Cryptanalysis of KECCAK

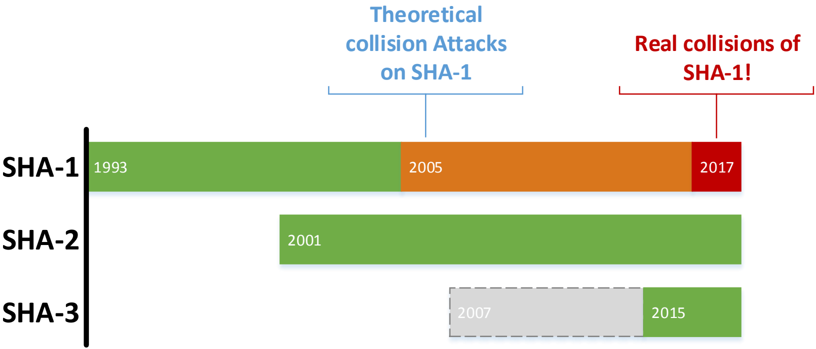
Ling Song

8 May 2021

Outlines

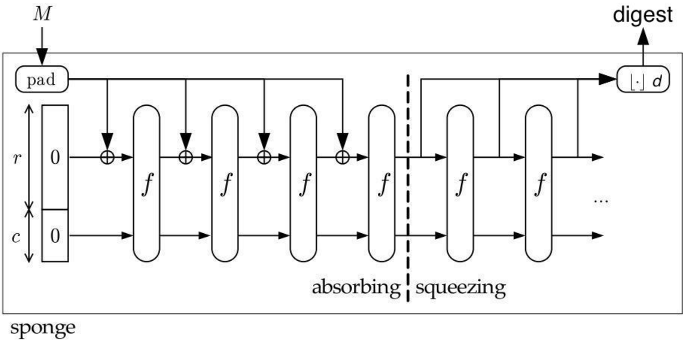
Outline

NIST Standards of Secure Hash Algorithm



SHA-3 Hash Function

The sponge construction [BDPV11]



- b-bit permutation f
- Two parameters: bitrate r, capacity c, and $b = r + c$.
- The message is padded and then split into r-bit blocks.

Instances of KECCAK and SHA-3

Based on the Sponge construction with a permutation called KECCAK-f (KECCAK-p):

- KECCAK versions
  - KECCAK$[c]$, $c = 2d, d =$224/256/384/512.
- SHA-3 versions
  - SHA3-n, $n =$224/256/384/512 and $c = 2n$, $d = n$.
  - SHAKEn (eXtendable Output Functions, XOFs)
    - (SHAKE = SHA + KEccak)
    - $n =$128/256, $c = 2n, d \leq 2n$.
- Instances of KECCAK challenge
  - KECCAK$[r, c, n_r, d]$ where d is the digest size, and n_r is the number of rounds.
  - For the category of collision challenges, $d = c = 160$.

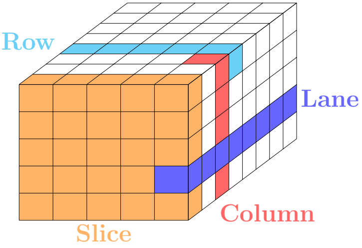
SHA-3 Hash Function

KECCAK-*f* permutation

- 1600 bits: seen as a 5×5 array of 64-bit lanes, $A[x,y], 0 \leq x,y < 5$
- 24 rounds
- each round R consists of five steps:

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

- χ : the only nonlinear operation, a 5-bit Sbox applies to each row.



http://www.iacr.org/authors/tikz/

SHA-3 Hash Function

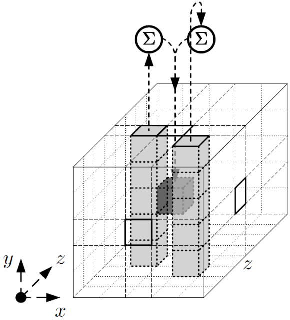
KECCAK permutation: $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

θ step: adding two columns to the current bit

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$$

$$D[x] = C[x-1] \oplus (C[x+1] \lll 1)$$

$$A[x,y] = A[x,y] \oplus D[x]$$



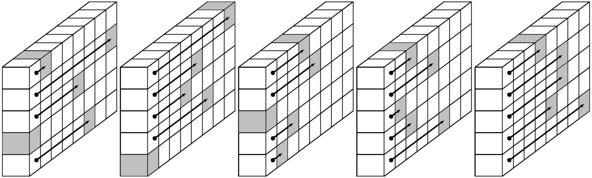
http://keccak.noekeon.org/

- The Column Parity kernel
  - If $C[x] = 0, 0 \leq x < 5$, then the state A is in the CP kernel.

