Unaligned Rebound Attack for KECCAK

Thomas Peyrin - NTU

joint work with **Alexandre Duc**, **Jian Guo** and **Lei Wei**

Workshop on Symmetric Cryptanalysis

Microsoft Research - Redmond, USA

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

Outline

Current status of the SHA−3 competition

In december 2010, the NIST announced the five SHA−3 **finalists:**
Blake, Grøstl, JH, KECCAK, Skein.

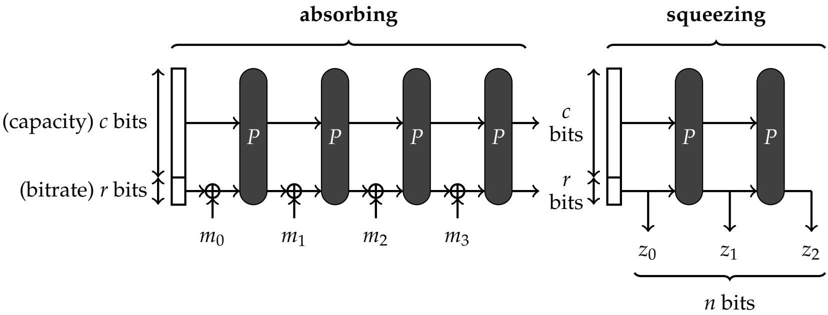
So far, none of them broken. It is very unlikely that this happens before the selection of the winner. So in order to compare their security, the cryptanalysts look for

* **"easier" attack models**:
  - near collisions
  - distinguishers (zero-sums, subspace, limited-birthday)
  - etc ...

* **reduced variants**:
  - lower number of rounds
  - only some internal function of the whole hash
  - etc ...

Here we will be analyzing **the reduced-round** KECCAK **internal permutations** in regards to **differential distinguishers**.

Orginial sponge functions [Bertoni et al. 2007]



A sponge function has been proven to be indifferentiable from a random oracle up to $2^{c/2}$ calls to the internal permutation P. However, **the best known generic attacks have the following complexity:**

- **Collision:** $\min\{2^{n/2}, 2^{c/2}\}$
- **Second-preimage:** $\min\{2^{n}, 2^{c/2}\}$
- **Preimage:** $\min\{2^{n}, 2^{c}, \max\{2^{n-r}, 2^{c/2}\}\}$

Previous cryptanalysis results on KECCAK

So far, the results on KECCAK [B+08]:

- **J.-P. Aumasson *et al.* (2009)**:
  zero-sum distinguishers up to 16 rounds of KECCAK-1600 internal permutation with complexity 2^{1024}.
- **P. Morawiecki and M. Srebrny (2010)**:
  small messages preimage attack using SAT solvers, up to 3 rounds.
- **D. Bernstein (2010)**:
  a (second)-preimage attack on 8 rounds with complexity $2^{511.5}$ and 2^{508} bits of memory.
- **C. Boura *et al.* (2010-2011)**:
  zero-sum partitions distinguishers to the full 24-round version of KECCAK-1600 internal permutation with complexity 2^{1590}.

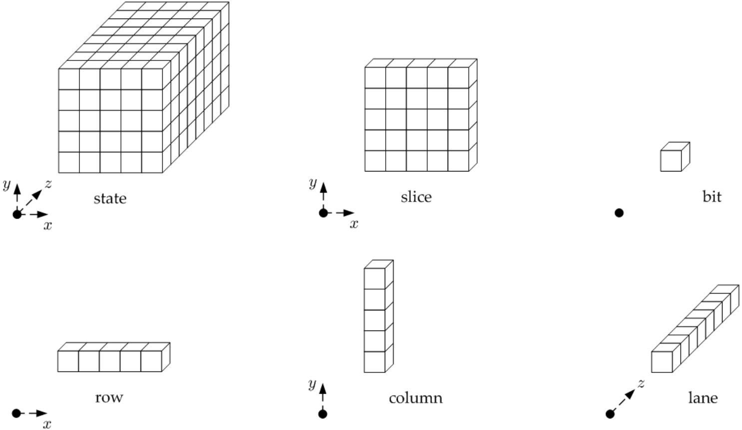
Previous cryptanalysis results on KECCAK

Motivation:

- the **zero-sum distinguishers** proposed can attack more rounds (or the same number of rounds with better complexity) than the distinguishers we will present here. However:
  - **their advantage to the generic complexity is very small** (always a factor about 2), while in our case the gap will be huge
  - zero-sums are **difficult to exploit** in order to get collisions for example, while in our case we use differential properties
  - **zero-sums partitions descriptions are in fact huge** without using full KECCAK rounds in the descriptions
- because it is difficult to apply on KECCAK, there is **no "differential analysis" provided by a third party yet**.
- **we focus on attacks with a complexity lower than $2^{b/2}$**

The KECCAK internal state

The b-bit **internal state of** KECCAK can be viewed as a **rectangular cuboid of** $5 \times 5 \times w$ **bits**.



