KU LEUVEN

COSIC

Rhythmic Keccak:
SCA Security and Low Latency in HW

Victor Arribas, Begül Bilgin, George Petrides,
Svetla Nikova, Vincent Rijmen

Rhythmic Keccak

Hackers can guess PINs using your smartphone's sensor data

Data from your smartphone sensors can reveal PINs and passwords to hackers and allow them to unlock your mobile devices, according to a study led by an Indian-origin scientist.

Source :https://www.financialexpress.com/industry/technology/hackers-can-guess-pins-using-your-smartphones-sensor-data/991535/

Rhythmic Keccak

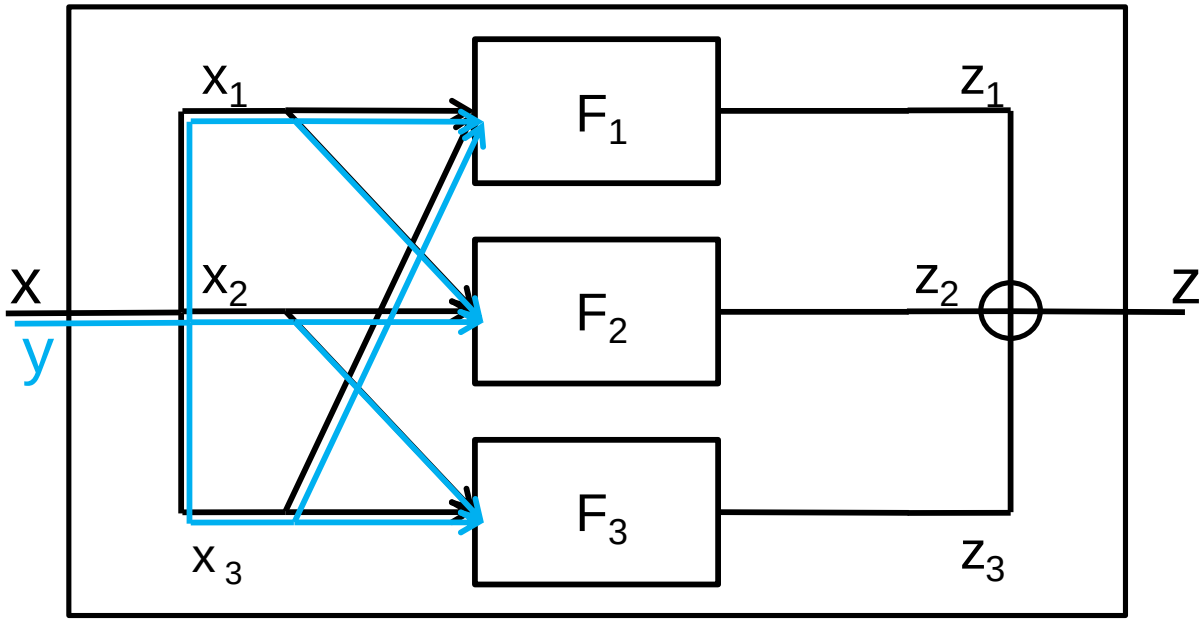
- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

Masking

Threshold Implementations (TI) [NRR06]



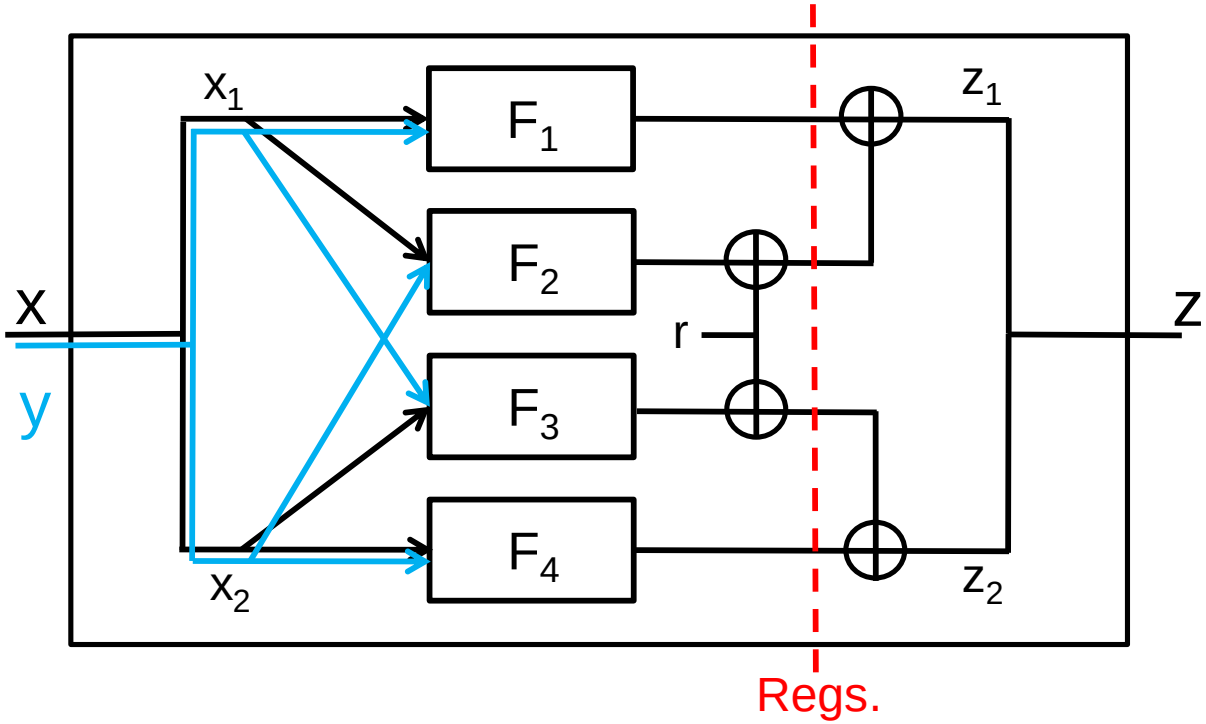
$s_{in} \geq t \cdot d + 1$ dth-order Non-Completeness [BGN+14]

[NRR06] *S. Nikova, C. Rechberger, and V. Rijmen. Threshold implementations against side-channel attacks and glitches. In ICICS 2006.*

[BGN+14] *B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. A more efficient AES threshold implementation. In AFRICACRYPT 2014.*

Masking

Domain Oriented Masking (DOM) [GMK16]



$s_{in} = d + 1$ Layer of registers Indep. inputs

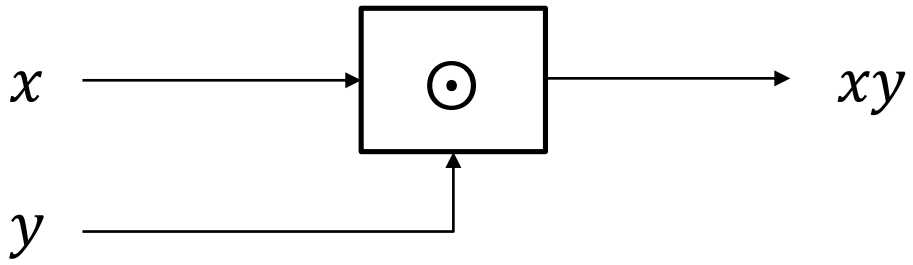
[GMK16] *H. Gross, S. Mangard, and T. Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In Cryptology ePrint, Report 2016/486.*