SHA-3 Hash Function

KECCAK permutation: $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

ρ step: lane level rotations, $A[x,y] = A[x,y] \lll r[x,y]$



http://keccak.noekeon.org/

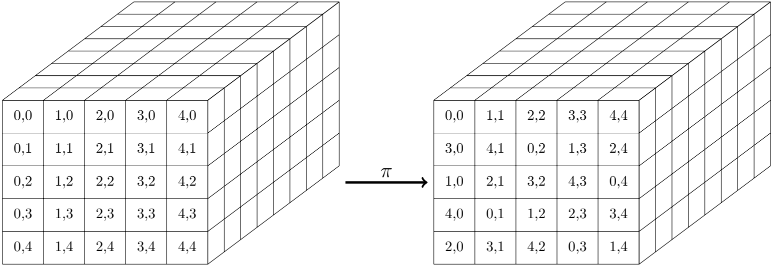
Rotation offsets $r[x,y]$

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=0$	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

SHA-3 Hash Function

KECCAK permutation: $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

π step: permutation on lanes



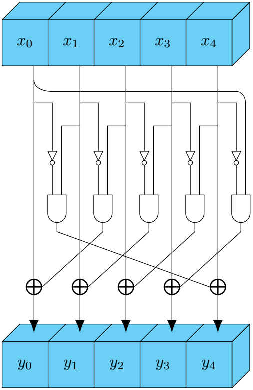
$$A[y, 2 * x + 3 * y] = A[x, y]$$

SHA-3 Hash Function

KECCAK permutation: $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

χ step: 5-bit S-boxes, nonlinear operation on rows

$$y_0 = x_0 + (x_1 + 1) \cdot x_2,$$
$$y_1 = x_1 + (x_2 + 1) \cdot x_3,$$
$$y_2 = x_2 + (x_3 + 1) \cdot x_4,$$
$$y_3 = x_3 + (x_4 + 1) \cdot x_0,$$
$$y_4 = x_4 + (x_0 + 1) \cdot x_1.$$



SHA-3 Hash Function

KECCAK permutation: $\iota \circ \chi \circ \pi \circ \rho \circ \theta$

ι step: adding a round constant to the state

Adding one round-dependent constant to the first "lane", to destroy the symmetry.

Without ι

- The round function would be symmetric.
- All rounds would be the same.
- Fixed points exist.
- Vulnerable to rotational attacks, slide attacks, ...

Description of SHA-3 (KECCAK)

Round function of KECCAK-f

Internal state A: a 5×5 array of 64-bit lanes

θ step $C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$
$D[x] = C[x-1] \oplus (C[x+1] \lll 1)$
$A[x,y] = A[x,y] \oplus D[x]$

ρ step $A[x,y] = A[x,y] \lll r[x,y]$
- The constants $r[x,y]$ are the rotation offsets.

π step $A[y, 2*x+3*y] = A[x,y]$

χ step $A[x,y] = A[x,y] \oplus ((A[x+1,y])\&A[x+2,y])$

ι step $A[0,0] = A[0,0] \oplus RC$
- $RC[i]$ are the round constants.

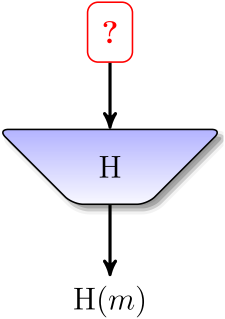
$L \triangleq \pi \circ \rho \circ \theta$

The only non-linear operation is χ step.

Outline

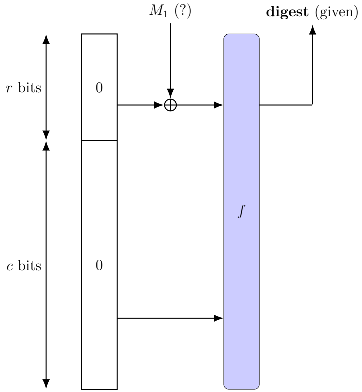
Security Requirements

Preimage



2^n

Preimage Attack: Strategy



- Simplest case: Given a d-bit digest, find an r-bit message block M_1.
- Padding and c bits capacity are out of control
- Permutation f is reduced

How to keep the Sbox χ linear

The expression of $b = \chi(a)$ is of algebraic degree 2:

$b_i = a_i + \overline{a_{i+1}} \cdot a_{i+2}$, for $i = 0, 1, \ldots, 4$.

How to keep the Sbox χ linear

The expression of $b = \chi(a)$ is of algebraic degree 2:

$b_i = a_i + \overline{a_{i+1}} \cdot a_{i+2}$, for $i = 0, 1, \ldots, 4$.

Observation

When there is no neighbouring variables in the input of an Sbox, then the application of χ does NOT increase algebraic degree.

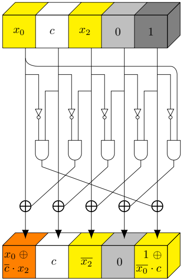
How to keep the Sbox χ linear

The expression of $b = \chi(a)$ is of algebraic degree 2:

$b_i = a_i + \overline{a_{i+1}} \cdot a_{i+2}$, for $i = 0, 1, \ldots, 4$.

Observation

When there is no neighbouring variables in the input of an Sbox, then the application of χ does NOT increase algebraic degree.



✓

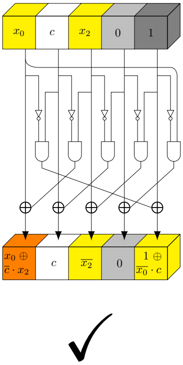
How to keep the Sbox χ linear

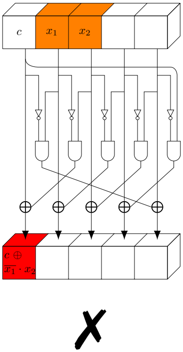
The expression of $b = \chi(a)$ is of algebraic degree 2:

$b_i = a_i + \overline{a_{i+1}} \cdot a_{i+2}$, for $i = 0, 1, \ldots, 4$.

