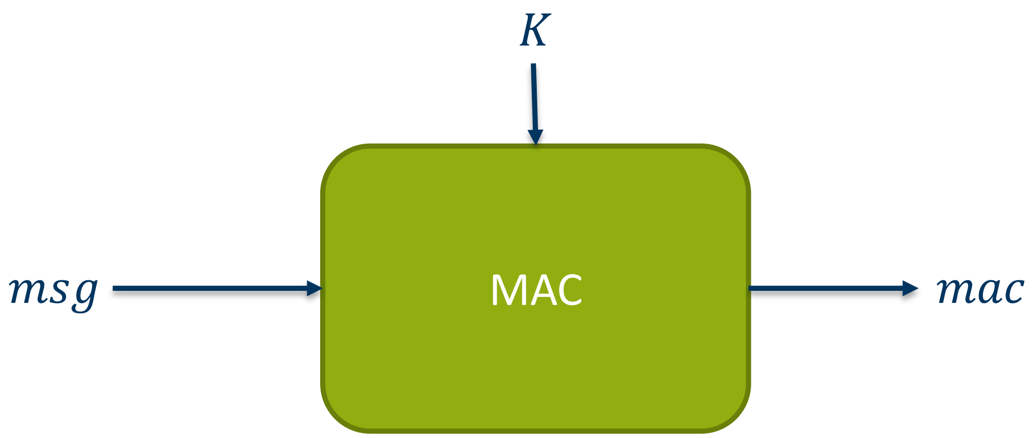
Shuffle and Mix: On the Diffusion of Randomness in TI of Keccak

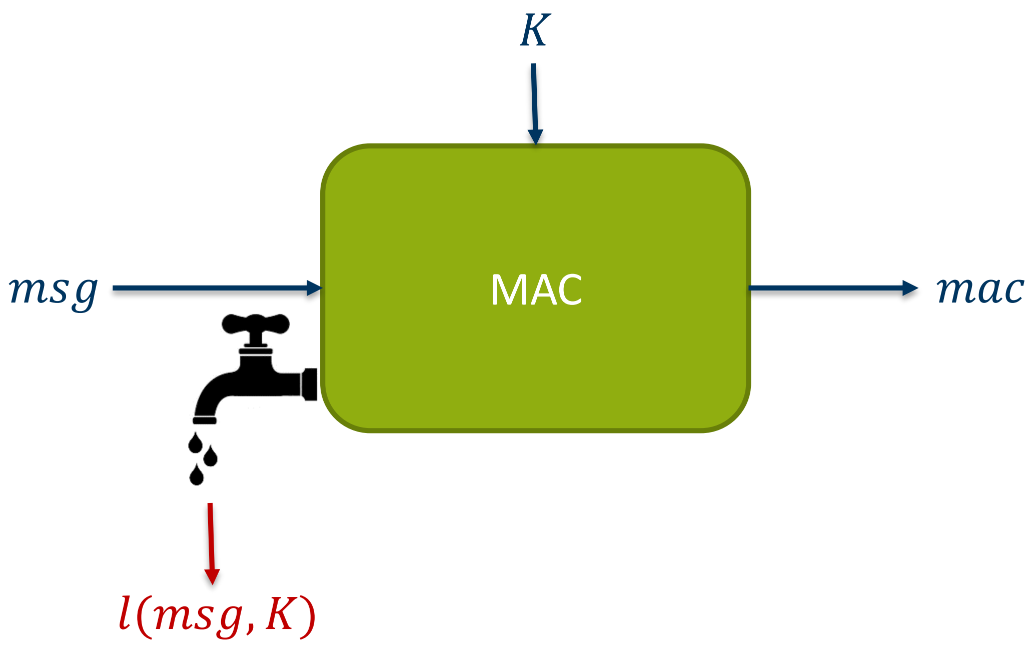
COSADE 2019, Darmstadt

Felix Wegener, Christian Baiker, Amir Moradi
Ruhr University Bochum, Horst Görtz Institute for IT-Security, Germany

Motivation



Motivation



Motivation

RUB

Countermeasures

Masking: Make intermediate value independent of secret
Hiding: Lower SNR

Masking

Boolean Masking

RUB

- Core Idea: Secret x ⟶ multiple shares $\mathrm{X} = (a, b, c)$:

$$x = a \oplus b \oplus c$$

Boolean Masking

- Core Idea: Secret x ⟶ multiple shares $\mathrm{X} = (a, b, c)$:

$$x = a \oplus b \oplus c$$

Boolean Masking

- Core Idea: Secret x ⟶ multiple shares $\mathrm{X} = (a, b, c)$:

$$x = a \oplus b \oplus c$$

- Problem: How to compute a function f on shared values?

Boolean Masking

- Core Idea: Secret x $\longrightarrow$ multiple shares $X = (a, b, c)$:

$$x = a \oplus b \oplus c$$

- Problem: How to compute a function f on shared values?
- In Hardware: Even more difficult due to glitches

Boolean Masking

RUB

- Core Idea: Secret x ⟶ multiple shares $\mathrm{X} = (a, b, c)$:

$$x = a \oplus b \oplus c$$

- Problem: How to compute a function f on shared values?
- In Hardware: Even more difficult due to glitches

Solution:
Threshold Implementations

Threshold Implementations

RUB

Three properties for first-order secure computations

- Correctness

$$A, B, C = F(a, b, c)$$
$$f(x) = A \oplus B \oplus C$$

Nikova, Rechberger, Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches, ICICS 2006

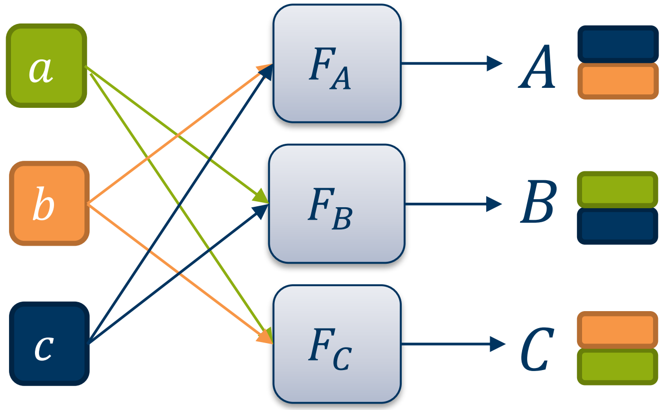
Threshold Implementations

Three properties for first-order secure computations

- Correctness

$$A, B, C = F(a, b, c)$$
$$f(x) = A \oplus B \oplus C$$

- Non-completeness



Nikova, Rechberger, Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches, ICICS 2006

Threshold Implementations

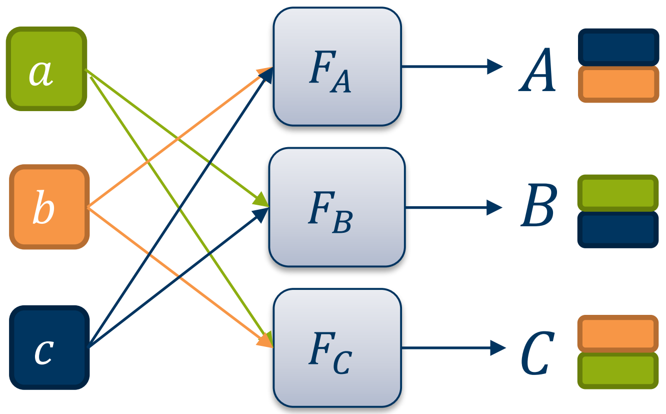
RUB

Three properties for first-order secure computations

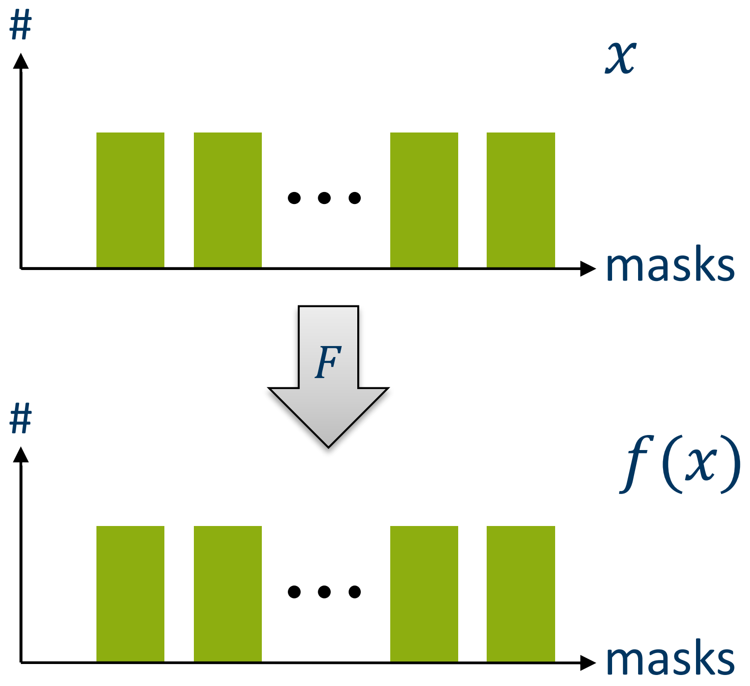
- Correctness

$$A, B, C = F(a, b, c)$$
$$f(x) = A \oplus B \oplus C$$

- Non-completeness



- Uniformity



Nikova, Rechberger, Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches, ICICS 2006

Why Uniformity?

RUB

- Locally:

Theorem:
If F is
- correct
- non-complete
- Input is masked uniformly

Then:
Evaluation is first-order secure

Why Uniformity?

RUB

- Locally:

Theorem:
If F is
- correct
- non-complete
- Input is masked uniformly

Then:
Evaluation is first-order secure

Uniform output <u>not</u> needed

Why Uniformity?

RUB

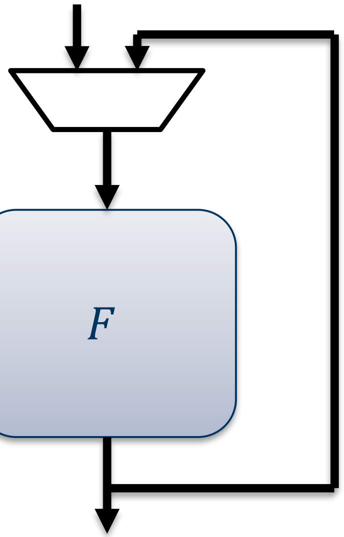
- Locally:

Theorem:
If F is
- correct
- non-complete
- Input is masked uniformly

Then:
Evaluation is first-order secure

Uniform output not needed

- Globally:



Iterated Round-function

Why Uniformity?

RUB

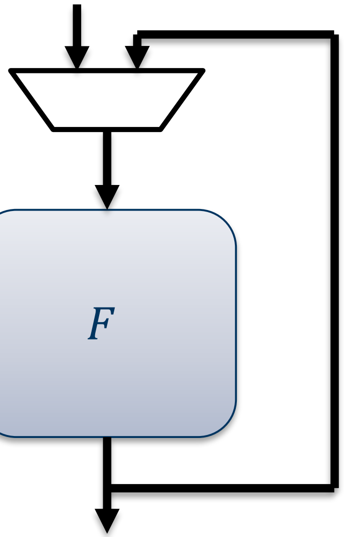
- Locally:

Theorem:
If F is
- correct
- non-complete
- Input is masked uniformly

Then:
Evaluation is first-order secure

Uniform output not needed

- Globally:



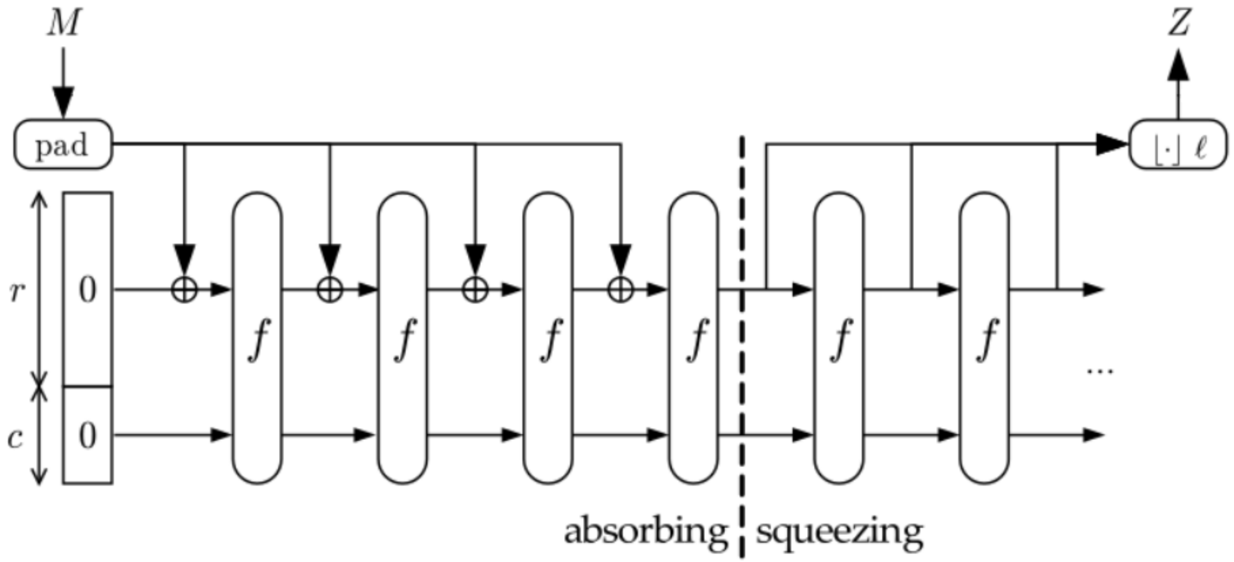
Iterated Round-function

Uniform output needed

Keccak

Keccak

- Sponge-based Hashfunction

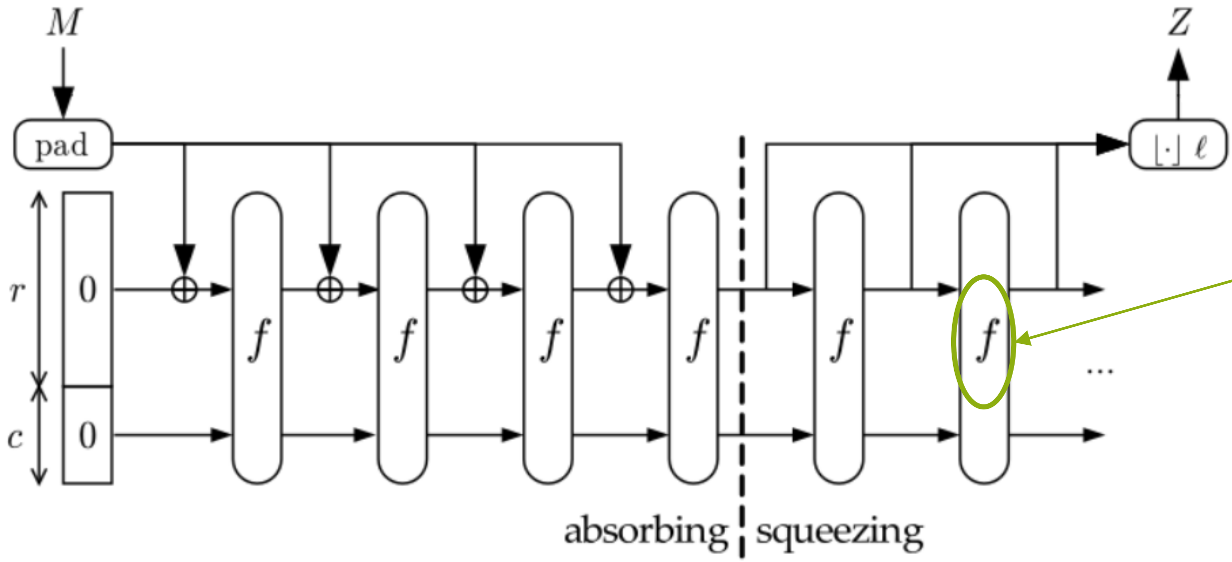


- SHA3 in 2015

Bertoni et al. Cryptographic Sponge Functions. Keccak.team

Keccak

- Sponge-based Hashfunction



- SHA3 in 2015

Keccak-*f*[b]:

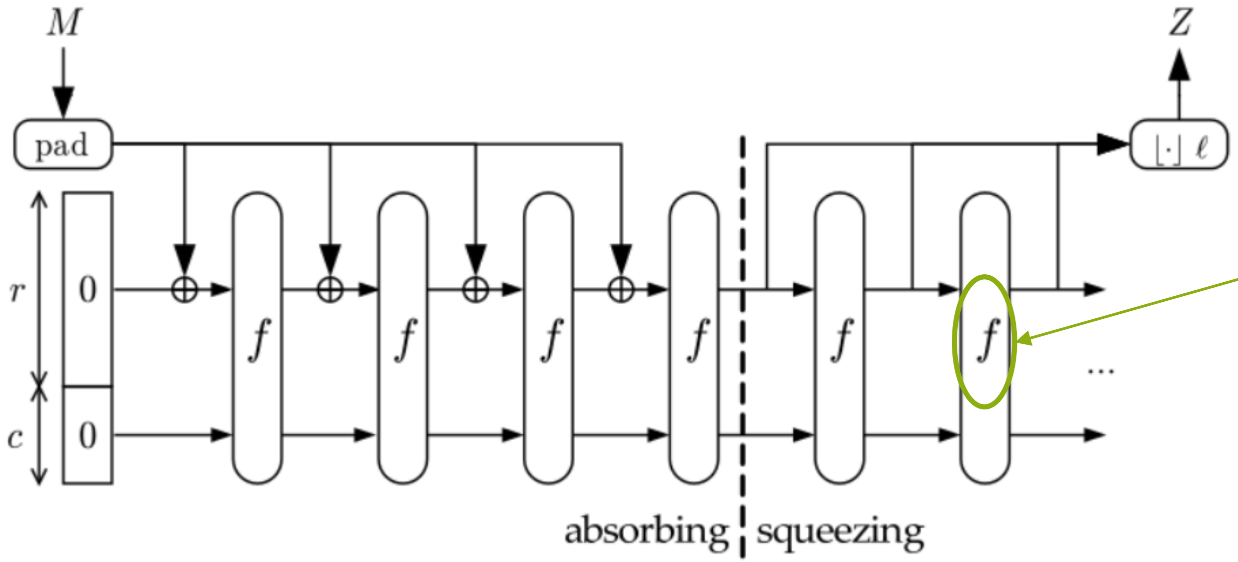
- $b = 25 \cdot 2^l,\ l = 0, \dots, 6$
- $n_r = 12 + 2l$
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$

Bertoni et al. Cryptographic Sponge Functions. Keccak.team

Keccak

RUB

- Sponge-based Hashfunction



- SHA3 in 2015

Keccak-*f*[b]:

- $b = 25 \cdot 2^l, \ l = 0, \dots, 6$
- $n_r = 12 + 2l$
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$

Here:
Keccak-*f*[200]
18 rounds

Bertoni et al. Cryptographic Sponge Functions. Keccak.team

How to mask Keccak-f?

Linear Layer

ρ

π

θ

ι

Linear Layer

RUB

Use linearity:

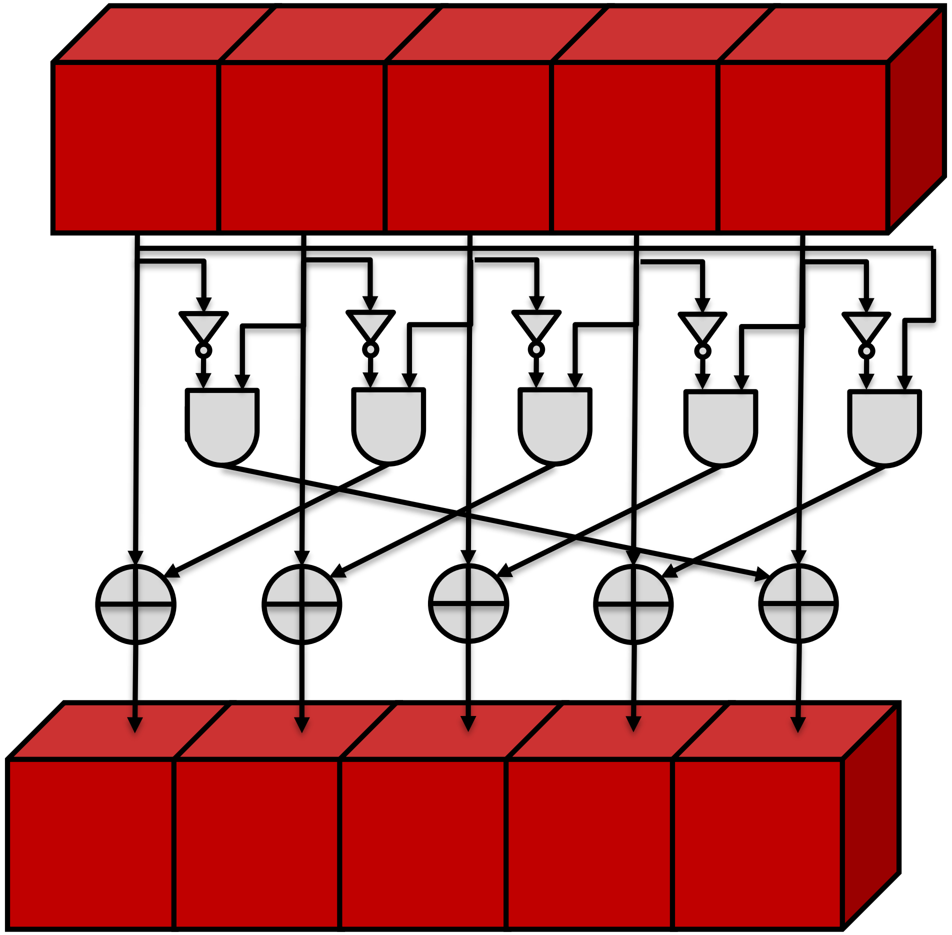
$$L(x_1 \oplus x_2 \oplus x_3) = L(x_1) \oplus L(x_2) \oplus L(x_3)$$

Replication without modification

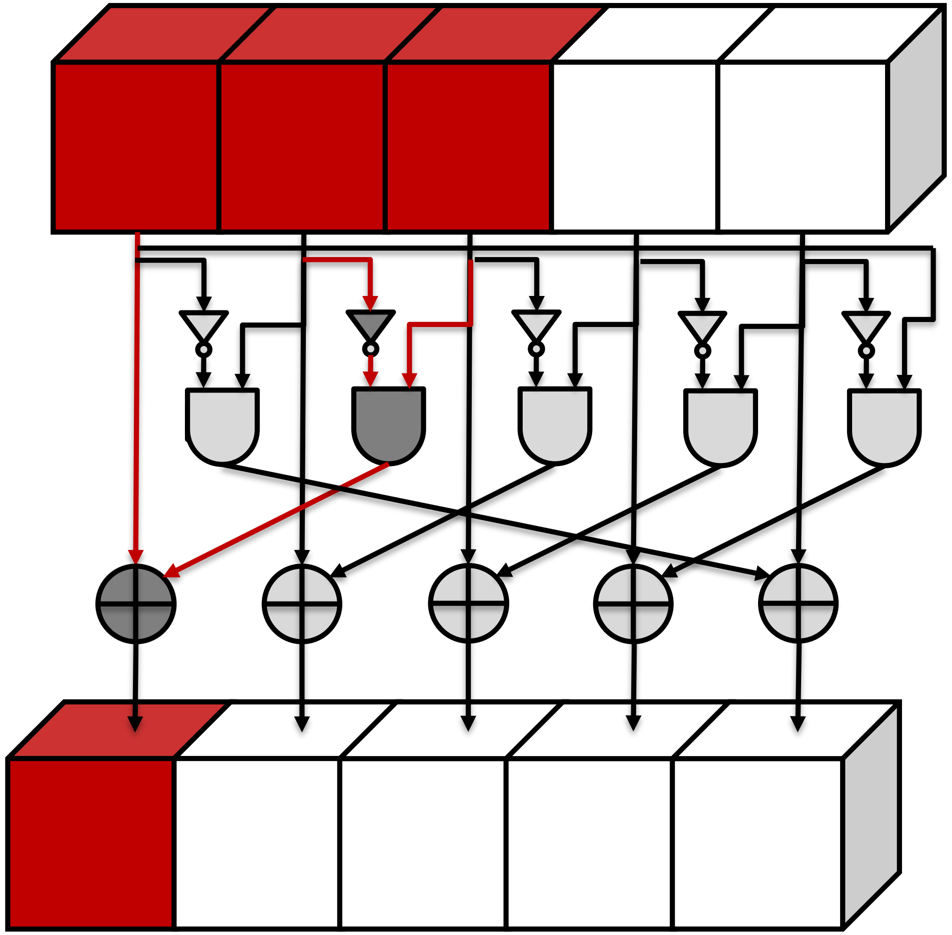
Non-linear Layer

χ

Non-linear Layer



Non-linear Layer

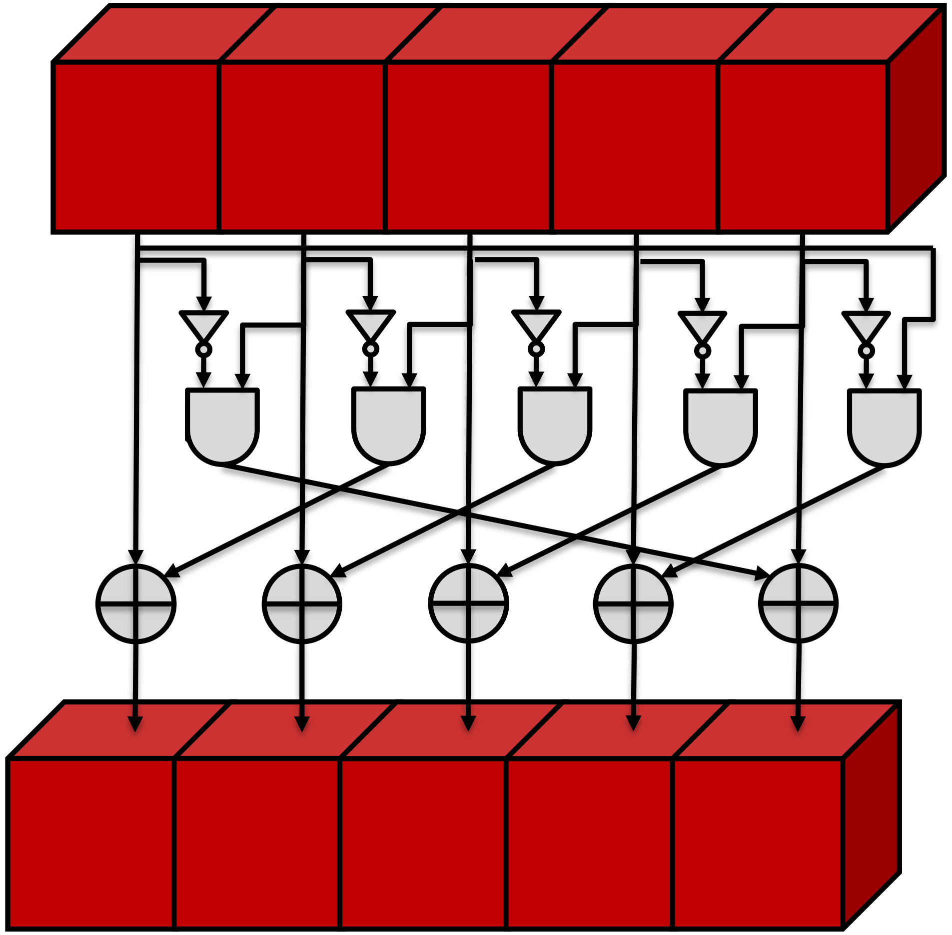


One Coordinate function:

$$y_0 = x_0 \oplus [(1 \oplus x_1) \land x_2]$$
$$= x_0 \oplus (x_1 \land x_2) \oplus x_2$$

Non-linear Layer

RUB



One Coordinate function:

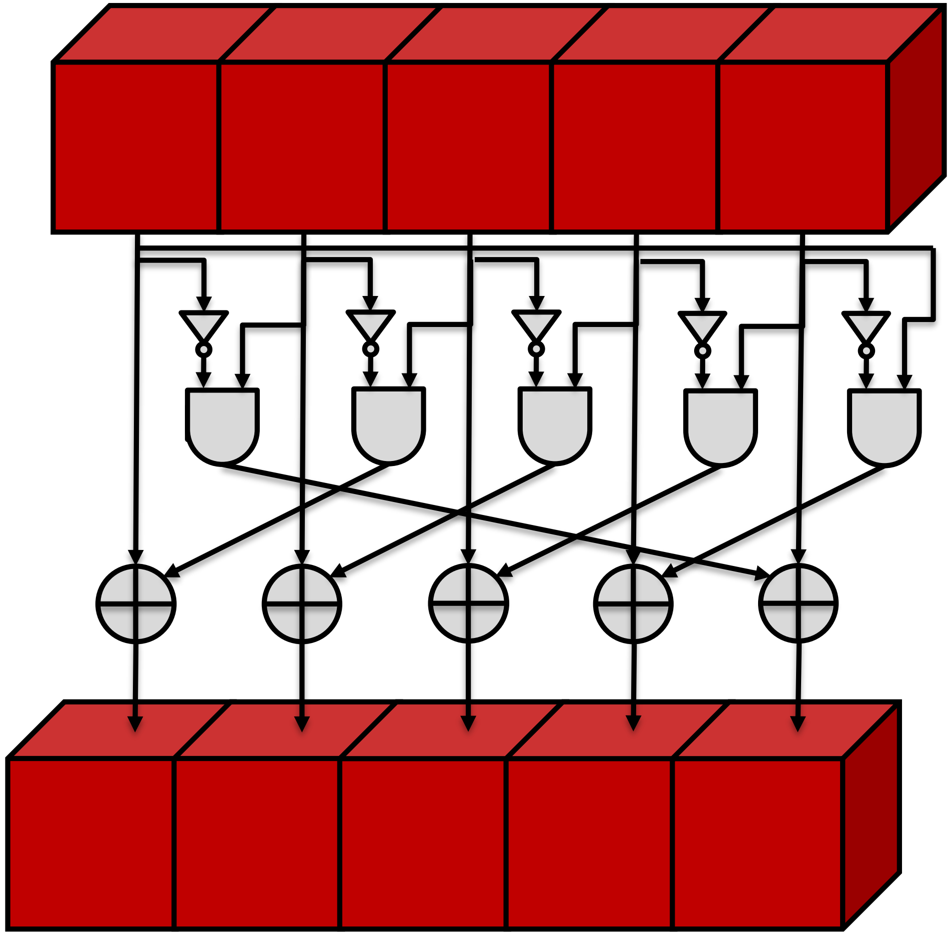
$$y_0 = x_0 \oplus [(1 \oplus x_1) \wedge x_2]$$
$$= x_0 \oplus (x_1 \wedge x_2) \oplus x_2$$

Direct Sharing of χ:

$$A_i = b_i \oplus (b_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge b_{i+2}) \oplus b_{i+2}$$
$$B_i = c_i \oplus (c_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge c_{i+2}) \oplus c_{i+2}$$
$$C_i = a_i \oplus (a_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge a_{i+2}) \oplus a_{i+2}$$

Bertoni, Daemen, Peeters, Van Assche: Keccak. EUROCRYPT 2013

Non-linear Layer



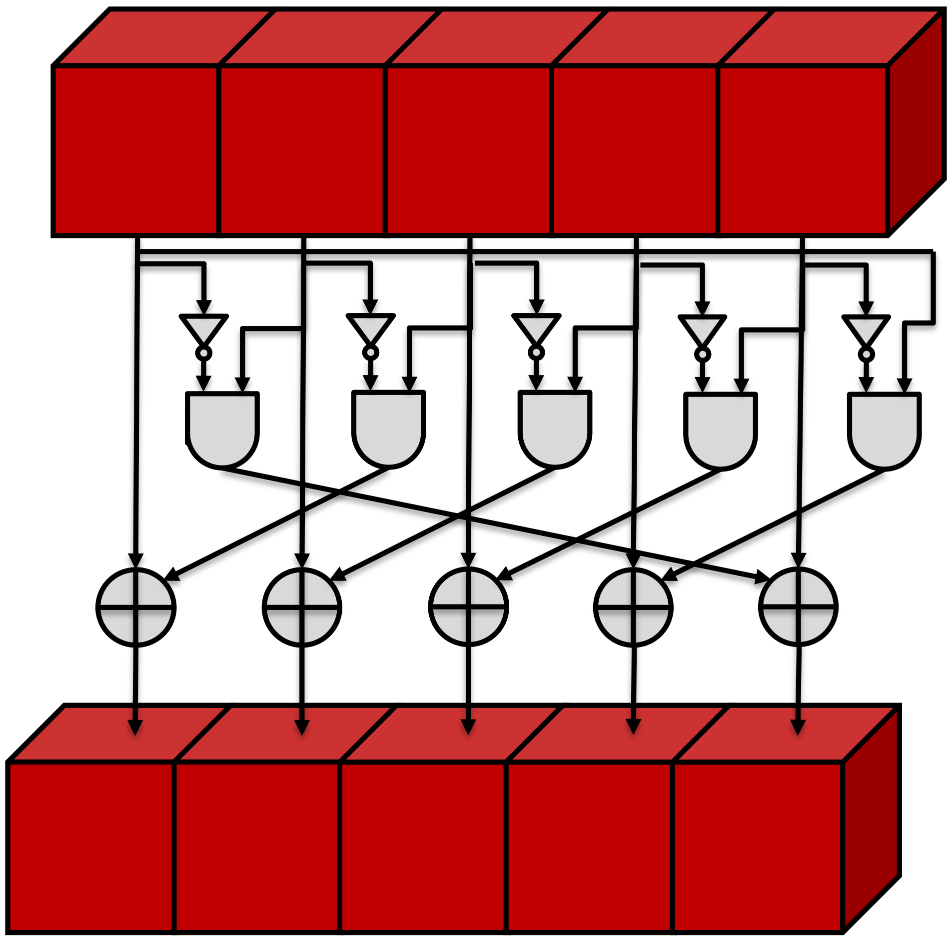
Direct Sharing of χ:

$$A_i = b_i \oplus (b_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge b_{i+2}) \oplus b_{i+2}$$
$$B_i = c_i \oplus (c_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge c_{i+2}) \oplus c_{i+2}$$
$$C_i = a_i \oplus (a_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge a_{i+2}) \oplus a_{i+2}$$

Bertoni, Daemen, Peeters, Van Assche: Keccak. EUROCRYPT 2013

Non-complete ✔

Non-linear Layer



Direct Sharing of χ:

$$A_i = b_i \oplus (b_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge b_{i+2}) \oplus b_{i+2}$$
$$B_i = c_i \oplus (c_{i+1} \wedge c_{i+2}) \oplus (c_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge c_{i+2}) \oplus c_{i+2}$$
$$C_i = a_i \oplus (a_{i+1} \wedge a_{i+2}) \oplus (a_{i+1} \wedge b_{i+2}) \oplus (b_{i+1} \wedge a_{i+2}) \oplus a_{i+2}$$

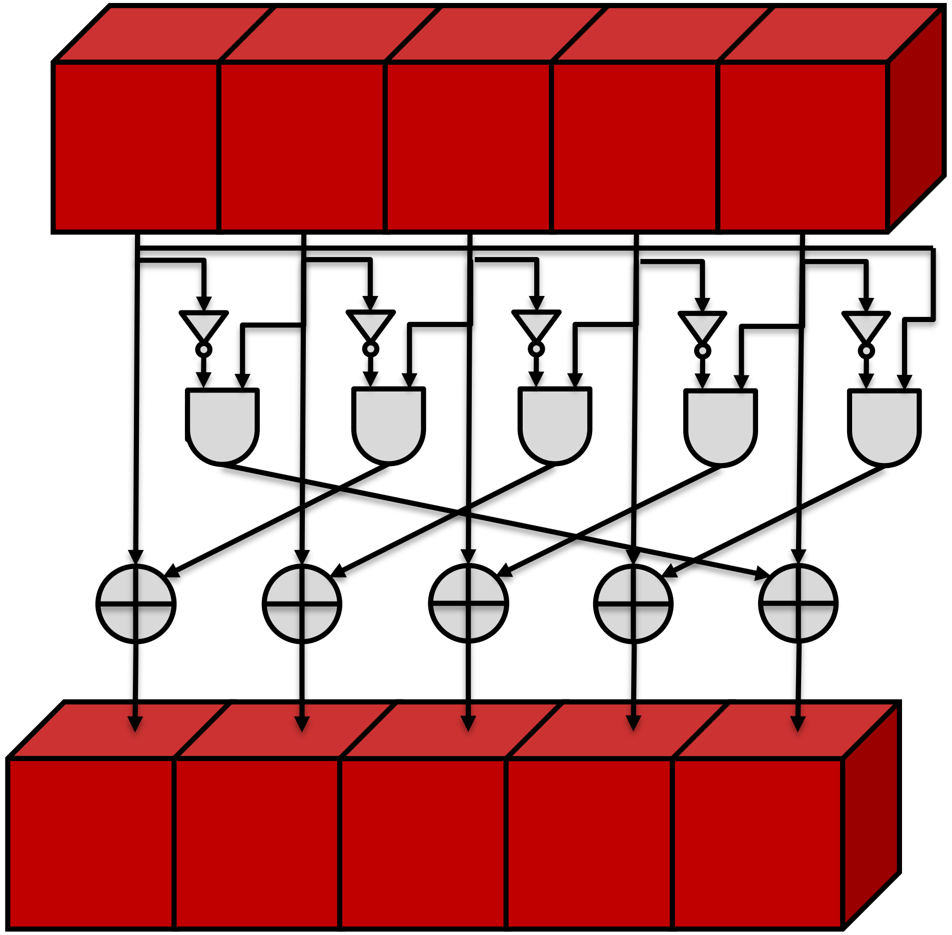
Bertoni, Daemen, Peeters, Van Assche: Keccak. EUROCRYPT 2013

Non-complete ✔

NOT Uniform ✘

Non-linear Layer

RUB



Direct Sharing of χ:

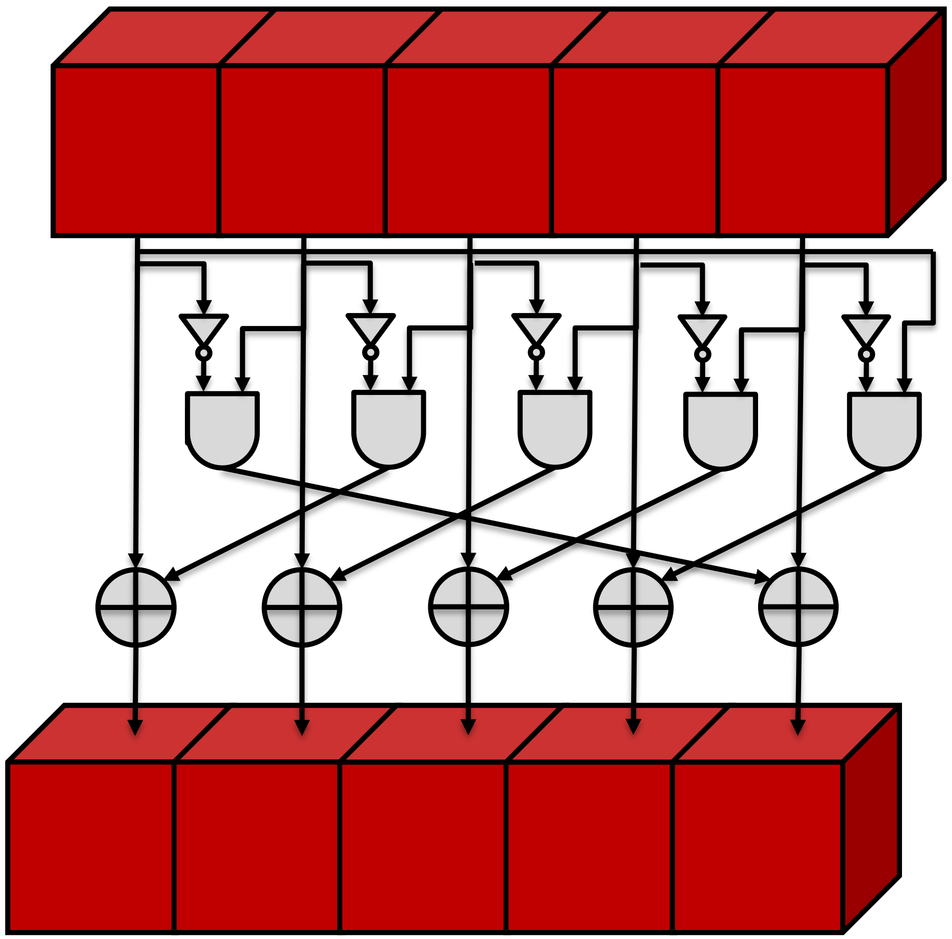
$$A_i = b_i \oplus (b_{i+1} \land b_{i+2}) \oplus (b_{i+1} \land c_{i+2}) \oplus (c_{i+1} \land b_{i+2}) \oplus b_{i+2}$$
$$B_i = c_i \oplus (c_{i+1} \land c_{i+2}) \oplus (c_{i+1} \land a_{i+2}) \oplus (a_{i+1} \land c_{i+2}) \oplus c_{i+2}$$
$$C_i = a_i \oplus (a_{i+1} \land a_{i+2}) \oplus (a_{i+1} \land b_{i+2}) \oplus (b_{i+1} \land a_{i+2}) \oplus a_{i+2}$$

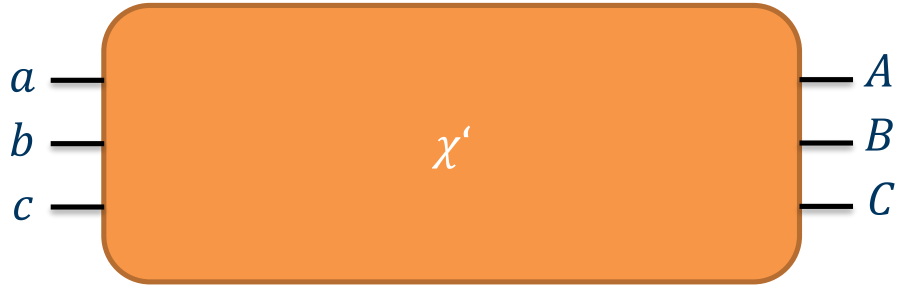
Bertoni, Daemen, Peeters, Van Assche: Keccak. EUROCRYPT 2013