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

Input dependencies

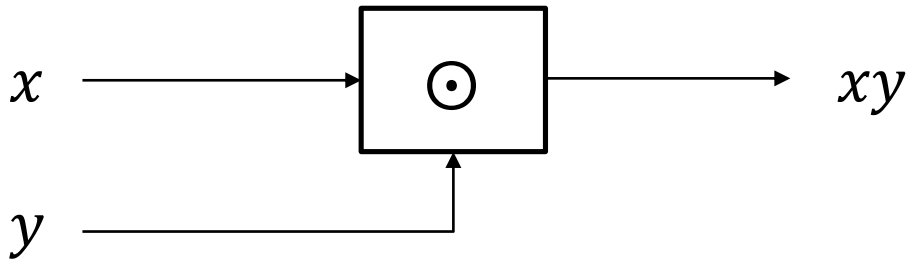
Example: calculating xy



$$x \odot y = \begin{bmatrix} y_1 \cdot x_1 & y_1 \cdot x_2 \\ y_2 \cdot x_1 & y_2 \cdot x_2 \end{bmatrix}$$

Input dependencies

Example: calculating xy

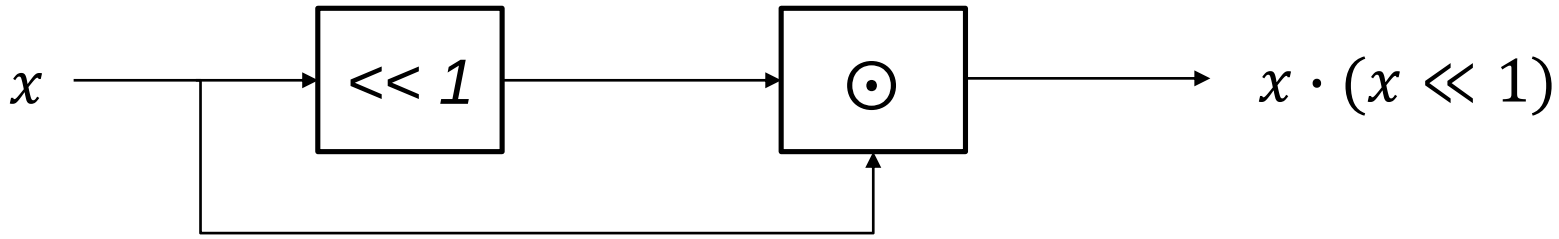


$$x \odot y = \begin{bmatrix} y_1 \cdot x_1 & \boldsymbol{y_1 \cdot x_2} \\ \boldsymbol{y_2 \cdot x_1} & y_2 \cdot x_2 \end{bmatrix}$$

Non-completeness OK

Input dependencies

Example: calculating $x \cdot (x \ll 1)$

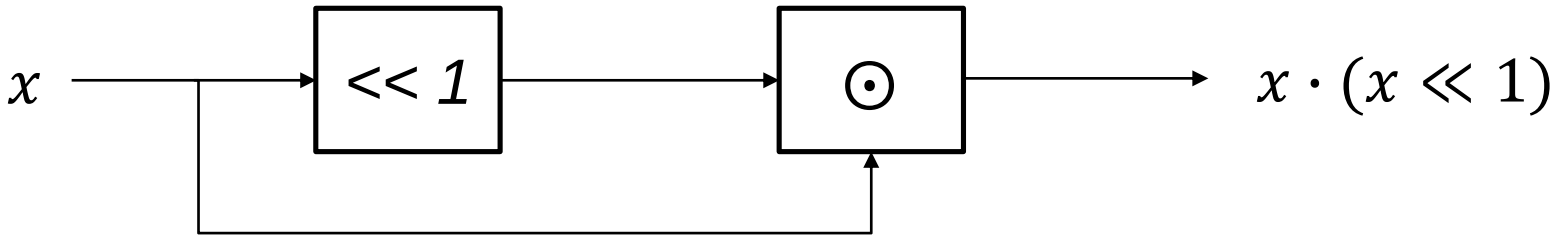


$$x = (x_1, x_2) \Longrightarrow x \ll 1 = (x_1 \ll 1, x_2 \ll 1)$$

$$x \odot (x \ll 1) = \begin{bmatrix} (x_1 \ll 1) \cdot x_1 & (x_1 \ll 1) \cdot x_2 \\ (x_2 \ll 1) \cdot x_1 & (x_2 \ll 1) \cdot x_2 \end{bmatrix}$$

Input dependencies

Example: calculating $x \cdot (x \ll 1)$



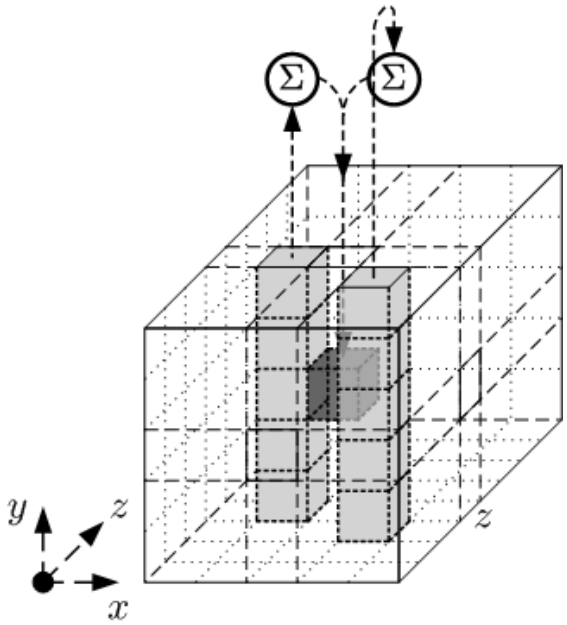
$$x = (x_1, x_2) \Longrightarrow x \ll 1 = (x_1 \ll 1, x_2 \ll 1)$$

$$x \odot (x \ll 1) = \begin{bmatrix} (x_1 \ll 1) \cdot x_1 & \boldsymbol{(x_1 \ll 1) \cdot x_2} \\ \boldsymbol{(x_2 \ll 1) \cdot x_1} & (x_2 \ll 1) \cdot x_2 \end{bmatrix}$$

