECCAK- 224/256	1	[GLS16]
2	KECCAK- 384/512	$2^{129}/2^{384}$	[GLS16]
3	KECCAK- 224/256/384/512	$2^{97}/2^{192}/2^{322}/2^{484}$	[GLS16]
4	KECCAK- 224/256	$2^{213}/2^{251}$	[GLS16]
4	KECCAK- 384/512	$2^{378}/2^{506}$	[MPS13]

Pre-image Attacks

Table: Preimage attack results.

No. of rounds	Hash length	Time Complexity	Reference
1	KECCAK- 224/256/384/512	1	[KRA18]
2	KECCAK- 224/256	2^{33}	[NPRM11]
2	KECCAK- 224/256	1	[GLS16]
2	KECCAK – 384/512	2^{129} / 2^{384}	[GLS16]
3	KECCAK- 224/256/384/512	$2^{97}/2^{192}/2^{322}/2^{484}$	[GLS16]
4	KECCAK- 224/256	$2^{213}/2^{251}$	[GLS16]
4	KECCAK- 384/512	$2^{378}/2^{506}$	[MPS13]

Outline

1. Introduction
2. Known Attacks
3. Our Contribution
4. Conclusion

Notations

- We will represent a state by 25 Lanes.

Notations

- We will represent a state by 25 Lanes.
- Each lane in a state will be represented by a variable which is a 64-bit array.(example: a_0)

Notations

- We will represent a state by 25 Lanes.
- Each lane in a state will be represented by a variable which is a 64-bit array.(example: a_0)
- A variable with a number in round bracket "$(\cdot)$" represents the shift of the bits in array towards MSB. (example: $a_0(4)$)

Notations

- We will represent a state by 25 Lanes.
- Each lane in a state will be represented by a variable which is a 64-bit array.(example: a_0)
- A variable with a number in round bracket "$(\cdot)$" represents the shift of the bits in array towards MSB. (example: $a_0(4)$)
- A variable with a number in square bracket "$[\cdot]$" represents the bit value of the variable at that index.(examples: $a_0[3]$)

2-round KECCAK

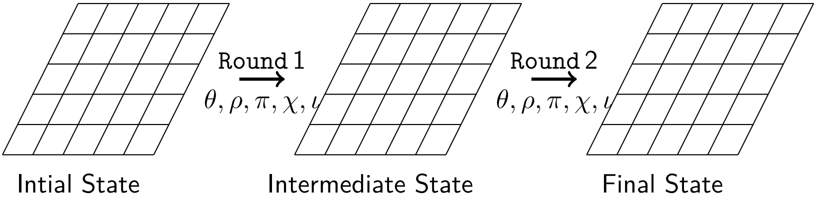

Figure: Two round of Keccak-384

Initial State

0	0	0	0	0
0	0	0	0	0
a_1	b_1	c_2	0	0
a_2	b_2	c_1	d_1	e_1
a_0	b_0	c_0	d_0	e_0

Figure: Setting of Initial State in the Attack

first round θ step mapping

- θ step mapping diffuses message bits to full state.

first round θ step mapping

- θ step mapping diffuses message bits to full state.
- Aim: Control the diffusion.

first round θ step mapping

- θ step mapping diffuses message bits to full state.
- Aim: Control the diffusion.
- How: Put condition on message bits.

first round θ step mapping

- θ step mapping diffuses message bits to full state.
- Aim: Control the diffusion.
- How: Put condition on message bits.
- Conditions to make column parity zero:

$$\begin{aligned} &a_2 = a_0 \oplus a_1, \quad b_2 = b_0 \oplus b_1, \quad c_2 = c_0 \oplus c_1 \\ &d_1 = d_0 \quad \text{and} \quad e_1 = e_0. \end{aligned} \tag{1}$$