Non-complete ✔

Partially Uniform

Non-linear Layer

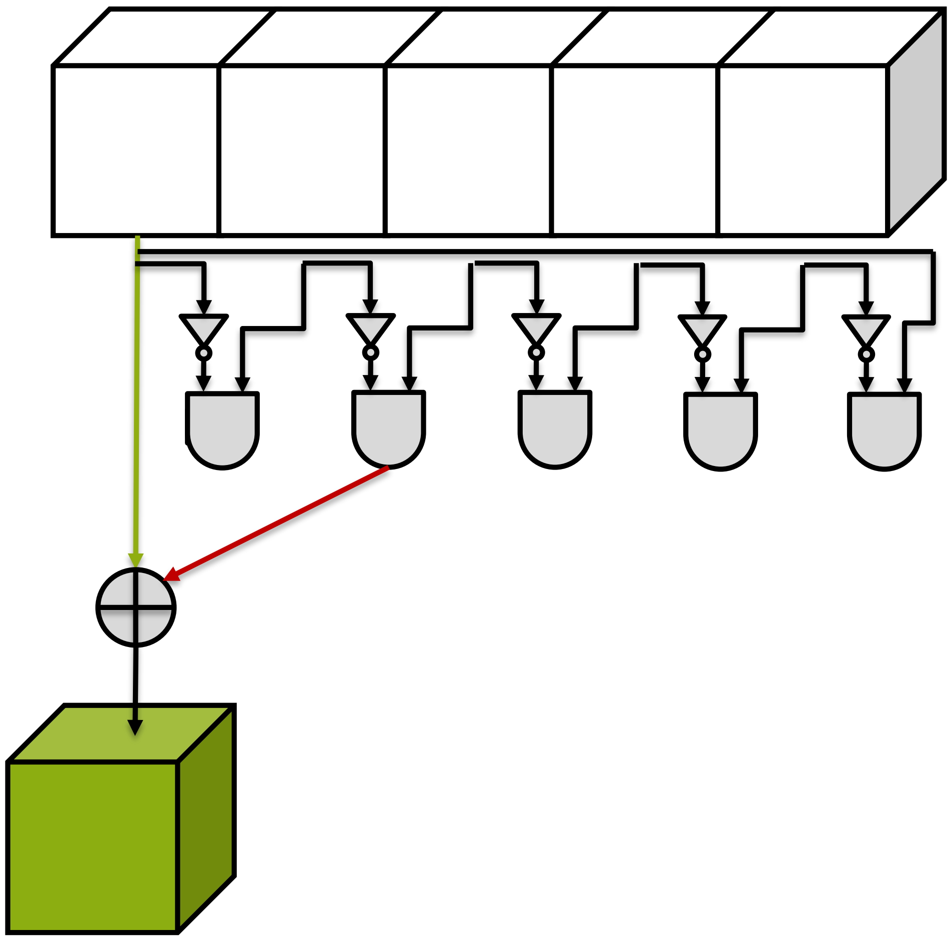


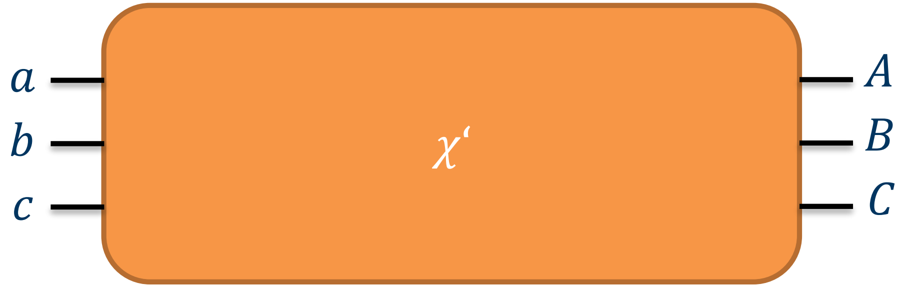


Non-complete ✔

Partially Uniform

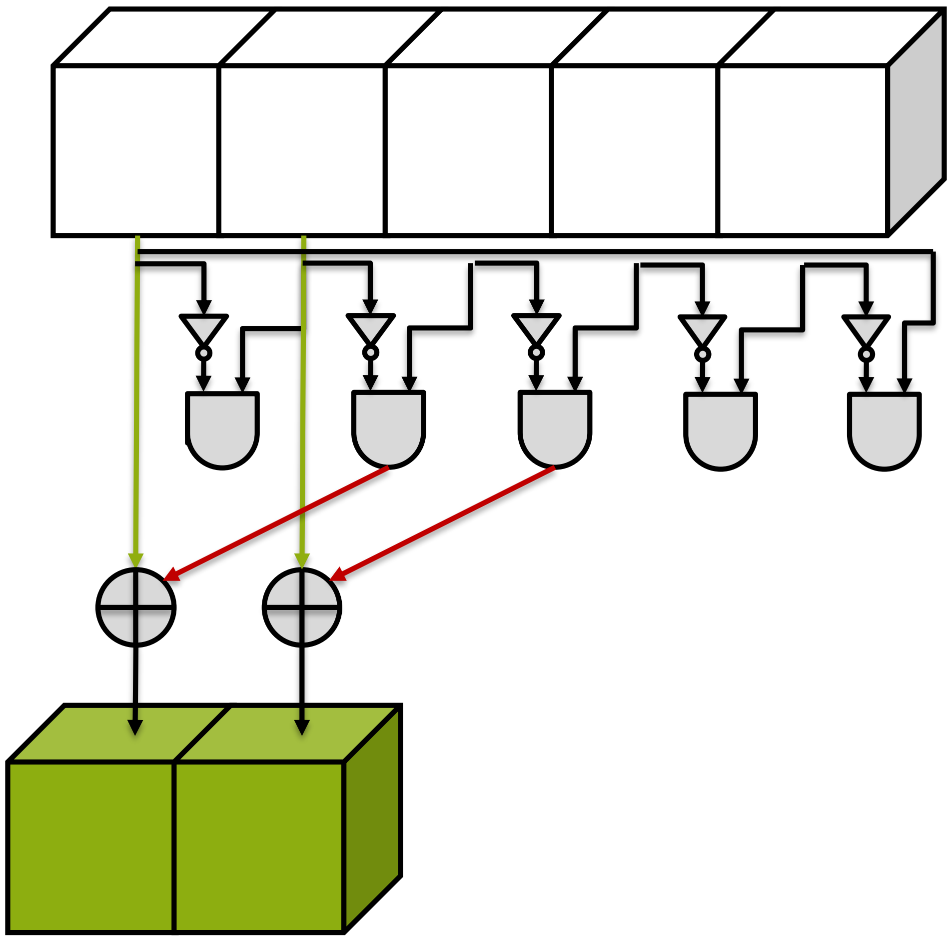
Non-linear Layer

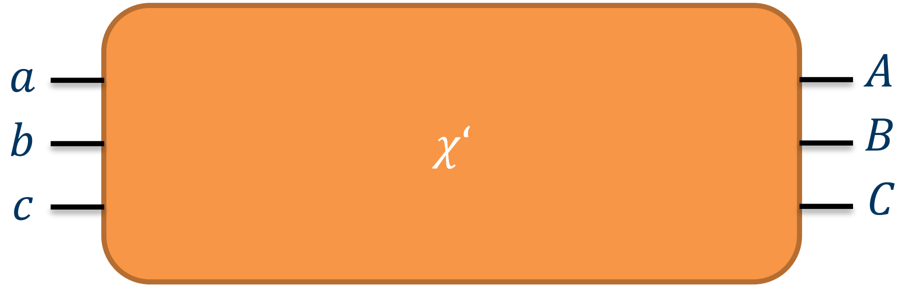




1 single bit: uniform

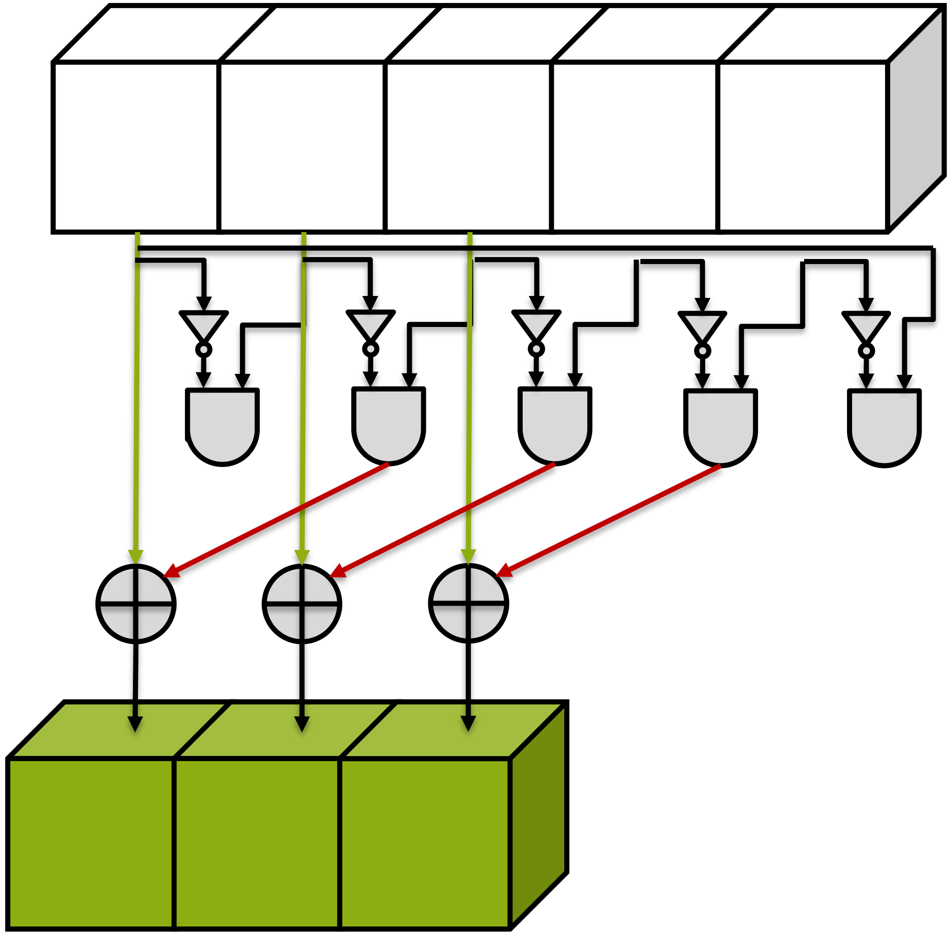
Non-linear Layer

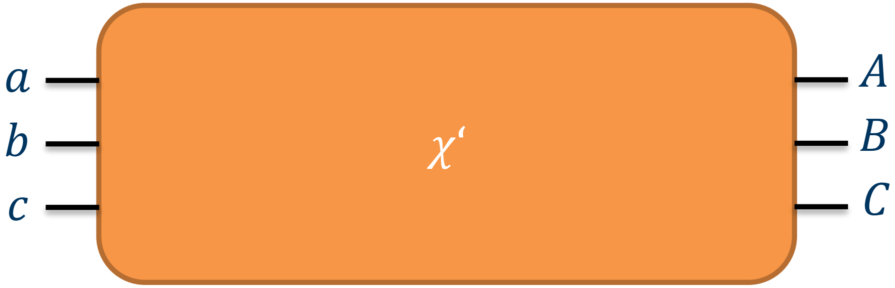




2 bits: jointly uniform

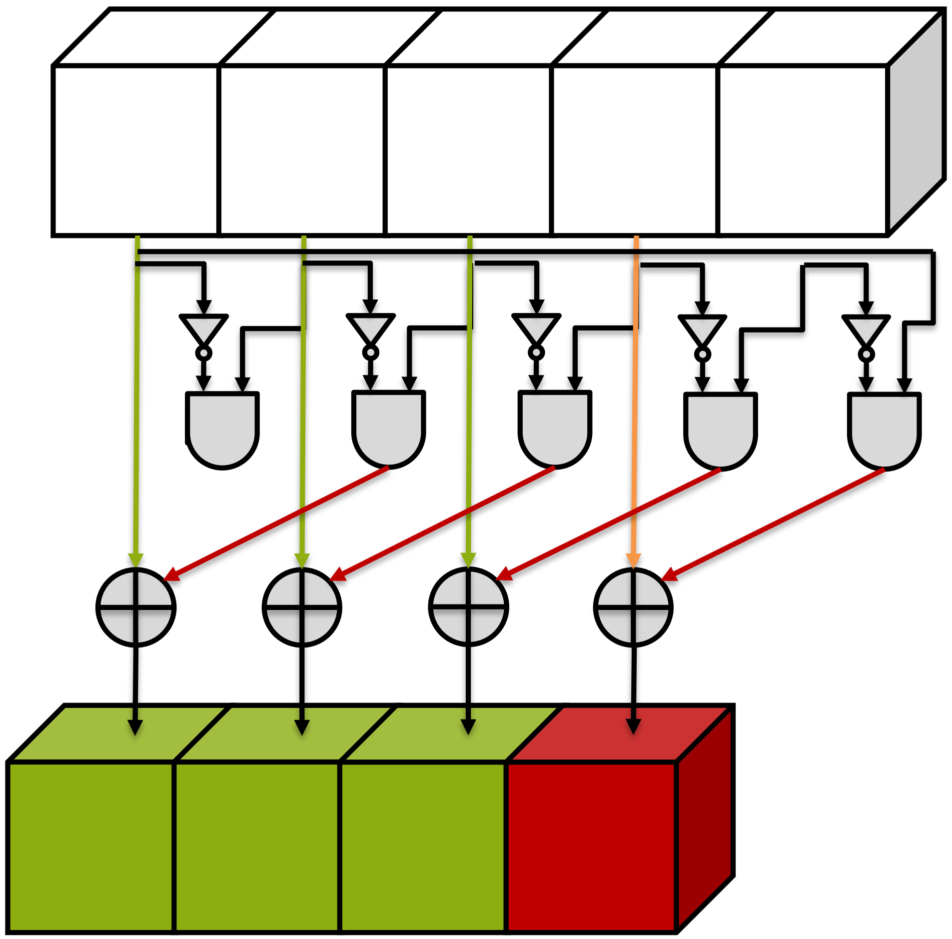
Non-linear Layer

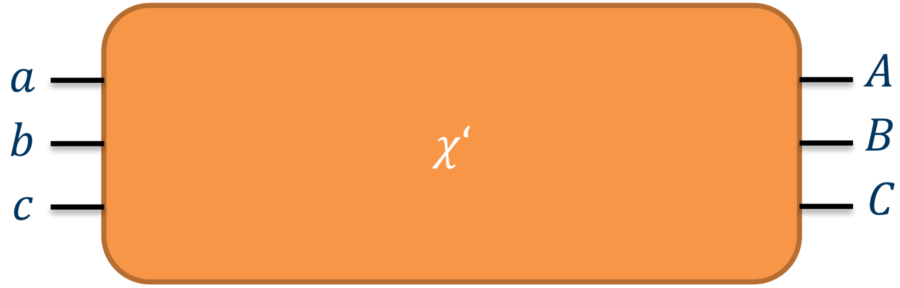




3 bits: jointly uniform

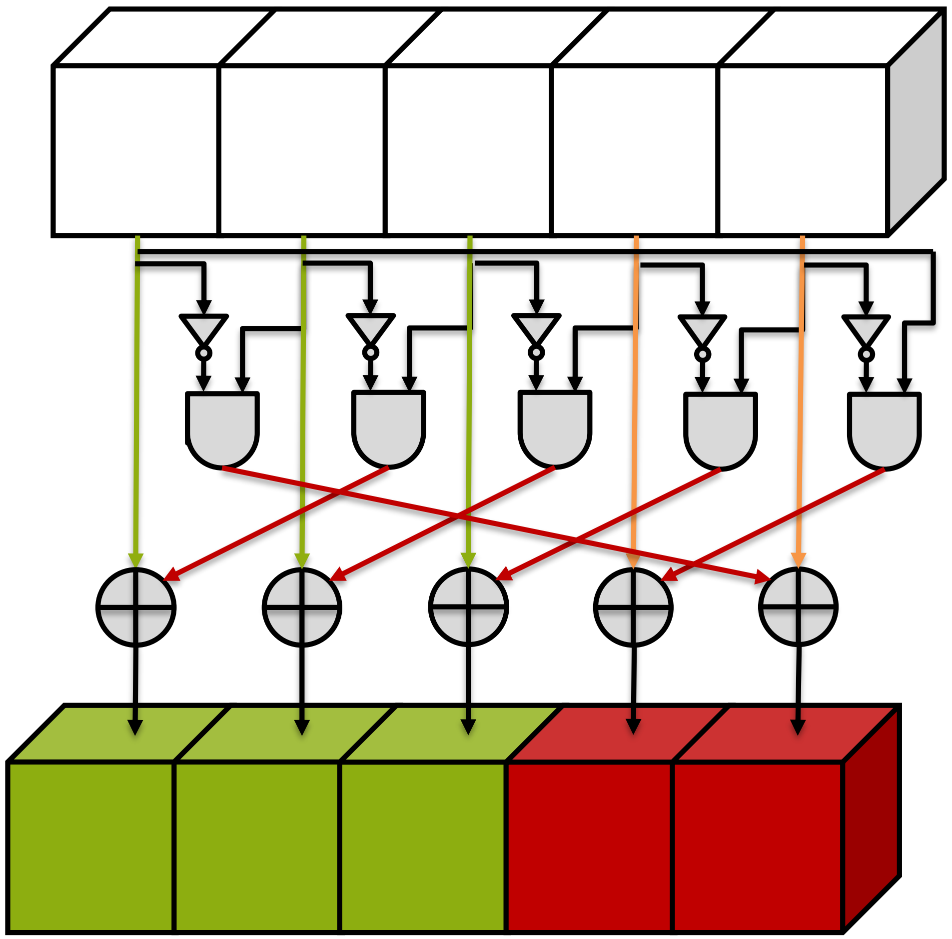
Non-linear Layer

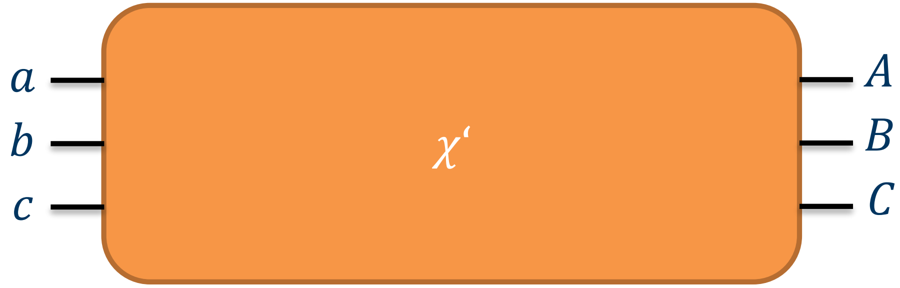




4 bits: not jointly uniform

Non-linear Layer



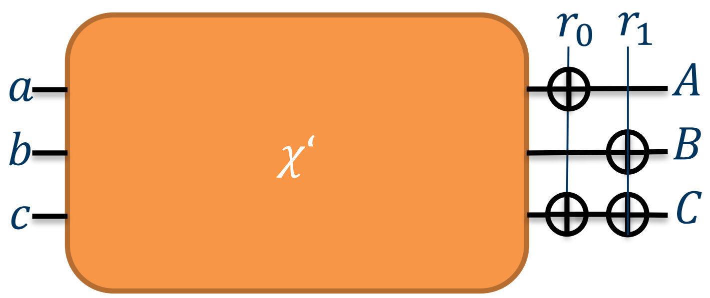


2 out of 5 bits not jointly uniform*

*Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

Fixing Non-Uniformity

Refresh with 4 bits of fresh randomness*

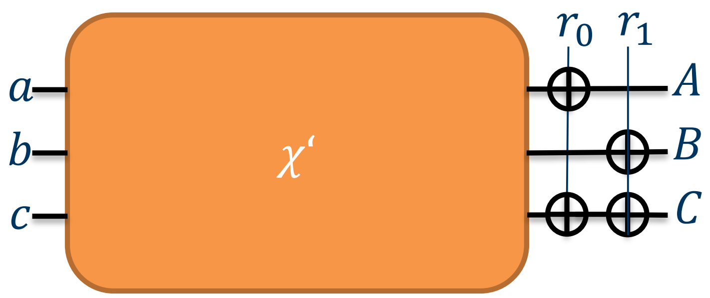


*Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

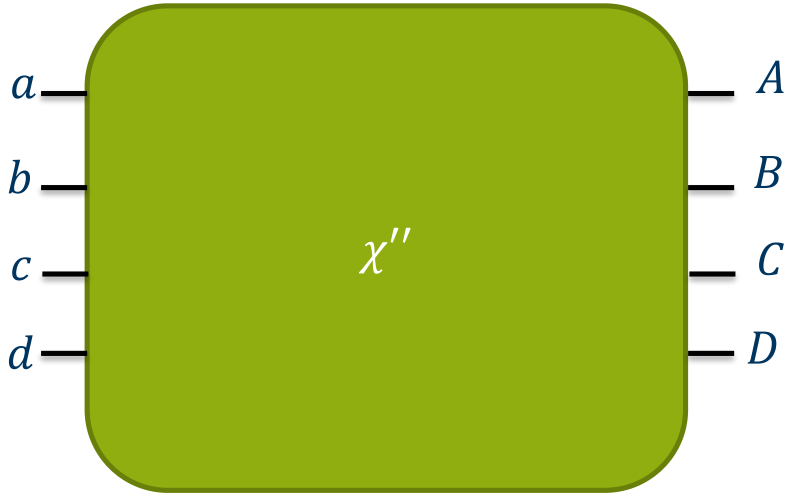
**Daemen. Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharings. CHES 2017

Fixing Non-Uniformity

