Unaligned Rebound Attack

Application to KECCAK

Alexandre Duc [1], Jian Guo [2], Thomas Peyrin [3] and Lei Wei [3]

[1] Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2] Institute for Infocomm Research, Singapore
[3] Nanyang Technological University, Singapore

21 March 2012

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Institute for Infocomm Research A*STAR

NANYANG TECHNOLOGICAL UNIVERSITY

The SHA-3 Competition

- Most standardized hash functions suffer from attacks
- NIST launched a SHA-3 competition
- December 2010: five finalists selected:
  BLAKE, Grøstl, JH, KECCAK, Skein
- None of them is broken yet → Important to perform cryptanalysis on them
- We focus on KECCAK (designed by Bertoni, Daemen, Peeters and Van Assche)

Outline

Our Goals

- Hard to find collision or preimage attacks
- We look for differential distinguishers
- on reduced-round versions of the internal permutation used in KECCAK (KECCAK-*f*)
- The Sponge proof relies on the fact that the internal permutation is ideal

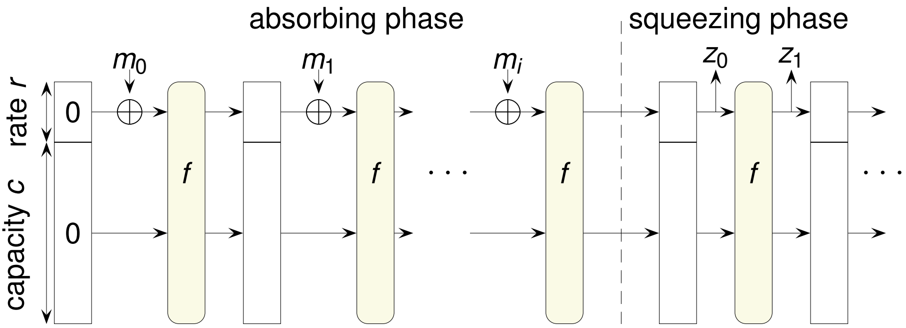
Previous Cryptanalysis Results on KECCAK

So far, the results on KECCAK are the following:

- **J.-P. Aumasson and W. Meier (2009)**:
  Zero-sum distinguishers up to 16 rounds of KECCAK-f[1600].
- **P. Morawiecki and M. Srebrny (2010)**:
  Preimage attack using SAT solvers on up to 3 rounds of KECCAK.
- **D. J. Bernstein (2010)**:
  A second-preimage attack on 8 rounds with high complexity.
- **C. Boura *et al.* (2010-2011)**:
  Zero-sum partitions distinguishers to the full 24-round version of KECCAK-f[1600].
- **M. Naya-Plasencia *et al.* (2011)** :
  Practical attacks on a small number of rounds.

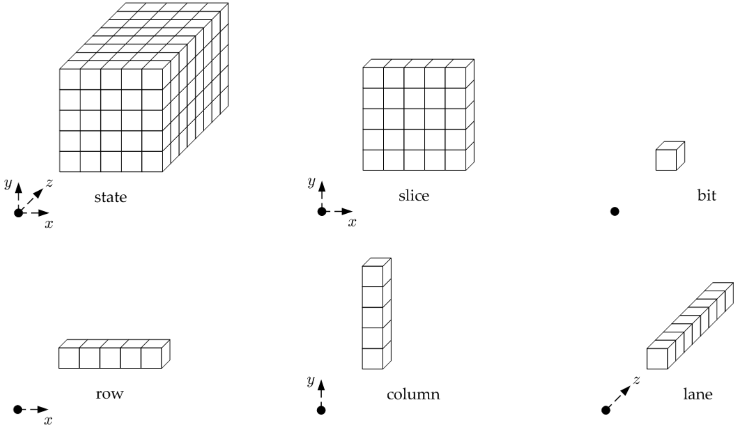
Outline

The Sponge Construction



The KECCAK-f State

- The b bit KECCAK-f state: a $5 \times 5 \times 2^\ell$ bit array
- 7 versions of KECCAK-f: $\ell = 0, \ldots, 6$ named KECCAK-$f[b]$