Observation

When there is no neighbouring variables in the input of an Sbox, then the application of χ does NOT increase algebraic degree.





How to keep χ^{-1} linear

$a = \chi^{-1}(b)$ is of algebraic degree 3: $a_i = b_i \oplus \overline{b_{i+1}} \cdot (b_{i+2} \oplus \overline{b_{i+3}} \cdot b_{i+4})$

Our Setting

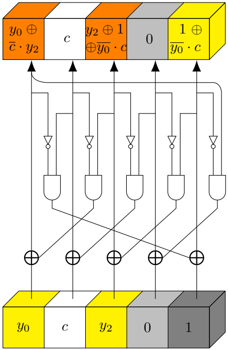
keep $y_3 = 0$, $y_4 = 1$, and y_1 constant, then χ^{-1} becomes linear.

How to keep χ^{-1} linear

$a = \chi^{-1}(b)$ is of algebraic degree 3: $a_i = b_i \oplus \overline{b_{i+1}} \cdot (b_{i+2} \oplus \overline{b_{i+3}} \cdot b_{i+4})$

Our Setting

keep $y_3 = 0, y_4 = 1$, and y_1 constant, then χ^{-1} becomes linear.



Properties of θ

Definition of θ operation:

$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$
$D[x] = C[x-1] \oplus (C[x+1] \lll 1)$
$A[x,y] = A[x,y] \oplus D[x]$

Properties of θ

Definition of θ operation:

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$$
$$D[x] = C[x-1] \oplus (C[x+1] \lll 1)$$
$$A[x,y] = A[x,y] \oplus D[x]$$

Properties:

When $C[x]$ is forced to be a constant, i.e., the sum of the all columns are kept to be constants, then θ acts the same as adding a constant.

Properties of θ

Definition of θ operation:

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$$
$$D[x] = C[x-1] \oplus (C[x+1] \lll 1)$$
$$A[x,y] = A[x,y] \oplus D[x]$$

Properties:

When $C[x]$ is forced to be a constant, i.e., the sum of the all columns are kept to be constants, then θ acts the same as adding a constant.

When differential attack is applied, and the sum of differences of all columns are kept to be zero ($C[x] = 0$ for all x), then θ acts the same as identity. This special structure is called CP-kernel (Column Parity).

θ acts like identity

When the sum of all columns are constants, θ acts like identity w.r.t. the variables.

x	c	c	c	c
$x+c$	c	c	c	c
c	c	c	c	c
c	c	c	c	c
c	c	c	c	c

$\theta =$ Identity $\longrightarrow$

$x+c$	c	c	c	c
$x+c$	c	c	c	c
c	c	c	c	c
c	c	c	c	c
c	c	c	c	c

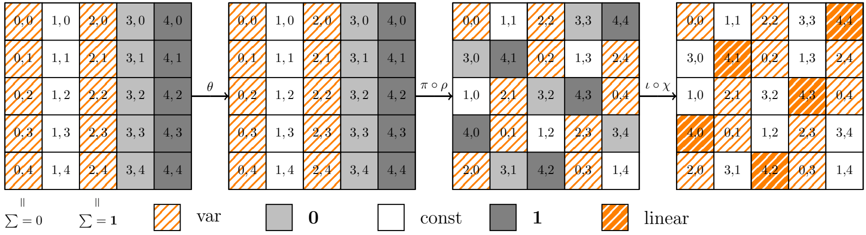
$\|$

$\sum = c$

c denotes a binary constant with value either 0 or 1.