State 1 to State 2

0	0	0	0	0
0	0	0	0	0
a_1	b_1	c_2	0	0
a_2	b_2	c_1	d_1	e_1
a_0	b_0	c_0	d_0	e_0

State 1

$$\xrightarrow{\theta, \rho, \pi}$$

$c_0(62)$	$d_1(55)$	0	0	0
$e_0(27)$	$a_2(36)$	$b_1(10)$	0	0
$b_0(1)$	$c_1(6)$	0	0	0
$d_0(28)$	$e_1(20)$	$a_1(3)$	0	0
$a_0(0)$	$b_2(44)$	$c_2(43)$	0	0

State 2

Figure: Diagram for 2-round preimage attack on KECCAK-384

Final State

y					
4	$\star$	$\star$	$\star$	$\star$	$\star$
3	$\star$	$\star$	$\star$	$\star$	$\star$
2	$\star$	$\star$	$\star$	$\star$	$\star$
1	h_5	$\star$	$\star$	$\star$	$\star$
0	h_0	h_1	h_2	h_3	h_4
	0	1	2	3	4
	x				

Figure: Final State

Observations

χ is a row dependent operation.

Observations

χ is a row dependent operation.

$$\boxed{a_0}\boxed{a_1}\boxed{a_2}\boxed{a_3}\boxed{a_4} \xrightarrow{\chi^{-1}} \boxed{a'_0}\boxed{a'_1}\boxed{a'_2}\boxed{a'_3}\boxed{a'_4}$$

Figure: Computation of χ^{-1}

Observations

χ is a row dependent operation.

$$\boxed{a_0}\boxed{a_1}\boxed{a_2}\boxed{a_3}\boxed{a_4} \xrightarrow{\chi^{-1}} \boxed{a'_0}\boxed{a'_1}\boxed{a'_2}\boxed{a'_3}\boxed{a'_4}$$

Figure: Computation of χ^{-1}

$$\boxed{a_0}\boxed{*}\boxed{*}\boxed{*}\boxed{*} \xrightarrow{\chi^{-1}} \boxed{a_0}\boxed{1}\boxed{*}\boxed{*}\boxed{*}$$