Refresh with 4 bits of fresh randomness*



Use 4 shares*



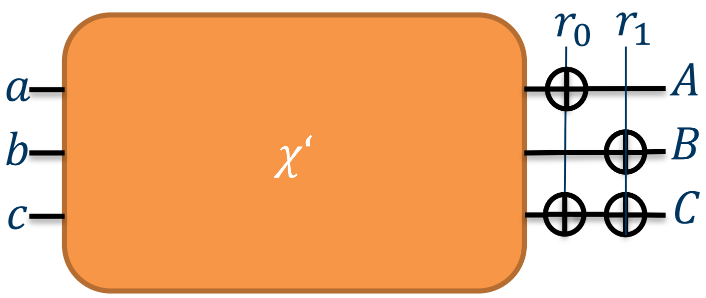
*Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

**Daemen. Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharings. CHES 2017

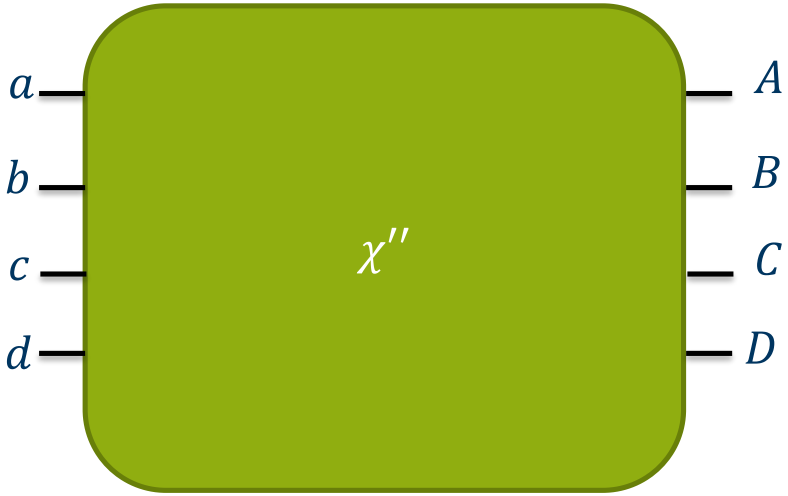
Fixing Non-Uniformity

RUB

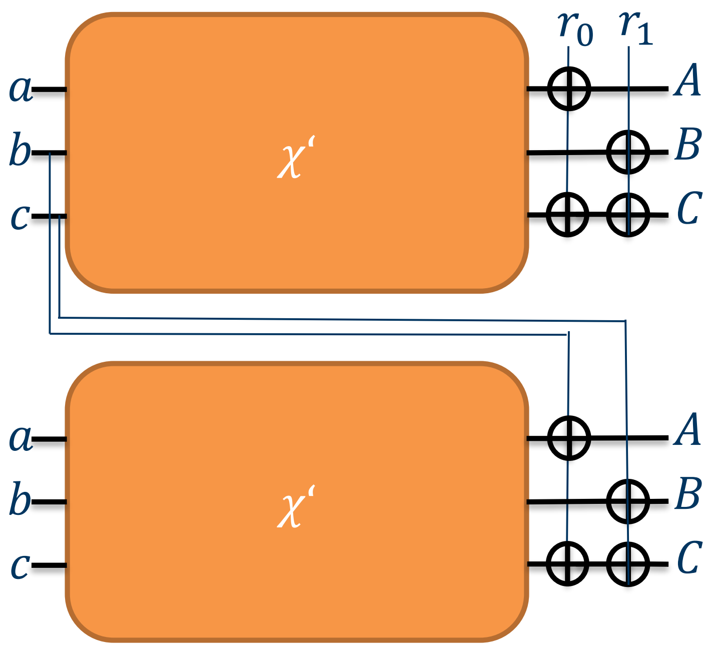
Refresh with 4 bits of fresh randomness*



Use 4 shares*



Changing of the Guards**

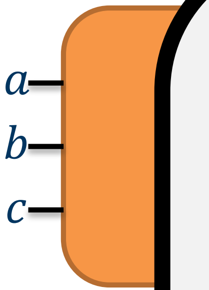


*Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

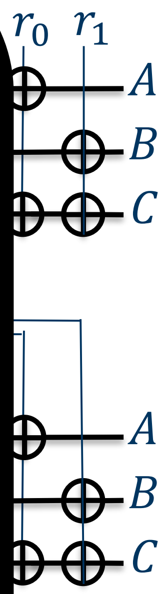
**Daemen. Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharings. CHES 2017

Fixing Non-Uniformity

Refresh with 4 bits of fresh randomness*



Changing of the Guards**



This Work: Don‘t fix it.

Consequences?

Use 4 sha

*Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

**Daemen. Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharings. CHES 2017

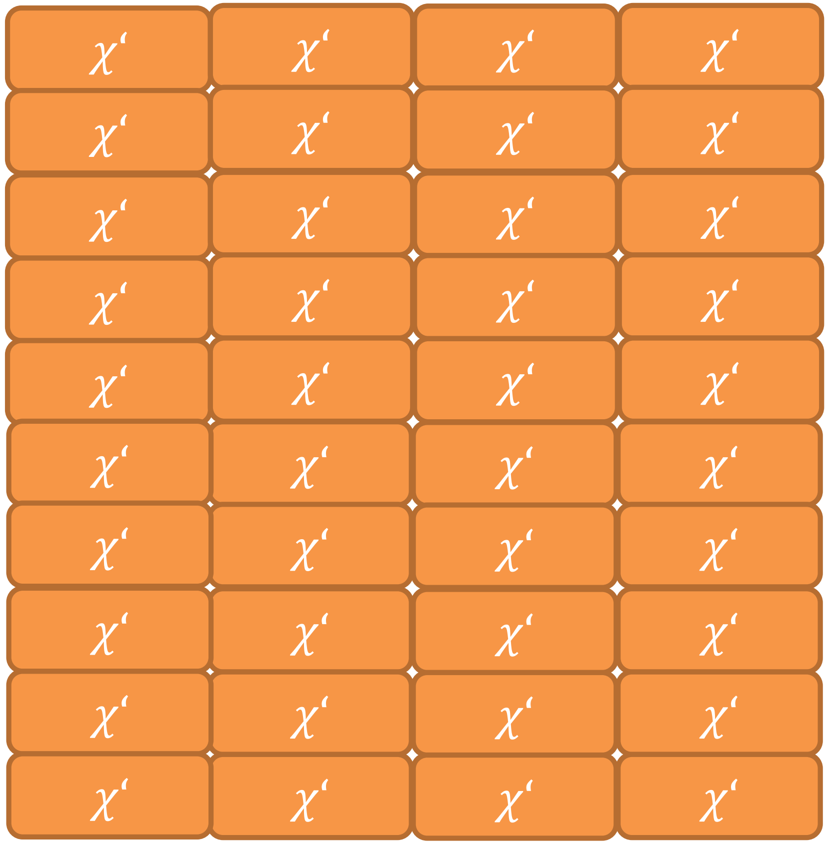
Hardware Target

Hardware Architecture

How many parallel S-boxes?

Serialized

Round-based



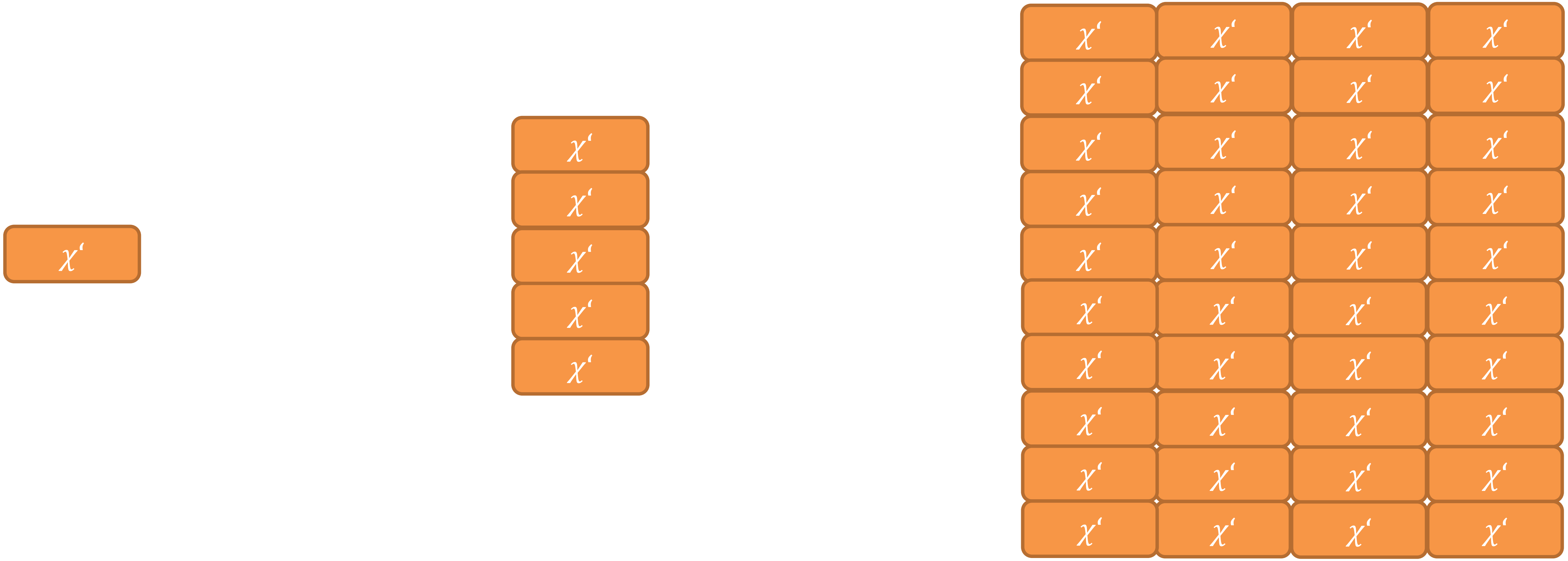
Hardware Architecture

How many parallel S-boxes?

