Unaligned Rebound Attack - Application to KECCAK

Jian Guo

joint with *Alexandre Duc*, *Thomas Peyrin* and *Lei Wei*

TCCM-CACR, 24 - 25 Nov 2012

A*STAR Institute for Infocomm Research

Outline

1. Introduction to Hash Functions
   - Introduction
   - Recent Events
   - the SHA-3 Competition
2. The Keccak Sponge function family
   - Compression Function Constructions
   - Keccak
3. Unaligned Rebound Attack
4. Conclusion

Outline

1 Introduction to Hash Functions
- Introduction
- Recent Events
- the SHA-3 Competition

2 The KECCAK Sponge function family
- Compression Function Constructions
- KECCAK

3 Unaligned Rebound Attack

4 Conclusion

What is hash

A typical order of corned beef hash, hashed with potatoes and carrots, or ‘groestl’.

What is hash

A typical order of corned beef hash, hashed with potatoes and carrots, or 'groestl'.

Cryptographic hash functions:

$$\text{arbitrary bit string} \xrightarrow{h} \text{fixed length string}$$

Applications

- digitial signature
- data integrity, checksum
- message authetication code
- random number generators
- digital forensics
- and more ...

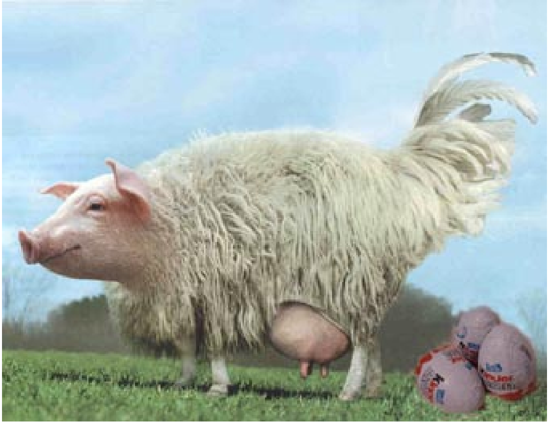
Required Properties

- **collision resistance**.
  It should be computationally difficult to
  find x, x' s.t. $x \neq x'$ and $h(x) = h(x')$. — $2^{n/2}$
- **preimage resistance**.
  Given $h(x)$ (not x), find x' s.t. $h(x) = h(x')$. — 2^n
- **second preimage resistance**.
  Given m, find m' s.t. $m \neq m'$ and $h(m) = h(m')$. — 2^{n-k}

Required Properties

- **collision resistance**.
  It should be computationally difficult to
  find x, x' s.t. $x \neq x'$ and $h(x) = h(x')$. — $2^{n/2}$
- **preimage resistance**.
  Given $h(x)$ (not x), find x' s.t. $h(x) = h(x')$. — 2^n
- **second preimage resistance**.
  Given m, find m' s.t. $m \neq m'$ and $h(m) = h(m')$. — 2^{n-k}
- **pseudo-randomness**.
  $h(key, \cdot)$ should **look like** a random oracle.
- **Unpredictability**.
  predict $h(key, x)$ for unqueried x's
- **Indifferentiability**.
  find "related" sets of input/output values.
- software/hardware **efficiency**
- and more ....

Expectations from Hash Functions



egg producing, wool providing, milk giving, pig

Recent Events in Hash Functions

- `MD5` and `SHA-1` were broken by Wang et al. in 2005.

Recent Events in Hash Functions

- `MD5` and `SHA-1` were broken by Wang et al. in 2005.
- In response, on 3rd March 2006, NIST recommended using `SHA-2` from 2010.

Recent Events in Hash Functions

- `MD5` and `SHA-1` were broken by Wang et al. in 2005.
- In response, on 3rd March 2006, NIST recommended using `SHA-2` from 2010.
- `SHA-2` follows similar design principle of `SHA-1`, call for `SHA-3` on 2nd Nov 2007.

The SHA−3 Competition

- 2007/11/02: call for submissions.
- 2008/10/31: submission deadline, 64 received.
- 2008/12/09: 51/64 were selected for Round 1.
- 2009/07/24: 14/51 were selected for Round 2.
- 2010/12/10: 5/14 were selected for Round 3.
- 2012/10/03: announcement of winner – KECCAK.

The SHA−3 Competition

- 2007/11/02: call for submissions.
- 2008/10/31: submission deadline, 64 received.
- 2008/12/09: 51/64 were selected for Round 1.
- **2009/02/25-28: 1st SHA−3 conference, KULeuven.**
- 2009/07/24: 14/51 were selected for Round 2.
- **2010/08/23-24: 2nd SHA−3 conference, UCSB.**
- 2010/12/10: 5/14 were selected for Round 3.
- **2012/03/22-23: 3rd SHA−3 conference, Washington.**
- 2012/10/03: announcement of winner – KECCAK.