Figure: Computation of χ^{-1}

χ and ι inverse

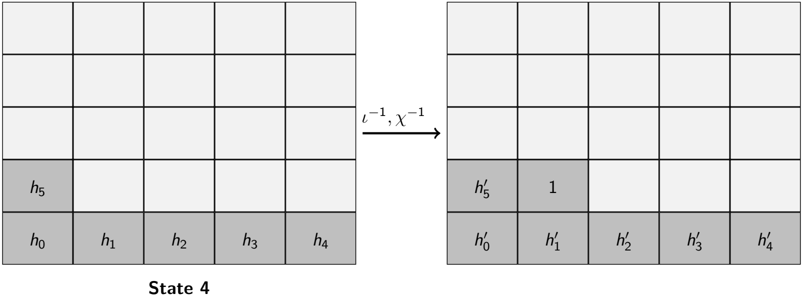

Figure: Diagram for 2-round preimage attack on KECCAK-384

State 4 to State 3

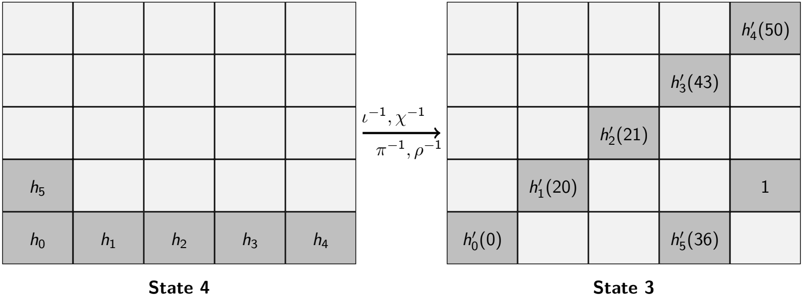

Figure: Diagram for 2-round preimage attack on KECCAK-384

State 1 to 4

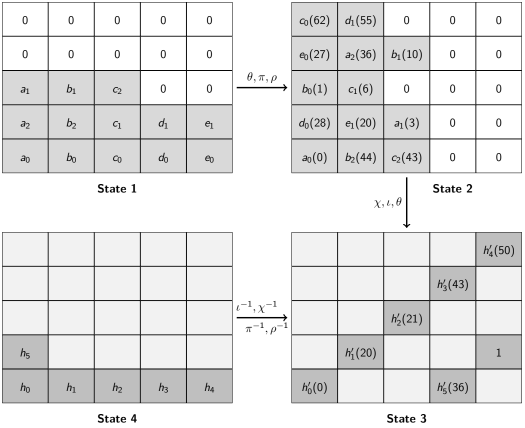

Figure: Diagram for 2-round preimage attack on KECCAK-384

State 2 to State 3

$c_0(62)$	$d_1(55)$	0	0	0
$e_0(27)$	$a_2(36)$	$b_1(10)$	0	0
$b_0(1)$	$c_1(6)$	0	0	0
$d_0(28)$	$e_1(20)$	$a_1(3)$	0	0
$a_0(0)$	$b_2(44)$	$c_2(43)$	0	0

State 2

$\xrightarrow{\chi, \iota, \theta}$

				$h'_4(50)$
			$h'_3(43)$	
		$h'_2(21)$		
	$h'_1(20)$			1
$h'_0(0)$			$h'_5(36)$	

State 3

Figure: Intermediate States in 2-round preimage attack on KECCAK-384

Description of Attack

- Fix d_0, d_1 as constants (with condition that $d_0 = d_1$).

Description of Attack

- Fix d_0, d_1 as constants (with condition that $d_0 = d_1$).
- In state 2, we have $11 \cdot 64$ free variables.

Description of Attack

- Fix d_0, d_1 as constants (with condition that $d_0 = d_1$).
- In state 2, we have $11 \cdot 64$ free variables.
- And we need to satisfy $7 \cdot 64$ Boolean equations and $4 \cdot 64$ column parity conditions.

Description of Attack

- Fix d_0, d_1 as constants (with condition that $d_0 = d_1$).
- In state 2, we have $11 \cdot 64$ free variables.
- And we need to satisfy $7 \cdot 64$ Boolean equations and $4 \cdot 64$ column parity conditions.
- So, we expect a solution.

Description of Attack

- Fix d_0, d_1 as constants (with condition that $d_0 = d_1$).
- In state 2, we have $11 \cdot 64$ free variables.
- And we need to satisfy $7 \cdot 64$ Boolean equations and $4 \cdot 64$ column parity conditions.
- So, we expect a solution.
- We do find the possible solution subspace.

Description of Attack

- In state 3, the values of i^{th}-slice depend on the $(i-1)^{\mathrm{th}}$ and i^{th}-slice of state 2.

Description of Attack

- In state 3, the values of i^{th}-slice depend on the $(i-1)^{\text{th}}$ and i^{th}-slice of state 2.
- We first find the set of input message bits which satisfy the small collection of consecutive slices of state 3.

Description of Attack

- In state 3, the values of i^{th}-slice depend on the $(i-1)^{\mathrm{th}}$ and i^{th}-slice of state 2.
- We first find the set of input message bits which satisfy the small collection of consecutive slices of state 3.
- We then merge the solutions to find message bits which satisfy large collection of consecutive slices.

Possible solutions for groups of 3 slices

- Consider a group of 3 slices (for example take the first 3 slices).

