Secure Hash Algorithm 3 (SHA3)

Ching Tzu Chen

Cryptanalysis Laboratory, Department of Computer Science, National Chiao Tung University

May 2016

Outline

- Introduction
- Sponge Construction
- Iteration function
- References

Introduction

- In 2006, NIST announced to hold *NIST hash function competition* to create a new hash standard SHA3
- In 2008 and 2009, Keccak was selected as one of the candidates and advanced to the last round in 2010
- In 2012, Keccak was selected as the winner of the SHA-3 competition
- In 2015, NIST announced that SHA-3 had become a hashing standard

Introduction

NIST didn't mean to replace SHA-2 by SHA-3 since no significant attack on SHA-2 has been revealed.

It's because of the total break on MD-5 and SHA-0, the theoretical attack on SHA-1, NIST sought to find an alternatively secure hash different from SHA-2, which become SHA-3

Introduction

Keccak won because of:

- High security, high quality
- Elegant, clean design
- Best hardware performance
- Design diversity from SHA-2

Sponge Construction

The basic scheme of SHA3 is based on Sponge construction, which can be defined by the 3 parameters:

1. f : The internal function used to process each input block
2. r : The size of bit of the input block, also called **bitrate**
3. pad : The padding algorithm

Sponge Construction

An input message of n bits will be partitioned into k fixed-size blocks of r bit each, say,

$$Input = P_0||P_1||...||P_{k-1}$$

where P_i is r-bit, $Input$ is n-bit, and so $n = r \times k$

Sponge Construction

In SHA-3, a padding is always added,
in the case of $n \mod r = 0$, a padding block of r bit is added.

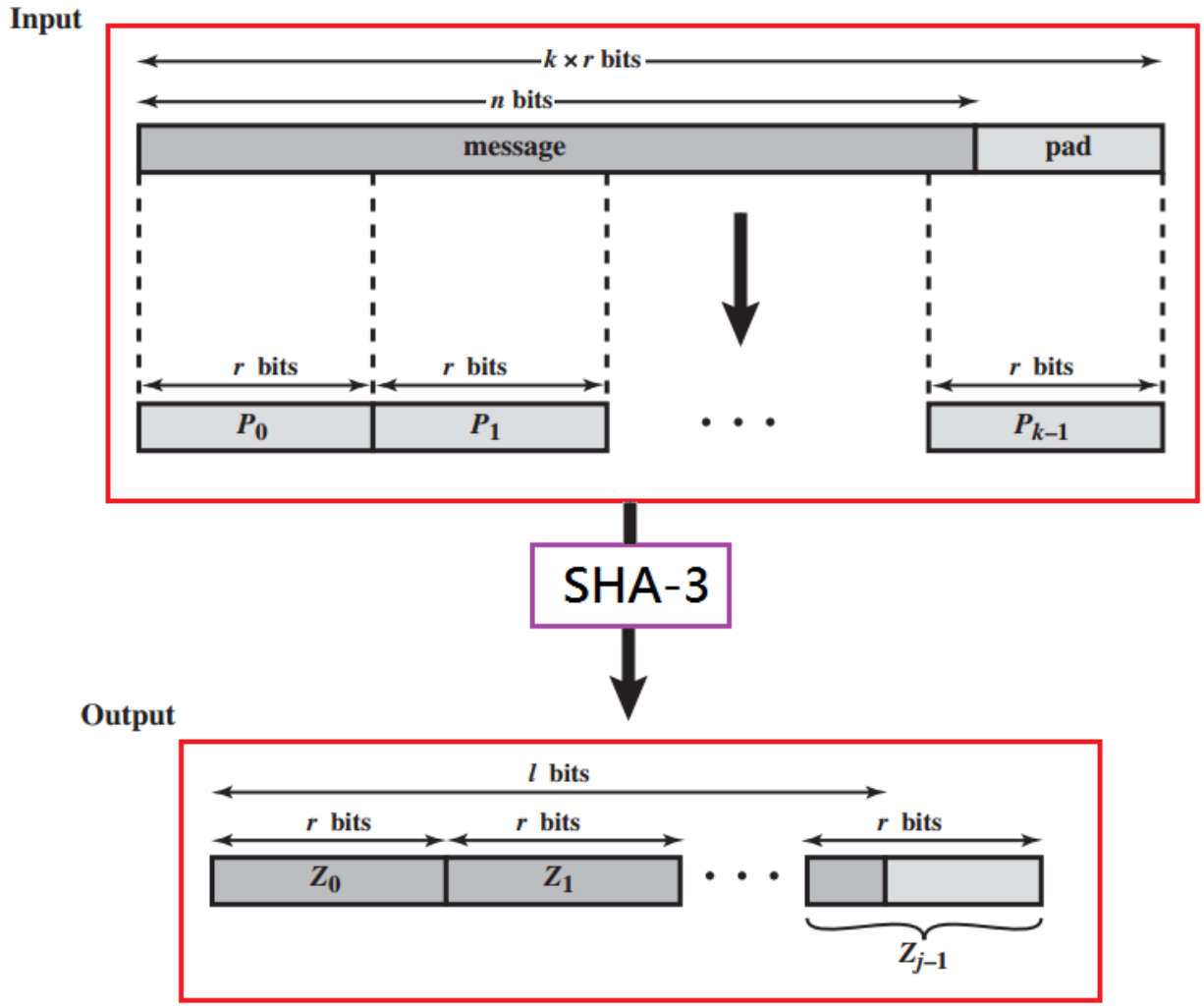
Two padding algorithms in sponge specification are provided:

1. **Simple padding**: Appends 10^* to the message M such that resulting length is $r \times k$ bits
2. **Multirate padding**: Appends 10^*1 to the message M such that resulting length is $r \times k$ bits

Sponge Construction

After processing, the Sponge function generates the sequence of blocks $Z_0, Z_1, ..., Z_{j-1}$ as output, where j is the number of output blocks such that $(j-1) \times r < l \leq j \times r$ if the desired output is l bits.

Sponge Construction



Sponge Construction

In each iteration of Sponge construction, the sponge operates on a state variable s of $b = r + c$ bits, which is initialized to all zeros.
Note that r is the bitrate and c is referred to as the **capacity**, which is the achievable level of security.

Sponge Construction

The Sponge construction consists of two phases:

1. **Absorbing phase**
2. **Squeezing phase**

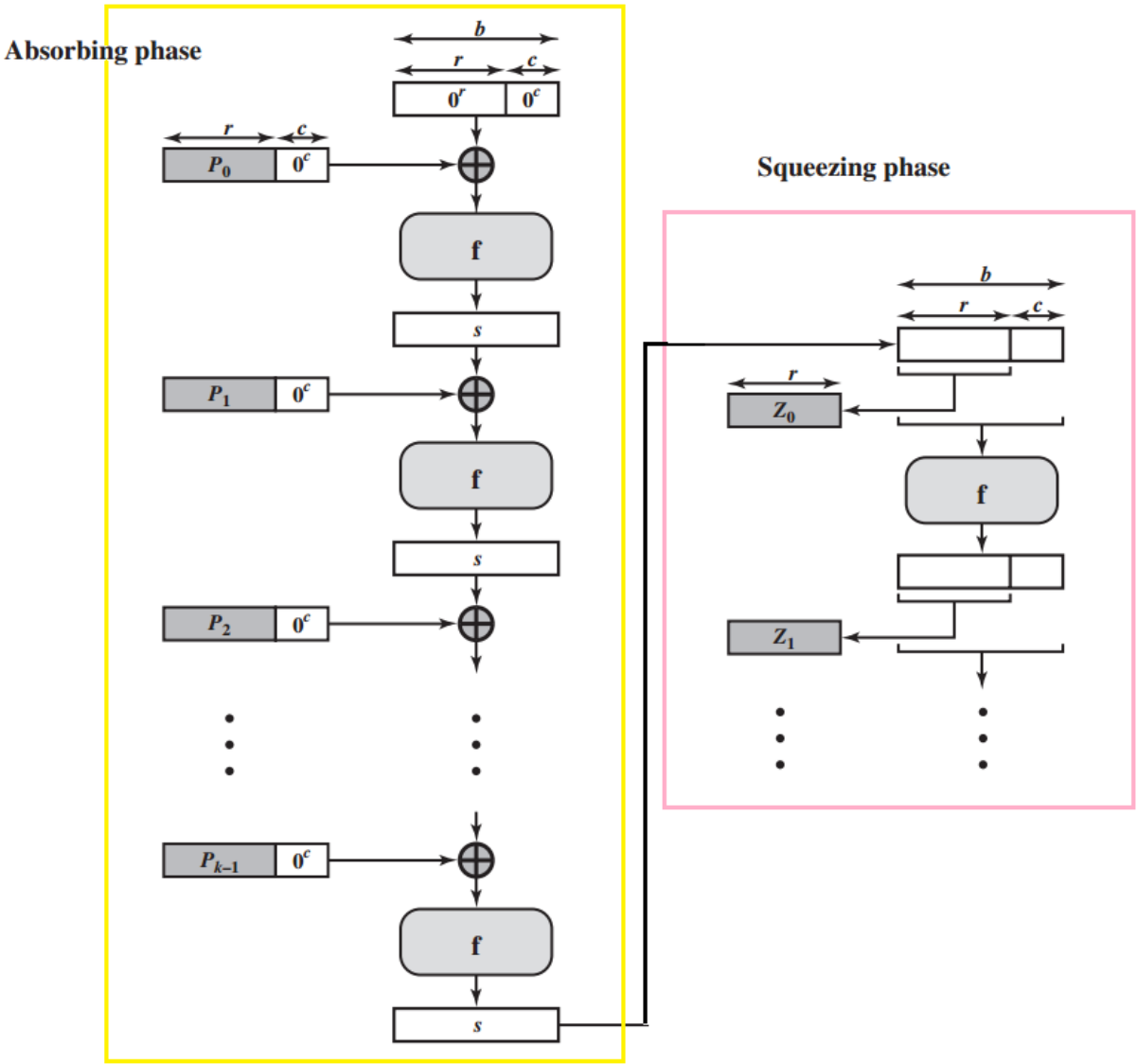
Sponge Construction

1. **Absorbing phase**: In each iteration, the input block to be processed is padded with zeroes to extend its length to b bits. Then XOR it with the state variable s, this forms b bits as input to the iteration function f, the output of f is the value of the next s.

Sponge Construction

2. **Squeezing phase**: If the desired l-bit output satisfies $l \leq r$, it will simply return the first l bits of s at the completion of the absorbing phase.
