Preimage Attacks on the Round-reduced KECCAK with Cross-linear Structures

Yao Sun

(Joint work with Ting Li, Maodong Liao, Dingkang Wang)

SKLOIS, Institute of Information Engineering, CAS

March 6, 2018

Cryptographic Hash Function Security

Types of cryptanalytic attack:

- Preimage attack: given h, find m s.t.

$$h = \mathrm{Hash}(m);$$

- Collision attack: find m_1, m_2 ($m_1 \neq m_2$), s.t.

$$\mathrm{Hash}(m_1) = \mathrm{Hash}(m_2);$$

- Second preimage attack: given m_1, find m_2 ($m_1 \neq m_2$), s.t.

$$\mathrm{Hash}(m_1) = \mathrm{Hash}(m_2).$$

Backgrounds of KECCAK

- In 2008, KECCAK submitted to SHA-3 competition.
- In 2011, KECCAK Crunchy Crypto Collision and Pre-image Contest.
- In 2012, KECCAK won the competition.
- In 2015, KECCAK standardized by NIST as SHA-3.

Team Keccak

Guido Bertoni[3], Joan Daemen[1,2], Michaël Peeters[1], Gilles Van Assche[1] and Ronny Van Keer[1]
[1]STMicroelectronics - [2]Radboud University - [3]Security Pattern

Standard Instances

Instance	used in FIPS 202 and SP 800-185 by
KECCAK[r=1344, c=256]	SHAKE128 [FIPS 202], cSHAKE128, KMAC128, KMACXOF128, TupleHash128, TupleHashXOF128, ParallelHash128, ParallelHashXOF128 [SP 800-185]
KECCAK[r=1152, c=448]	SHA3-224 [FIPS 202]
KECCAK[r=1088, c=512]	SHAKE256, SHA3-256 [FIPS 202], cSHAKE256, KMAC256, KMACXOF256, TupleHash256, TupleHashXOF256, ParallelHash256, ParallelHashXOF256 [SP 800-185]
KECCAK[r=832, c=768]	SHA3-384 [FIPS 202]
KECCAK[r=576, c=1024]	SHA3-512 [FIPS 202]

- The SHA-3 standard: 224, 256, 384, and 512;
- SHAKE128/256.

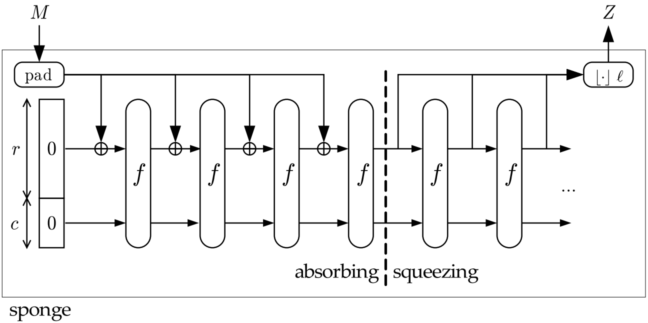
Main Contributions

1. Present two types of cross-linear structures
   - $1 \rightarrow 3$ cross-linear structures;
   - $2 \rightarrow 7$ cross-linear structures.
2. Break KECCAK $[r = 240, c = 160, n_r = 3]$ Preimage Challenge
   - Complexity is 2^{45}.
3. Improved preimage attacks on 3-round KECCAK -256/ SHA3-256/ SHAKE256
   - Complexitiy of KECCAK -256: $2^{192}{}_{[1]} \longrightarrow 2^{150}$.

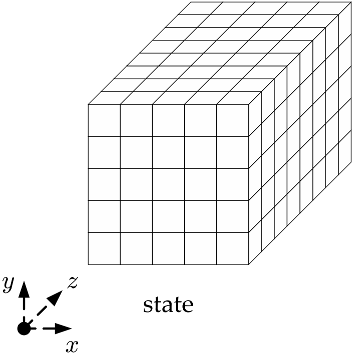
[1] J. Guo, M. Liu, and L. Song. Linear structures: Applications to cryptanalysis of round reduced KECCAK . *ASIACRYPT 2016.*