Possible solutions for groups of 3 slices

- Consider a group of 3 slices (for example take the first 3 slices).
- It contains the following message bits
  - $a_0[0, 1, 2]$, $a_1[3, 4, 5]$, $a_2[36, 37, 38]$
  - $b_0[1, 2, 3]$, $b_1[10, 11, 12]$, $b_2[44, 45, 46]$
  - $c_0[62, 63, 0]$, $c_1[6, 7, 8]$, $c_2[43, 44, 45]$
  - $e_0[27, 28, 29]$, $e_1[20, 21, 22]$

Possible solutions for groups of 3 slices

- Consider a group of 3 slices (for example take the first 3 slices).
- It contains the following message bits
  - $a_0[0, 1, 2]$, $a_1[3, 4, 5]$, $a_2[36, 37, 38]$
  - $b_0[1, 2, 3]$, $b_1[10, 11, 12]$, $b_2[44, 45, 46]$
  - $c_0[62, 63, 0]$, $c_1[6, 7, 8]$, $c_2[43, 44, 45]$
  - $e_0[27, 28, 29]$, $e_1[20, 21, 22]$
- Once we fix these message bits in the State 2, the slice 1 and slice 2 of State 3 get fixed.

Possible solutions for groups of 3 slices

- Consider a group of 3 slices (for example take the first 3 slices).
- It contains the following message bits
  - $a_0[0, 1, 2]$, $a_1[3, 4, 5]$, $a_2[36, 37, 38]$
  - $b_0[1, 2, 3]$, $b_1[10, 11, 12]$, $b_2[44, 45, 46]$
  - $c_0[62, 63, 0]$, $c_1[6, 7, 8]$, $c_2[43, 44, 45]$
  - $e_0[27, 28, 29]$, $e_1[20, 21, 22]$
- Once we fix these message bits in the State 2, the slice 1 and slice 2 of State 3 get fixed.
- Furthermore there is no dependency between these message bits.

Possible solutions for groups of 3 slices

- Consider a group of 3 slices (for example take the first 3 slices).
- It contains the following message bits
  - $a_0[0,1,2]$, $a_1[3,4,5]$, $a_2[36,37,38]$
  - $b_0[1,2,3]$, $b_1[10,11,12]$, $b_2[44,45,46]$
  - $c_0[62,63,0]$, $c_1[6,7,8]$, $c_2[43,44,45]$
  - $e_0[27,28,29]$, $e_1[20,21,22]$
- Once we fix these message bits in the State 2, the slice 1 and slice 2 of State 3 get fixed.
- Furthermore there is no dependency between these message bits.
- Thus the total possible solutions for this 3-slice $= 2^{33-2\cdot 7} = 2^{19}$.

Possible solutions for groups of 6 slices

- This is obtained by merging two groups of 3 slices.

Possible solutions for groups of 6 slices

- This is obtained by merging two groups of 3 slices.
- Consider, for example, the first two 3-slices (first 6 slices). It contains the following message bits:
  - $a_0[0-5]$, $a_1[3-8]$, $a_2[36-41]$
  - $b_0[1-6]$, $b_1[10-15]$, $b_2[44-49]$
  - $c_0[62-3]$, $c_1[6-11]$, $c_2[43-48]$
  - $e_0[27-32]$, $e_1[20-25]$

Possible solutions for groups of 6 slices

- This is obtained by merging two groups of 3 slices.

- Consider, for example, the first two 3-slices (first 6 slices). It contains the following message bits:
  - $a_0[0-5]$, $a_1[3-8]$, $a_2[36-41]$
  - $b_0[1-6]$, $b_1[10-15]$, $b_2[44-49]$
  - $c_0[62-3]$, $c_1[6-11]$, $c_2[43-48]$
  - $e_0[27-32]$, $e_1[20-25]$

- During merging, we get to compute the bit values of slice 3 of the State 3 as well.

Possible solutions for groups of 6 slices

- This is obtained by merging two groups of 3 slices.
- Consider, for example, the first two 3-slices (first 6 slices). It contains the following message bits:
  - $a_0[0-5]$, $a_1[3-8]$, $a_2[36-41]$
  - $b_0[1-6]$, $b_1[10-15]$, $b_2[44-49]$
  - $c_0[62-3]$, $c_1[6-11]$, $c_2[43-48]$
  - $e_0[27-32]$, $e_1[20-25]$
- During merging, we get to compute the bit values of slice 3 of the State 3 as well.
- Since, we already have the correct bit values of slice 3 of the State 3, and there is no dependency between the above message bit variables, we end up having total possible solutions $= 2^{2\cdot19-7} = 2^{31}$.