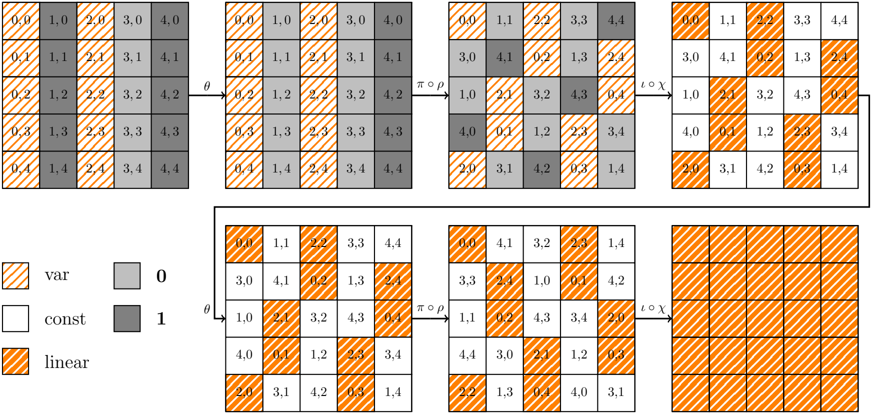
Linear Structure

Keeping $1 + 1$ rounds being linear with the degree of freedom up to 512

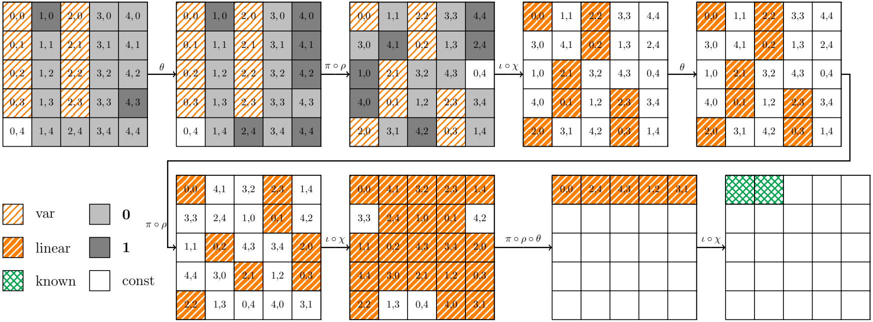


Linear Structure

Keeping 1 + 2 rounds being linear with the degree of freedom up to 194



Preimage Attack on 3-Round SHAKE128 (1)



64 ∗ 2 variables, 64 ∗ 2 quadratic equations.
Solving systems of non-linear equations is hard.

Setting up linear equations from the output of χ

Bilinear structure of χ

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

$$b_0 = a_0 \oplus \overline{a_1} \cdot a_2 \tag{1}$$

$$b_1 = a_1 \oplus \overline{a_2} \cdot a_3 \tag{2}$$

Setting up linear equations from the output of χ

Bilinear structure of χ

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

$$b_0 = a_0 \oplus \overline{a_1} \cdot a_2 \tag{1}$$

$$b_1 = a_1 \oplus \overline{a_2} \cdot a_3 \tag{2}$$

Multiplying a_2 to both sides of (2), one obtains:

$$b_1 \cdot a_2 = (a_1 \oplus \overline{a_2} \cdot a_3) \cdot a_2 = a_1 \cdot a_2 \tag{3}$$

Setting up linear equations from the output of χ

Bilinear structure of χ

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

$$b_0 = a_0 \oplus \overline{a_1} \cdot a_2 \quad (1)$$

$$b_1 = a_1 \oplus \overline{a_2} \cdot a_3 \quad (2)$$

Multiplying a_2 to both sides of (2), one obtains:

$$b_1 \cdot a_2 = (a_1 \oplus \overline{a_2} \cdot a_3) \cdot a_2 = a_1 \cdot a_2 \quad (3)$$

and thus according to (1) we obtain

$$b_0 = a_0 \oplus \overline{b_1} \cdot a_2 \quad (4)$$

Given two consecutive bits of the output of χ, one linear equation on the input bits can be set up.

Setting up linear equations from the output of χ

Bilinear structure of χ

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

$$b_0 = a_0 \oplus \overline{a_1} \cdot a_2 \tag{1}$$
$$b_1 = a_1 \oplus \overline{a_2} \cdot a_3 \tag{2}$$

Multiplying a_2 to both sides of (2), one obtains:

$$b_1 \cdot a_2 = (a_1 \oplus \overline{a_2} \cdot a_3) \cdot a_2 = a_1 \cdot a_2 \tag{3}$$

and thus according to (1) we obtain

$$b_0 = a_0 \oplus \overline{b_1} \cdot a_2 \tag{4}$$

Given two consecutive bits of the output of χ, one linear equation on the input bits can be set up.

Preimage attack on 3-round SHAKE128

$64 * 2$ variables, 64 **linear equations**.

Setting up more linear equations

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

Setting 1

Guess $a_{i+1} = 0$ or 1, then b_i becomes linear.

Setting up more linear equations

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

Setting 1

Guess $a_{i+1} = 0$ or 1, then b_i becomes linear.

Setting 2

$b_i = a_i$ holds with probability 0.75 when input bit a_j is uniformly distributed, for all $i \in \{0, ..., 4\}$.

Setting up more linear equations

χ: $b_i = a_i \oplus \overline{a_{i+1}} \cdot a_{i+2}$, and specially we have

Setting 1

Guess $a_{i+1} = 0$ or 1, then b_i becomes linear.

Setting 2

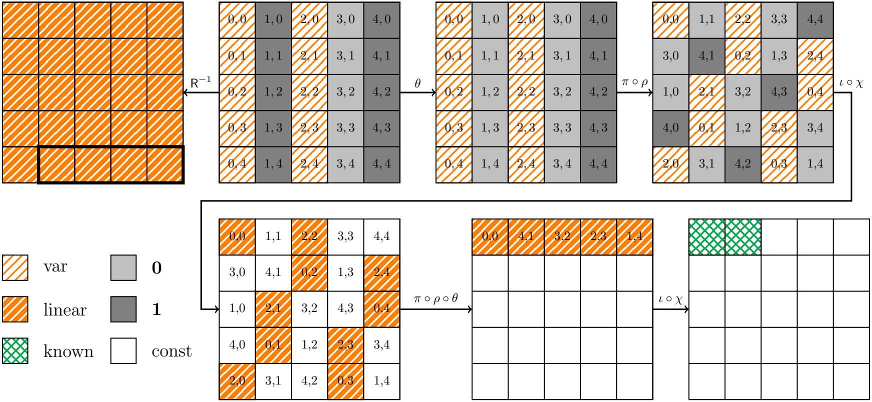
$b_i = a_i$ holds with probability 0.75 when input bit a_j is uniformly distributed, for all $i \in \{0, ..., 4\}$.