The KECCAK-f Internal Permutation

- b-bit KECCAK **round permutation** R_r applied on n_r rounds
- $n_r = 12 + 2\ell$
- 24 rounds for KECCAK-$f[1600]$
- R_r is divided into 5 substeps
- $R_r = \iota_r \circ \chi \circ \pi \circ \rho \circ \theta$

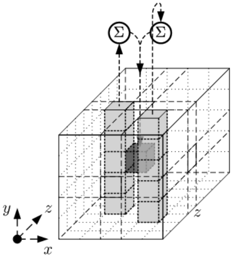
The θ Permutation

$$R_r = \iota_r \circ \chi \circ \pi \circ \rho \circ \boldsymbol{\theta}$$

The θ permutation

Linear mapping that provides a high level of diffusion

$$a[x][y][z] \leftarrow a[x][y][z] + \sum_{i=0}^{4} a[x-1][i][z] + \sum_{i=0}^{4} a[x+1][i][z-1] \, .$$

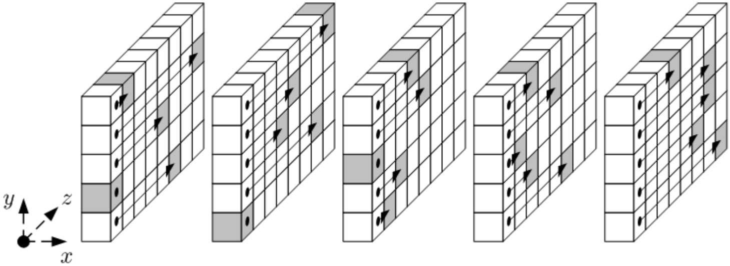


The ρ Permutation

$$R_r = \iota_r \circ \chi \circ \pi \circ \boldsymbol{\rho} \circ \theta$$

The ρ permutation

Linear mapping that provides inter-slice diffusion.
Each lane is rotated by a constant depending on x and y



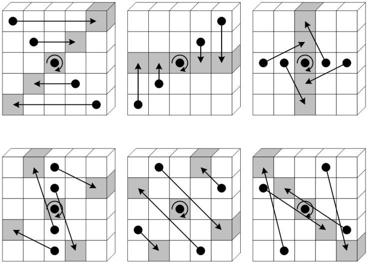
The π Permutation

$$R_r = \iota_r \circ \chi \circ \boldsymbol{\pi} \circ \rho \circ \theta$$

The π permutation

Rotation within a slice. Breaks column alignment.

Bit at position (x', y') is moved to $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$.



The χ Permutation

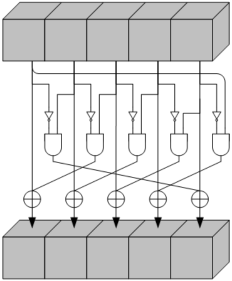
$$R_r = \iota_r \circ \chi \circ \pi \circ \rho \circ \theta$$

The χ permutation

Only non-linear layer

$s = 5 \times 2^\ell$ Sboxes (one per row)

$$a[x] \leftarrow a[x] + ((\neg a[x+1]) \wedge a[x+2])$$



The ι_r Permutation

$$R_r = \boldsymbol{\iota_r} \circ \chi \circ \pi \circ \rho \circ \theta$$

- Depends on the round number
- Addition of round constants to the first lane $a[0][0][.]$
- Breaks the symmetry of the rounds
- For differential cryptanalysis we ignore it