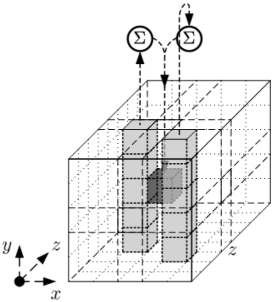
The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

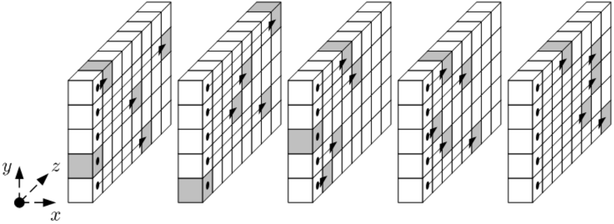
- **θ: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)**



The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

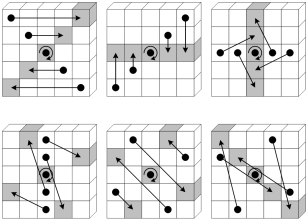
- $\boldsymbol{\theta}$: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)
- **$\boldsymbol{\rho}$: linear mapping that provides diffusion between the slices of the state through intra-lane bit translations**



The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

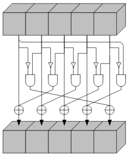
- $\boldsymbol{\theta}$: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)
- $\boldsymbol{\rho}$: linear mapping that provides diffusion between the slices of the state through intra-lane bit translations
- **$\boldsymbol{\pi}$: linear mapping that provides diffusion in the state through transposition of the lanes.**



The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

- $\boldsymbol{\theta}$: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)
- $\boldsymbol{\rho}$: linear mapping that provides diffusion between the slices of the state through intra-lane bit translations
- $\boldsymbol{\pi}$: linear mapping that provides diffusion in the state through transposition of the lanes.
- $\boldsymbol{\chi}$**: non-linear mapping similar to** $s = 5w$ **Sboxes applied independently to each 5-bit row of the state**



The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

- $\boldsymbol{\theta}$: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)
- $\boldsymbol{\rho}$: linear mapping that provides diffusion between the slices of the state through intra-lane bit translations
- $\boldsymbol{\pi}$: linear mapping that provides diffusion in the state through transposition of the lanes.
- $\boldsymbol{\chi}$: non-linear mapping similar to $s = 5w$ Sboxes applied independently to each 5-bit row of the state
- $\boldsymbol{\iota}$**: adds round-dependant constants to the lane** $a[0][0][\cdot]$**. We can forget about this layer since completely transparent in terms of differential paths.**

The KECCAK internal permutation

The b-bit KECCAK **internal permutation** P applies R rounds (for $b = 1600$ we have $R = 24$ rounds), each composed of the five following layers:

- $\boldsymbol{\theta}$: linear mapping that provides diffusion for the state (the xor of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$ is xored to the bit $a[x][y][z]$)
- $\boldsymbol{\rho}$: linear mapping that provides diffusion between the slices of the state through intra-lane bit translations
- $\boldsymbol{\pi}$: linear mapping that provides diffusion in the state through transposition of the lanes.
- $\boldsymbol{\chi}$: non-linear mapping similar to $s = 5w$ Sboxes applied independently to each 5-bit row of the state
- $\boldsymbol{\iota}$: adds round-dependant constants to the lane $a[0][0][\cdot]$. We can forget about this layer since completely transparent in terms of differential paths.

One round is now composed of:

- **a linear layer $\lambda = \pi \circ \rho \circ \theta$**
- **a non-linear Sbox layer χ**

Outline

Introduction

Building differential paths for KECCAK

The rebound attack

The unaligned rebound attack for KECCAK

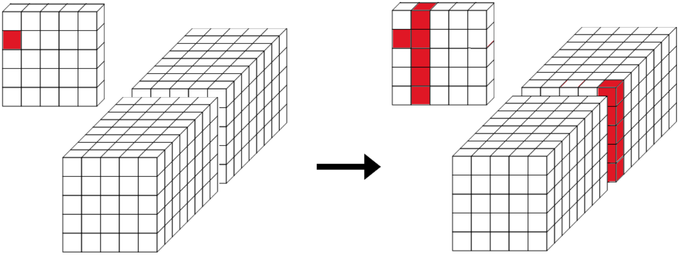
Results and future works

The diffusion in KECCAK

Diffusion in KECCAK mostly provided by θ, since:

- π and ρ layers only change bit positions
- diffusion of the Sboxes in χ layer is very small.

Good diffusion of θ:

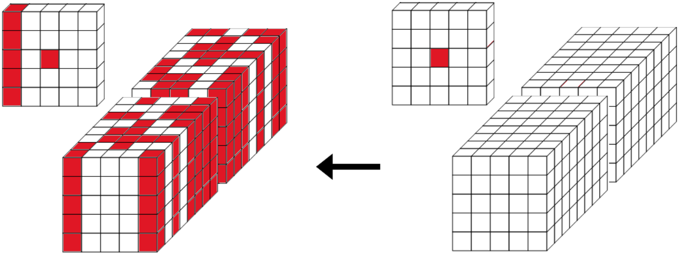


The diffusion in KECCAK

Diffusion in KECCAK mostly provided by θ, since:

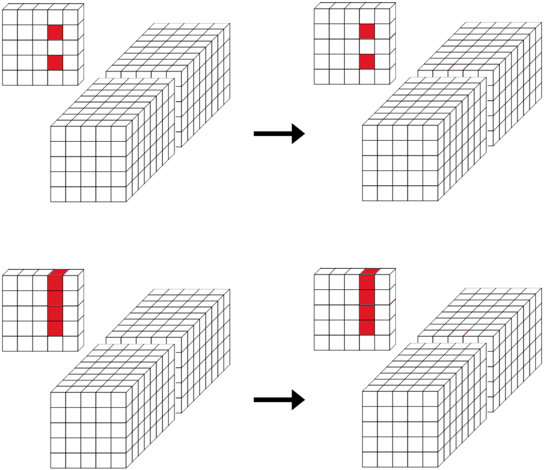
- π and ρ layers only change bit positions
- diffusion of the Sboxes in χ layer is very small.

Excellent diffusion of θ^{-1}:



The column parity kernel for θ

An even number of active bits gives no diffusion through θ (column parity kernel, CPK):



The differential path search for KECCAK

Our goal is of course to **minimize as much as possible the effect of the diffusion**. When looking for a bitwise differential path, the branching in the search only comes from χ (for a given input, all valid transitions have the same success probability through the Sbox).