Possible solutions for groups of 12 slices

- Similar to the case of 6-slice, the solution for a 12-slice is obtained by merging two 6-slices.

Possible solutions for groups of 12 slices

- Similar to the case of 6-slice, the solution for a 12-slice is obtained by merging two 6-slices.
- Consider, for example, the first 12 slices. It contain the following message bits:
  - $a_0[0-11]$, $a_1[3-14]$, $a_2[36-47]$
  - $b_0[1-12]$, $b_1[10-21]$, $b_2[44-55]$
  - $c_0[62-9]$, $c_1[6-17]$, $c_2[43-54]$
  - $e_0[27-38]$, $e_1[20-31]$

Possible solutions for groups of 12 slices

- Similar to the case of 6-slice, the solution for a 12-slice is obtained by merging two 6-slices.
- Consider, for example, the first 12 slices. It contain the following message bits:
  - $a_0[0-11]$, $a_1[3-14]$, $a_2[36-47]$
  - $b_0[1-12]$, $b_1[10-21]$, $b_2[44-55]$
  - $c_0[62-9]$, $c_1[6-17]$, $c_2[43-54]$
  - $e_0[27-38]$, $e_1[20-31]$
- As before, here we again get the values of slice 6 of State 3.

Possible solutions for groups of 12 slices

- Similar to the case of 6-slice, the solution for a 12-slice is obtained by merging two 6-slices.
- Consider, for example, the first 12 slices. It contain the following message bits:
  - $a_0[0-11]$, $a_1[3-14]$, $a_2[36-47]$
  - $b_0[1-12]$, $b_1[10-21]$, $b_2[44-55]$
  - $c_0[62-9]$, $c_1[6-17]$, $c_2[43-54]$
  - $e_0[27-38]$, $e_1[20-31]$
- As before, here we again get the values of slice 6 of State 3.
- But, in contrast to 6-slice, the bit variables are not independent. The bit variables $e_0[27-31]$ and $e_1[27-31]$ are dependent.

Possible solutions for groups of 12 slices

- Similar to the case of 6-slice, the solution for a 12-slice is obtained by merging two 6-slices.
- Consider, for example, the first 12 slices. It contain the following message bits:
  - $a_0[0-11]$, $a_1[3-14]$, $a_2[36-47]$
  - $b_0[1-12]$, $b_1[10-21]$, $b_2[44-55]$
  - $c_0[62-9]$, $c_1[6-17]$, $c_2[43-54]$
  - $e_0[27-38]$, $e_1[20-31]$
- As before, here we again get the values of slice 6 of State 3.
- But, in contrast to 6-slice, the bit variables are not independent. The bit variables $e_0[27-31]$ and $e_1[27-31]$ are dependent.
- Hence, the total possible solutions $= 2^{2\cdot 31-5-7} = 2^{50}$.

Possible solutions for groups of 24 slices

- This is obtained by merging two groups of 12 slices.

Possible solutions for groups of 24 slices

- This is obtained by merging two groups of 12 slices.
- For example, consider the first 24 slices i.e.,
  - $a_0[0-23], \quad a_1[3-26], \quad a_2[36-59]$
  - $b_0[1-24], \quad b_1[10-33], \quad b_2[44-3]$
  - $c_0[62-21], \quad c_1[6-29], \quad c_2[43-2]$
  - $e_0[27-50], \quad e_1[20-43]$

Possible solutions for groups of 24 slices

- This is obtained by merging two groups of 12 slices.
- For example, consider the first 24 slices i.e.,
  - $a_0[0-23]$, $a_1[3-26]$, $a_2[36-59]$
  - $b_0[1-24]$, $b_1[10-33]$, $b_2[44-3]$
  - $c_0[62-21]$, $c_1[6-29]$, $c_2[43-2]$
  - $e_0[27-50]$, $e_1[20-43]$
- This is very much similar to the 12 slice solution. In this case we get 7 dependencies and hence the total number of possible solutions is equal to $2^{2\cdot 50-7-7}=2^{86}$.