Preimage attack on 3-round SHAKE128

Setting 1 $64 * 2$ variables, $64 + 32 + 32$ linear equations (guess 32 bits), complexity 2^{32}

Setting 2 $64 * 2$ variables, $64 + 64$ linear equations, complexity $\frac{1}{0.75^{64}} = 2^{26.6}$ (by changing the constant part)

Preimage Attack on 3-Round SHAKE128 (2)



The degree of freedom: $64 * (10 - 2 - 4) - 6 = 250$
The complexity is 1.

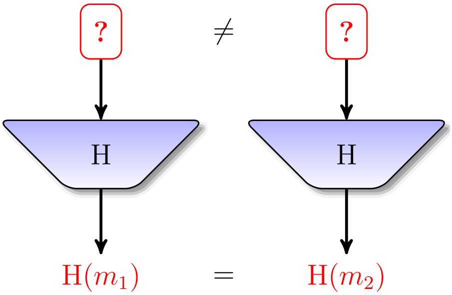
Summary of preimage attacks on SHA-3

Target	#Rounds	Time
SHAKE128	3	1
	4	2^{106}
SHA3-224	2	2^{33}
	3	2^{39}
	4	2^{207}
SHA3-256 / SHAKE256	2	2^{33}
	3	2^{82}
	4	2^{239}
SHA3-384	3	2^{323}
	4	2^{378}
SHA3-512	2	2^{256}
	3	2^{482}
	4	2^{506}

Outline

Security Requirements

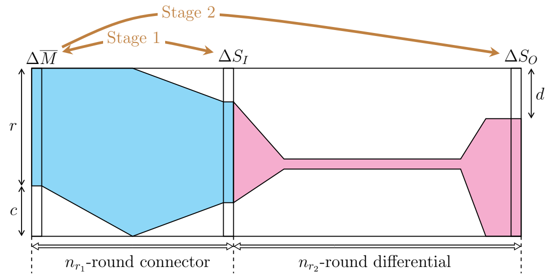
Collision



$\mathbf{2^{n/2}}$

Overview

$(n_{r_1} + n_{r_2})$-round collision attacks



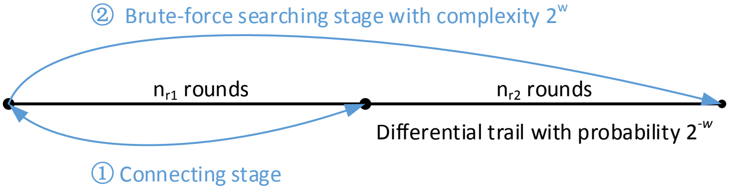
- n_{r_2}**-round differential**: $\Delta S_I \rightarrow \Delta S_O$
- n_{r_1}**-round connector**: A certain procedure which produces message pairs (M_1, M_2) such that

$$\mathtt{R}^{n_{r_1}}(\overline{M_1}||0^c) + \mathtt{R}^{n_{r_1}}(\overline{M_2}||0^c) = \Delta S_I, \quad (\mathtt{R}^i : i \text{ iterations of } \mathtt{R})$$

Overview

$(n_{r_1} + n_{r_2})$-round collision attacks

- Two stages:
  - *Connecting stage.*
    - Construct an n_{r_1}-round connector and get a subspace of messages bypassing the first n_{r_1} rounds.
  - *Brute-force searching stage.*
    - Find a colliding pair following the n_{r_2}-round differential trail from the subspace by brute force.



1-round connector by Dinur *et al.*

Collision attacks on 4-round KECCAK-224/256 (FSE 2012)

- 1-round connector + 3-round differential trail

Properties of KECCAK S-box

Property 1. Given $(\delta^{in}, \delta^{out})$, $V = \{x : \mathtt{S}(x) + \mathtt{S}(x + \delta^{in}) = \delta^{out}\}$ is an affine subspace.

Example

Let $(\delta^{in}, \delta^{out}) = (01, 01)$, then $\mathtt{DDT}(01, 01) = 8$ and $V = \{10, 11, 14, 15, 18, 19, 1C, 1D\}$ is a 3-dimensional affine subspace, defined by

$$\begin{cases} x_1 = 0, \\ x_4 = 1. \end{cases}$$

1-round connector by Dinur *et al.*

Property 2. Given δ^{out}, $T = \{\delta^{in} : \texttt{DDT}(\delta^{in}, \delta^{out}) > 0\}$ contains at least five 2-dimensional affine subspaces.

Example

Suppose $\delta^{out} = 01$. Then, $T = \{01, 09, 0\text{B}, 11, 15, 19, 1\text{B}, 1\text{D}, 1\text{F}\}$.
Among T there are nine 2-dimensional affine subspaces.