The core algorithm is simple:

- **Precomputation:** for every possible slice input difference, we precompute and store the best differential transitions through χ, i.e. the ones that will minimize the diffusion through the next θ (favor CPK, low Hamming weight).
- **Keep repeating:**
  - start with a difference in a_1 composed of only k CPK, with k small
  - compute forward by choosing random candidates among the best slice transitions
  - if the current path tested is good, compute one round backward (about $2k$ active sboxes)

$$a_0 \xleftarrow{\lambda^{-1}} b_0 \xleftarrow{\chi^{-1}} \mathbf{a_1} \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda} b_3 \cdots$$

Differential paths results on KECCAK

Table: Best differential path results for each version of KECCAK internal permutations, for 1 up to 5 rounds (red = new results).

b	best differential path probability		
	1 rd	2 rds	3 rds
100	2^{-2} (2)	2^{-8} (4 - 4)	2^{-19} (4 - 8 - 7)
200	2^{-2} (2)	2^{-8} (4 - 4)	2^{-20} (4 - 8 - 8)
400	2^{-2} (2)	2^{-8} (4 - 4)	2^{-24} (8 - 8 - 8)
800	2^{-2} (2)	2^{-8} (4 - 4)	**2^{-32}** (4 - 4 - 24)
1600	2^{-2} (2)	2^{-8} (4 - 4)	**2^{-32}** (4 - 4 - 24)

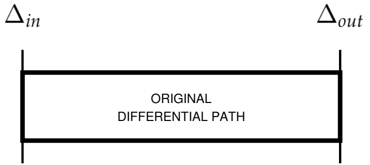
b	best differential path probability	
	4 rds	5 rds
100	2^{-30} (4 - 8 - 10 - 8)	2^{-54} (4 - 8 - 10 - 8 - 24)
200	2^{-46} (11 - 9 - 8 - 8)	**2^{-121}** (20 - 16 - 22 - 22 - 41)
400	**2^{-84}** (16 - 14 - 12 - 42)	**2^{-245}** (16 - 14 - 12 - 42 - 161)
800	**2^{-109}** (12 - 12 - 12 - 73)	**2^{-459}** (12 - 12 - 12 - 73 - 350)
1600	**2^{-142}** (12 - 12 - 12 - 106)	**2^{-709}** (16 - 16 - 16 - 114 - 547)

Simple distinguishers

Obvious distinguisher:
for a differential path $\Delta_{in} \leftrightarrow \Delta_{out}$ with success probability $P > 2^{-b}$ (the generic algorithm finds such a pair with complexity 2^b)

Use the freedom degrees (+1 round):
add an extra round for free to the left (or to the right) by fixing the Sboxes values for this round. Same overall complexity (same generic complexity)

Add an extra round to the left and to the right (+2 rounds):
without controlling the new differential transitions (i.e. same complexity). This will increase the amount of reacheable input and output differences (from 1 to IN and 1 to OUT) and therefore reduce the generic complexity (limited-birthday distinguishers [GP10]): $\max\{\sqrt{2^b/IN}, \sqrt{2^b/OUT}, 2^b/(IN \cdot OUT)\}$

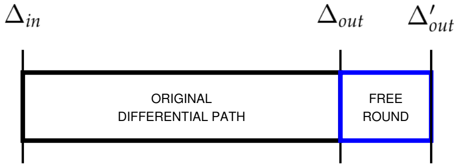


Simple distinguishers

Obvious distinguisher:
for a differential path $\Delta_{in} \leftrightarrow \Delta_{out}$ with success probability $P > 2^{-b}$ (the generic algorithm finds such a pair with complexity 2^b)

Use the freedom degrees (+1 round):
add an extra round for free to the left (or to the right) by fixing the Sboxes values for this round. Same overall complexity (same generic complexity)

Add an extra round to the left and to the right (+2 rounds):
without controlling the new differential transitions (i.e. same complexity). This will increase the amount of reacheable input and output differences (from 1 to IN and 1 to OUT) and therefore reduce the generic complexity (limited-birthday distinguishers [GP10]): $\max\{\sqrt{2^b/IN}, \sqrt{2^b/OUT}, 2^b/(IN \cdot OUT)\}$

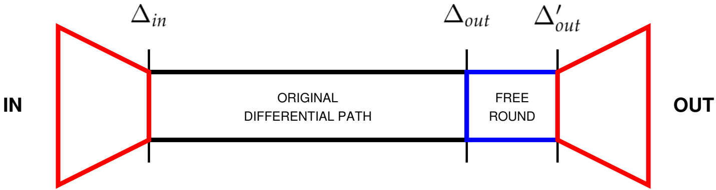


Simple distinguishers

Obvious distinguisher:
for a differential path $\Delta_{in} \leftrightarrow \Delta_{out}$ with success probability $P > 2^{-b}$ (the generic algorithm finds such a pair with complexity 2^b)

Use the freedom degrees (+1 round):
add an extra round for free to the left (or to the right) by fixing the Sboxes values for this round. Same overall complexity (same generic complexity)

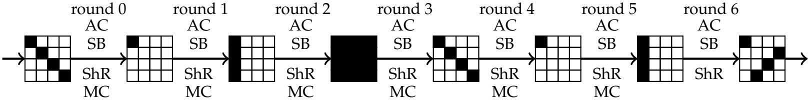
Add an extra round to the left and to the right (+2 rounds):
without controlling the new differential transitions (i.e. same complexity). This will increase the amount of reacheable input and output differences (from 1 to *IN* and 1 to *OUT*) and therefore reduce the generic complexity (limited-birthday distinguishers [GP10]): $\max\{\sqrt{2^b/IN}, \sqrt{2^b/OUT}, 2^b/(IN \cdot OUT)\}$