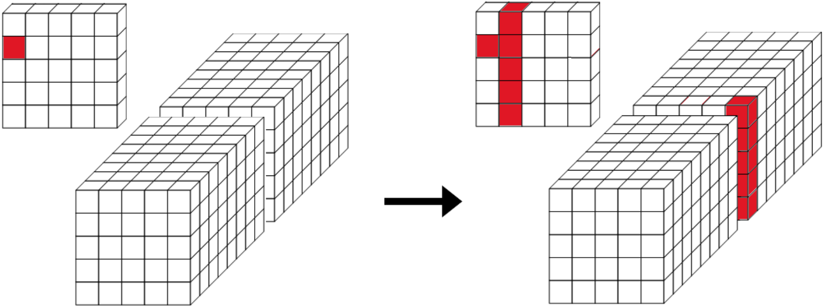
Summary

- We have one linear layer $\rightarrow \lambda := \pi \circ \rho \circ \theta$
- One non-linear layer χ
- One round constant layer that we ignore ι_r

Outline

Diffusion in KECCAK

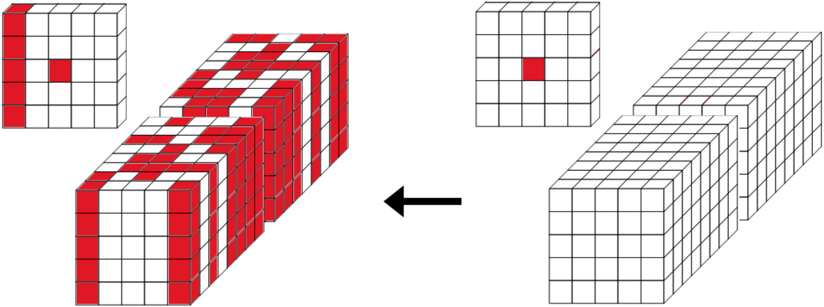
- Diffusion comes mostly from θ
- π and ρ move bits around
- χ has a very slow diffusion



Diffusion of θ (at most 11 new active bits)

Diffusion in KECCAK

- Diffusion comes mostly from θ
- π and ρ move bits around
- χ has a very slow diffusion

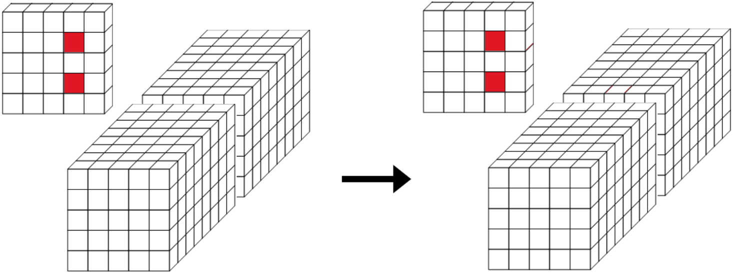


Diffusion of θ^{-1} (half of the bits are active in average)

The Column-Parity Kernel

$$\theta: \quad a[x][y][z] \leftarrow a[x][y][z] + \sum_{i=0}^{4} a[x-1][i][z] + \sum_{i=0}^{4} a[x+1][i][z-1]\ .$$

Even number of active bits in every column → no diffusion through θ



We call the set of such states the column-parity kernel (CPK)

Path Search Algorithm

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

- We start with random state in the CPK with $\leq k$ active columns
- We compute forward taking random “best” slice transition
- By “best”, we mean a transition that maximizes the number of columns with even parity and with lowest Hamming weight
- If path has best DP : one round backwards

Differential paths results on KECCAK

b	best differential path probability					
	1 rd	2 rds		3 rds		
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	
400	**2^{-84}**	(16 - 14 - 12 - 42)	2^{-216}	(16 - 32 - 40 - 32 - 96)
800	**2^{-109}**	(12 - 12 - 12 - 73)	2^{-432}	(32 - 64 - 80 - 64 - 192)
1600	**2^{-142}**	(12 - 12 - 12 - 106)	**2^{-709}**	(16 - 16 - 16 - 114 - 547)

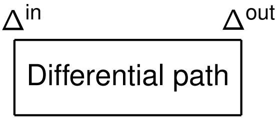
- Three round paths with 2^{-32} are best we can hope (see next talk)
- Path with 2^{-709} was independently improved by M. Naya-Plasencia *et al.* to 2^{-510}.