Non-completeness FAILS

Input deps. In Keccak

Dependencies introduced by θ



Round-based implementations

Non-completeness flaw in round-based implement. [GMS17] using DOM

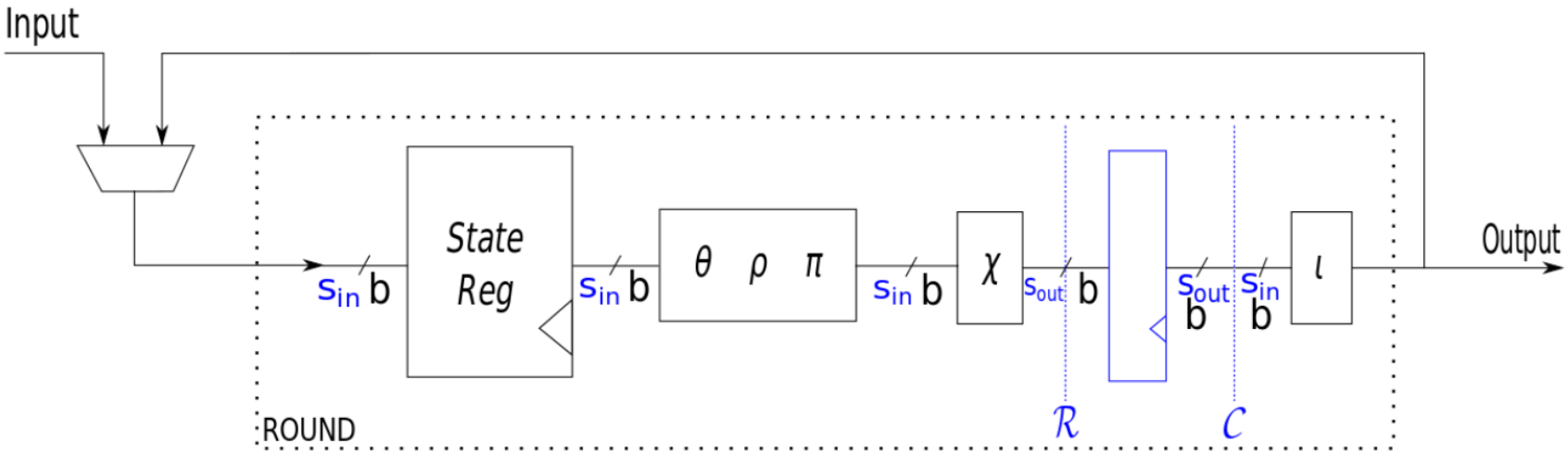
Potentially exploitable leakage

[GMS17] *H. Gross, D. Schaffenrath, and S. Mangard. Higher-order side-channel protected implementations of Keccak. In Euromicro DSD 2017.*

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

DOM-indep. Keccak

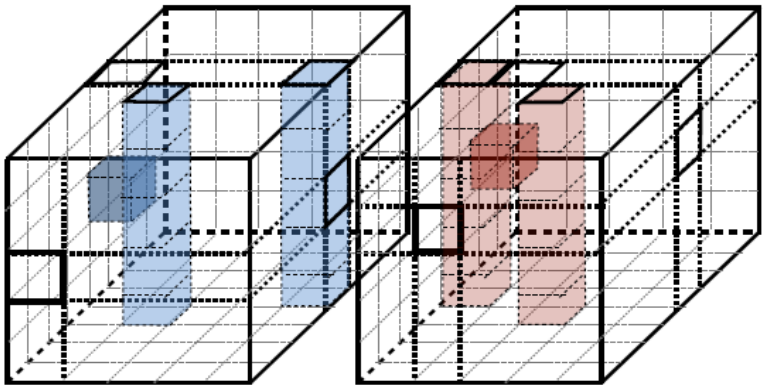


Non-completeness broken when computing χ

Keccak

$$\theta_1 \quad : S_2^{\theta}[3,0,4] \rightarrow S_2[3,0,4] \oplus \bigoplus_{y'=0}^{4} S_2[2,y',4] \oplus \bigoplus_{y'=0}^{4} S_2[4,y',3]$$

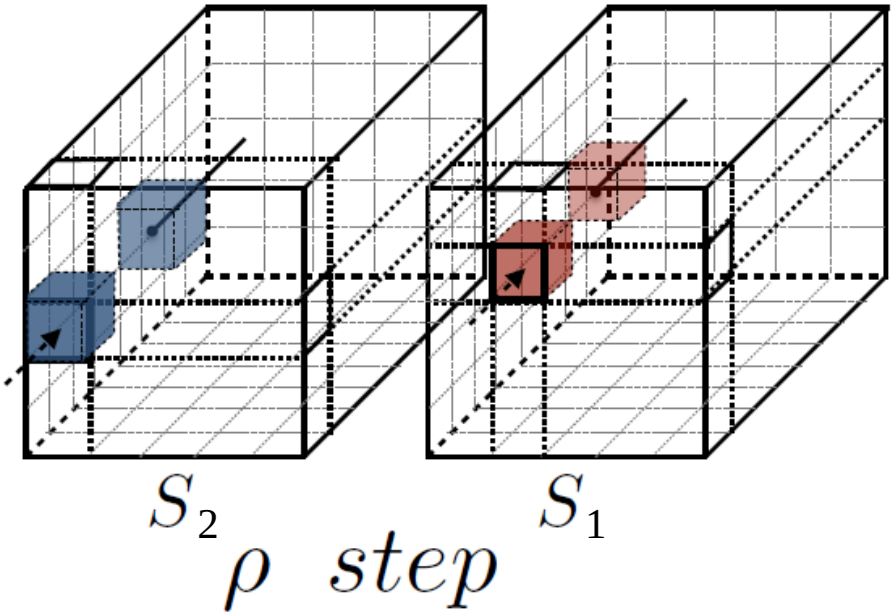
$$S_1^{\theta}[4,1,4] \rightarrow S_1[4,1,4] \oplus \bigoplus_{y'=0}^{4} S_1[3,y',4] \oplus \sum_{y'=0}^{4} S_1[0,y',3]$$



θ *step*

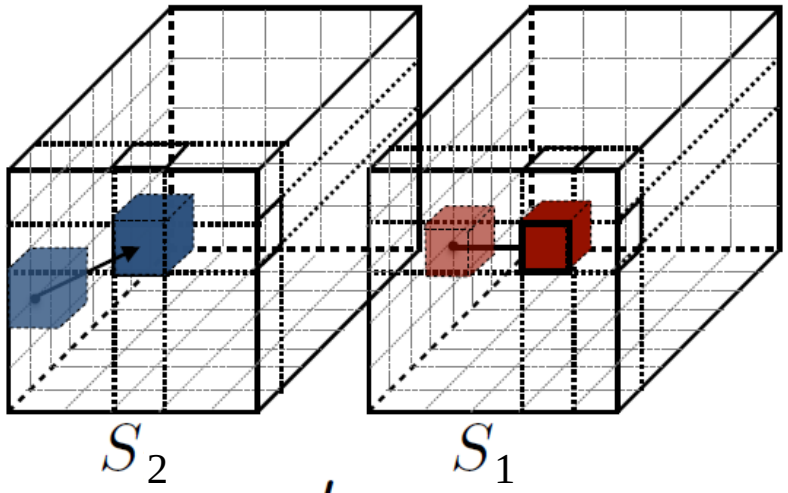
Keccak

$$\rho \quad : \quad S_2^{\rho}[3,0,0] \leftarrow S_2^{\theta}[3,0,4]$$
$$S_1^{\rho}[4,1,0] \leftarrow S_1^{\theta}[4,$$



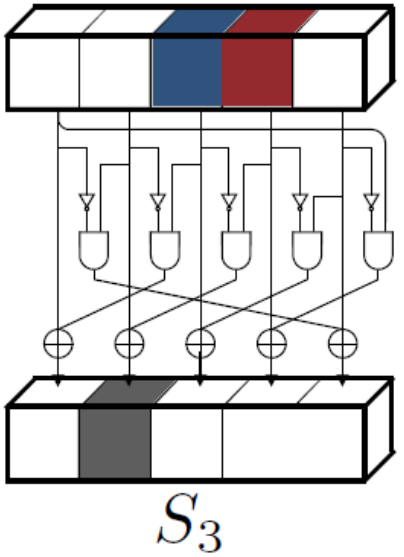
Keccak

$$\pi \quad : \quad S_2^{\pi}[0,1,0] \leftarrow S_2^{\rho}[3,0,0]$$
$$S_1^{\pi}[1,1,0] \leftarrow S_1^{\rho}[4,1,0]$$



π *step*

Total of 112 bits found (200 bit state) [ANR17]



χ *step*

$$\chi \quad : \quad S_3^{\chi}[4,1,0] \leftarrow S_2^{\pi}[0,1,0]\, S_1^{\pi}[1,1,0] \oplus r$$