Outline

Rebound attack and improvements

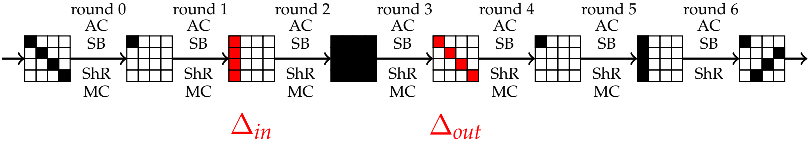
The **rebound attack** [M+09] (example with AES-like permutation):



Rebound attack and improvements

The **rebound attack** [M+09] (example with AES-like permutation):

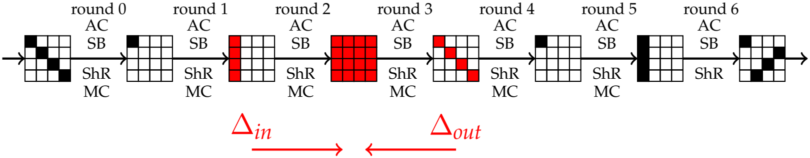
- **step 1:** choose input difference Δ_{in} and output difference Δ_{out} of the inbound phase ...



Rebound attack and improvements

The **rebound attack** [M+09] (example with AES-like permutation):

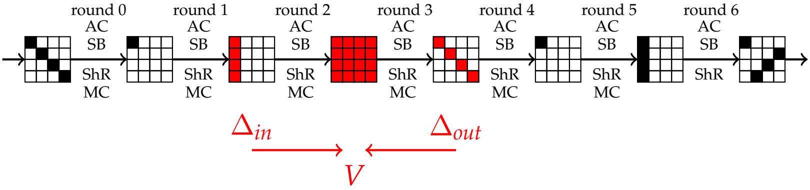
- **step 1:** choose input difference Δ_{in} and output difference Δ_{out} of the inbound phase ...
- **step 2:** ...and propagate those **differences** forward and backward up to the middle layer of Sboxes, until reaching a differential match (with probability p_{match})



Rebound attack and improvements

The **rebound attack** [M+09] (example with AES-like permutation):

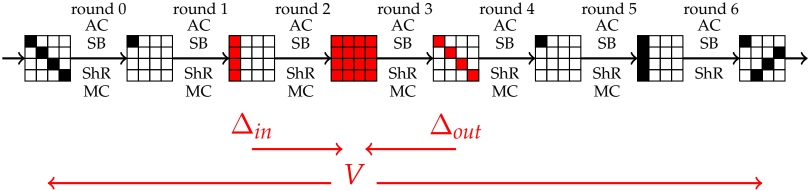
- **step 1:** choose input difference Δ_{in} and output difference Δ_{out} of the inbound phase ...
- **step 2:** ...and propagate those **differences** forward and backward up to the middle layer of Sboxes, until reaching a differential match (with probability p_{match})
- **step 3:** once a differential match obtained, deduce and generate all the N_{match} valid Sbox **values** V



Rebound attack and improvements

The **rebound attack** [M+09] (example with AES-like permutation):

- **step 1:** choose input difference Δ_{in} and output difference Δ_{out} of the inbound phase ...
- **step 2:** ...and propagate those **differences** forward and backward up to the middle layer of Sboxes, until reaching a differential match (with probability p_{match})
- **step 3:** once a differential match obtained, deduce and generate all the N_{match} valid Sbox **values** V
- **step 4:** propagate the **values and differences** forward and backward and check if the differential path is entirely verified (with probability p_F and p_B)



Complexity and improvements

The **overall complexity** is $\frac{1}{p_{\text{match}}} \cdot \left\lceil \frac{1}{p_F \cdot p_B \cdot N_{\text{match}}} \right\rceil + \frac{1}{p_B \cdot p_F}$, since:

- we need to start with a least p_{match}^{-1} pairs of differences for the inbound before finding a differential match in the middle
- we need to generate at least $p_B^{-1} \cdot p_F^{-1}$ valid inbound values in order to find a solution for the entire path

Some **improvements** exist:

- **Super-Sbox [L+09,GP10]:** merge two rounds in the middle in order to build a layer of bigger Sboxes (gain of one round)
- **Non-full active [S+10]:** do no necessarily use a full active state in the middle (lower complexity)

Why rebound is hard on KECCAK?

Our goal: take the best differential path on x rounds of KECCAK, and merge it using the rebound to create a $(2x+1)$-round one (we hope for 9 rounds at max for a complexity $< 2^{512}$).

But there are **many problems** for KECCAK:

- there is (by far !) **not enough differential paths** with good probability
- **the differential match probability of the KECCAK Sbox depends on the input and output difference mask** (see its DDT) ...
- ... but fortunately the distribution of output difference probabilities is the same when the **input difference hamming weight** is fixed

Moreover, **the improvements will not apply**:

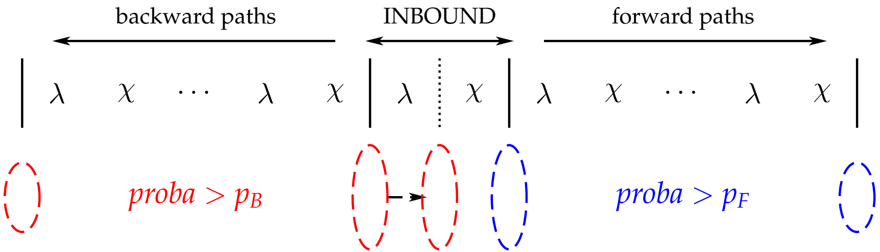
- **alignement in KECCAK is bad** (see designers recent article at ECRYPT HASH3), thus the Super-Sbox improvement cannot be used
- we will see later that it is very hard to build non-full active differential paths using rebound technique