Otherwise, The first r bits of s are retained as block Z_0, then s is updated by executing with f, and to obtain the next r bits of s as Z_1, until there're j blocks such that $(j-1) \times r < l \leq j \times r$. Then returns the first l bits of the concatenation of these Z_i.

Sponge Construction



Sponge Construction

Algorithm The sponge construction SPONGE[f, pad, r]
Require: $r < b$

Interface: $Z = \text{sponge}(M, \ell)$ with $M \in Z_2^*$, integer $\ell > 0$ and $Y \in Z_2^\ell$
$P = M \| \text{pad}[r](|M|)$
$s = 0^b$
for $i = 0$ to $|P|_r - 1$ **do**
  $s = s \oplus (Pi \| 0^{b-r})$
  $s = f(s)$
end for
$Z = \lfloor s \rfloor_r$
while $|Z|_r \; r < \ell$ **do**
  $s = f(s)$
  $Z = Z \| \lfloor s \rfloor_r$
end while
return $\lfloor Z \rfloor_\ell$

Notation:

- $|M|$: the length in bits of a bit string M
- $|M|_x$: the number of fix-length block with length x in M
- $\lfloor M \rfloor_l$: the truncation of M to its first l bits

Sponge Construction

SHA-3 parameters

Message Digest Size	224	256	384	512
Message Size	no maximum	no maximum	no maximum	no maximum
Block Size (bitrate *r*)	1152	1088	832	576
Word Size	64	64	64	64
Number of Rounds	24	24	24	24
Capacity *c*	448	512	768	1024
Collision Resistance	2^{112}	2^{128}	2^{192}	2^{256}
Second Preimage Resistance	2^{224}	2^{256}	2^{384}	2^{512}

The above are all measured in bits

The default value for SHA-3 are $c = 1024$, $r = 576$ and therefore $b = 1600$

Iteration function

Internal function f takes a 1600-bit variable s as input, s is organized as a $5 \times 5 \times 64$ array a, The 64-bit units are referred as **lanes**.