[ANR17] *V. Arribas, S. Nikova and Vincent Rijmen. VerMI: Verification Tool for Masked Implementations. In Cryptology ePrint, Report 2017/1227.*

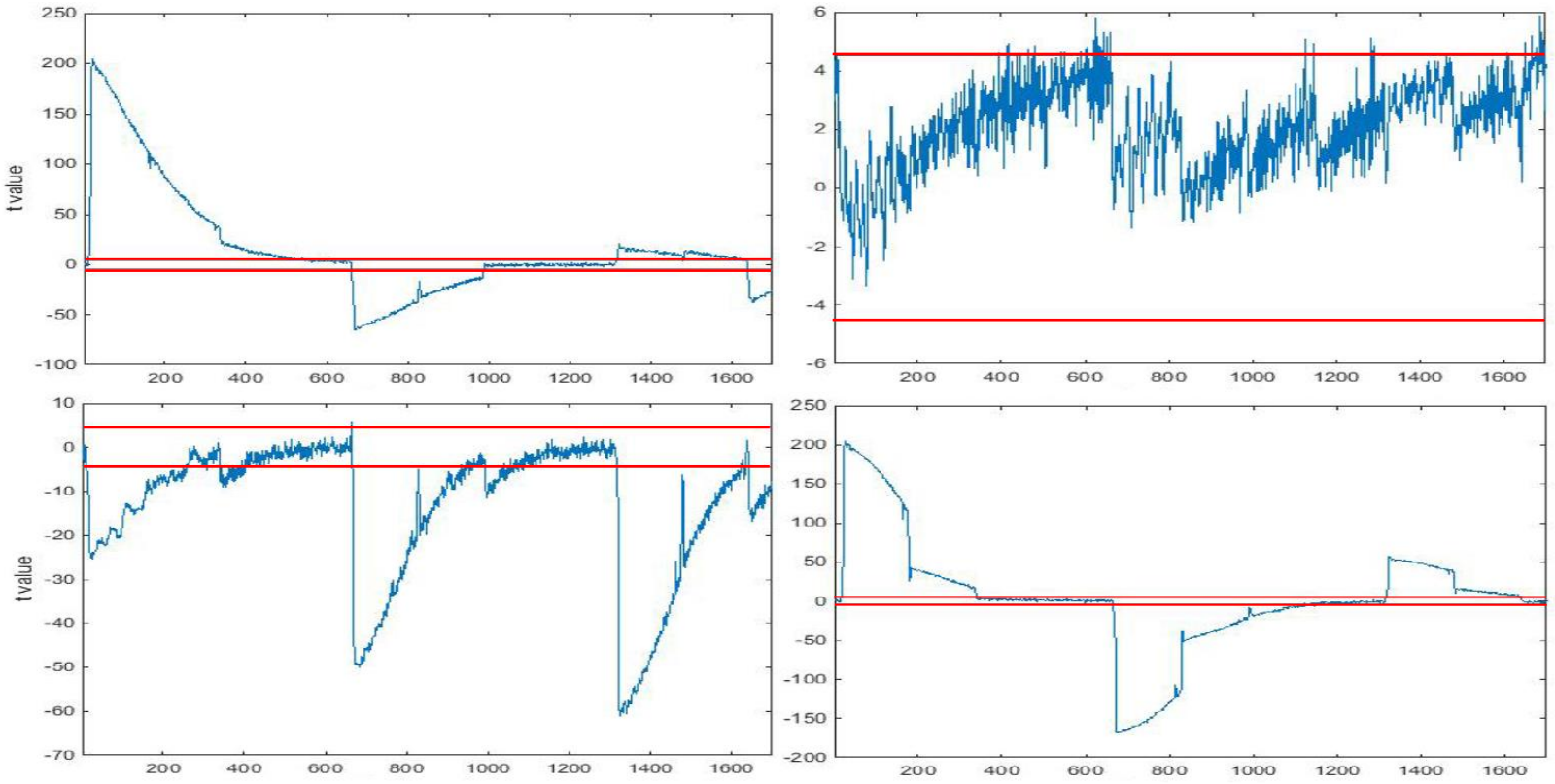
DOM-Keccak

Masks Off

Masks On

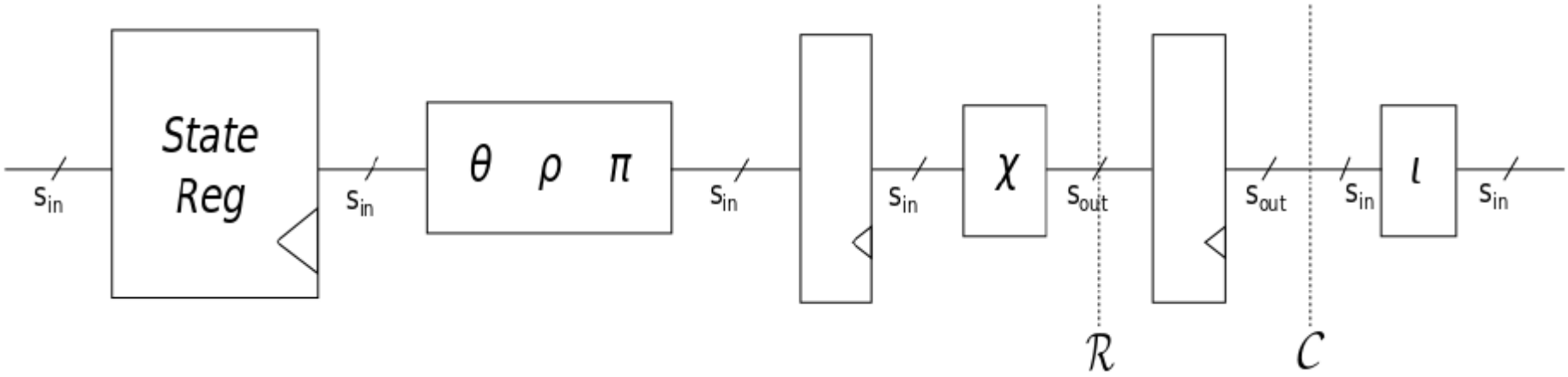
1st-order

2nd-order



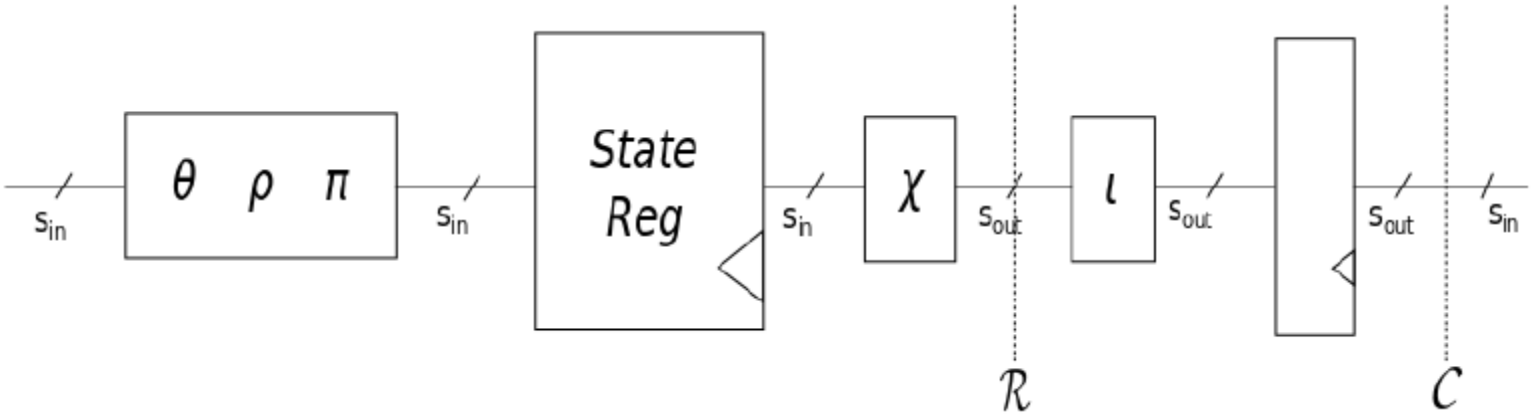
TVLA $t_{value} \leq |4.5|$ 55M traces

NC-Keccak



Non-completeness fulfilled

NC-Keccak

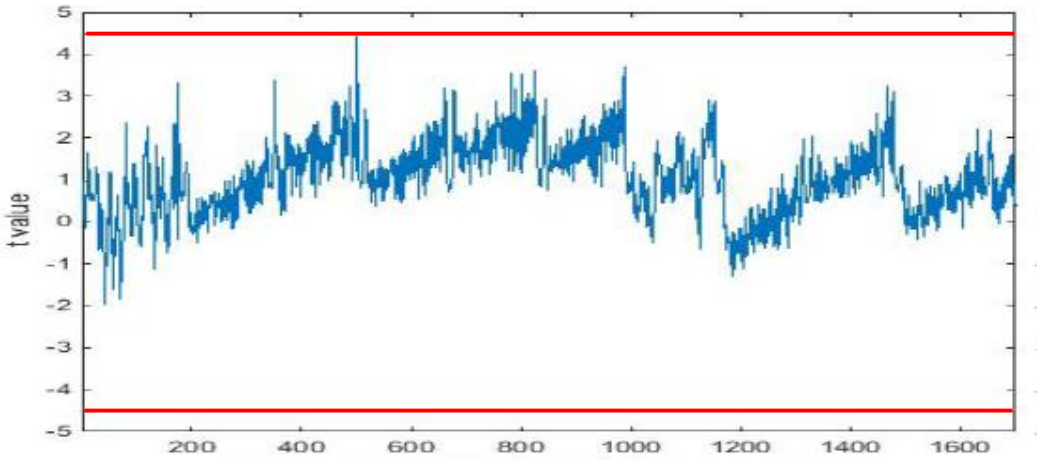


Non-completeness fulfilled