The KECCAK Sbox DDT

Δ_{in} \ Δ_{out}	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00	32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
01	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-
02	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-
03	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-
04	-	-	-	-	8	8	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
05	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4
06	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-
07	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2
08	-	-	-	-	-	-	-	-	8	-	8	-	8	-	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
09	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4
0A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4
0B	-	4	4	-	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	4	4	-
0C	-	-	-	-	-	-	-	-	4	4	4	4	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0D	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-
0E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2
0F	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-
11	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-
12	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4
13	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	-	-	4	4	4	4	-	-	-	-	4	4
15	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-
16	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-
17	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-
19	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2
1A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-
1B	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-
1C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1D	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-
1E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-
1F	-	2	2	-	2	-	-	2	2	-	-	2	-	2	2	-	2	-	-	2	-	2	2	-	-	2	2	-	2	-	-	2

Outline

Our roadmap

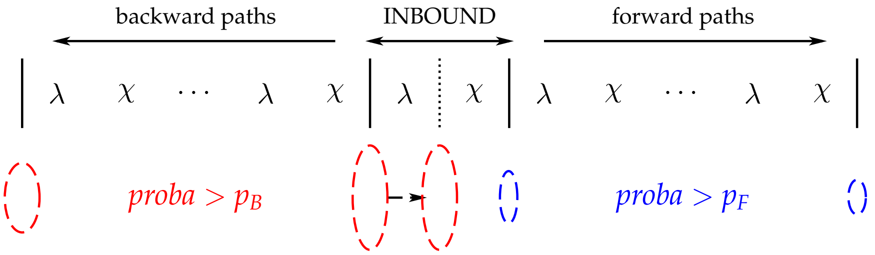


We consider an inbound composed of one KECCAK round

Due to the very good diffusion of θ^{-1}, **the amount of forward paths will be small**. In order to have a chance to find at least one match for the inbound, **we will need a lot of backward paths**

In the following, we will focus on the case KECCAK-1600 but our framework allows to apply the unaligned rebound attack on any version.

Our roadmap



We consider an inbound composed of one KECCAK round

Due to the very good diffusion of θ^{-1}, **the amount of forward paths will be small**. In order to have a chance to find at least one match for the inbound, **we will need a lot of backward paths**

In the following, we will focus on the case KECCAK-1600 but our framework allows to apply the unaligned rebound attack on any version.

Balls and bucket problem

In order for a differential match to happen during the inbound, we first need the exact same set of Sboxes to be active forward and backward.

We modeled this with a **limited capacity balls and buckets problem**:

Theorem

Given a set B of s buckets of capacity 5 in which we throw x_B balls and a set F of s buckets of capacity 5 in which we throw x_F balls, the probability that B and F have the same pattern of empty buckets is given by

$$p_{pattern}(s, x_B, x_F) = \frac{1}{\binom{5s}{x_B}\binom{5s}{x_F}} \sum_{i=0}^{s} b_{\text{bucket}}(x_B, s-i) b_{\text{bucket}}(x_F, s-i) \binom{s}{i} ,$$

where $b_{\text{bucket}}(x, s) = \sum_{i=\lceil n/5 \rceil}^{s} (-1)^i \binom{s}{i}\binom{5i}{n}$ *if* $s \leq n \leq 5s$ *and 0 otherwise. The average number* $n_{pattern}$ *of non-empty buckets if both experiments results follow the same pattern is given by*

$$n_{pattern}(s, x_B, x_F) = \frac{\sum_{i=0}^{s} b_{\text{bucket}}(x_B, s-i) b_{\text{bucket}}(x_F, s-i) \binom{s}{i} (s-i)}{\sum_{i=0}^{s} b_{\text{bucket}}(x_B, s-i) b_{\text{bucket}}(x_F, s-i) \binom{s}{i}} .$$

Balls and bucket problem

In order for a differential match to happen during the inbound, we first need the exact same set of Sboxes to be active forward and backward.

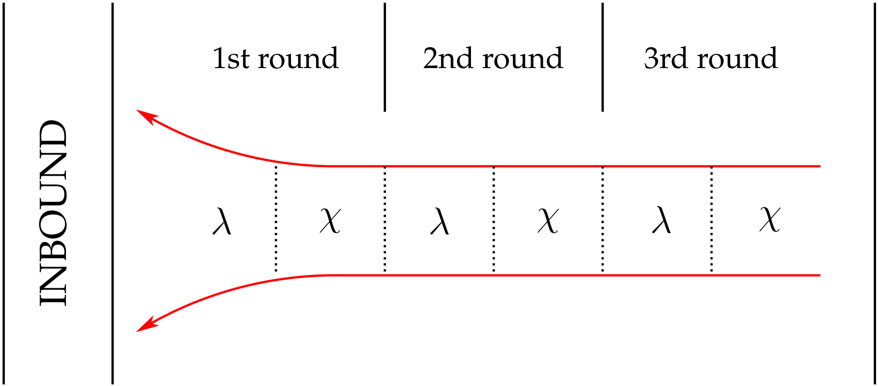
We modeled this with a **limited capacity balls and buckets problem**:

Theorem

Conclusion: for our range of difference bit Hamming weights (not too small) on the input and output of the inbound

- it is very likely that a match on the active Sboxes pattern happens ($p_{pattern}$ is high)
- when it happens, it is very likely that **all sboxes are active** ($n_{pattern} = s$).

