New Techniques for Searching Differential Trails in KECCAK

Guozhen Liu, Weidong Qiu, Yi Tu

Nanyang Technological University, Singapore
Shanghai Jiao Tong University, China

Overview

1. Introduction
   - Brief Description of KECCAK-f[1600]
   - Previous Works on Differential Trail Search
2. New 3-Round Trial Core Search Strategy
   - Classification of Search Space
   - Ideal Improvement Assumption
   - General Search Algorithm
   - Summary of Search Result

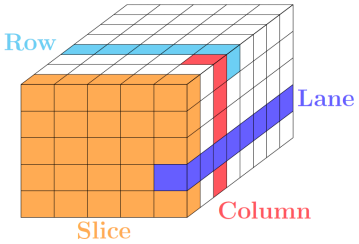
KECCAK-f[1600] - the SHA3 Permutation

KECCAK-f[1600] permutation uses XOR, AND and NOT operations in its round function.

- The state size is 1600 bits, organized as a 5×5 array of 64-bit lanes with (x, y, z) coordinates.
- Each round consists of 5 steps, i.e., the linear θ, ρ, π, ι operation, and the nonlinear χ.

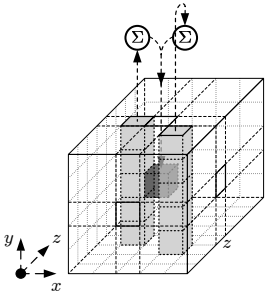
$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- 24 rounds.



Round Function of KECCAK-f[1600]

$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$

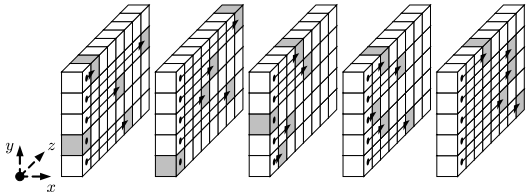


- θ step adds two columns to current bit position (x,y,z).
- column sum $c[x][z] = \bigoplus_{y=0}^{4} a[x][y][z]$

$$a[x][y][z] = c[x-1][z] \oplus a[x][y][z] \oplus c[x+1][z-1]$$

Round Function of KECCAK-f[1600]

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$



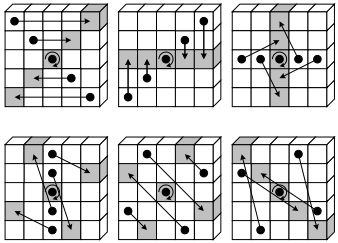
Rotation offsets $r[x, y]$

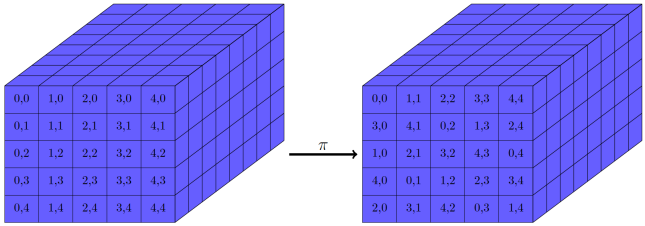
	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=0$	0	1	62	28	27
$y=1$	36	44	6	55	20
$y=2$	3	10	43	25	39
$y=3$	41	45	15	21	8
$y=4$	18	2	61	56	14

- ρ step: lane-level rotation. It rotates the 64 bits of each lane by a specific *offset*, which is determined by the coordinates [x,y] of the lane.

Round Function of KECCAK-f[1600]

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$





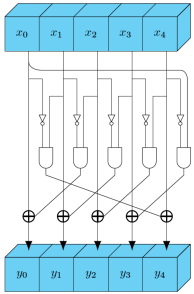
- π step: permutation on lanes. It rearranges the 25 bits of each slice.

$$a[y][2x + 3y][z] = a[x][y][z].$$

Round Function of KECCAK-f[1600]

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

$$\begin{aligned}
y_0 &= x_0 \oplus (x_1 \oplus 1) \cdot x_2 \\
y_1 &= x_1 \oplus (x_2 \oplus 1) \cdot x_3 \\
y_2 &= x_2 \oplus (x_3 \oplus 1) \cdot x_4 \\
y_3 &= x_3 \oplus (x_4 \oplus 1) \cdot x_0 \\
y_4 &= x_4 \oplus (x_0 \oplus 1) \cdot x_1
\end{aligned}$$



- χ is the only nonlinear component. It is a row wise 5-bit Sbox.

Round Function of KECCAK-f[1600]

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- ι step: add a round constant to the state
- Add a round-dependent constant to the first lane to destroy the symmetry.
- Since it has no effect on this kind of differential trail search, we ignore it.

Previous Results on Exhaustive Trail Search of KECCAK-f[1600]

Differential Propagation Analysis from [DVA12]

- 3-round trails with propagation weight below $T_3 = 36$ are searched completely.
- Lower bound of 6-round trails is 74.