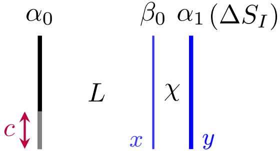
$$\begin{cases} \delta_0^{in} = 1 \\ \delta_1^{in} = 0 \\ \delta_2^{in} = 0 \end{cases} \leftrightarrow T_0 = \{01, 09, 11, 19\}$$

$$\begin{cases} \delta_0^{in} = 1 \\ \delta_1^{in} = 0 \\ \delta_2^{in} + \delta_4^{in} = 0 \end{cases} \leftrightarrow T_1 = \{01, 09, 15, 1D\}$$

$$\vdots$$

$$\begin{cases} \delta_0^{in} = 1 \\ \delta_3^{in} = 1 \\ \delta_4^{in} = 1 \end{cases} \leftrightarrow T_8 = \{19, 1B, 1D, 1F\}$$

1-round connector



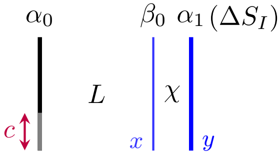
- Choose β_0 s.t. $\Pr(\beta_0 \to \alpha_1) > 0$.
- Derive the solution set V for x.
- For $x \in V$,

$$\chi(x) + \chi(x + \beta_0) = \alpha_1$$

always holds.

[1] p denotes the minimal number of fixed padding bit(s).

1-round connector



- Choose β_0 s.t. $\Pr(\beta_0 \rightarrow \alpha_1) > 0$.
- Derive the solution set V for x.
- For $x \in V$,

$$\chi(x) + \chi(x + \beta_0) = \alpha_1$$

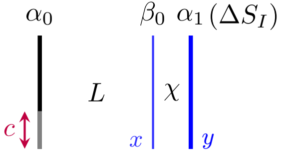
always holds.

How about the $(c + p)$-bit[1] initial constraints?

[1] p denotes the minimal number of fixed padding bit(s).

1-round connector by Dinur *et al.*

The target difference algorithm



- **Difference phase**: find a subspace of input difference β_0 to χ
  - Choose an affine subspace of input differences for each active S-box (*using Property 2*).
  - These β_0s should be compatible with the last $(c+p)$-bit initial difference.
- **Value phase**: by fixing β_0, obtain a subspace of input values to χ that lead to the target difference ΔS_I (*using Property 1*)
  - These input values should be compatible with the last $(c+p)$-bit initial value.

Example

$$? \quad x \xrightarrow{\text{S-box}} y$$

$$? \quad \delta^{in} \xrightarrow{\text{S-box}} \delta^{out} = 01$$

Example

$$? \quad x \xrightarrow{\text{S-box}} \quad y$$

$$? \quad \delta^{in} \xrightarrow{\text{S-box}} \delta^{out} = 01$$

Initialization:

E_Δ: over β_0, initialized with $c + p$ equations concerning the initial difference;

E_M: over x, initialized with $c + p$ equations concerning the initial value.

Example

$$? \quad x \xrightarrow{\text{S-box}} y$$

$$? \quad \delta^{in} \xrightarrow{\text{S-box}} \delta^{out} = 01$$

Initialization:

E_Δ: over β_0, initialized with $c + p$ equations concerning the initial difference;

E_M: over x, initialized with $c + p$ equations concerning the initial value.

Difference phase: Choose a subspace for δ^{in} from $T = \{01, 09, 0B, 11, 15, 19, 1B, 1D, 1F\}$.

$$\begin{cases} \delta_0^{in} = 1 \\ \delta_1^{in} = 0 \\ \delta_2^{in} = 0 \end{cases} \leftrightarrow T_0 = \{01, 09, 11, 19\}$$

$$\vdots$$

$$\begin{cases} \delta_0^{in} = 1 \\ \delta_1^{in} = 0 \\ \delta_2^{in} + \delta_4 = 0 \end{cases} \leftrightarrow T_1 = \{01, 09, 15, 1D\}$$

$$\begin{cases} \delta_0^{in} = 1 \\ \delta_3^{in} = 1 \\ \delta_4^{in} = 1 \end{cases} \leftrightarrow T_8 = \{19, 1B, 1D, 1F\}$$

Suppose T_0 is compatible with E_Δ and is chosen by adding it to E_Δ.

Example

Value phase: From T_0, choose an exact value for δ^{in}.
Suppose 01 is chosen. This means

(1)

$$\begin{cases} \delta_3^{in} = 0 \\ \delta_4^{in} = 0 \end{cases}$$

is compatible with E_Δ and added to E_Δ.

Example

Value phase: From T_0, choose an exact value for δ^{in}.
Suppose 01 is chosen. This means

(1)

$$\begin{cases} \delta_3^{in} = 0 \\ \delta_4^{in} = 0 \end{cases}$$

is compatible with E_Δ and added to E_Δ.

(2)

$$\begin{cases} x_1 = 0 \\ x_4 = 1 \end{cases}$$

is compatible with E_M and added to E_M. It constrains x to

$$V = \{10, 11, 14, 15, 18, 19, 1C, 1D\}$$

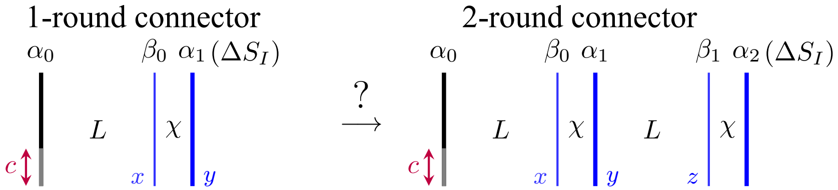
With $x \in V$, Pr(01 $\rightarrow$ 01)=1 for this S-box.