Simple Distinguishers

Easy distinguisher: fixed input/output difference

Generic complexity

Mapping a fixed input/output difference: 2^b

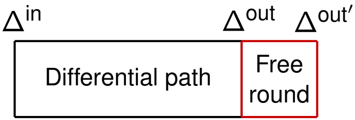


Simple Distinguishers

One free round: choose value for each of the Sboxes
$\rightarrow$Use freedom degrees

Generic complexity

Mapping a fixed input/output difference: 2^b



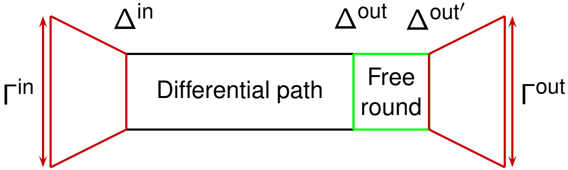
Simple Distinguishers

Map a set of input differences to a set of output differences:

Generic complexity

Limited birthday distinguisher (Gilbert and Peyrin):

$$\max\left\{\min\left\{\sqrt{2^b/\Gamma^{\text{in}}}, \sqrt{2^b/\Gamma^{\text{out}}}\right\}, \frac{2^b}{\Gamma^{\text{in}} \times \Gamma^{\text{out}}}\right\}$$



Outline

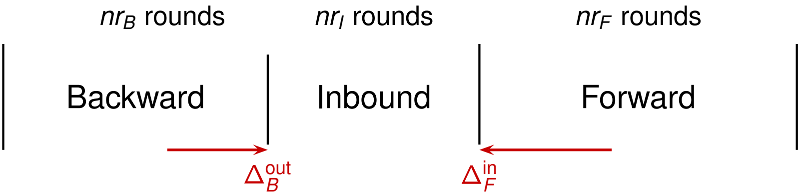
The Rebound Attack

- Proposed first by Mendel *et al.* in 2009.
- We divide the rounds into three parts

nr_B rounds	nr_I rounds	nr_F rounds
Backward	Inbound	Forward

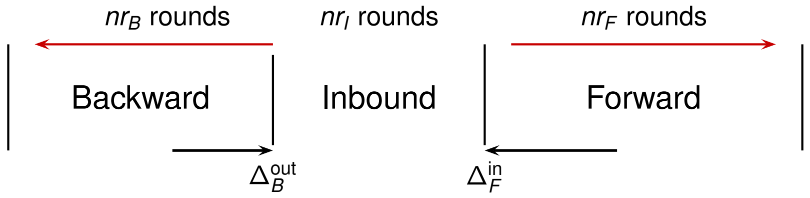
The Rebound Attack

- Proposed first by Mendel *et al.* in 2009.
- Inbound Phase: find matching differences with probability p_{match}. Usually all Sboxes active in the middle



The Rebound Attack

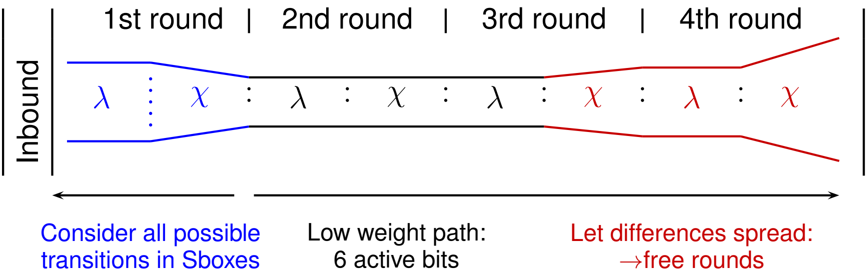
- Proposed first by Mendel *et al.* in 2009.
- Outbound Phase: generate N_{match} values from this match and propagate backward and forward with probability p_B and p_F



Rebound Attack is Hard on KECCAK

- We tried to apply the rebound directly with the 4-round path
  $\rightarrow$ Would give 9 rounds with complexity $< 2^{512}$
- *Not enough differential paths* to perform the inbound
- KECCAK has *weak alignment*: impossible to exploit truncated differentials or Super-Sboxes
- DDT: *fixed input difference* $\rightarrow$ all possible output differences occur with same probability
- Number of possible output differences depends strongly on the *Hamming weight of the input*