The forward paths

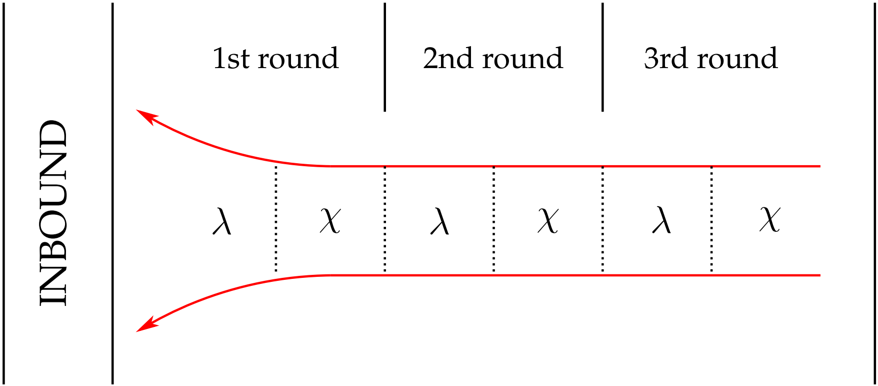


Active bits

log2 proba

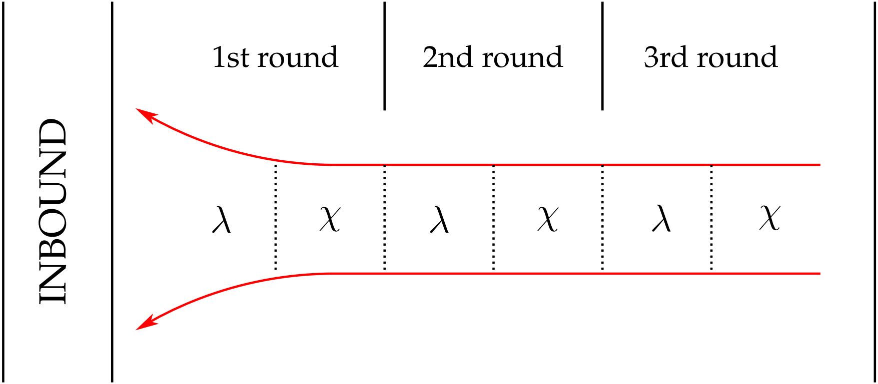
Number of paths

The forward paths



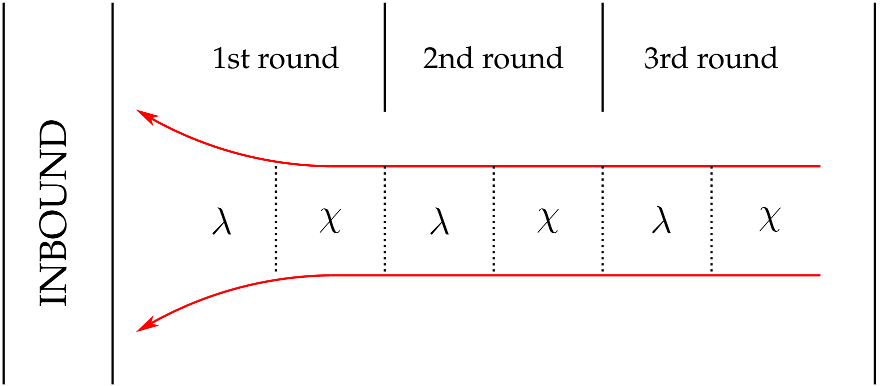
Active bits		$6 \leftarrow 6$		$6 \leftarrow 6$		$6 \rightarrow 6$	
log2 proba		-12		-12		-12	
Number of paths	2^6		2^6		2^6		2^6

The forward paths



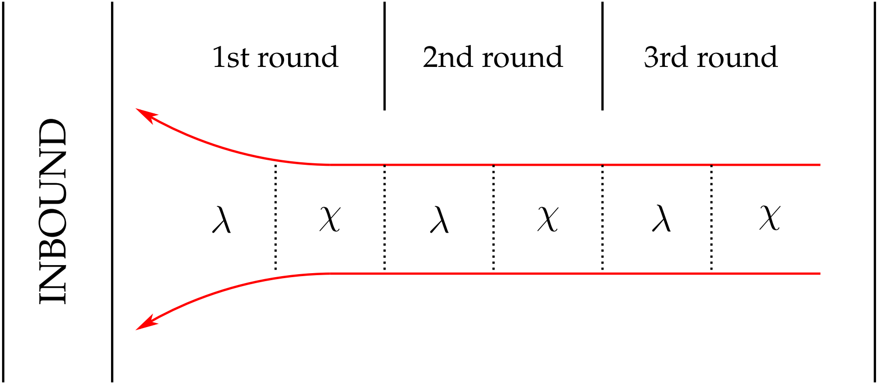
Active bits		$6 \leftarrow 6$		$6 \leftarrow 6$		$6 \rightarrow *$	
log2 proba		-12		-12		0	
Number of paths	2^6		2^6		2^6		2^{18}