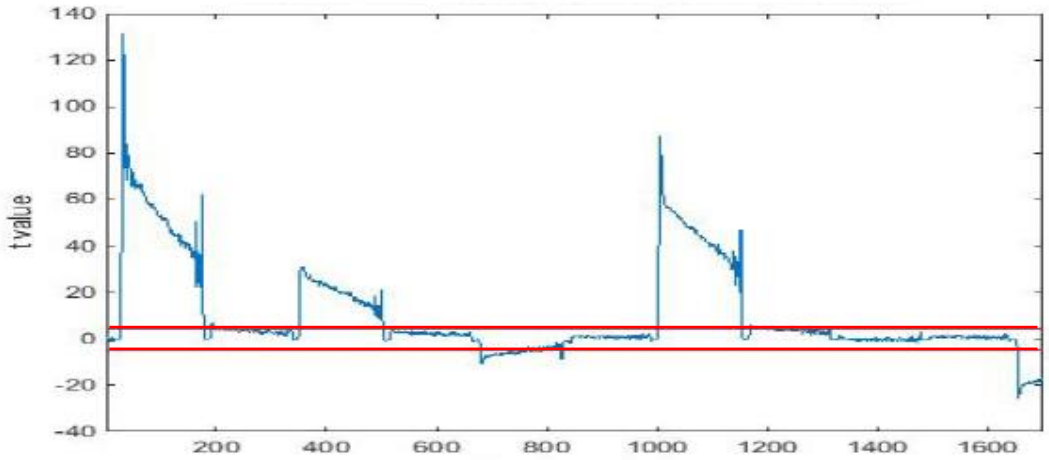
NC-Keccak

Masks On

1st-order



2nd-order



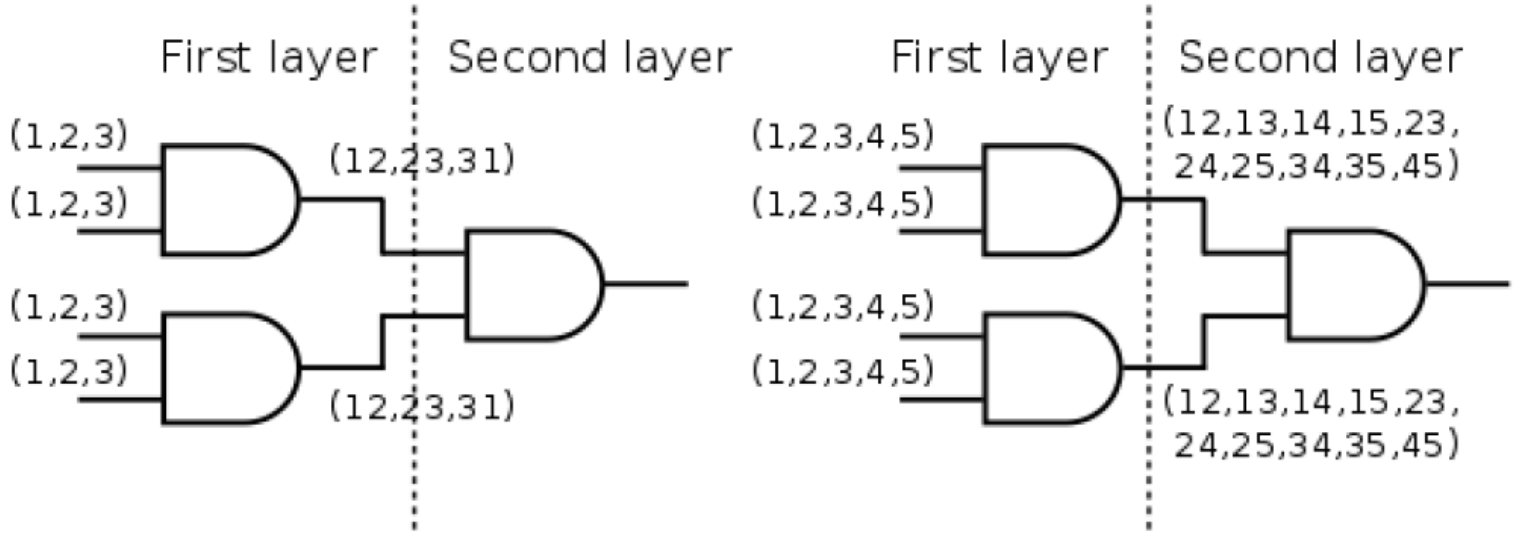
TVLA $t_{value} \leq |4.5|$ 55M traces

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

Unrolling

Halve number of cycles on protected implementations



Quartic operation (out of quadratics): $F = R^2 \circ R^1$

Unrolling

Extending methodology: $F = R^N \circ \cdots \circ R^1$

$$d_{R^i} = \begin{cases} d_F & \text{if } i = N \\ t_{R^{i+1}} * d_{R^{i+1}} & \text{if } i < N \end{cases}$$