Preimage Attacks on the Round-reduced KECCAK with Cross-linear Structures

Sponge Construction



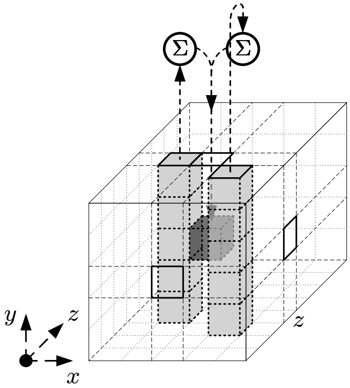
KECCAK-f permutations



- KECCAK-$f[b]$: 5×5 L-bits lanes, for $b = 400$, $L = 16$
- 24 rounds
- each round consists 5 operations:

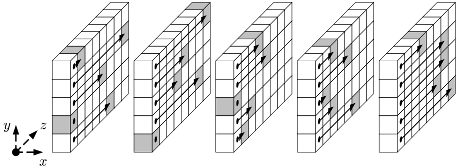
$$\iota \circ \chi \circ \pi \circ \rho \circ \theta,$$

KECCAK-f permutations



$$\theta : A[x, y, z] = A[x, y, z] \oplus \bigoplus_{j=0}^{4} (A[x-1, j, z] \oplus A[x+1, j, z-1]),$$

KECCAK -f permutations

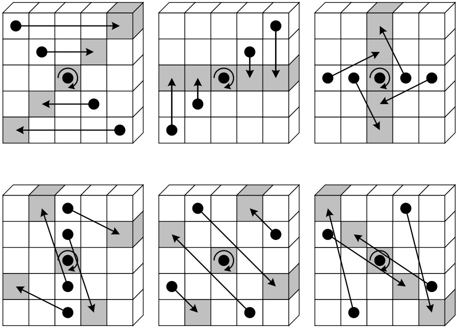


$$\rho : A[x, y, z] = A[x, y, (z + r[x, y])],$$

	x=3	x=4	x=0	x=1	x=2
y=2	153	231	3	10	171
y=1	55	276	36	300	6
y=0	28	91	0	1	190
y=4	120	78	210	66	253
y=3	21	136	105	45	15

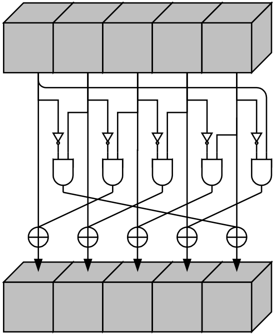
Table: The offsets of ρ.

KECCAK-f permutations



$$\pi : A[y, 2x+3y, z] = A[x, y, z],$$

KECCAK-f permutations



- $\chi : A[x,y,z] = A[x,y,z] \oplus ((\sim A[x+1,y,z])\&A[x+2,y,z])$,

  The only non-linear operation.
- $\iota : A[0,0,z] = A[0,0,z] \oplus RC[z]$. No impacts on preimage attacks.

KECCAK Crunchy Crypto Preimage and Collision Contest

- Reduced-round instances challenges by KECCAK Team.

KECCAK $[r = b - c, c = 160]$, $b \in \{200, 400, 800, 1600\}$, n_r=1,2,$\cdots$,12.

Summary of the results: The best preimage solution was on 4 rounds and was submitted by Meicheng Liu and Jian Guo in December 2016. The best collision was on 6 rounds and was submitted by Ling Song, Guohong Liao and again Jian Guo in February 2017. Remarkably, the smaller versions are harder to break. Although they have a smaller state, they offer much less degrees of freedom, especially relative to the capacity that is the same for all versions.

KECCAK Crunchy Crypto Preimage and Collision Contest

Preimage challenge status on Mar. 6, 2018, `https://keccak.team/crunchy_contest.html`