Serialized

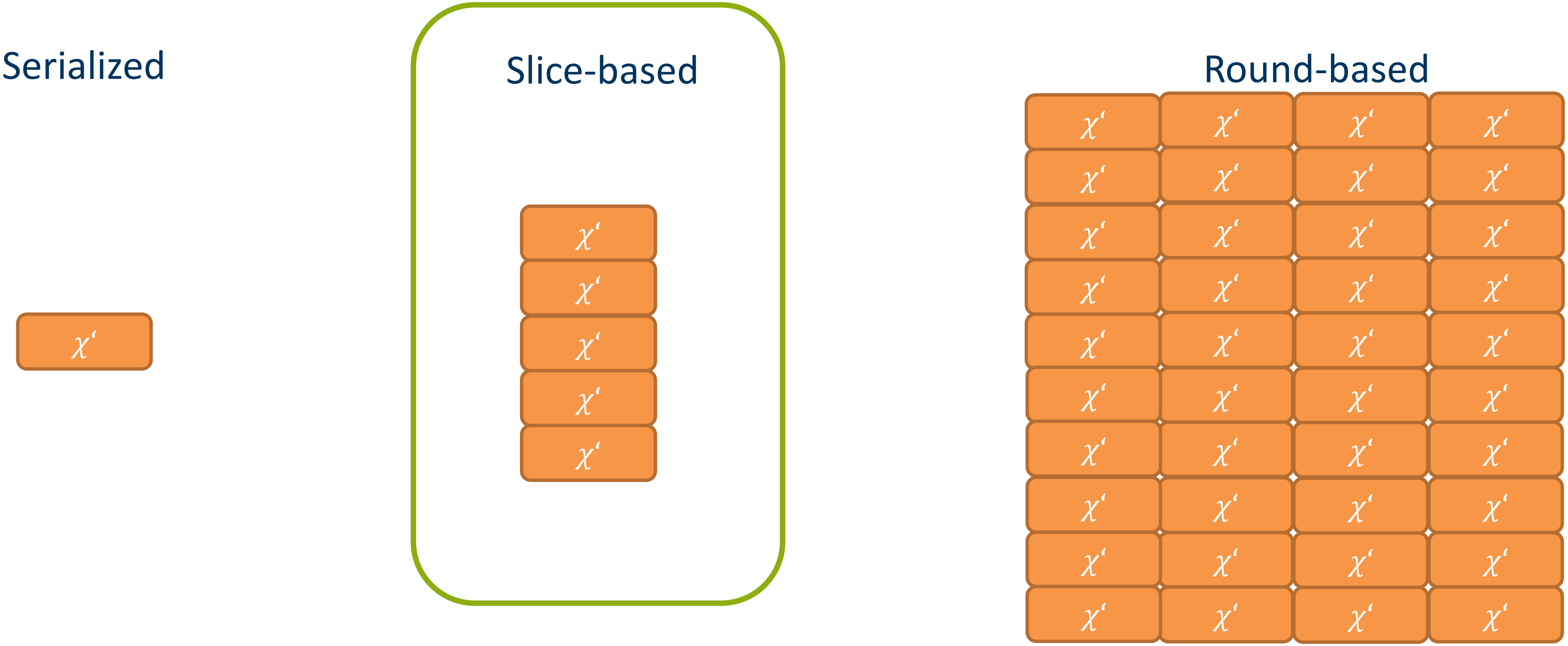
Slice-based

Round-based



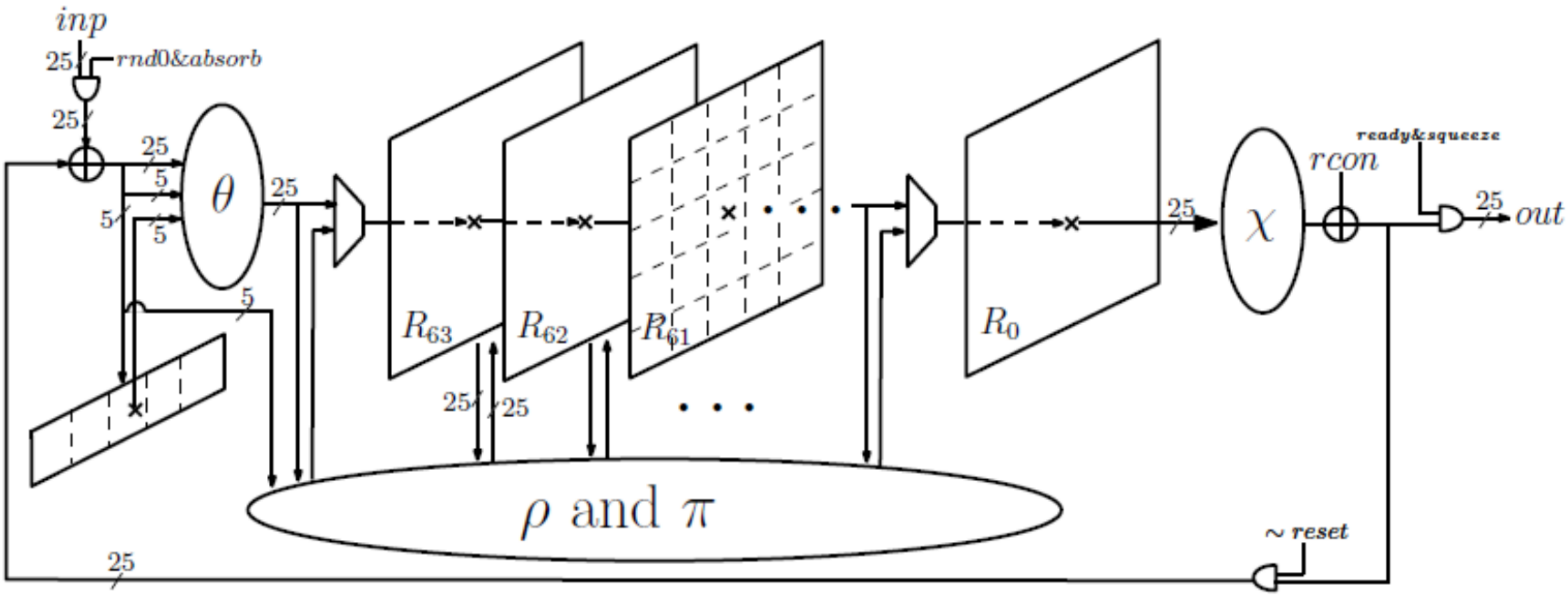
Hardware Architecture

How many parallel S-boxes?



Hardware Architecture

- Slice-Serial: 5 parallel χ evaluations
- Special treatment: θ applied to slice 0



Bilgin et al. Efficient and First-Order DPA Resistant Implementations of Keccak, CARDIS 2013

Leakage Evaluation

SCA-Measurements

RUB

Evaluation methodology:

- Non-specific T-test „fixed vs. Random“
  - over entire 200bit state
  - with 100 million traces
- Each trace: entire last round

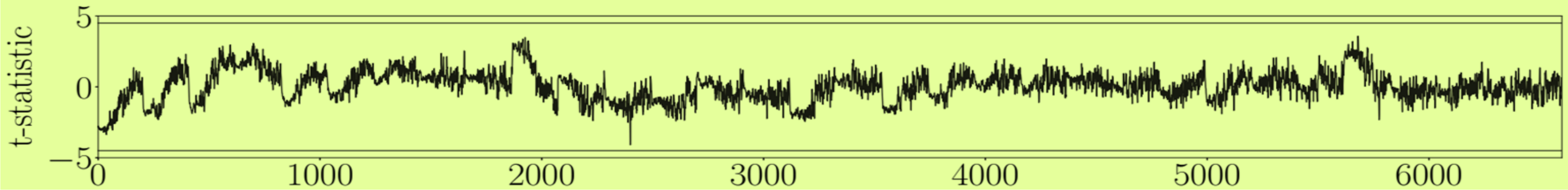
Measurement Setup:

- SAKURA-G board @ 1.5Mhz
- Picoscope 6402 @ 625 MS/s
- Amplifier: ZFL-100LN+ (Mini-Circuits)

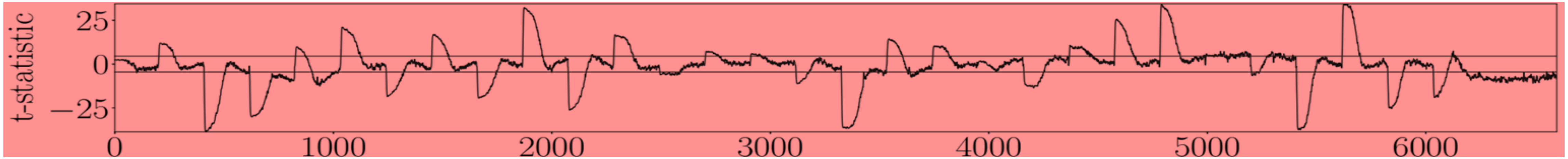
Schneider, Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations, CHES 2015

18 Rounds of Keccak

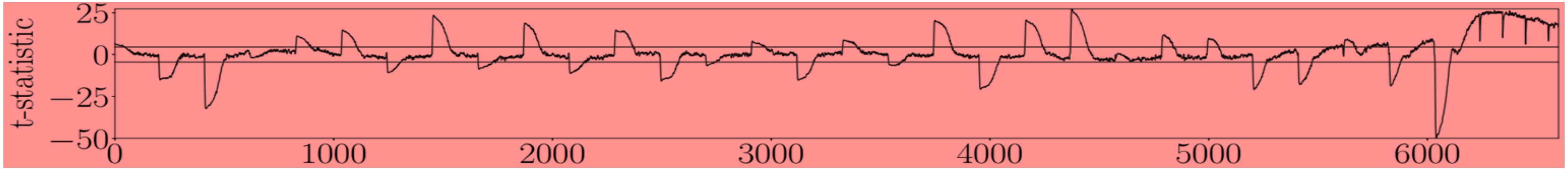
1. order over time



2. order over time

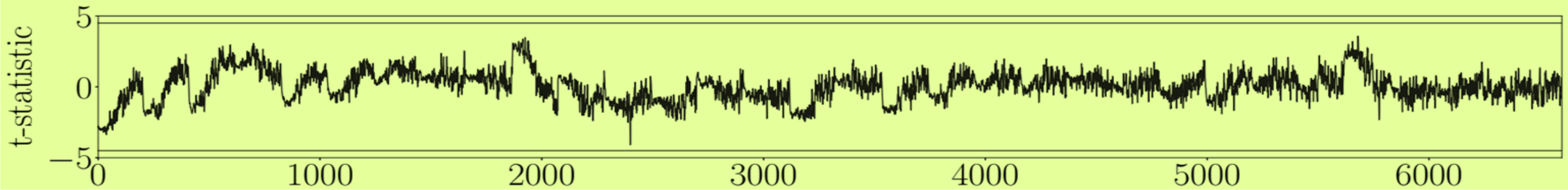


3. order over time

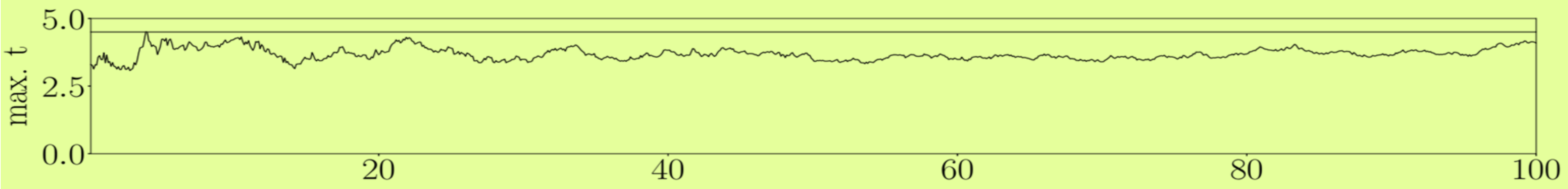


18 Rounds of Keccak

1. order over time

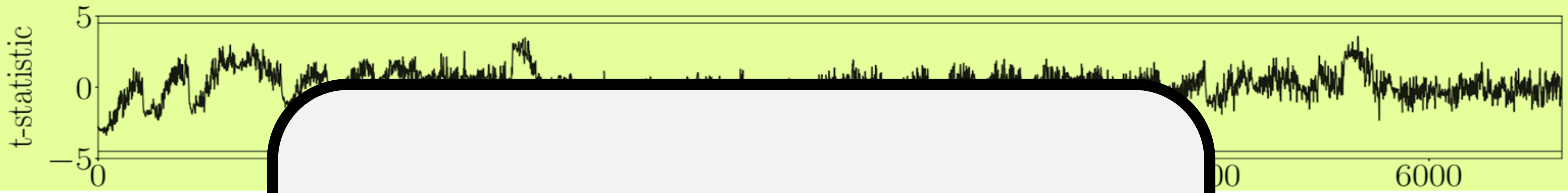


1. order over traces



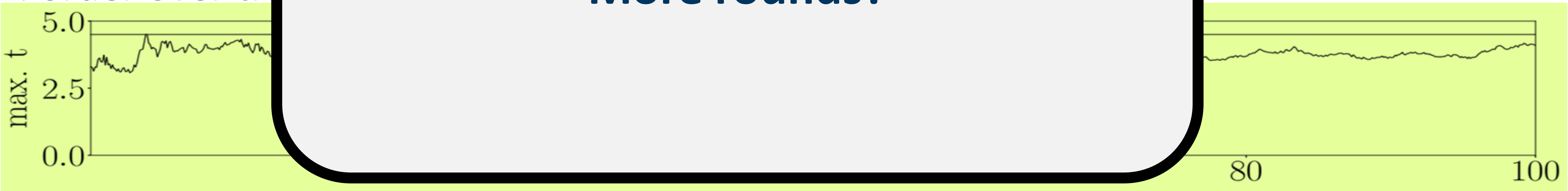
18 Rounds of Keccak

1. order over time



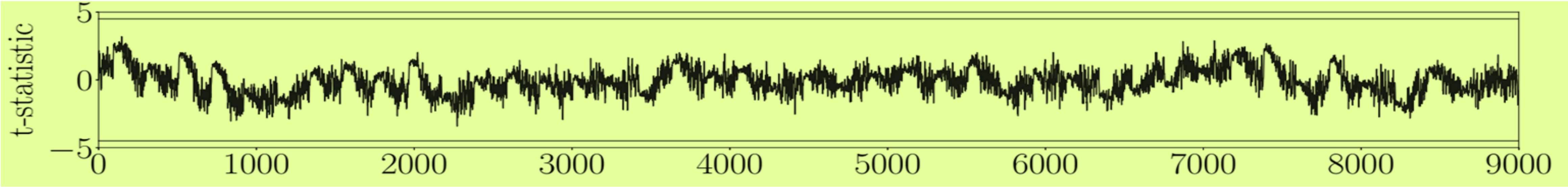
Works fine.
More rounds?