Possible solutions for group of 48 slices

- This is done by merging two groups of 24 slices.

Possible solutions for group of 48 slices

- This is done by merging two groups of 24 slices.
- For example, take first 48 slices. It contain the following message bits for a_0, a_1, a_2

  1^{st} group :

$$\left.\begin{aligned} a_0 &\rightarrow 0,\ 1,\ 2, \ldots,\ 23 \\ a_1 &\rightarrow 3,\ 4,\ 5, \ldots,\ 26 \\ a_2 &\rightarrow 36,\ 37,\ 38, \ldots,\ 59 \end{aligned}\right\} \tag{2}$$

  2^{nd} group :

$$\left.\begin{aligned} a_0 &\rightarrow 24,\ 25,\ 26, \ldots,\ 47 \\ a_1 &\rightarrow 27,\ 28,\ 29, \ldots,\ 50 \\ a_2 &\rightarrow 60,\ 61,\ 62, \ldots,\ 19 \end{aligned}\right\}. \tag{3}$$

Possible solutions for group of 48 slices

- This is done by merging two groups of 24 slices.
- For example, take first 48 slices. It contain the following message bits for a_0, a_1, a_2

  1^{st} group :

$$\left.\begin{aligned} a_0 &\rightarrow 0,\ 1,\ 2, \ldots,\ 23 \\ a_1 &\rightarrow 3,\ 4,\ 5, \ldots,\ 26 \\ a_2 &\rightarrow 36,\ 37,\ 38, \ldots,\ 59 \end{aligned}\right\} \tag{2}$$

  2^{nd} group :

$$\left.\begin{aligned} a_0 &\rightarrow 24,\ 25,\ 26, \ldots,\ 47 \\ a_1 &\rightarrow 27,\ 28,\ 29, \ldots,\ 50 \\ a_2 &\rightarrow 60,\ 61,\ 62, \ldots,\ 19 \end{aligned}\right\}. \tag{3}$$

- Dependent variables are $a_0[36-47, 3-19]$, $a_1[36-47, 3-19]$ and $a_2[36-47, 3-19]$.

Possible solutions for group of 48 slices

- This is done by merging two groups of 24 slices.
- For example, take first 48 slices. It contain the following message bits for a_0, a_1, a_2
  1^{st} group :

$$\left.\begin{aligned} a_0 &\rightarrow 0,\ 1,\ 2, \ldots, 23 \\ a_1 &\rightarrow 3,\ 4,\ 5, \ldots, 26 \\ a_2 &\rightarrow 36,\ 37,\ 38, \ldots, 59 \end{aligned}\right\} \tag{2}$$

  2^{nd} group :

$$\left.\begin{aligned} a_0 &\rightarrow 24,\ 25,\ 26, \ldots, 47 \\ a_1 &\rightarrow 27,\ 28,\ 29, \ldots, 50 \\ a_2 &\rightarrow 60,\ 61,\ 62, \ldots, 19 \end{aligned}\right\}. \tag{3}$$

- Dependent variables are $a_0[36-47, 3-19]$, $a_1[36-47, 3-19]$ and $a_2[36-47, 3-19]$.
- Total dependent variables are $(29+23+24+7) = 83$