For convenience, we denotes:

- $a[x, y, z]$: an individual bit in state array.
- $L[x, y]$: a 5×5 matrix where each entry is a 64-bit lane.

The mapping between the bits of s and bits of a is :

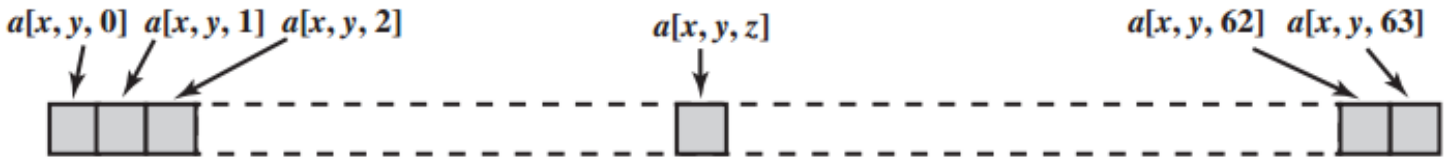
$$s[64(5y + x) + z] = a[x, y, z]$$

Iteration function

State variable as 5 × 5 matrix A of 64-bit words

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[4, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

Bit labeling of 64-bit words



Iteration function

Structure of f

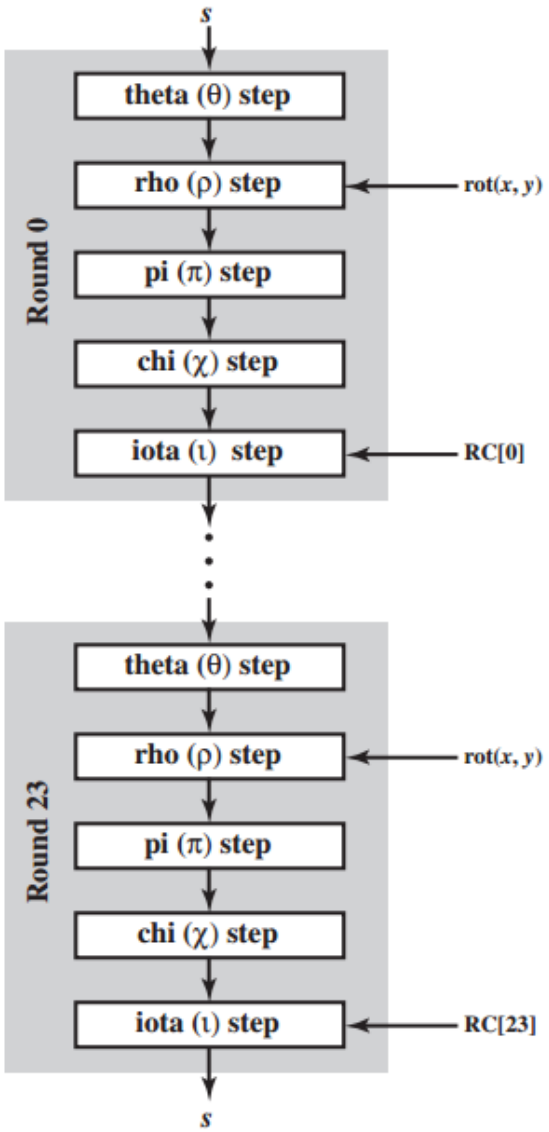
f takes 1600-bit state variable as input, and converts it into a 5×5 matrix of 64-bit lanes, this matrix then passes through 24 rounds of processing, each round consists of 5 steps.
Suppose a round is denoted by a function R, then we have

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

Iteration function

Structure of f

Function	Type
θ	Substitution
ρ	Permutation
π	Permutation
χ	Substitution
ι	Substitution



Iteration function

θ step function

For a bit in $a[x, y, z]$, θ does the following:

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

Note that **Addition** and **XOR** have the same effect on $\mathbb{Z}_2$

Iteration function

More clear explanation of θ

Let $C[x] = \sum_{i=0}^{4} L[x, i]$

- $\sum_{y=0}^{4} a[(x-1), y, z]$: Performing XOR of the lanes in column $(x-1) \mod 5$ to form $C[x-1]$
- $\sum_{y=0}^{4} a[(x+1), y, (z-1)]$: Performing XOR of the lanes in column $(x+1) \mod 5$, and then rotate bits in lanes such that the bit in position z is mapped into position $z+1 \mod 64$, this forms $ROT(C[x+1], 1)$