Sharing scheme to use not specified

Refreshing for multi-variate security not provided

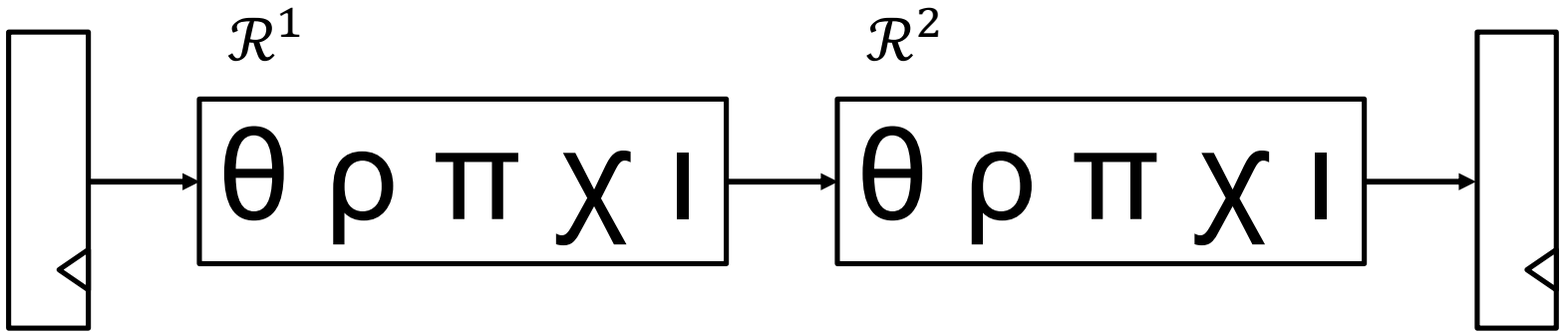
Might not be optimal

Starting point

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

Low latency Keccak



$d_2 = 2$ $\quad$ $d_F = d_2 = 1$

1. $s_{in} = 5; s_{out} = 10$ $\quad$ $s_{in} = 10; s_{out} = 5$

2. $s_{in} = 6; s_{out} = 6$ $\quad$ $s_{in} = 6;\ \ s_{out} = 6$

Low latency Keccak

$6 \rightarrow 6 \rightarrow 6$

$\mathcal{R}^1$

$$S'_1 = f_1(S_1, S_2, S_3)$$
$$S'_2 = f_2(S_1, S_4, S_5)$$
$$S'_3 = f_3(S_1, S_4, S_6)$$
$$S'_4 = f_4(S_2, S_5, S_6)$$
$$S'_5 = f_5(S_3, S_5, S_6)$$
$$S'_6 = f_6(S_2, S_3, S_4)$$

$\mathcal{R}^2$

$$S''_1 = f_1(S'_4, S'_5, S'_6) \quad (\text{Missing input } S_1)$$
$$S''_2 = f_2(S'_2, S'_3, S'_5) \quad (\text{Missing input } S_2)$$
$$S''_3 = f_3(S'_2, S'_3, S'_4) \quad (\text{Missing input } S_3)$$
$$S''_4 = f_4(S'_1, S'_4, S'_5) \quad (\text{Missing input } S_4)$$
$$S''_5 = f_5(S'_1, S'_3, S'_6) \quad (\text{Missing input } S_5)$$
$$S''_6 = f_6(S'_1, S'_2, S'_6) \quad (\text{Missing input } S_6)$$

Rhythmic Keccak

- Masking
- Input dependencies
- Implementations & Evaluation
- Unrolled implementations
- Speeding up Keccak
- Evaluation

