Unaligned Rebound Attack
Application to KECCAK

Alexandre Duc\textsuperscript{1}, Jian Guo\textsuperscript{2}, Thomas Peyrin\textsuperscript{3} and Lei Wei\textsuperscript{3}

\textsuperscript{1} Ecole Polytechnique Fédérale de Lausanne, Switzerland
\textsuperscript{2} Institute for Infocomm Research, Singapore
\textsuperscript{3} Nanyang Technological University, Singapore

21 March 2012
Introduction

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

1. Introduction
2. Keccak
3. Differential Path Search
4. The Rebound Attack
5. Results and Further Work
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

1. Introduction
2. Keccak
3. Differential Path Search
4. The Rebound Attack
5. Results and Further Work
The Sponge Construction

absorbing phase

squeezing phase

\[ m_0 \quad m_1 \quad m_i \quad z_0 \quad z_1 \]

rate \( r \)
capacity \( c \)
The KECCAK\textit{-f} State

- The $b$ bit KECCAK\textit{-f} state: a $5 \times 5 \times 2^\ell$ bit array
- 7 versions of KECCAK\textit{-f}: $\ell = 0, \ldots, 6$ named KECCAK\textit{-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 $\theta$ Permutation

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

The $\theta$ 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 $\rho$ Permutation

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

The $\rho$ permutation

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

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

The $\pi$ 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}.$$

---

Duc, Guo, Peyrin, Wei (EPFL, I2R, NTU)  
Unaligned Rebound Attack  
21.03.2012 12 / 35
The $\chi$ Permutation

\[ R_r = \iota_r \circ \chi \circ \pi \circ \rho \circ \theta \]

The $\chi$ permutation

Only non-linear layer
\[ s = 5 \times 2^\ell \text{ Sboxes (one per row)} \]

\[ a[x] \leftarrow a[x] + ((\neg a[x + 1]) \land a[x + 2]) \]
The $\iota_r$ Permutation

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

- 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 $\chi$
- One round constant layer that we ignore $\nu_r$
Outline

1. Introduction
2. KECCAK
3. Differential Path Search
4. The Rebound Attack
5. Results and Further Work
Diffusion in KECCAK

- Diffusion comes mostly from $\theta$
- $\pi$ and $\rho$ move bits around
- $\chi$ has a very slow diffusion

Diffusion of $\theta$ (at most 11 new active bits)
Diffusion in KECCAK

- Diffusion comes mostly from $\theta$
- $\pi$ and $\rho$ move bits around
- $\chi$ has a very slow diffusion

Diffusion of $\theta^{-1}$ (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 $\rightarrow$ no diffusion through $\theta$

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

\[ a_0 \xleftarrow{\lambda^{-1}} b_0 \xleftarrow{\chi^{-1}} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda} b_3 \ldots \]

- 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

<table>
<thead>
<tr>
<th>( b )</th>
<th>1 rd</th>
<th>2 rds</th>
<th>3 rds</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>( 2^{-2} ) (2)</td>
<td>( 2^{-8} ) (4 - 4)</td>
<td>( 2^{-24} ) (8 - 8 - 8)</td>
</tr>
<tr>
<td>800</td>
<td>( 2^{-2} ) (2)</td>
<td>( 2^{-8} ) (4 - 4)</td>
<td>( 2^{-32} ) (4 - 4 - 24)</td>
</tr>
<tr>
<td>1600</td>
<td>( 2^{-2} ) (2)</td>
<td>( 2^{-8} ) (4 - 4)</td>
<td>( 2^{-32} ) (4 - 4 - 24)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( b )</th>
<th>4 rds</th>
<th>5 rds</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>( 2^{-84} ) (16 - 14 - 12 - 42)</td>
<td>( 2^{-216} ) (16 - 32 - 40 - 32 - 96)</td>
</tr>
<tr>
<td>800</td>
<td>( 2^{-109} ) (12 - 12 - 12 - 73)</td>
<td>( 2^{-432} ) (32 - 64 - 80 - 64 - 192)</td>
</tr>
<tr>
<td>1600</td>
<td>( 2^{-142} ) (12 - 12 - 12 - 106)</td>
<td>( 2^{-709} ) (16 - 16 - 16 - 114 - 547)</td>
</tr>
</tbody>
</table>

- 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: $2^b$

Differential path

$\Delta^{\text{in}}$  $\Delta^{\text{out}}$
Simple Distinguishers

One free round: choose value for each of the Sboxes

→ Use freedom degrees

Generic complexity

Mapping a fixed input/output difference: $2^b$

$\Delta_{\text{in}} \quad \Delta_{\text{out}} \quad \Delta_{\text{out'}}$

Differential path  Free round
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

1. Introduction
2. KECCAK
3. Differential Path Search
4. The Rebound Attack
5. Results and Further Work
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 $p_{\text{match}}$. Usually all Sboxes active in the middle.

\[
\begin{align*}
\text{Backward} & \quad \text{Inbound} & \quad \text{Forward} \\
\Delta^\text{out}_B & \quad & \Delta^\text{in}_F \\
nr_B \text{ rounds} & \quad nr_I \text{ rounds} & \quad nr_F \text{ rounds}
\end{align*}
\]
The Rebound Attack

- Proposed first by Mendel *et al.* in 2009.