New techniques for trail search [MDVA17]

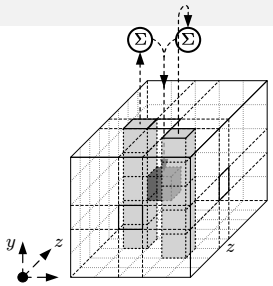
- 3-round trail cores with threshold propagation weight $T_3 = 45$ are searched exhaustively.
- Lower bound on propagation weight of 4/5/6-round trails are improved accordingly.

Our results

We set $T_3 = 53$ for our search strategy. There is no theoretical proof for a satisfactory lower bound, but we indeed found many new trail cores.

θ Property and 3-Round Trail Core

- **Column Parity p of state** α is the parity of all columns, i.e., $p = P(\alpha)$.
- **In CP Kernel and out CP Kernel.** If $p = 0$, θ has no effect on α, α is called in CP Kernel denoted as $|K|$, otherwise, it's out CP Kernel, denoted as $|N|$.
- We use **parity** and **Kernel** to represent column parity and column parity kernel.



3-round trail core

$$\beta_0 \dashrightarrow \alpha_1 \xrightarrow{\lambda} \beta_1 \xrightarrow{\chi} \alpha_2 \xrightarrow{\lambda} \beta_2$$

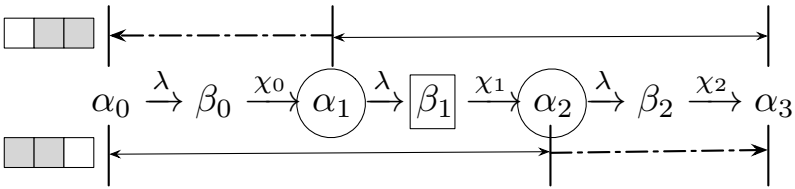
A 3-round trail core is denoted by (α_1, α_2) or (β_1, β_2).

Target 3-round trail cores

The 3-round trail core (β_1, β_2) with propagation weight $w^{rev}(\alpha_1)$[a] $+ w(\beta_1) + w(\beta_2) \leq T_3$.

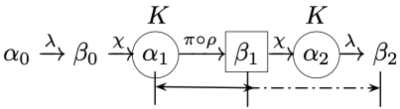
[a] $w^{rev}(\alpha_1)$ refers to the optimal weight of β_0 which can propagate to α_1

Classification of 3-Round Trail Core



1. According to whether α_1 and α_2 are **in Kernel**, 3-round trail cores can be classified into 4 categories.
   - $|K|K|$ trail cores, both α_1 and α_2 are in Kernel.
   - $|N|K|$ and $|N|N|$ trail cores, with always α_1 out Kernel. (In our work, trail cores with either of the features are covered by the same strategy.)
   - $|K|N|$ trail cores with only α_2 in Kernel.
2. For the last two cases, the search strategy are quite similar. But for $|N|K|$ and $|N|N|$ trails, the trail core search starts from out Kernel α_1, and from out Kernel α_2 for $|K|N|$ trails.

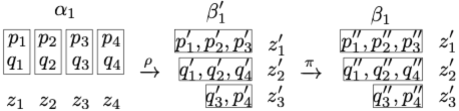
Search strategy for $|K|K|$ trail cores



1. First prepare all the theoretical candidate β_1 structures for in Kernel α_1 with m orbitals[1]. Store them in a look up table.
2. According to β_1 can propagate to α_1 which is in Kernel through $\lambda^{-1} = \rho^{-1} \circ \pi^{-1}$, construct the possible α_1
3. Based on the relationship between α_1 and β_1, filter α_1, and extend forward by one round to obtain the target three round trails

[1] A group of 2 active bits in the same column is called an *orbital*

An Example - $|K|K|$ Trail Search Algorithm



- 4 orbitals at α_1 propagate to 3 slices at β_1 with {3,3,2} pattern
- From the look up table, we enumerate all the possible valid slice for z'_1 to obtain p''_1, p''_2 and p''_3. Through $\lambda^{-1} = \theta^{-1} \circ \rho^{-1} \circ \pi^{-1}$, p_1, p_2, and p_3 are determined. Then q_1, q_2, q_3 can be enumerated according to the orbital relation.
- Through $\pi \circ \rho \circ \theta$, q''_3 is determined. According to the valid 2-bit slices stored in the look up table, p''_4 can be obtained, so p_4 is fixed, after that, q_4 can be enumerated according to the orbital relation.
- Until now, all the four orbitals with 8 bits are determined. Then we filter α_1 by checking q''_1, q''_2 and q''_4 are all at slice z'_2 or not and they result in in kernel slice at α_2 or not.
- Extend one round to get the target three round trail cores.