$\mathbf{6 \rightarrow 6 \rightarrow 6}$

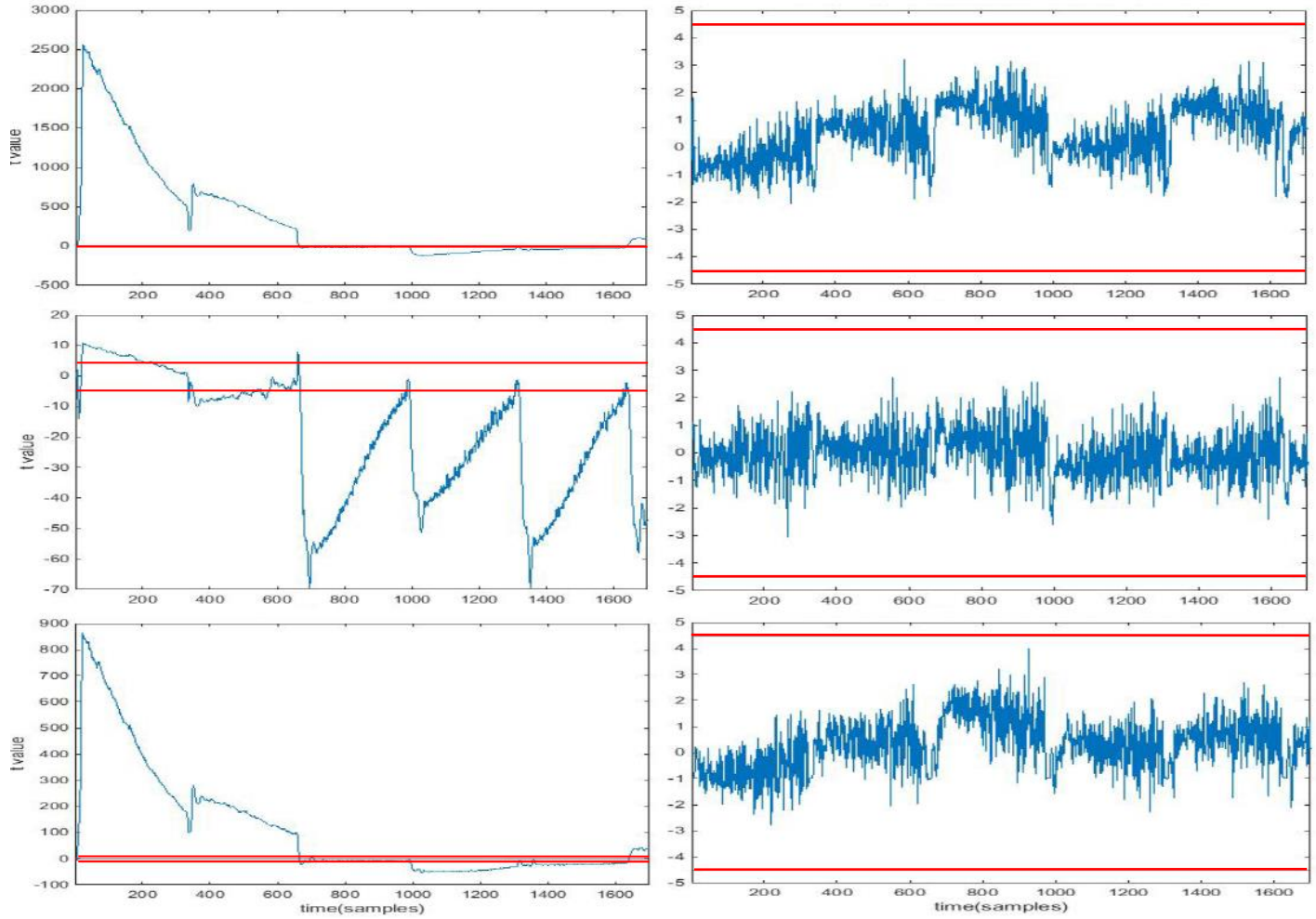
Masks Off

Masks On

1st-order

2nd-order

3rd-order



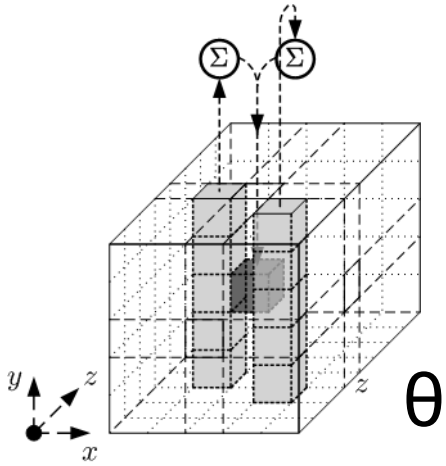
TVLA $t_{value} \leq |4.5|$ 55M traces

Conclusions

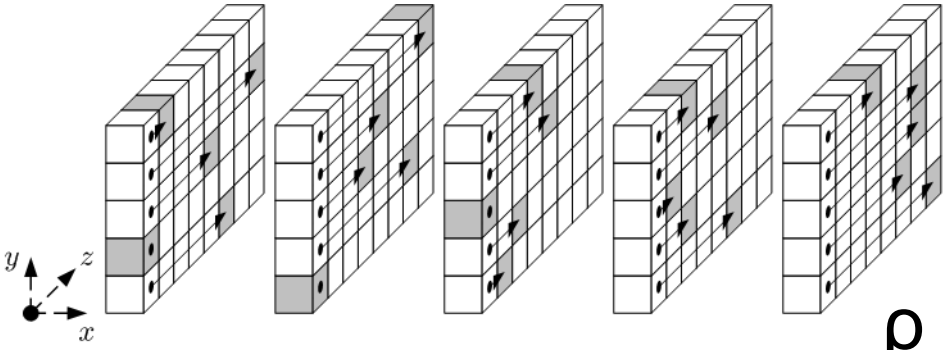
- A flaw was discovered in previous work round-based implementation of Keccak
- Causes are analyzed and fixed, with a practical evaluation
- A methodology to secure unwrapped implementations
- Faster Keccak implementation with 70,12 kGE and 20,61 ns per computation

Thank you!

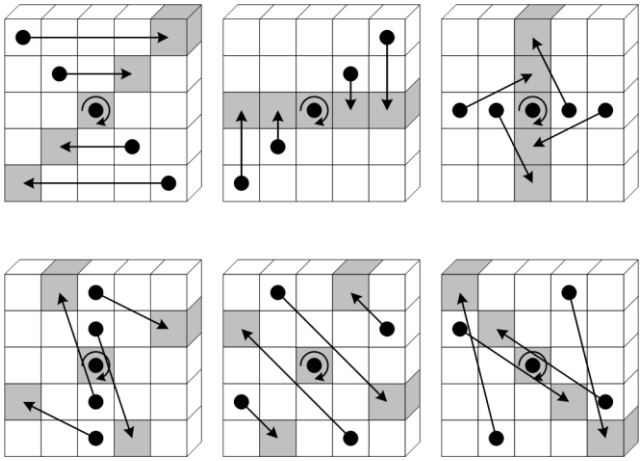
Keccak



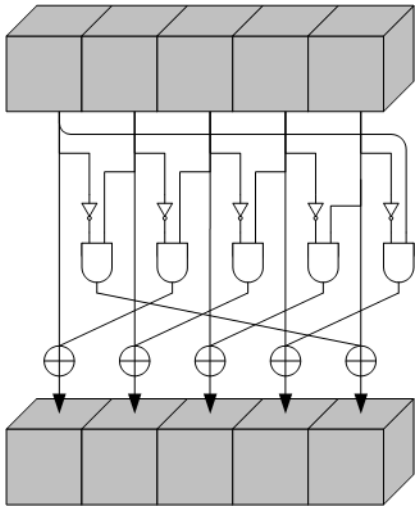
θ



ρ



π



χ

ι

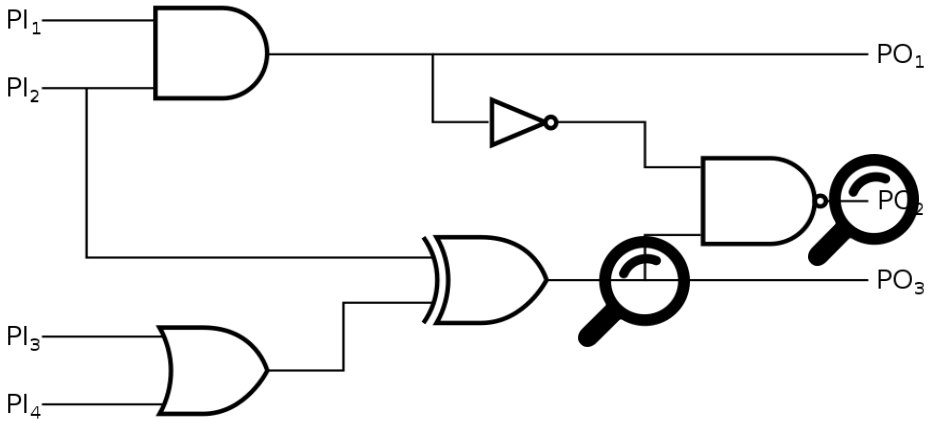
Masking

Based on secret sharing

$$x = x_1 \perp x_2 \perp \ldots \perp x_s$$

Boolean: $\perp \Rightarrow \oplus$

d-Probing Model [ISW03]



d^{th}-order SCA Security [DFS15]

[ISW03] *Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In CRYPTO 2003*
[DFS15] *A. Duc, S. Faust, and F.-X. Standaert. Making Masking Security Proofs Concrete. In EUROCRYPT 2015*

Evaluation

SAKURA board ⇒ Spartan-6 FPGA

Frequency = 3.072 MHz

Test Vector Leakage Assessment (TVLA) [CMG+13]

Security ⇒ $t_{value} \leq |4.5|$

55 million traces collected

[GMS17] *J. Cooper, E. De Mulder, G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. Test vector leakage assessment (TVLA) methodology in practice. In ICMC 2013.*