KECCAK[$r = 40, c = 160, n_r = 2$]	?	02 4a 55 18 e1 e9 5d b5 32 19
KECCAK[$r = 240, c = 160, n_r = 2$]	found by Paweł Morawiecki	7a b8 98 1a da 8f db 60 ae fd
KECCAK[$r = 640, c = 160, n_r = 2$]	found by Paweł Morawiecki	82 8d 4d 09 05 0e 06 35 07 5e
KECCAK[$r = 1440, c = 160, n_r = 2$]	found by Paweł Morawiecki	63 90 22 0e 7b 5d 32 84 d2 3e
KECCAK[$r = 40, c = 160, n_r = 3$]	?	d8 ed 85 69 2a fb ee 4c 99 ce
KECCAK[$r = 240, c = 160, n_r = 3$]	found by Yao Sun and Ting Li	5c 9d 5e 4b 38 5e 9c 4f 8e 2e
KECCAK[$r = 640, c = 160, n_r = 3$]	found by Jian Guo and Meicheng Liu	00 7b b5 c5 99 80 66 0e 02 93
KECCAK[$r = 1440, c = 160, n_r = 3$]	found by Jian Guo and Meicheng Liu	06 25 a3 46 28 c0 cf e7 6c 75
KECCAK[$r = 40, c = 160, n_r = 4$]	?	74 2c 7e 3c d9 46 1d 0d 03 4e
KECCAK[$r = 240, c = 160, n_r = 4$]	?	0d d2 5e 6d e2 9a 42 ad b3 58
KECCAK[$r = 640, c = 160, n_r = 4$]	?	75 1a 16 e5 e4 95 e1 e2 ff 22
KECCAK[$r = 1440, c = 160, n_r = 4$]	found by Meicheng Liu and Jian Guo	7d aa d8 07 f8 50 6c 9c 02 76

We found a preimage in 5 days with 8 GPU cards:

53 73 e0 75 3d ec af 5b 2e c1
00 00 00 00 00 00 00 00 00 00
53 73 e0 75 3d ec af 5b 2e c1.

1. Backgrounds
2. Specifications of KECCAK and KECCAK Crunchy Crypto Contest
3. Main Ideas and Cross-linear Structures
4. Preimage Attacks on Challenge KECCAK [r=240,c=160,n_r=3]
5. Preimage Attacks on KECCAK -256/SHA3-256/SHAKE-256

Main ideas

How to find a preimage via algebraic approach?

How to find a preimage via algebraic approach?

1. Construct non-linear systems and solve by Groebner basis?
   - High computation complexity;
   - Only deployed on CPU.

Main ideas

How to find a preimage via algebraic approach?

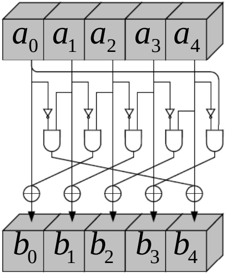
1. Construct non-linear systems and solve by Groebner basis?
   - High computation complexity;
   - Only deployed on CPU.
2. Totally linearizing the system by enumeration?
   - High complexity for enumeration.

Main ideas

How to find a preimage via algebraic approach?

1. Construct non-linear systems and solve by Groebner basis?
   - High computation complexity;
   - Only deployed on CPU.
2. Totally linearizing the system by enumeration?
   - High complexity for enumeration.
3. Partially linearize the system: Solving + Verification.
   - Balancing the computation and evaluation complexity;
   - Can be sped up by GPU.

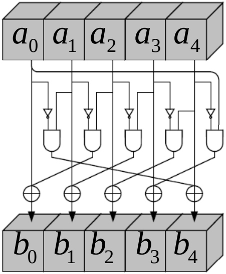
Linearize System from KECCAK



$$\chi \text{ step} : b_i = a_i + (a_{i+1} + 1) \cdot a_{i+2}$$

1. Linearizing all the 5 b_i's needs to know at least 3 values of a_i.
   - Freedom is not enough in some instances.

Linearize System from KECCAK



$$\chi \text{ step} : b_i = a_i + (a_{i+1} + 1) \cdot a_{i+2}$$

1. Linearizing all the 5 b_i's needs to know at least 3 values of a_i.
   - Freedom is not enough in some instances.
2. Linearize parts of b_i's $\longmapsto$ Cross-linear structures.

Cross-Linear Structures

A Cross-linear Structure

$$a \cdot b + \text{ Linear polynomail} = Constant,$$

$$c \cdot d + \text{ Linear polynomail} = Constant,$$

$$b \cdot e + \text{ Linear polynomail} = Constant,$$

where a, b, c, d, e are linear polynomials.

Cross-Linear Structures

A Cross-linear Structure

$$a \cdot b + \text{ Linear polynomail} = \textit{Constant},$$

$$c \cdot d + \text{ Linear polynomail} = \textit{Constant},$$

$$b \cdot e + \text{ Linear polynomail} = \textit{Constant},$$

where a, b, c, d, e are linear polynomials.