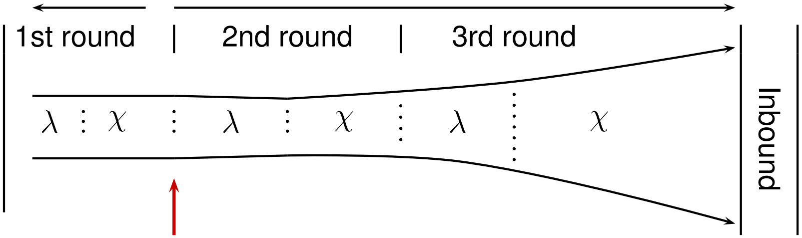
Forward Paths



Backward Paths

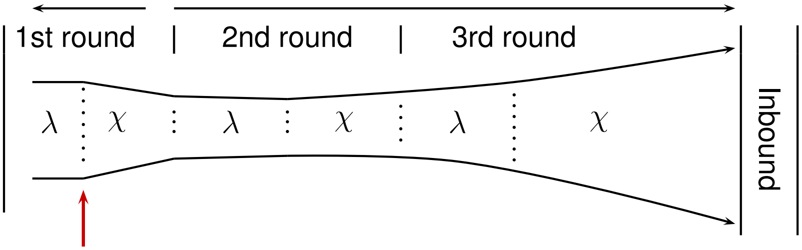
- We need *enough differential paths* for the inbound.
- We need *differential paths with good DP* for the outbound.

Backward Paths Generation



We start in the CPK with X active columns and 2 active bits each

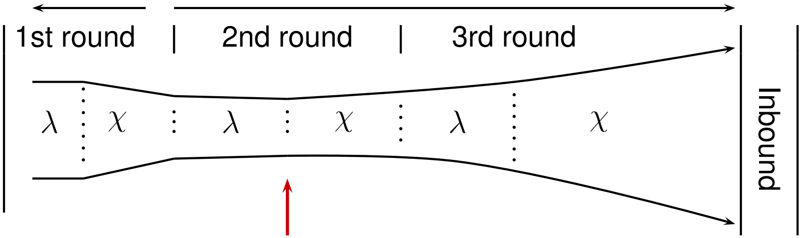
Backward Paths Generation



We let the differences spread in the first round

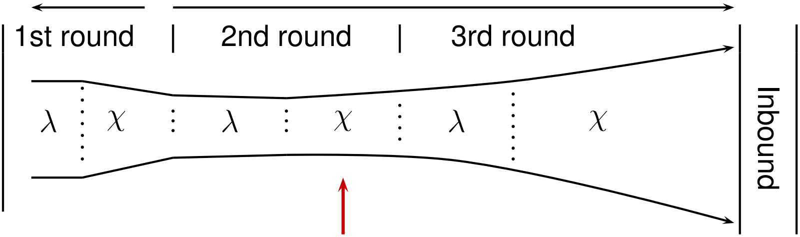
$\rightarrow$ Round for free

Backward Paths Generation



We keep the paths with at most one active bit per Sbox.

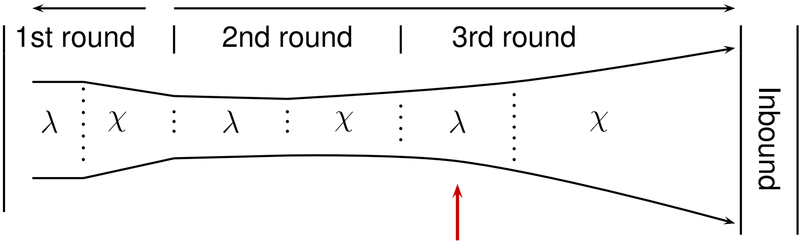
Backward Paths Generation



If HW=1 at input of Sbox, there always exists an output difference with HW=1 and two differences with HW=2.