Outline

1. Introduction to Hash Functions
   - Introduction
   - Recent Events
   - the SHA−3 Competition
2. The KECCAK Sponge function family
   - Compression Function Constructions
   - KECCAK
3. Unaligned Rebound Attack
4. Conclusion

Compression Function Constructions

- block size based, e.g., $H' = E_K(H) \oplus H$, `MD5`, `SHA-1`, `SHA-2`

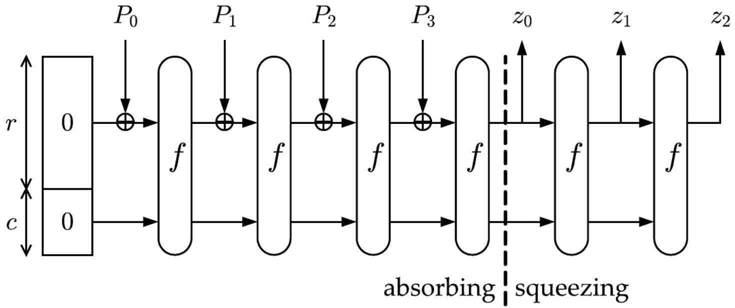
Compression Function Constructions

- block size based, e.g., $H' = E_K(H) \oplus H$, `MD5`, `SHA-1`, `SHA-2`
- hard problem based — the security of the hash function can be reduced to some hard mathematical problem, e.g., VSH, SWIFFT

Compression Function Constructions

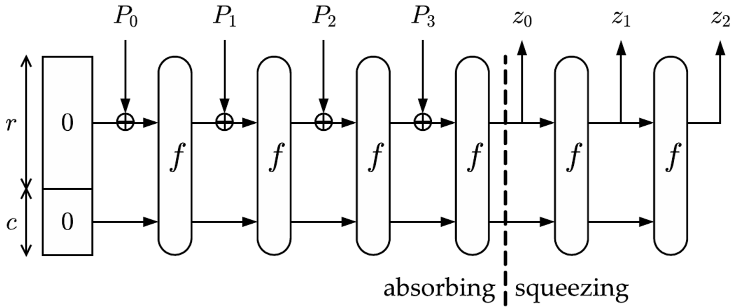
- block size based, e.g., $H' = E_K(H) \oplus H$, `MD5`, `SHA-1`, `SHA-2`
- hard problem based — the security of the hash function can be reduced to some hard mathematical problem, e.g., VSH, SWIFFT
- permutation based, e.g., KECCAK, JH

The Sponge Construction



f: a permutation; r: message rate, i.e., bit size for each message block; c: capacity; z_i: hash outputs;

The Sponge Construction



f: a permutation; r: message rate, i.e., bit size for each message block; c: capacity; z_i: hash outputs;

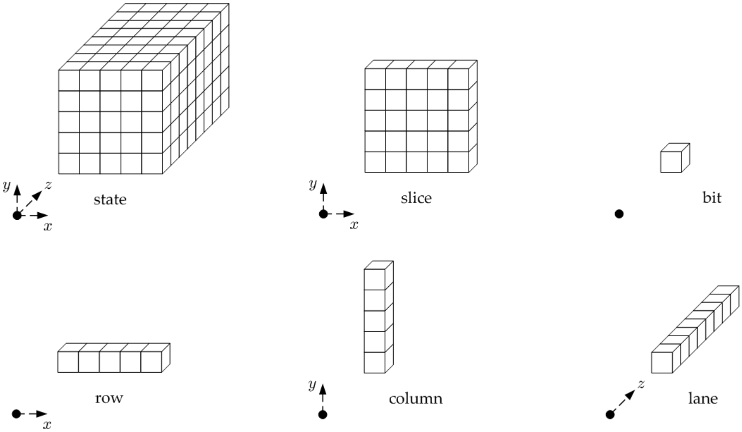
A sponge function is indifferentiable from a random oracle if the permutation is ideal.