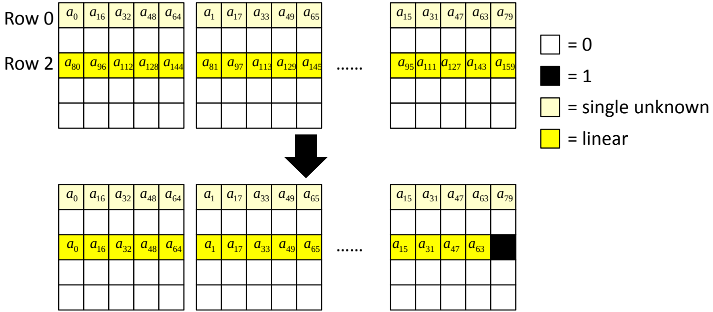
- Characteristics:
  - Each equation has only 1 product of linear polynomials.
  - b appears across different equations.
  - Guessing the value of b:

$$b = \textit{Constant},$$

$$a \cdot \textit{Constant} + \text{ Linear polynomail} = \textit{Constant},$$

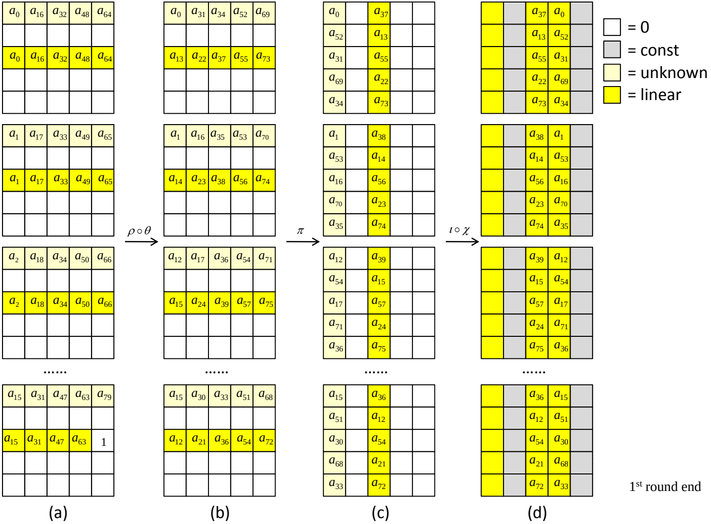
$$\textit{Constant} \cdot e + \text{ Linear polynomail} = \textit{Constant}.$$

Setting Initial Status

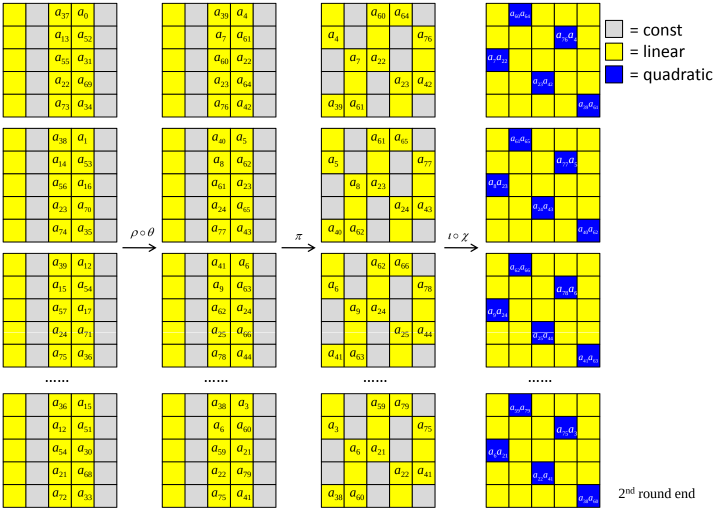


- Row 0 and Row 2 : variables.
- Row 1 : constants.
- To avoid the mixture of bits brought by the operation θ :
  - $a_{159} = 1$;
  - $a_i = a_{i+80} + c$, $c = a_{159} + a_{79}$.
- Each variable a_i appears in two places!