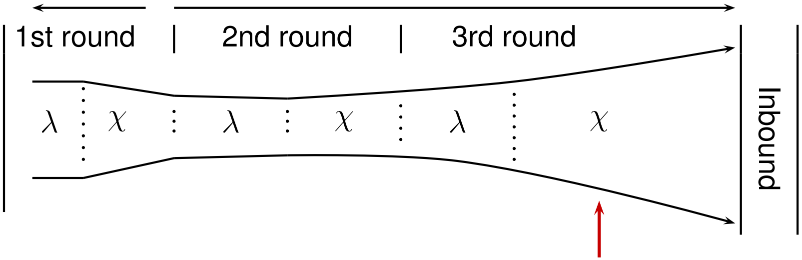
We select k $1 \mapsto 2$ transitions. Remaining transitions : $1 \mapsto 1$

Backward Paths Generation



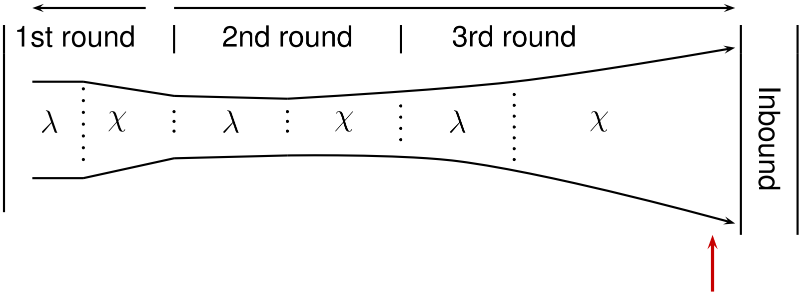
Expansion through θ
$\rightarrow$ Much more active bits.

Backward Paths Generation



We keep the paths that have a “good” DP

Backward Paths Generation



We want all Sboxes active to simplify analysis

Inbound Complexity

- We need to compute the probability of having a match p_{match} for the inbound
- We could use the average probability that a transition is possible
- Incorrect in practice
- Depends on the input Hamming weight: $4/31$ for $\text{Hw} = 1$, $16/31$ for $\text{Hw} = 4$
- Separation into Hamming weight classes: for every possible input Hamming weight, we compute the probability of a match

Outbound Complexity Problems

- We need to compute the number of values N_{match} we can generate from a match
- Same idea
- Number of solutions *decreases exponentially* with the Hamming weight
- Probability of having a match *increases exponentially*
- Average number of solutions not possible: we expect only one match

Outbound Complexity

- We call N_w the expected number of solutions when the input Hamming weight is w
- Same analysis (we consider all Hamming weight distributions)
- We select a w_{max}: *highest Hamming weight we can afford*
- $N_{\text{match}} \geq N_{w_{\text{max}}}$
- We need to update p_{match}: a match occur only below w_{max}

Finding Parameters

- We need to set X, k and the bound on the DP p_B for the backward paths
- With the best parameters we found, we get

Complexity of $2^{491.47}$ for 8 rounds (4 forward, 3 backward, 1 inbound)

Generic complexity is $\geq 2^{1057.6}$.

Outline

Overall Results

Table: Best differential distinguishers complexities for each version of KECCAK internal permutations, for 4 up to 8 rounds

b	best differential distinguishers complexity				
	4 rds	5 rds	6 rds	7 rds	8 rds
100	2^2	2^8	2^{19}	-	-
200	2^2	2^8	2^{20}	2^{46}	-
400	2^2	2^8	2^{24}	2^{84}	-
800	2^2	2^8	2^{32}	2^{109}	-
1600	2^2	2^8	2^{32}	2^{142}	$\mathbf{2^{491.47}}$

Our model and our method have been **verified in practice** on KECCAK-$f[100]$
We obtained a 6 round rebound attack with complexity $2^{28.76}$

Further Work

Use the differential path search algorithm for

- the **collision/preimage** KECCAK **“crunchy” challenges**:
  $\rightarrow$ We found collisions for 1 and 2-round challenges
- **differential distinguisher on the hash function**

Analyze other functions with our framework

Thank You!