Parity-Bare State and Subspace

1. Enumerating out Kernel α
   - A group of out Kernel states α share the same parity p, i.e., each parity stands for a subspace of α, denoted by V_p.
   - Under each parity p, there are a group of states called *parity-bare states* that can represent all other states in V_p. Other states can be generated by adding **orbitals** to the *parity-bare states*.
   - Thus, out Kernel states in V_p can be covered by enumerating *parity-bare states*.
2. The space and subspace of out Kernel states
   - Any out Kernel state α can represent a set of states simply through adding **orbitals** to it. The subspace represented by α is denoted as V_α.
   - The space of out Kernel states can be divided into subspaces represented by out Kernel α, i.e.,

$$V_p := V_{\alpha_1} \bigcup V_{\alpha_2} \bigcup \cdots \bigcup V_{\alpha_n}$$

How to cover the search space

The search space is all the out Kernel states α.

The ideal representative of subspace V_α

For each subspace V_α of V_p, an ideal representative state α' is generated based on α.

- The *ideal representative* state generally does not exist in reality.
- It represents the optimal number of active rows of 3-round trail cores, of all states in V_p, indicating the lower bound of the whole subpace.
- Thus, if the ideal representative of a subspace cannot meet the weight requirement T_3, the whole subspace can be safely discarded.

Viability Check and Ideal Improvement Assumption

Ideal Improvement Assumption

The ideal improvement assumption on out Kernel states α assumes that

- for $|N|K|$ and $|N|N|$ trails, α_1 can be optimally improved at β_2 in terms of number of active rows with least number of orbitals added to it;
- for $|K|N|$ trails, α_2 can be optimally compensated with an in Kernel α_1 with the least number of orbitals added to it.

Basically, the ideal representative of subspace V_α is obtained when α is ideally improved.

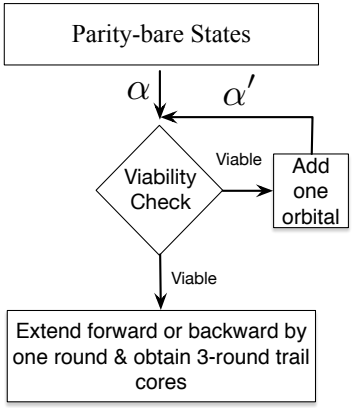
Viability Check

The process of generating the *ideal representative state* of a subspace and deciding whether to delete it is called **viability check**. The out Kernel state α that passes the viability check is called **viable**. Thus searching 3-round trail cores equals to generating all viable out Kernel states.

The Complete Process of 3-Round Trail Core Search

A general strategy to efficiently cover the search space:

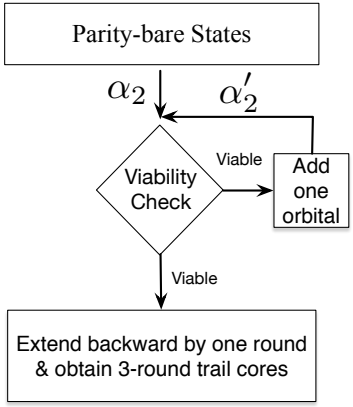
1. All candidate parities are prepared. For each candidate parity, the corresponding *parity-bare* states are enumerated.
2. For each *parity-bare* state, conduct viability check on it and generate the viable states.
3. For all viable α, add one orbital to it and conduct viability check on the newly generated α'. Repeat the process until there is no viable states anymore.
4. For all the viable α, extend them forward or backward and collect the target 3-round trail cores.



An Example - $|K|N|$ Trail Search Algorithm

$$\alpha_1 \xleftarrow{\lambda^{-1}} \beta_1 \xleftarrow{\chi^{-1}} \alpha_2 \xrightarrow{\lambda} \beta_2$$

1. The search starts from out Kernel α_2.
2. The ideal improvement assumption states that
   - for each active rows at α_2, rather than consider only the compatible β_1, it assumes all the 31 patterns are legal;
   - for any α_2, with the superset of β_1, it assumes that α_2 always have in Kernel α_1. If its original active rows cannot make α_1 in Kernel, it can be improved to be in Kernel by adding orbitals to α_2;
   - when adding orbitals to α_2, it assumes the least number of row increase on α_2 and β_2.
3. Conduct the viability check and add one orbital to viable α_2. Repeat the process on viable α_2.
4. Extend all the collected viable α_2 backward to in Kernel α_1 by one round.



Brief Summary of Result

	$\|K\|K\|$	$\|N\|K\|$	$\|N\|N\|$	$\|K\|N\|$
T_3 in [DVA12]	40	36	36	36
T_3 in [MDVA17]	45	45	45	45
our T_3	53	53	53	53
Minimal Weight	35	46	48	32
Time Complexity	2^{42}	2^{40}	2^{40}	2^{45}

Thanks for your attention!

References


Joan Daemen and Gilles Van Assche.
Differential propagation analysis of keccak.
In *Fast Software Encryption*, pages 422–441. Springer, 2012.


Silvia Mella, Joan Daemen, and Gilles Van Assche.
New techniques for trail bounds and application to differential trails in keccak.
IACR Transactions on Symmetric Cryptology, 2017(1):329–357, 2017.