The forward paths



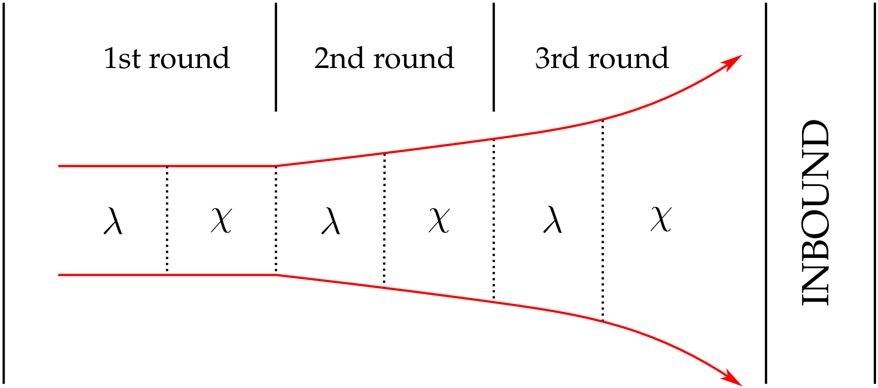
Active bits		$* \leftarrow 6$		$6 \leftarrow 6$		$6 \rightarrow *$	
log2 proba		[-24,-12]		-12		0	
Number of paths	2^{25}		2^{6}		2^{6}		2^{18}

The forward paths



Active bits	320 act. sboxes		$* \leftarrow 6$		$6 \leftarrow 6$		$6 \rightarrow *$	
log2 proba			[-24,-12]		-12		0	
Number of paths	$2^{23.3}$	2^{25}		2^{6}		2^{6}		2^{18}

The backward paths

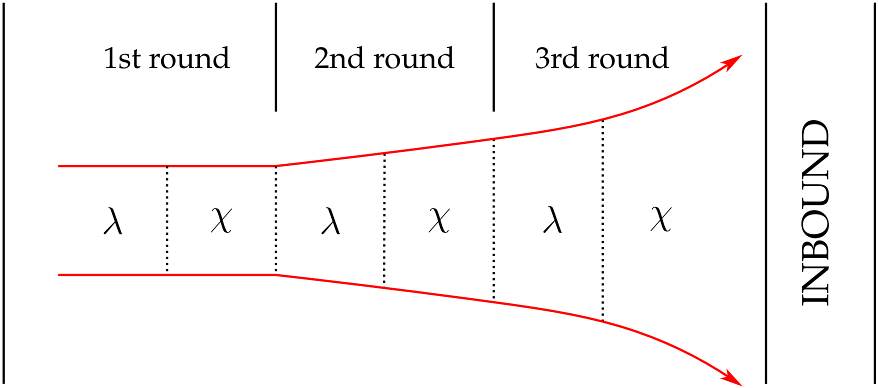


Active bits

log2 proba

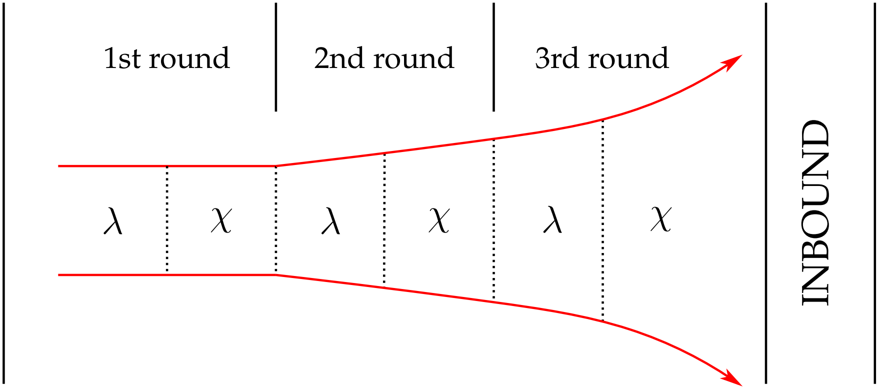
Number of paths

The backward paths



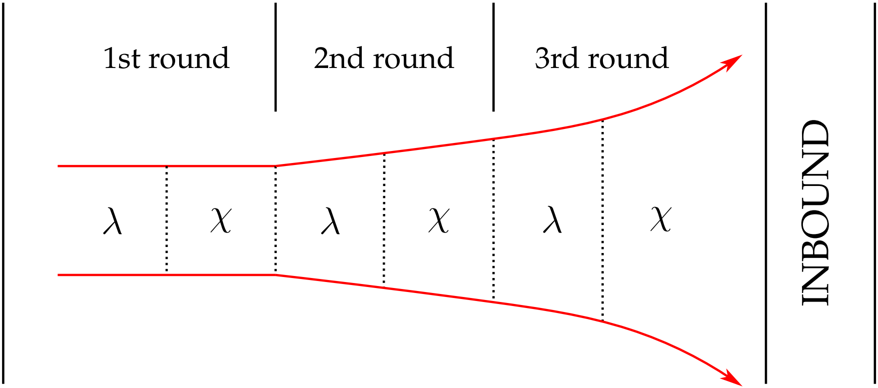
	$16 \leftarrow 16$		$16 \rightarrow 16$		**Active bits**
	-32		-32		**log2 proba**
$2^{77.7}$		$2^{77.7}$		$2^{77.7}$	**Number of paths**

The backward paths



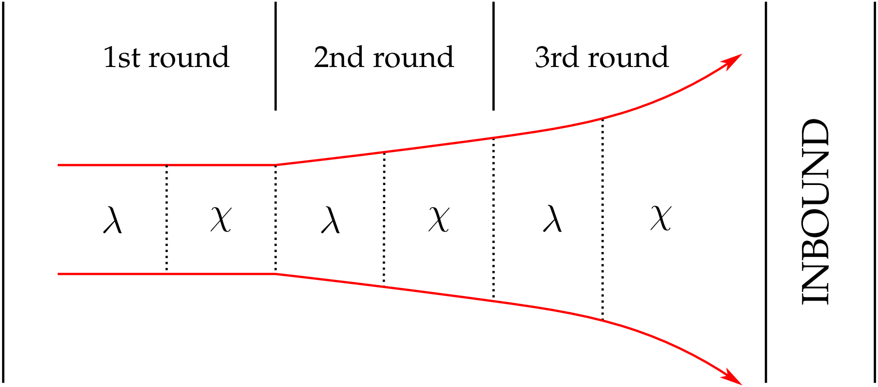
	$* \leftarrow 16$	$16 \rightarrow 16$		**Active bits**
	0	-32		**log2 proba**
$\leq 2^{128.4}$		$2^{77.7}$	$2^{77.7}$	**Number of paths**