- **Outbound Phase:** generate \(N_{\text{match}}\) values from this match and propagate backward and forward with probability \(p_B\) and \(p_F\)

\[
\begin{align*}
nr_B \text{ rounds} & \quad \Delta^\text{out}_B \quad nr_I \text{ rounds} \\
\text{Backward} & \quad \text{Inbound} & \quad \text{Forward} \\
& \quad \Delta^\text{in}_F \quad nr_F \text{ rounds}
\end{align*}
\]
Rebound Attack is Hard on KECCAK

- We tried to apply the rebound directly with the 4-round path → 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* → all possible output differences occur with same probability

- Number of possible output differences depends strongly on the Hamming weight of the input
Forward Paths

Consider all possible transitions in Sboxes

Low weight path:
6 active bits

Let differences spread:
→free rounds
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
→ Round for free
Backward Paths Generation

We keep the paths with at most one active bit per Sbox.
If $\text{HW}=1$ at input of Sbox, there always exists an output difference with $\text{HW}=1$ and two differences with $\text{HW}=2$.

We select $k \ 1 \leftrightarrow 2$ transitions. Remaining transitions : $1 \leftrightarrow 1$
Backward Paths Generation

Expansion through $\theta$

$\rightarrow$ 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 $p_{\text{match}}$ 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 $Hw = 1$, 16/31 for $Hw = 4$

- 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 $N_{\text{match}}$ 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 $w_{\text{max}}$: highest Hamming weight we can afford
- $N_{\text{match}} \geq N_{w_{\text{max}}}$
- We need to update $\rho_{\text{match}}$: a match occur only below $w_{\text{max}}$
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

1. Introduction
2. KECCAK
3. Differential Path Search
4. The Rebound Attack
5. Results and Further Work
Results and Further Work

Overall Results

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

<table>
<thead>
<tr>
<th>b</th>
<th>4 rds</th>
<th>5 rds</th>
<th>6 rds</th>
<th>7 rds</th>
<th>8 rds</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>$2^2$</td>
<td>$2^8$</td>
<td>$2^{19}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>200</td>
<td>$2^2$</td>
<td>$2^8$</td>
<td>$2^{20}$</td>
<td>$2^{46}$</td>
<td>-</td>
</tr>
<tr>
<td>400</td>
<td>$2^2$</td>
<td>$2^8$</td>
<td>$2^{24}$</td>
<td>$2^{84}$</td>
<td>-</td>
</tr>
<tr>
<td>800</td>
<td>$2^2$</td>
<td>$2^8$</td>
<td>$2^{32}$</td>
<td>$2^{109}$</td>
<td>-</td>
</tr>
<tr>
<td>1600</td>
<td>$2^2$</td>
<td>$2^8$</td>
<td>$2^{32}$</td>
<td>$2^{142}$</td>
<td>$2^{491.47}$</td>
</tr>
</tbody>
</table>

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:
  → We found collisions for 1 and 2-round challenges

- differential distinguisher on the hash function

Analyze other functions with our framework
Thank You!
Finding Parameters (technical details)

- We need to set $X$, $k$ and the bound on the DP $p_B$ for the backward paths

- For $X = 8$, $k = 8$ and $p_B = 2^{-450}$, we can generate $2^{477.98}$ differences

- $p_B = 2^{-450}$ and $p_F = 2^{-36}$
  - $\Rightarrow$ we need $N_{\text{match}} \geq 2^{486} \Rightarrow w_{\text{max}} = 1000$

- This leads to $p_{\text{match}} = 2^{-491.47}$
Finding Parameters (technical details)

- We need to set $X$, $k$ and the bound on the DP $\rho_B$ for the backward paths

- For $X = 8$, $k = 8$ and $\rho_B = 2^{-450}$, we can generate $2^{477.98}$ differences

- $\rho_B = 2^{-450}$ and $\rho_F = 2^{-36}$
  $\rightarrow$ we need $N_{\text{match}} \geq 2^{486} \rightarrow w_{\text{max}} = 1000$

- This leads to $\rho_{\text{match}} = 2^{-491.47}$

$\Gamma_{B}^{\text{out}} = 2^{468.17}$, $\Gamma_{F}^{\text{in}} = 2^{23.3} \rightarrow 2^{491.47}$ couples for inbound $\checkmark$
Finding Parameters (technical details)

- We need to set $X$, $k$ and the bound on the DP $p_B$ for the backward paths.

- For $X = 8$, $k = 8$ and $p_B = 2^{-450}$, we can generate $2^{477.98}$ differences.

- $p_B = 2^{-450}$ and $p_F = 2^{-36}$
  $\rightarrow$ we need $N_{\text{match}} \geq 2^{486} \rightarrow w_{\text{max}} = 1000$

- This leads to $p_{\text{match}} = 2^{-491.47}$

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

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

Separation into Hamming weight classes

\[ p_{\text{match}} := \Pr[\text{match}|\text{full}] \]
\[ = \sum_w \Pr[H_{\text{total}} = w|\text{full}] \times \Pr[\text{match}|H_{\text{total}} = w, \text{full}] \]

Measured probability at the input of the Sboxes
Inbound Complexity

Separation into Hamming weight classes

\[ p_{\text{match}} := \Pr[\text{match}|\text{full}] = \sum_w \Pr[H_{\text{total}} = w|\text{full}] \times \Pr[\text{match}|H_{\text{total}} = w, \text{full}] \]

We consider all possible Hamming weight distributions: \( c_i \) Sboxes with Hamming weight \( i \)