Status of the First Round

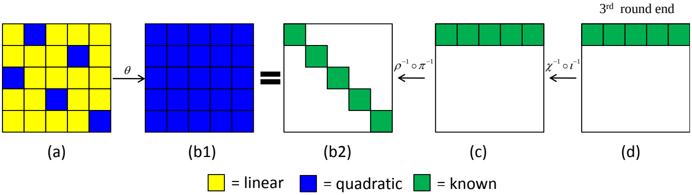


2nd Round—Guess the values of sums of Col 2 and 3



(a) (b) (c) (d)

3rd Round—Formulation of the system



$$
\begin{cases}
a_{32+[7+i]} \cdot a_{48+[13+i]} + a_{48+[11+i]} \cdot a_{64+[15+i]} + lin = c, \\
a_{[7+i]} \cdot a_{16+[6+i]} + a_{16+[6+i]} \cdot a_{32+[9+i]} + lin = c, \\
a_{48+[12+i]} \cdot a_{64+[i]} + a_{[3+i]} \cdot a_{64+[11+i]} + lin = c, \\
a_{16+[7+i]} \cdot a_{32+[10+i]} + a_{32+[6+i]} \cdot a_{48+[12+i]} + lin = c, \\
a_{[4+i]} \cdot a_{64+[12+i]} + a_{[6+i]} \cdot a_{16+[5+i]} + a_{32+[7+i]} \cdot a_{48+[13+i]} + lin = c.
\end{cases}
$$

where $[k]$: $k \mod L(=16)$, lin: linear polynomial, c: constant value.

Simplifying Quadratic Equations

Simplified by linear algebraic operations:

$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{16+[13+i]} \cdot a_{32+[i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{32+[12+i]} \cdot a_{48+[2+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{64+[15+i]} \cdot a_{[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{48+[5+i]} \cdot a_{64+[9+i]} + a_{48+[3+i]} \cdot a_{64+[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \qquad (1)
\end{aligned}
$$

Simplifying Quadratic Equations

Simplified by linear algebraic operations:

$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c,\\
a_{16+[13+i]} \cdot a_{32+[i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c,\\
a_{32+[12+i]} \cdot a_{48+[2+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c,\\
a_{64+[15+i]} \cdot a_{[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c,\\
a_{48+[5+i]} \cdot a_{64+[9+i]} + a_{48+[3+i]} \cdot a_{64+[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \qquad (1)
\end{aligned}
$$

Inter-reduce Equation (1) for $i = 0, 1, \cdots, 15$:

$$a_{48+[i]} \cdot a_{64+[4+i]} + lin = c. \qquad (2)$$

Simplifying Quadratic Equations

Simplified by linear algebraic operations:

$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{16+[13+i]} \cdot a_{32+[i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{32+[12+i]} \cdot a_{48+[2+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{64+[15+i]} \cdot a_{[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c, \\
a_{48+[5+i]} \cdot a_{64+[9+i]} + a_{48+[3+i]} \cdot a_{64+[7+i]} + a_{48+[i]} \cdot a_{64+[4+i]} + lin = c,
\end{aligned}
\tag{1}
$$

Inter-reduce Equation (1) for $i = 0, 1, \cdots, 15$:

$$a_{48+[i]} \cdot a_{64+[4+i]} + lin = c. \tag{2}$$

Substituting Equation (2) back:

$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + lin &= c, \\
a_{16+[13+i]} \cdot a_{32+[i]} + lin &= c, \\
a_{32+[12+i]} \cdot a_{48+[2+i]} + lin &= c, \\
a_{64+[15+i]} \cdot a_{[7+i]} + lin &= c, \\
a_{48+[i]} \cdot a_{64+[4+i]} + lin &= c.
\end{aligned}
$$

$1 \rightarrow 3$ Cross-linear Structures

$1 \rightarrow 3$ Cross-linear Structures

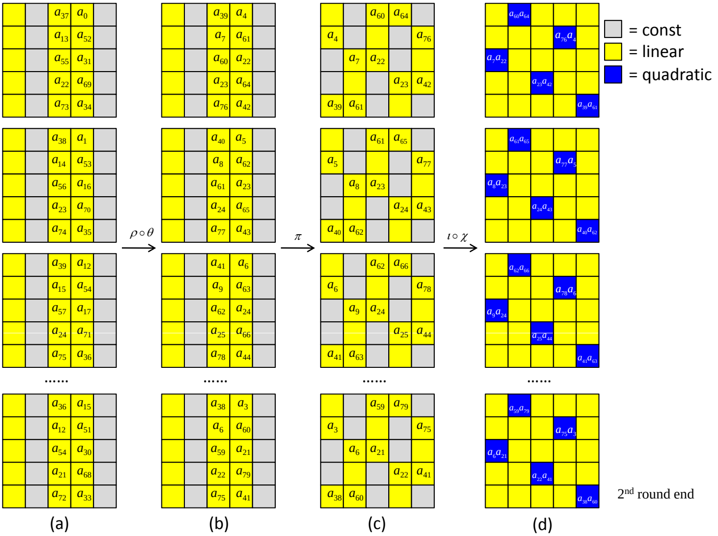
$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + lin &= c,\\
a_{16+[13+i]} \cdot a_{32+[i]} + lin &= c,\\
a_{32+[12+i]} \cdot a_{48+[2+i]} + lin &= c,\\
a_{64+[15+i]} \cdot a_{[7+i]} + lin &= c,\\
a_{48+[i]} \cdot a_{64+[4+i]} + lin &= c.
\end{aligned}
$$