Iteration function

Illustration of θ

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[3, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

$$L[2, 3] \leftarrow C[1] \oplus L[2, 3] \oplus \mathrm{ROT}(C[3], 1)$$

Iteration function

ρ step function

The ρ function is defined as follows:

$$\rho : a[x, y, z] \leftarrow a[x, y, z] \text{ if } x = y = 0$$
$$\rho : a[x, y, z] \leftarrow a[x, y, (z - \frac{(t+1)(t+2)}{2}) \mod 64] \text{ otherwise}$$

where t satisfying $0 \leq t < 24$ and

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}^t \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \in {\mathbb{Z}_5}^{2\times 1}$$

Iteration function

More clear explanation of ρ

- The lane $L[0, 0]$ is unaffected, other lanes will be performed by a circular bit shift
- t used to compute the amount of circular bit shift and which lane is assigned which shift value
- As t varies from 0 to 23, each iteration will be performed the respective bit shift value $\frac{(t+1)(t+2)}{2} \mod 64$

An example of ρ

Suppose $t = 3$, we have

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mod 5 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

so $a[1, 2, z]$ will be shifted to $a[1, 2, (z - \frac{4 \times 5}{2}) \mod 64]$ for all $z = 0, 1, ...63$

Iteration function

Rotation value of ρ

t	$g(t) \bmod 64$	x, y
0	1	1, 0
1	3	0, 2
2	6	2, 1
3	10	1, 2
4	15	2, 3
5	21	3, 3
6	28	3, 0
7	36	0, 1
8	45	1, 3
9	55	3, 1
10	2	1, 4
11	14	4, 4

t	$g(t) \bmod 64$	x, y
12	27	4, 0
13	41	0, 3
14	56	3, 4
15	8	4, 3
16	25	3, 2
17	43	2, 2
18	62	2, 0
19	18	0, 4
20	39	4, 2
21	61	2, 4
22	20	4, 1
23	44	1, 1

Note: $g(t) = (t + 1)(t + 2)/2$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bmod 5$$

Rotation values by word position in matrix

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	18	2	61	56	14
$y = 3$	41	45	15	21	8
$y = 2$	3	10	43	25	39
$y = 1$	36	44	6	55	20
$y = 0$	0	1	62	28	27

Iteration function

π step function

The π function is defined as follows:

$$\pi\colon\ L[x, y] \leftarrow L[x^*, y^*]$$

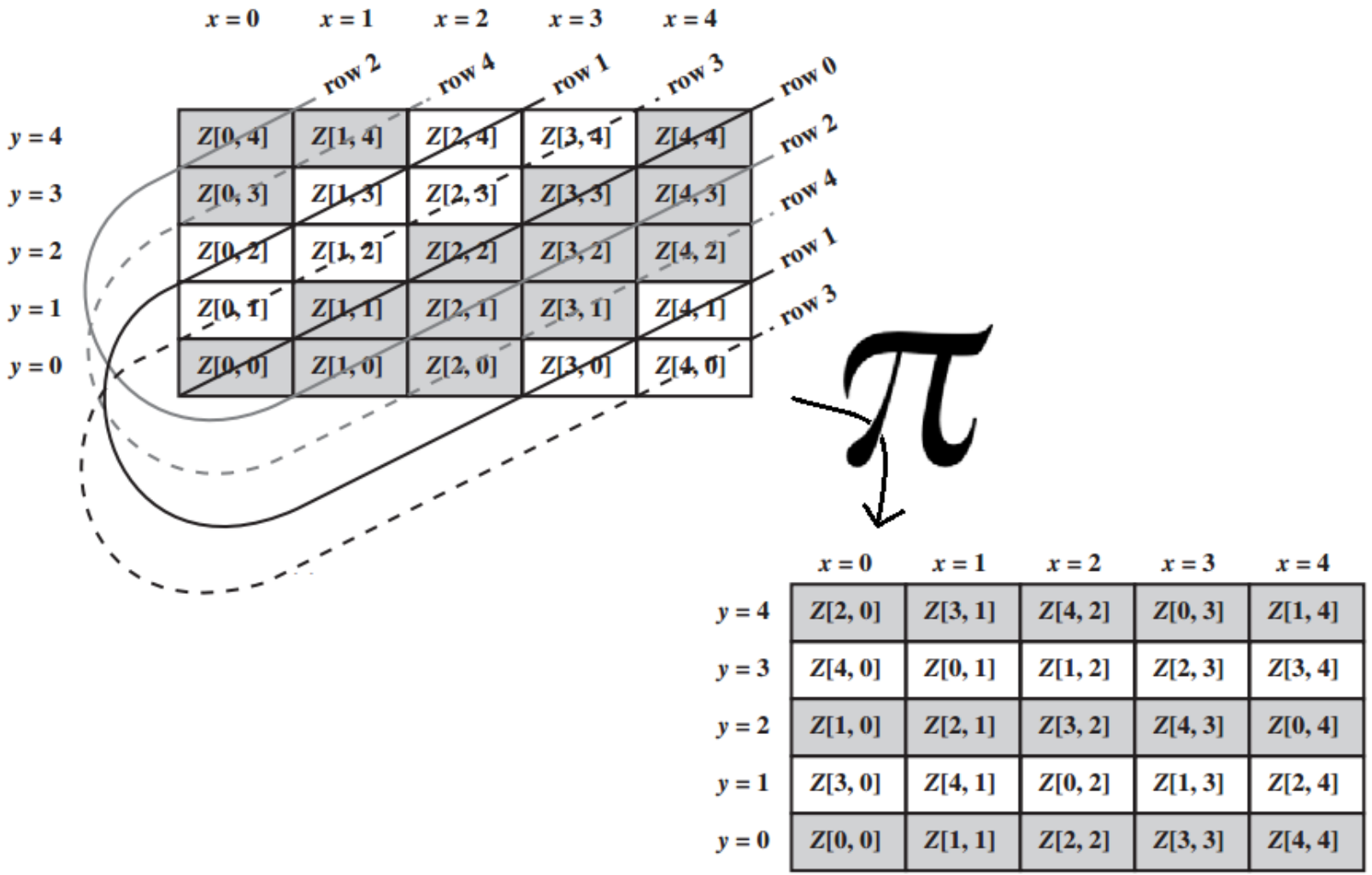
$$\text{where } \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \end{bmatrix} \mod 5$$

π simply permutes the 5×5 matrix by:

- The new x position is the old y position
- The new y position is determined by $(2x + 3y) \mod 5$

Iteration function

Illustration of π



Iteration function

χ step function

The χ function is defined as follows:

$$\chi\colon\ a[x,y,z] \leftarrow$$
$$a[x,y,z] \oplus (\mathrm{NOT}(a[x+1,y,z])\ \mathrm{AND}\ a[x+2,y,z])$$

Note that χ it's only function in SHA-3 which provides the non-linear mapping on bits.

Iteration function

Illustration of χ

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[4, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

$$L[2, 3] \leftarrow L[2, 3] \oplus \left(\overline{L[3, 3]} \text{ AND } L[4, 3] \right)$$

Iteration function

ι step function

The ι function is defined as follows:

$$\iota: \ L[0,0] \leftarrow L[0,0] \oplus \mathrm{RC}[i]$$

where $0 \leq i < 24$ and RC is the array of 64−bit round constants.

Note that ι only XOR RC with the first lane of the state variable. And $L[0,0]$ will be XORed with different value of $\mathrm{RC}[i]$ in each round.

Iteration function

RC array

Round Constants in SHA-3

Round	Constant (hexadecimal)	Round	Constant (hexadecimal)
0	0000000000000001	12	000000008000808B
1	0000000000008082	13	800000000000008B
2	800000000000808A	14	8000000000008089
3	8000000080008000	15	8000000000008003
4	000000000000808B	16	8000000000008002
5	0000000080000001	17	8000000000000080
6	8000000080008081	18	000000000000800A
7	8000000000008009	19	800000008000000A
8	000000000000008A	20	8000000080008081
9	0000000000000088	21	8000000000008080
10	0000000080008009	22	0000000080000001
11	000000008000000A	23	8000000080008008

References

[1] https://en.wikipedia.org/wiki/SHA-3

[2] Cryptography and Network Security 6th edition by William Stallings
(All figures in the slides are excerpted from it.)