The KECCAK Sponge hash function family

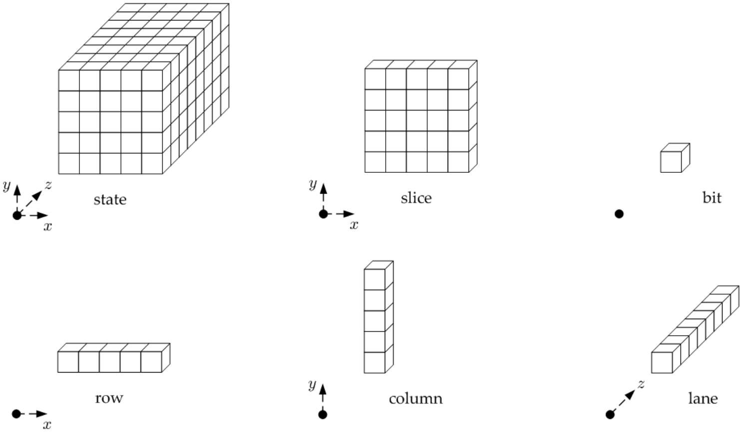
- one 1600-bit permutation
- specified $224, 256, 384, 512$ bit outputs, and ideally supports all output sizes.
- ARX (Addition-Rotation-Xor) design with 5-bit ‘ARX sbox’
- 24 rounds with identical round function up to a difference of constant addition.

The KECCAK Sponge hash function family

The 1600-bit state can be viewed a cube of 5*x*5*x*64.



The 1600-bit state can be viewed a cube of $5x5x64$.



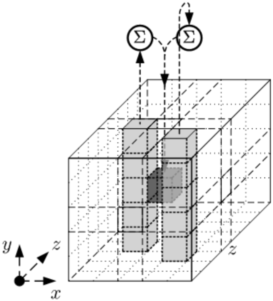
24 rounds, each consists of $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

The KECCAK round function

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

The KECCAK round function

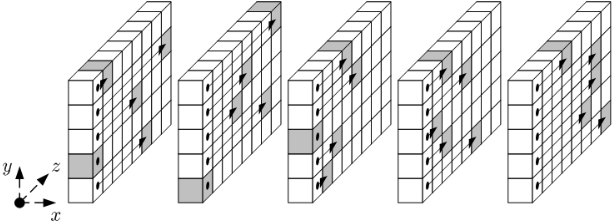
$\iota \circ \chi \circ \pi \circ \rho \circ \boldsymbol{\theta}$



a[x][y][z] = a[x][y][z] + $\sum$ a[x-1][·][z] + $\sum$ a[x+1][·][z-1]
provides inter-slice diffusion, linear

The KECCAK round function

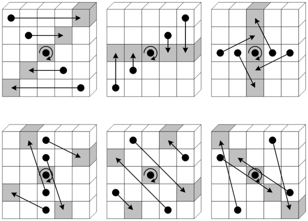
$\iota \circ \chi \circ \pi \circ \boldsymbol{\rho} \circ \theta$



provides intra-lane diffusion, linear, no difference increases for differential path

The KECCAK round function

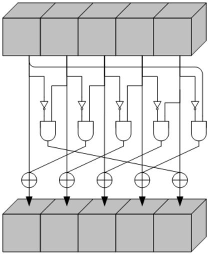
$\iota \circ \chi \circ \boldsymbol{\pi} \circ \rho \circ \theta$



provides intra-slice diffusion, linear, no difference increases for differential path

The KECCAK round function

$\iota \circ \boldsymbol{\chi} \circ \pi \circ \rho \circ \theta$



the only non-linear layer, can be viewed as 5-bit sbox

The KECCAK round function

$\boldsymbol{\iota} \circ \chi \circ \pi \circ \rho \circ \theta$

ι: adding a round dependent constant to the first lane, to distinguish each round, and resists slide attack etc. Not essential for differential path

Overall: we can write $\iota \circ \chi \circ (\pi \circ \rho \circ \theta)$ as $\iota \circ \chi \circ \lambda$, and consider only $\chi \circ \lambda$ for differential paths.

Outline

1 Introduction to Hash Functions
- Introduction
- Recent Events
- the SHA-3 Competition

2 The KECCAK Sponge function family
- Compression Function Constructions
- KECCAK

3 Unaligned Rebound Attack

4 Conclusion

Building High Probability Differential Path

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

Properties: θ propagates the differences slowly, i.e., one bit affects at most 11 other bits; however one bit difference θ^{-1} affects half of the state, around 800 bits.

Building High Probability Differential Path

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

Properties: θ propagates the differences slowly, i.e., one bit affects at most 11 other bits; however one bit difference θ^{-1} affects half of the state, around 800 bits.

Furthermore, if we can keep even number (0, 2, 4) of difference bits in each column, then the $\sum$ terms can be removed, and θ acts like identity. This is called column parity kernel (CPK).

Building High Probability Differential Path

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

Properties: θ propagates the differences slowly, i.e., one bit affects at most 11 other bits; however one bit difference θ^{-1} affects half of the state, around 800 bits.