Summary of the 1-Round Connector

- Without the initial E_M and E_Δ, these two phases always succeed.
- The greater the capacity c is, the more difficult it is for the algorithm to succeed.
- Construct a connector by processing linear equations.

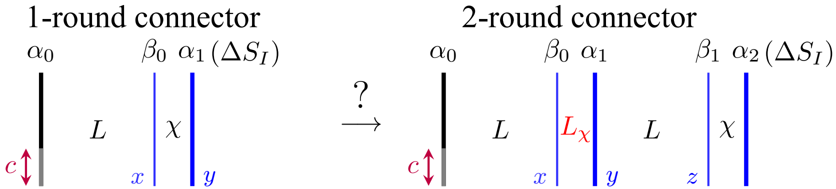
2-Round Connectors

Extending the 1-round connector



2-Round Connectors

Extending the 1-round connector



(Partially) linearize the first round.

S-box linearization

Linearizable subspaces

Definition (Linearizable subspaces)

Given an S-box $\mathtt{S}(\cdot)$, linearizable subspaces are input subspaces V, for which $\exists A, b$, s.t. $\forall x \in V, \mathtt{S}(x) = A \cdot x + b$.

Example

For an input subspace $V = \{0, 1, 4, 5\}$ which is defined by $\{x_1 = 0, x_3 = 0, x_4 = 0\}$, the S-box is equivalent to the linear transformation

$$y = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot x.$$

S-box linearization

Linearizable subspaces

- The largest linearizable subspace is of dimension 2.
- There are totally 80 2-dimensional linearizable affine subspaces.

Table: Linearizable affine subspaces of KECCAK S-box

{**0, 1, 4, 5**}	{2, 3, 6, 7}	{0, 1, 8, 9}	{4, 5, 8, 9}
{1, 2, 9, A}	{0, 3, 8, B}	{1, 3, 9, B}	{2, 3, A, B}
{0, 1, C, D}	{4, 5, C, D}	{8, 9, C, D}	{4, 6, C, E}
{4, 7, C, F}	{5, 7, D, F}	{2, 3, E, F}	{6, 7, E, F}
{0, 2, 10, 12}	{8, A, 10, 12}	{1, 3, 11, 13}	{9, B, 11, 13}
{1, 5, 10, 14}	{2, 4, 12, 14}	{0, 4, 11, 15}	{1, 5, 11, 15}
{10, 11, 14, 15}	{0, 6, 10, 16}	{2, 6, 12, 16}	{3, 7, 12, 16}
{C, E, 14, 16}	{1, 7, 11, 17}	{2, 6, 13, 17}	{3, 7, 13, 17}
{D, F, 15, 17}	{12, 13, 16, 17}	{10, 11, 18, 19}	{14, 15, 18, 19}
{8, A, 18, 1A}	{10, 12, 18, 1A}	{11, 12, 19, 1A}	{10, 13, 18, 1B}
{9, B, 19, 1B}	{11, 13, 19, 1B}	{12, 13, 1A, 1B}	{16, 17, 1A, 1B}
{9, D, 18, 1C}	{A, C, 1A, 1C}	{8, C, 19, 1D}	{9, D, 19, 1D}
{10, 11, 1C, 1D}	{14, 15, 1C, 1D}	{18, 19, 1C, 1D}	{8, E, 18, 1E}
{B, F, 1A, 1E}	{4, 6, 1C, 1E}	{C, E, 1C, 1E}	{14, 16, 1C, 1E}
{9, F, 19, 1F}	{A, E, 1B, 1F}	{B, F, 1B, 1F}	{14, 17, 1C, 1F}
{D, F, 1D, 1F}	{15, 17, 1D, 1F}	{12, 13, 1E, 1F}	{16, 17, 1E, 1F}
{0, 2, 8, A}	{6, 7, A, B}	{5, 6, D, E}	{A, B, E, F}
{0, 4, 10, 14}	{3, 5, 13, 15}	{4, 6, 14, 16}	{5, 7, 15, 17}
{0, 2, 18, 1A}	{1, 3, 19, 1B}	{8, C, 18, 1C}	{B, D, 1B, 1D}
{A, E, 1A, 1E}	{15, 16, 1D, 1E}	{5, 7, 1D, 1F}	{1A, 1B, 1E, 1F}

S-box linearization

Linearizable Subspace and DDT

Observation

For an active KECCAK *S-box,* $V = \{x : \mathtt{S}(x) + \mathtt{S}(x + \delta^{in}) = \delta^{out}\}$

1. *if* $\mathtt{DDT}(\delta^{in}, \delta^{out}) = 2$ *or* 4*, then* V *is a linearizable affine subspace.*
2. *if* $\mathtt{DDT}(\delta^{in}, \delta^{out}) = 8$*, then among* V *there are six 2-dimensional subsets* $W_i \subset V, i = 0, \cdots, 5$ *such that* W_i *are linearizable affine subspaces.*

Example

$\mathtt{DDT}(01, 01) = 8$, $V = \{10, 11, 14, 15, 18, 19, 1C, 1D\}$, w_i's are

$$\{10, 11, 14, 15\}, \{10, 11, 18, 19\}, \{10, 11, 1C, 1D\},$$
$$\{14, 15, 18, 19\}, \{14, 15, 1C, 1D\}, \{18, 19, 1C, 1D\}.$$

Drawback of S-box Linearization

- Each 5-bit S-box allows a linearizable subspace of dimension at most 2.
- Full linearization of two rounds is impossible, since 3/5 degree of freedom is lost in each round of linearization. Hence 3-round connectors are impossible.