- Each a_i (cross-linear factor) appears in 2 different equations.
- Guessing the value of *any* 1 cross-linear factor, we will obtain 3 linear equations.

Complexity Analysis on KECCAK[r=240,c=160,n_r=3]

- Guessing 48 values of linear polynomials.
- Solving linear system with 80 variables and 82 equations.
- Verifying on the other polynomials takes constant time.
- The preimage attack costs 2^{48} operations totally.

2nd Round—Guess the values of sums of Col 2 and 3



$2 \rightarrow 7$ Cross-linear Structures

Quadratic equations

$$
\begin{aligned}
\boxed{a_{[14+i]} \cdot a_{16+[13+i]}} + lin &= c,\\
\boxed{a_{16+[13+i]} \cdot a_{32+[i]}} + lin &= c,\\
a_{32+[12+i]} \cdot a_{48+[2+i]} + lin &= c,\\
a_{48+[i]} \cdot a_{64+[4+i]} + lin &= c,\\
a_{64+[15+i]} \cdot a_{[7+i]} + lin &= c.
\end{aligned}
$$

Linear equations obtained by constant-sums in 2nd Round

$$
\begin{aligned}
a_{[13+i]} + a_{16+[6+i]} + a_{32+[5+i]} + a_{48+[7+i]} + a_{64+[9+i]} &= c,\\
\boxed{a_{[i]} + a_{16+[15+i]} + a_{32+[2+i]}} + a_{48+[4+i]} + a_{64+[5+i]} &= c.
\end{aligned}
$$

$2 \to 7$ Cross-linear Structures

$$a_{[14+i]} + a_{16+[13+i]} + a_{32+[i]} + a_{48+[2+i]} + a_{64+[3+i]} = c, \quad (3)$$

$$a_{[14+i]} \cdot a_{16+[13+i]} + lin = c, \quad (4)$$

$$a_{16+[13+i]} \cdot a_{32+[i]} + lin = c, \quad (5)$$

Simplify (4) by (3):

$$a_{16+[13+i]} \cdot a_{32+[i]} + a_{16+[13+i]} \cdot (a_{48+[2+i]} + a_{64+[3+i]}) + lin = c.$$

Simplify by (5):

$$a_{16+[13+i]} \cdot (a_{48+[2+i]} + a_{64+[3+i]}) + lin = c.$$

$2 \to 7$ Cross-linear Structures

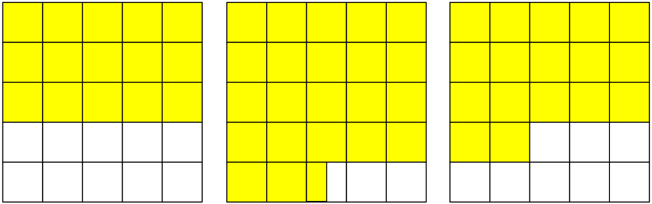
$2 \to 7$ Cross-linear Structures

$$
\begin{aligned}
a_{[14+i]} \cdot a_{16+[13+i]} + lin &= c, \\
a_{16+[13+i]} \cdot a_{32+[i]} + lin &= c, \\
a_{32+[12+i]} \cdot \boxed{a_{48+[2+i]}} + lin &= c, \\
\boxed{a_{48+[i]}} \cdot \boxed{a_{64+[4+i]}} + lin &= c, \\
\boxed{a_{64+[15+i]}} \cdot a_{[7+i]} + lin &= c, \\
a_{16+[13+i]} \cdot (\boxed{a_{48+[2+i]}} + \boxed{a_{64+[3+i]}}) + lin &= c.
\end{aligned}
$$

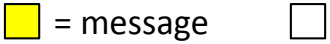
$2 \to 7$ Cross-linear Structures:

- Guessing $a_{48+[2+i]}$ and $a_{64+[3+i]}$, 7 linear equations are obtained.
- The complexity of preimage attack costs reduced to 2^{45} operations.