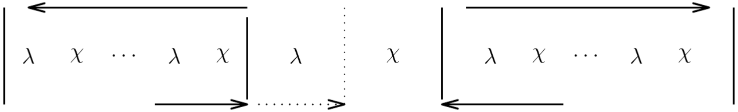
Furthermore, if we can keep even number (0, 2, 4) of difference bits in each column, then the $\sum$ terms can be removed, and θ acts like identity. This is called column parity kernel (CPK).

Keep the differential path in the CPK as much as possible! This is possible for at most 3 rounds of KECCAK.

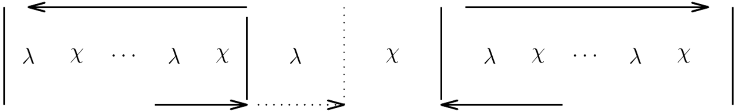
aiming for a differential path with 7 rounds like:
$3R \longrightarrow R \longleftarrow 3R$.

$$\left|\ \lambda \quad \chi \quad \cdots \quad \lambda \quad \chi\ \right| \quad \lambda \quad \vdots \quad \chi \quad \left|\ \lambda \quad \chi \quad \cdots \quad \lambda \quad \chi\ \right|$$

Unaligned Rebound Attack to KECCAK

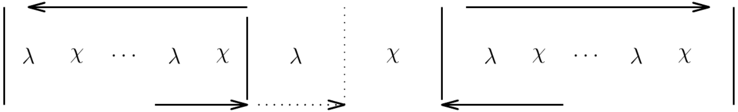


- differential path for each side is of 3 rounds.



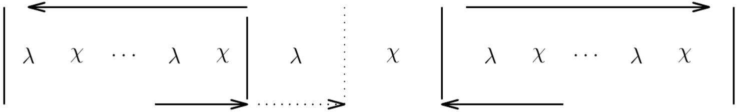
- differential path for each side is of 3 rounds.
- generate lots of differential paths independently from both sides, then find a possible difference match in the middle

Unaligned Rebound Attack to KECCAK



- differential path for each side is of 3 rounds.
- generate lots of differential paths independently from both sides, then find a possible difference match in the middle
- once a difference match is found, generate all possible solution, and find one confines the differential path in both sides.

Unaligned Rebound Attack to KECCAK



- differential path for each side is of 3 rounds.
- generate lots of differential paths independently from both sides, then find a possible difference match in the middle
- once a difference match is found, generate all possible solution, and find one confines the differential path in both sides.

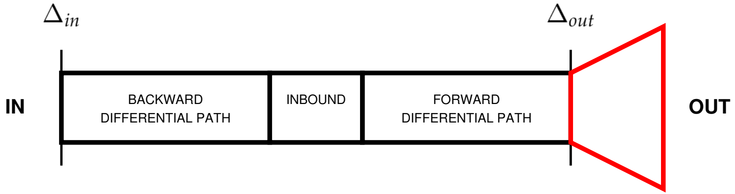
By carefully examine the tradeoff between differential probability and number of differential paths possible, we obtained 7-round differential path with complexity $2^{491.5}$.

Limited Birthday Distinguishers

Given a set of input difference Δ_{in} with size IN and a set of output difference Δ_{out} with size OUT, a bijective function f with b bits, the complexity to find a pair (x, x'), such that $x \oplus x' \in \Delta_{in}$ and $f(x) \oplus f(x') \in \Delta_{out}$, is $\max\{\sqrt{2^b/IN}, \sqrt{2^b/OUT}, 2^b/(IN \cdot OUT)\}$.

If an algorithm finds such a pair faster, we call it a distinguisher.

Application to KECCAK



©Thomas Peyrin

- Complexity $2^{491.5}$
- $IN \leq 2^{128.4}$, $OUT \leq 2^{414}$, generic limited birthday attack comes with complexity $2^{1057.6}$.
- hence distinguisher for 8 rounds.

Outline

1. Introduction to Hash Functions
   - Introduction
   - Recent Events
   - the SHA-3 Competition
2. The KECCAK Sponge function family
   - Compression Function Constructions
   - KECCAK
3. Unaligned Rebound Attack
4. Conclusion

Attack Comparisons

- 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):
  preimage attack using SAT solvers, 3 rounds.
- D. Bernstein (2010):
  (second)-preimage attack on 8 rounds with complexity $2^{511.5}$ and 2^{508} bits of memory, using low algebraic degree.
- 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}, using low algebriac degree. Improved by Duan and Lai in 2012 to 2^{1575}.
- I. Dinur (2012): 4-round collision and 5-round near collision for KECCAK-224 and KECCAK-256.
- Ours (2012): 8-round distinguisher with complexity $2^{491.5}$.

End of Talk

Thank you!

Questions?