The backward paths



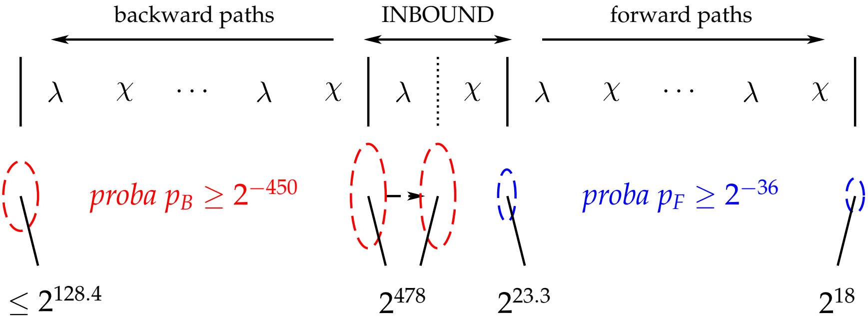
		$* \leftarrow 16$		$16 \rightarrow 24$		**Active bits**
		0		-32		**log2 proba**
$\leq 2^{128.4}$			$2^{77.7}$		$2^{99.4}$	**Number of paths**

The backward paths



Active bits		$* \leftarrow 16$		$16 \rightarrow 24$		$*$	
log2 proba		0		-32		≥ -418	
Number of paths	$\leq 2^{128.4}$		$2^{77.7}$		$2^{99.4}$		2^{478}

Overall complexity

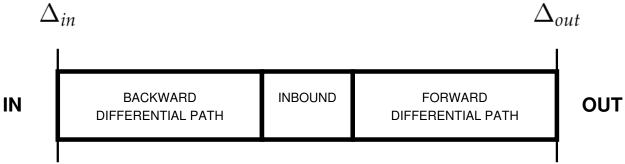


The **differential matching probability** is $p_{\mathsf{match}} = 2^{-491.5}$

The **number of solutions obtained per match** is $N_{\mathsf{match}} = 2^{486.8}$

The total complexity is $2^{491.5}$ computations

Distinguishers on KECCAK-1600 permutation



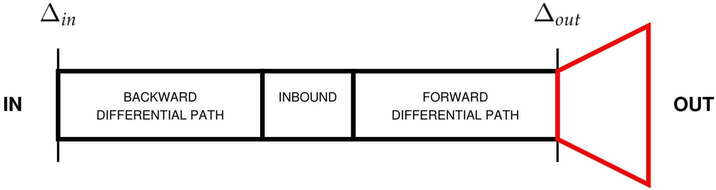
Limited birthday problem on a 1600-bit permutation with:

- $|IN| \leq 2^{128.4}$
- $|OUT| = 2^{18}$

We have a **generic complexity** of $2^{1453.6} > 2^{491.5}$ computations.

⇒ 7 rounds can be distinguished

Distinguishers on KECCAK-1600 permutation



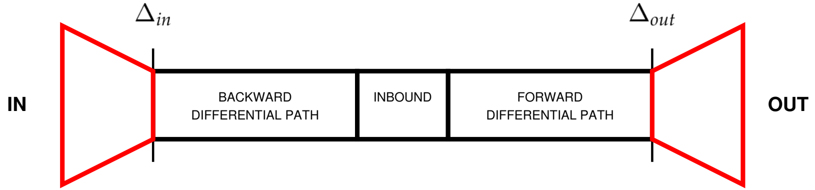
Limited birthday problem on a 1600-bit permutation with:

- $|IN| \leq 2^{128.4}$
- $|OUT| \leq 2^{414}$

We have a **generic complexity** of $2^{1057.6} > 2^{491.5}$ computations.

$\Rightarrow$ **8 rounds can be distinguished**

Distinguishers on KECCAK-1600 permutation



Limited birthday problem on a 1600-bit permutation with:

- $|IN| \leq 2^{1142.8}$
- $|OUT| \leq 2^{414}$

We have a **generic complexity** of $2^{228.6} < 2^{491.5}$ computations.

⇒ 9 rounds cannot be distinguished

Outline

Overall results

Table: Best differential distinguishers complexities for each version of KECCAK internal permutations, for 1 up to 8 rounds.

b	best differential distinguishers complexity							
	1 rd	2 rds	3 rds	4 rds	5 rds	6 rds	7 rds	8 rds
100	1	1	1	2^2	2^8	2^{19}	-	-
200	1	1	1	2^2	2^8	2^{20}	2^{46}	-
400	1	1	1	2^2	2^8	2^{24}	2^{84}	-
800	1	1	1	2^2	2^8	2^{32}	2^{109}	-
1600	1	1	1	2^2	2^8	2^{32}	2^{142}	$\mathbf{2^{491.5}}$

Our method and our model have been **verified in practice** on reduced versions of KECCAK.

Future works

Use the differential path search tool and the unaligned rebound for

- the **recent collision/preimage** `KECCAK` **challenges**:
  - the variants with little number of rounds seem clearly reacheable (we already found collisions for 1 and 2-round challenges)
  - we need to find a smart way to use the freedom degrees when several blocks are needed
- **differential distinguisher on the hash function**, so far we have:
  - 3-round fixed-IV distinguisher
  - 5-round chosen-IV distinguisher

Analyze other functions with our framework:

- PRESENT
- SPONGENT
- JH

Thank you !