Non-Full S-box Linearization

Two Observations - 1

Observation

For a non-active KECCAK S-box, when $U_i \neq \texttt{1F}$,

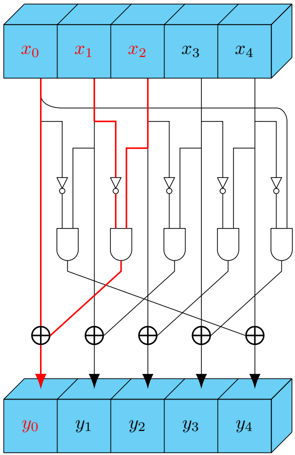
1. *if $U_i = 0$, it does not require any linearization;*
2. *if $U_i \in T, T = \{\texttt{01}, \texttt{02}, \texttt{04}, \texttt{08}, \texttt{10}, \texttt{03}, \texttt{06}, \texttt{0C}, \texttt{11}, \texttt{18}\}$, at least 1 equation should be added to E_M to linearize the output bit(s) of the S-box marked by U_i;*
3. *otherwise, at least 2 equations should be added to E_M to linearize the output bits of the S-box marked by U_i.*

Example

Suppose $U_i = 1$.

Linearization of $y_0 = x_0 + (x_1 + 1) \cdot x_2$

No.	constraint	linear mapping
1	$x_1 = 0$	$y_0 = x_0 + x_2$
2	$x_1 = 1$	$y_0 = x_0$
3	$x_2 = 0$	$y_0 = x_0$
4	$x_2 = 1$	$y_0 = x_0 + x_1 + 1$
5	$x_1 + x_2 = 0$	$y_0 = x_0$
6	$x_1 + x_2 = 1$	$y_0 = x_0 + x_2$



Non-Full S-box Linearization

Two Observations - 2

Observation

Given $(\delta^{in}, \delta^{out})$ *such that* $\texttt{DDT}(\delta^{in}, \delta^{out}) = 8$, *4 out of 5 output bits are already linear if the input is chosen from the solution set* $V = \{x \mid \texttt{S}(x) + \texttt{S}(x + \delta^{in}) = \delta^{out}\}$.

Example

$\texttt{DDT}(\texttt{01}, \texttt{01}) = 8$ and $V = \{\texttt{10}, \texttt{11}, \texttt{14}, \texttt{15}, \texttt{18}, \texttt{19}, \texttt{1C}, \texttt{1D}\}$. The algebraic expressions of the S-box are reduced to

$$\begin{aligned} y_0 &= x_0 + x_2, \\ y_1 &= (x_2 + 1) \cdot x_3, \\ y_2 &= x_2 + x_3 + 1, \\ y_3 &= x_3, \\ y_4 &= 1. \end{aligned}$$

Non-Full S-box Linearization

Table: #equations added to E_M to partially linearize an S-box

non-active		active	
U_i	#equations	DDT	#equations
1F	3	2	4
0	0	4	3
T	1	8	2,3
others	2		

- Less degrees of freedom are consumed for non-full S-box linearizations.

Timings for Practical Collision Attacks

Table: Collision attacks using 2-/3-round connectors

Target $[r, c, d]$	n_r	Searching Complexity	Searching Time	Connecting Time
KECCAK[1440,160,160]	5	2^{40}	2.48h	9.6s
	6	$2^{51.14}$	**112h** †	4.5h‡
KECCAK[640,160,160]	5	2^{35}	2.67h	30m
SHAKE128	5	2^{39}	30m	25m
SHA3-224	5	$2^{36.7}$	29h	11.7h
SHA3-256	5	$2^{36.7}$	45.6h	428.8h

† Use the GPU implementation: 3 GTX970 GPUs for KECCAK[1440,160,160] and 1 GTX1070 GPU for SHA3-224.
‡ For constructing the 2-round connector.

Summary of Collision Attacks

Target[r, c, d]	n_r	Complexity
KECCAK[1024]	3	Practical
KECCAK[768]	3	Practical
KECCAK[768]	4	2^{147}
KECCAK[512]	5	Practical
KECCAK[448]	5	Practical
SHA3-256	5	Practical
SHA3-224	5	Practical
SHAKE128	5	Practical
KECCAK[1440, 160, 160]	6	Practical
KECCAK[640, 160, 160]	5	Practical
KECCAK[240, 160, 160]	4	Practical
KECCAK[40, 160, 160]	1	Practical
KECCAK[40, 160, 160]	2	2^{73}

Outline

Summary

- Linearization is widely used in both collision and preimage attacks.
- Using GPU
  - Searching good differential trails
  - Solving systems of linear equations
- Main results
  - Preimages can be found for up to 4 (out of 24) rounds.
  - Collisions can be found for up to 6 (out of 24) rounds.
- Require new ways of exploiting degrees of freedom.

Thank you for your attention!

Q & A