Possible solutions for group of 48 slices

- This is done by merging two groups of 24 slices.
- For example, take first 48 slices. It contain the following message bits for a_0, a_1, a_2
  1^{st} group :
  $$\left.\begin{array}{l} a_0 \rightarrow 0,\ 1,\ 2, \ldots,\ 23 \\ a_1 \rightarrow 3,\ 4,\ 5, \ldots,\ 26 \\ a_2 \rightarrow 36,\ 37,\ 38, \ldots,\ 59 \end{array}\right\} \quad (2)$$
  2^{nd} group :
  $$\left.\begin{array}{l} a_0 \rightarrow 24,\ 25,\ 26, \ldots,\ 47 \\ a_1 \rightarrow 27,\ 28,\ 29, \ldots,\ 50 \\ a_2 \rightarrow 60,\ 61,\ 62, \ldots,\ 19 \end{array}\right\}. \quad (3)$$
- Dependent variables are $a_0[36-47, 3-19]$, $a_1[36-47, 3-19]$ and $a_2[36-47, 3-19]$.
- Total dependent variables are $(29+23+24+7) = 83$
- Total possible solutions $= 2^{2 \cdot 86 - 83 - 7} = 2^{82}$.

Final solutions

- Thus, using above method, we find the possible solutions for 48-slices, 12-slices and 4-slices.

Final solutions

- Thus, using above method, we find the possible solutions for 48-slices, 12-slices and 4-slices.

- For finding the solution for the last consicutive 16 slices, we merge possible solution of its consituent 12-slice and 4-slice.

Final solutions

- Thus, using above method, we find the possible solutions for 48-slices, 12-slices and 4-slices.
- For finding the solution for the last consicutive 16 slices, we merge possible solution of its consituent 12-slice and 4-slice.
- Final solution space is obtained by merging the solution space of first 48 slices and the last 16 slices.

Final solutions

- Thus, using above method, we find the possible solutions for 48-slices, 12-slices and 4-slices.

- For finding the solution for the last consicutive 16 slices, we merge possible solution of its consituent 12-slice and 4-slice.

- Final solution space is obtained by merging the solution space of first 48 slices and the last 16 slices.

- Space complexity of the attack $= 2^{87}$.

Final solutions

- Thus, using above method, we find the possible solutions for 48-slices, 12-slices and 4-slices.
- For finding the solution for the last consicutive 16 slices, we merge possible solution of its consituent 12-slice and 4-slice.
- Final solution space is obtained by merging the solution space of first 48 slices and the last 16 slices.
- Space complexity of the attack $= 2^{87}$.
- Time complexity of the attack $= 2^{88}$.

Implementation of attack on KECCAK$[b = 400, c = 192]$

- We have implemented the attack for 2-round KECCAK$[b = 400, c = 192]$.

Implementation of attack on $\textsc{Keccak}[b = 400, c = 192]$

- We have implemented the attack for 2-round $\textsc{Keccak}[b = 400, c = 192]$.
- Machine Specification: 132GB RAM, 20 Cores.

Implementation of attack on KECCAK$[b = 400, c = 192]$

- We have implemented the attack for 2-round KECCAK$[b = 400, c = 192]$.
- Machine Specification: 132GB RAM, 20 Cores.
- The average running time = 60 minutes.

Implementation of attack on KECCAK$[b = 400, c = 192]$

- We have implemented the attack for 2-round KECCAK$[b = 400, c = 192]$.
- Machine Specification: 132GB RAM, 20 Cores.
- The average running time = 60 minutes.
- URL: `https://github.com/nickedes/keccak`

Outline

Conclusion

- We have presented a preimage attack on the 2 rounds of round reduced KECCAK-384.

Conclusion

- We have presented a preimage attack on the 2 rounds of round reduced KECCAK-384.
- It is not yet practical but close to it.

Conclusion

- We have presented a preimage attack on the 2 rounds of round reduced KECCAK-384.
- It is not yet practical but close to it.
- Future work: Variant(s) of this attack for more round of KECCAK.

Thank You

Questions?

References I

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.

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.

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.

References II

María Naya-Plasencia, Andrea Röck, and Willi Meier.
Practical analysis of reduced-round keccak.
In *International Conference on Cryptology in India*, pages 236–254. Springer, 2011.