DOM-Keccak

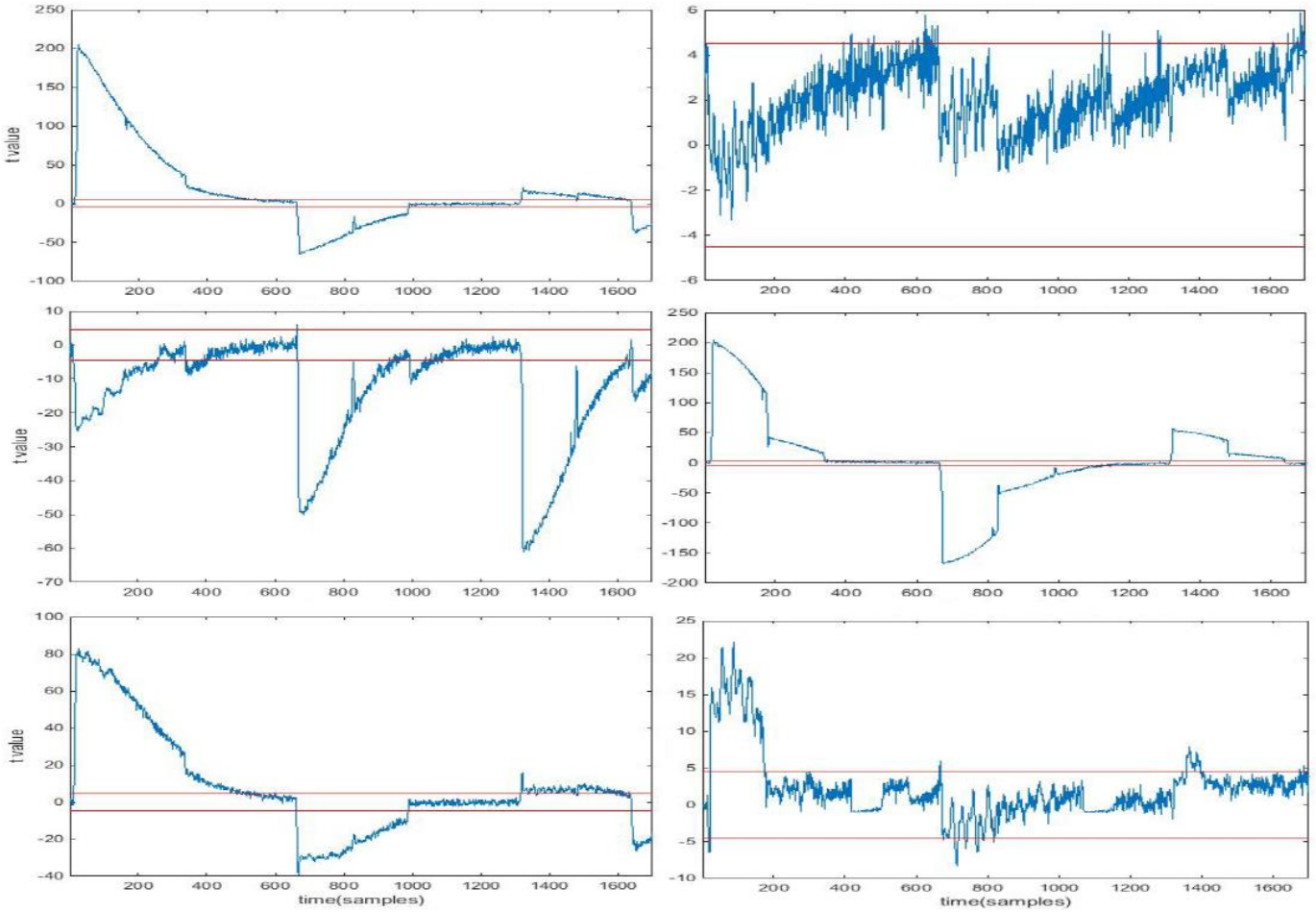
Masks Off

Masks On

1st-order

2nd-order

3rd-order



TVLA $t_{value} \leq |4.5|$ 55M traces

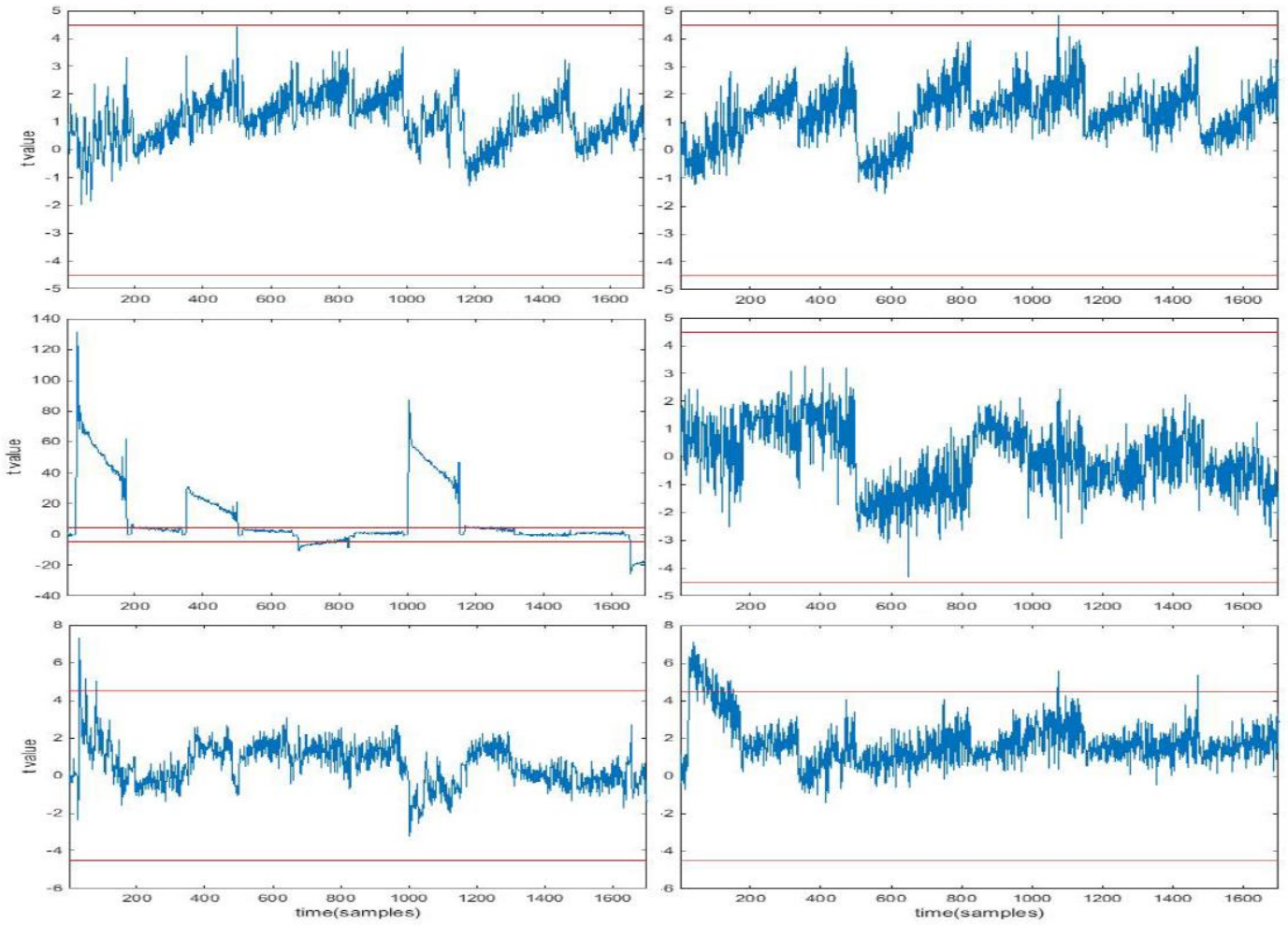
NC-Keccak

1st-order Implem. 2nd-order implem.

1st-order

2nd-order

3rd-order



TVLA $t_{value} \leq |4.5|$ 55M traces

Performance

DESIGN	AREA(kGE) χ^1	χ^2	θ^1	θ^2	State	Total	Cycles	Max.Freq. (MHz)
$5 \rightarrow 10 \rightarrow 5$	13.53	59.89	4.28	9.75	6.7	99.34	9	395.25
$6 \rightarrow 6 \rightarrow 6$	22.40	22.16	5.66	5.66	8.03	70.12	9	436.7
Previous work*								
3sh [BDN+14]	-	-	-	-	27.2	116.6	25	592
4sh [BDN+14]	-	-	-	-	36.3	139.4	24	588
D.c. [GSM17b]**	-	-	-	-	38.9	100.5	48	803.9
Pipel. [GSM17b]**	-	-	-	-	36.8	111.8	72	837.5

* Implementations of KECCAK-f[1600]
** Results gathered with library UMC 130nm