KECCAK-256

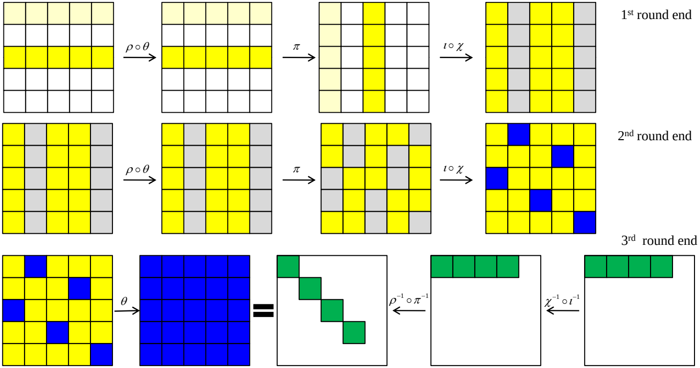


(a) KECCAK[r = 240, c = 160] (b) KECCAK[r = 1440, c = 160] (c) KECCAK[r = 1088, c = 512]



The initial statuses of KECCAK-256 and KECCAK[r=240,c=160] are quite similar.

KECCAK-256



- Output length : 256 bits (4 lanes).
- Number of original quadratic equations: $4 \times 64 = 256$;
- Each equation has 2 quadratic terms.

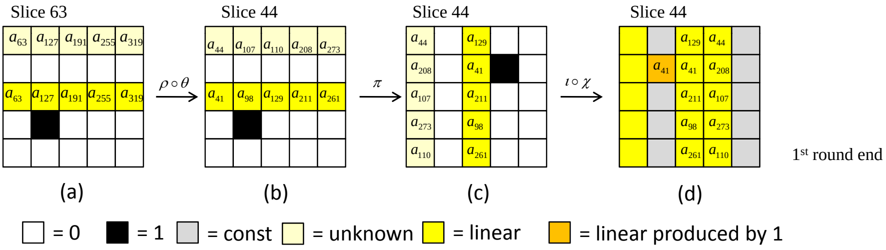
Original quadratic equations

$$\begin{aligned}
a_{128+[26+i]} \cdot a_{192+[48+i]} + a_{192+[62+i]} \cdot a_{256+[18+i]} + lin &= c,\\
a_{[58+i]} \cdot a_{64+[41+i]} + a_{64+[41+i]} \cdot a_{128+[12+i]} + lin &= c,\\
a_{192+[63+i]} \cdot a_{256+[19+i]} + a_{256+[30+i]} \cdot a_{[38+i]} + lin &= c,\\
a_{64+[42+i]} \cdot a_{128+[13+i]} + a_{128+[25+i]} \cdot a_{192+[47+i]} + lin &= c.
\end{aligned}$$

To construct 1 →3 cross-linear structures:

- Guessing $a_1, a_3, \cdots, a_{63}$.

How to deal with the Padding?



To avoid the quadratic term after first round by guessing a_{41}.

Enumerating 150 variables, we obtain:

- 320 variables;
- 322 linear equations;
- Complexity is 2^{150}, the previous known best attack costs 2^{192}.

SHA3-256/SHAKE256

Similar to the method of KECCAK-256, we reduce the complexity of preimage attack on SHA3-256/SHAKE256 to $2^{151}/2^{153}$.

Summary

1. Present two types of cross-linear structures.
   - $1 \rightarrow 3$ cross-linear structures;
   - $2 \rightarrow 7$ cross-linear structures.
2. Break KECCAK $[r = 240, c = 160, n_r = 3]$ Preimage Challenge
   - Complexity: 2^{48} by $1 \rightarrow 3$ cross-linear structures.
   - Complexity: 2^{45} by $2 \rightarrow 7$ cross-linear structures.
3. Reduce the complexity of preimage attacks on 3-round KECCAK -256/ SHA3-256/ SHAKE256 to $2^{150}/2^{151}/2^{153}$.

Thanks for your attention!