1. order over tra

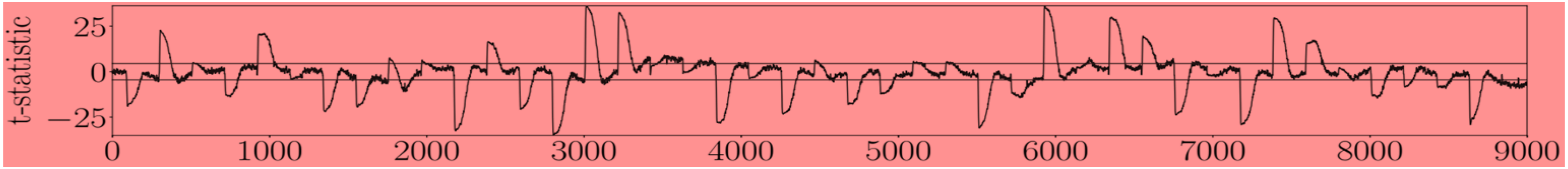


1800 Rounds of Keccak

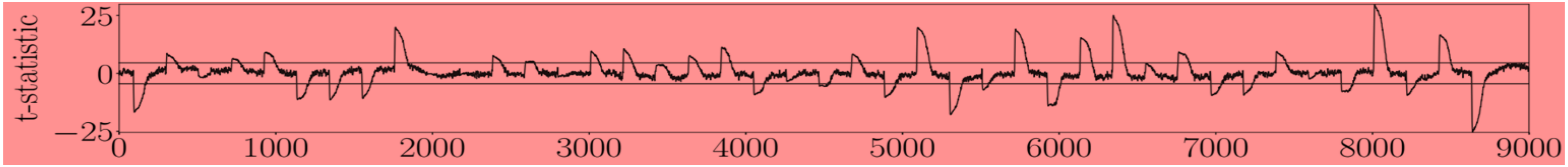
1. order over time



2. order over time

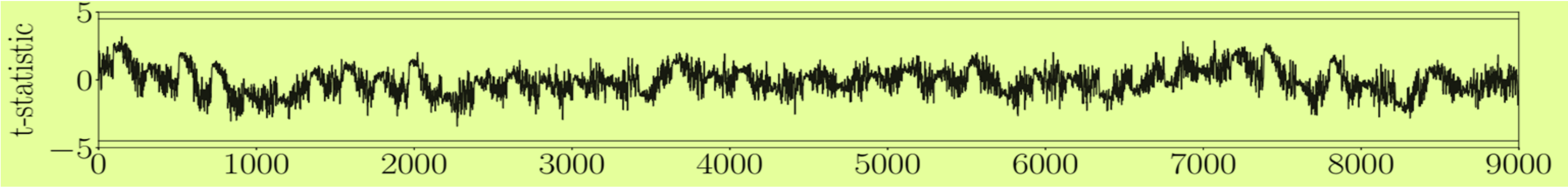


3. order over time

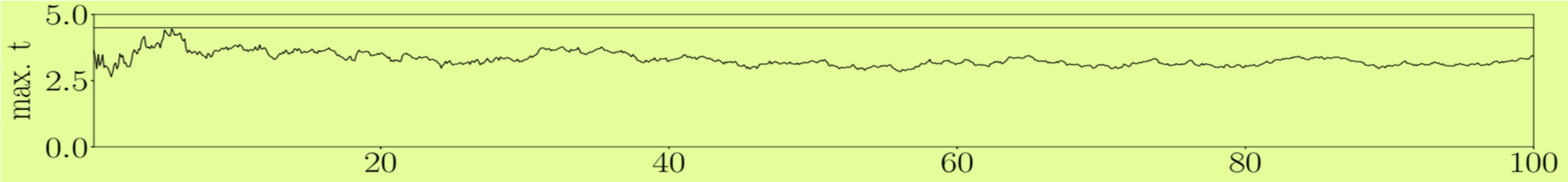


1800 Rounds of Keccak

1. order over time

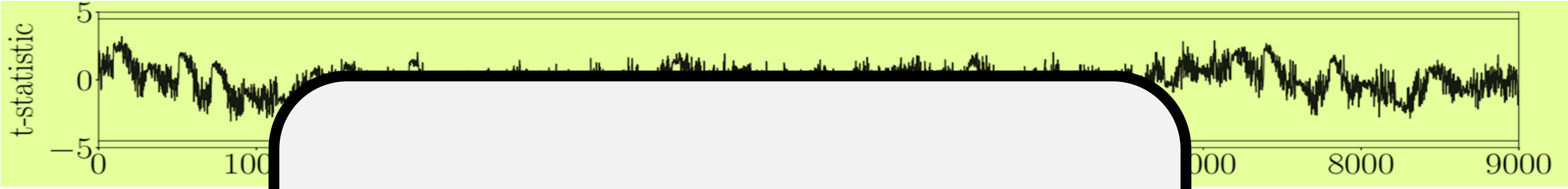


1. order over traces

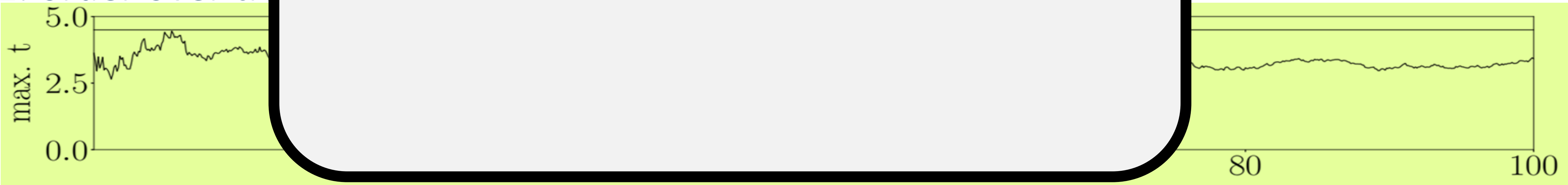


1800 Rounds of Keccak

1. order over time

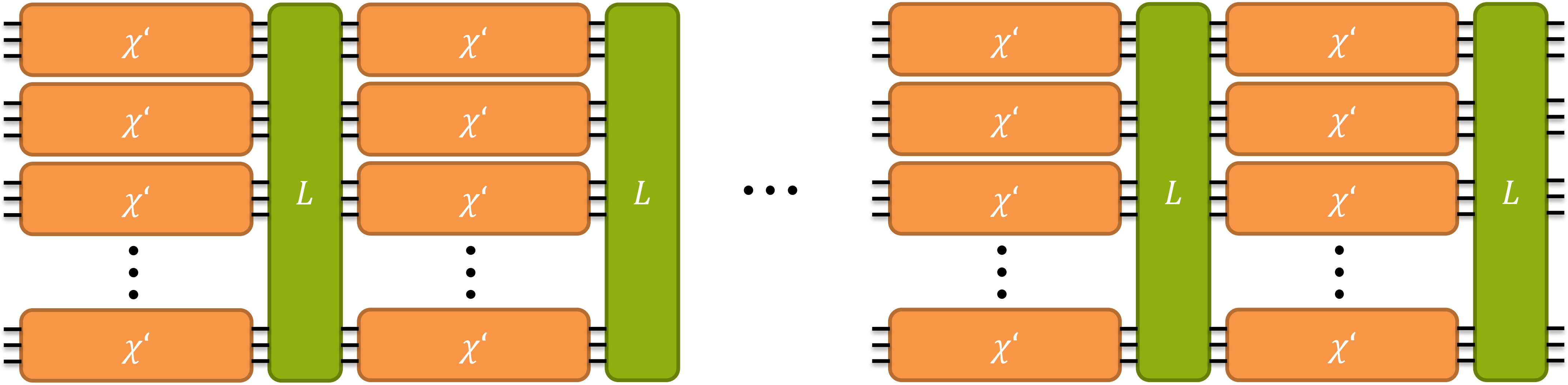


1. order over tra



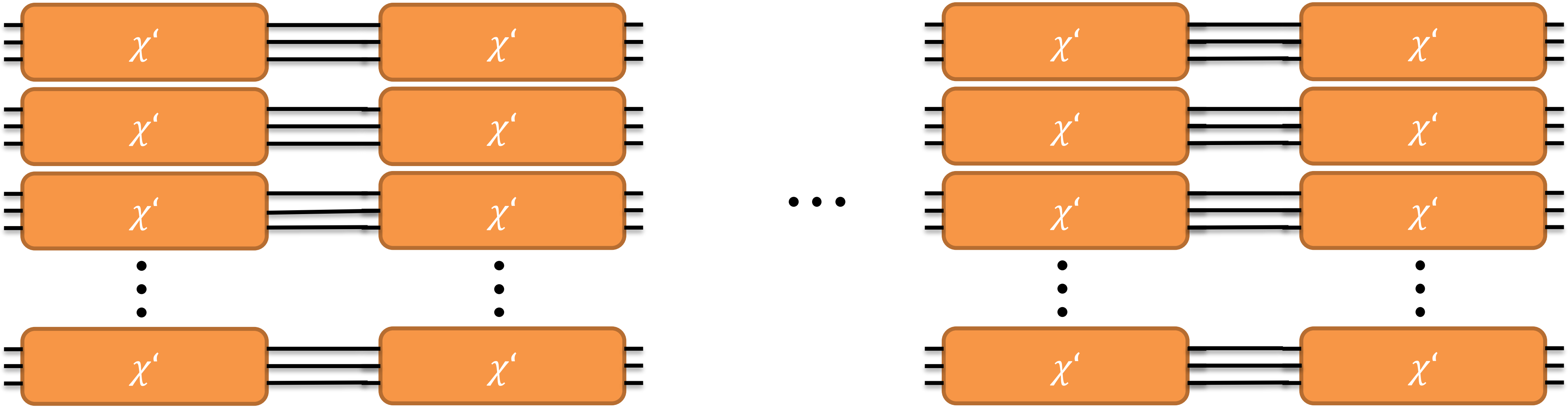
Origin of entropy?

Source of Diffusion: Linear Layer



Experiment: Remove Linear Layer

RUB



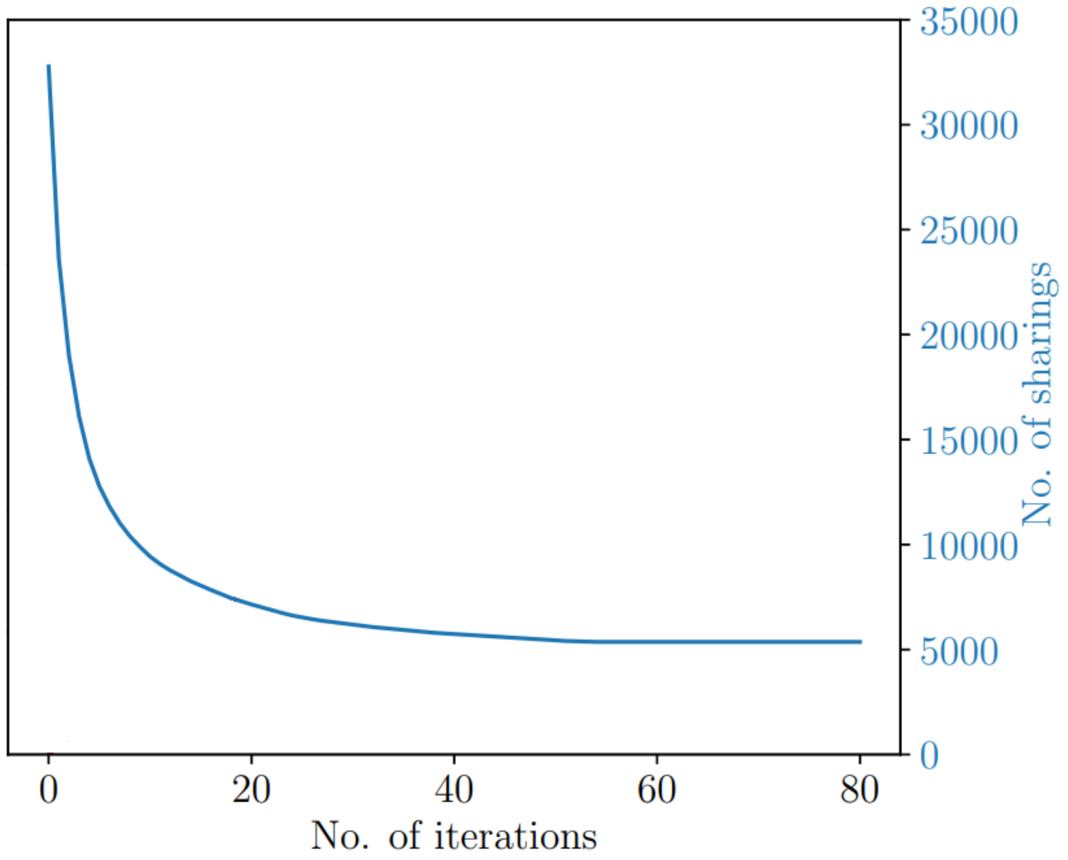
Simulation Part I

- Compute one instance of χ' on all 2^{15} inputs
- Feed outputs back into it
- Stop when plateau reached

Simulation Part I

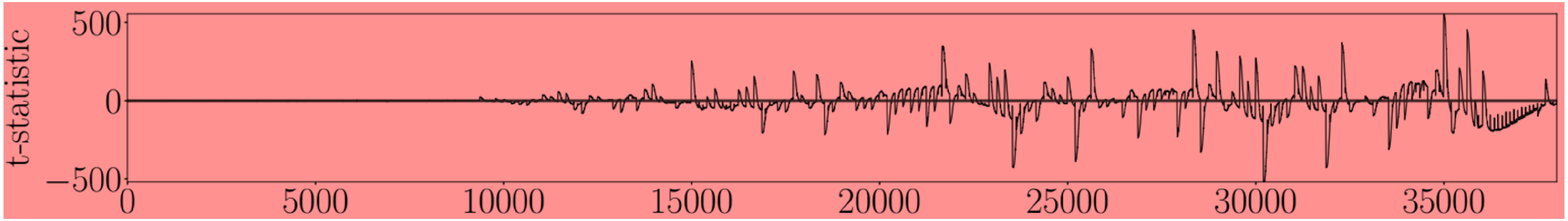
- Compute one instance of χ' on all 2^{15} inputs
- Feed outputs back into it
- Stop when plateau reached

Result:

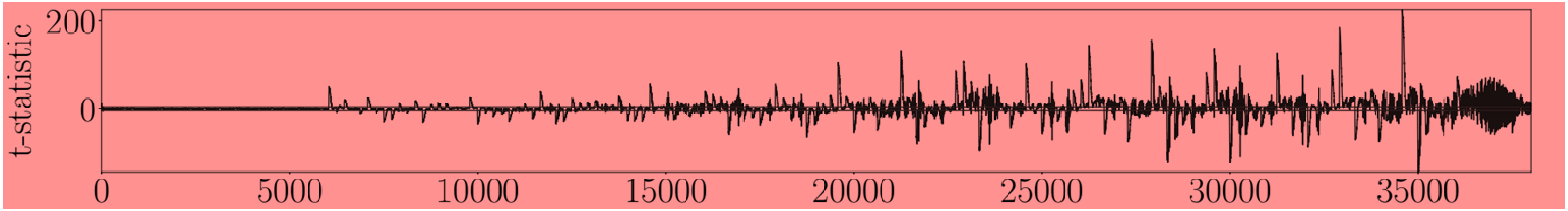


18 Rounds of χ'

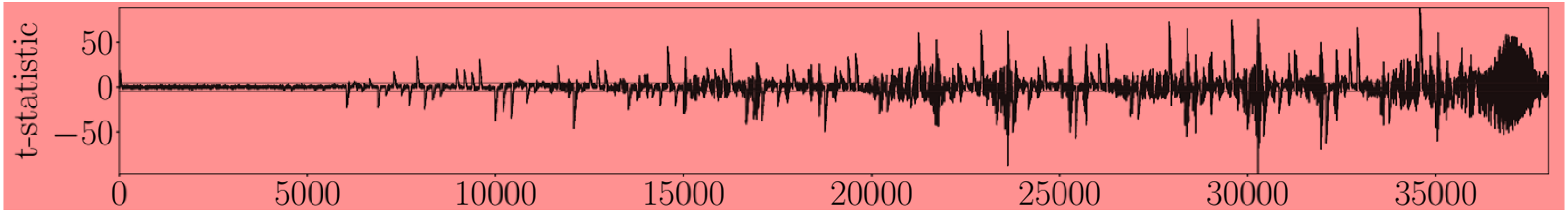
1. order over time



2. order over time



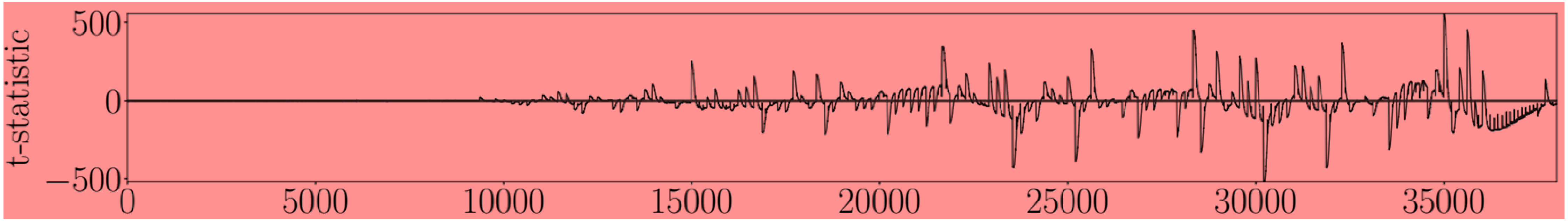
3. order over time



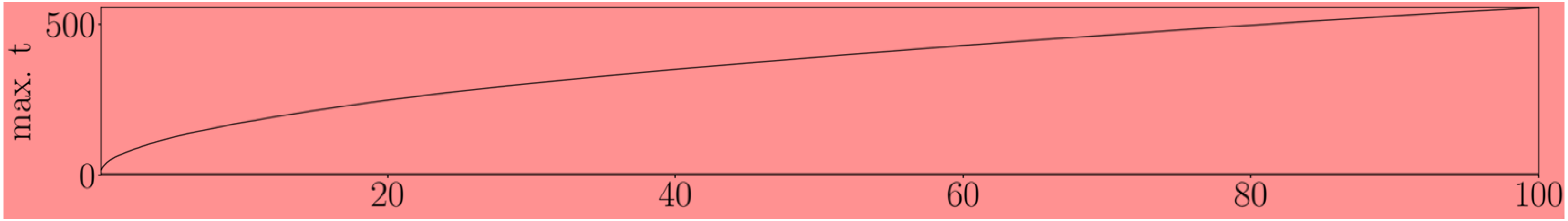
18 Rounds of χ'

RUB

1. order over time



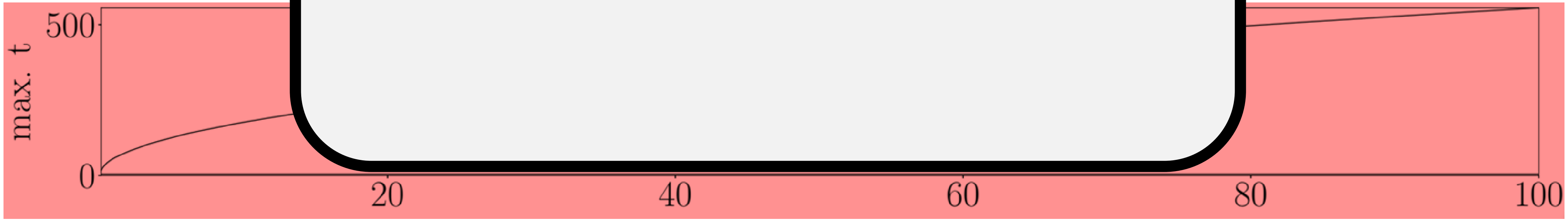
1. order over traces



18 Rounds of χ'

1. order over time

1. order over tra



How much diffusion is needed?

Linear Layer: Shuffling and Mixing

RUB

ρ

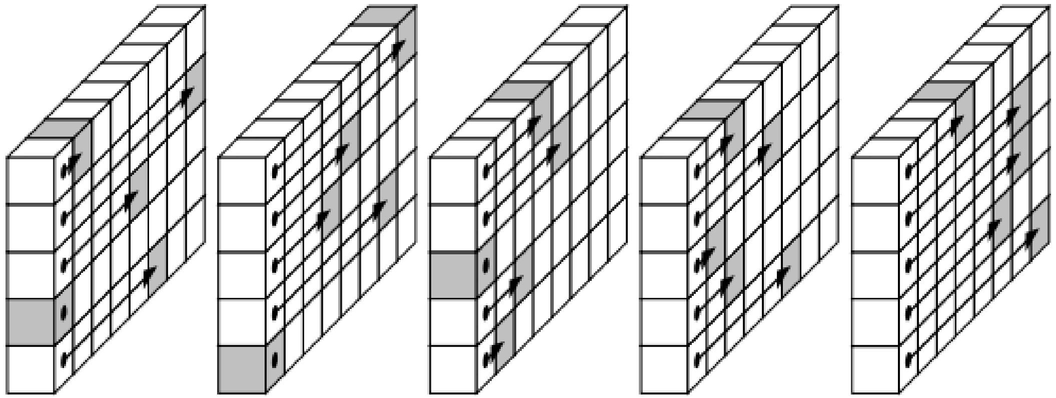
π

θ

ι

Linear Layer: Shuffling and Mixing

RUB



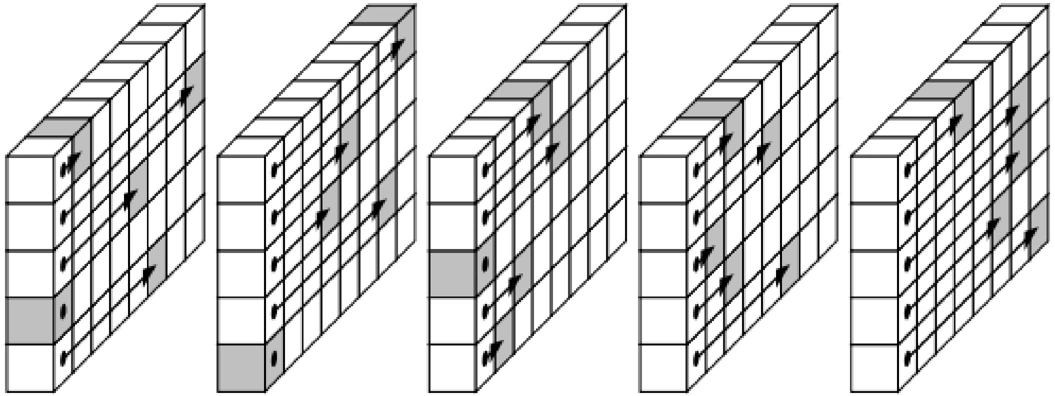
π

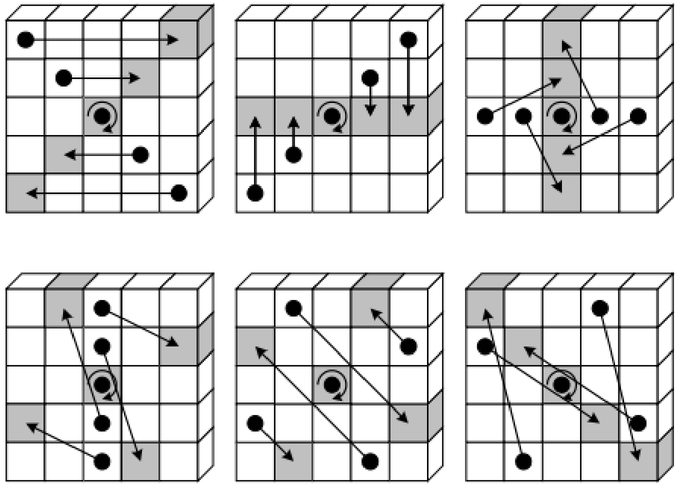
θ

ι

Bertoni et al. The Keccak Reference

Linear Layer: Shuffling and Mixing



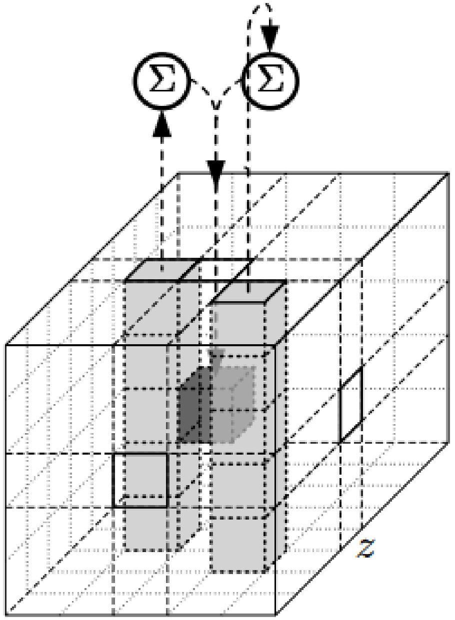


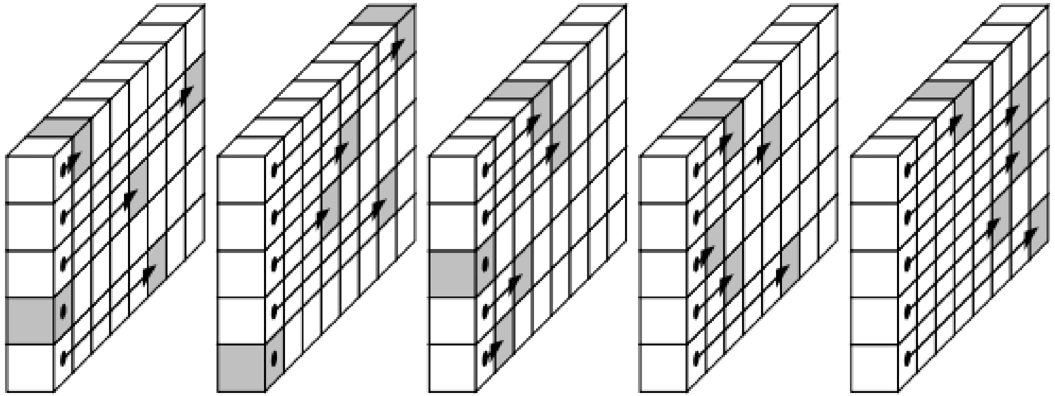
θ

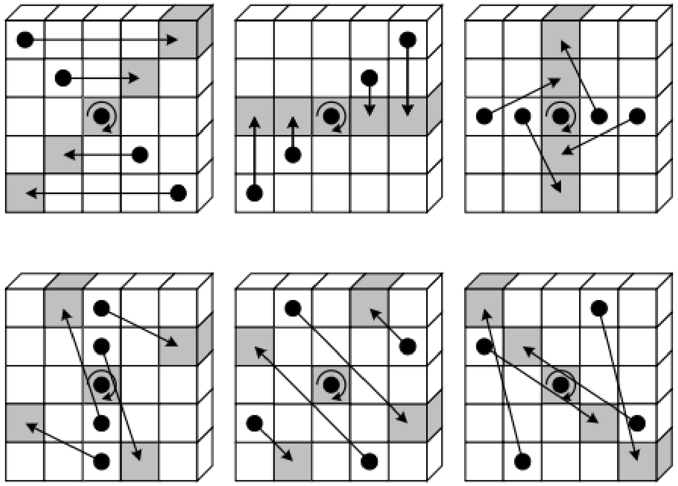
ι

Bertoni et al. The Keccak Reference

Linear Layer: Shuffling and Mixing



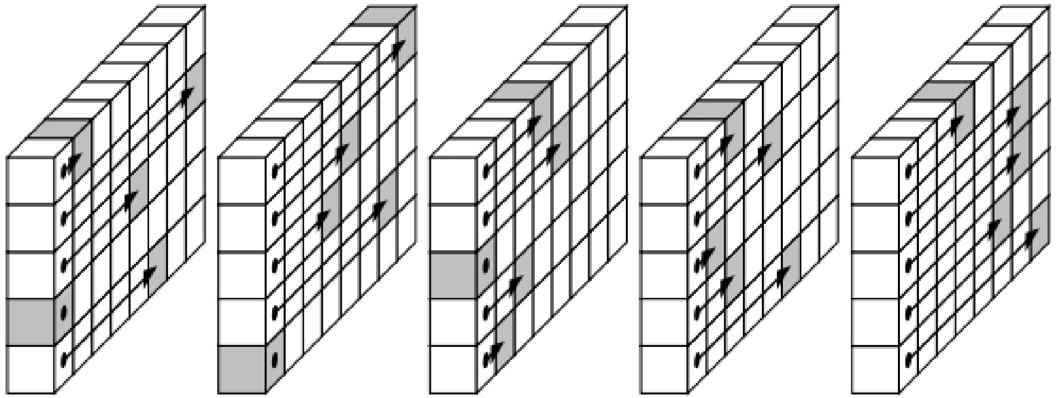


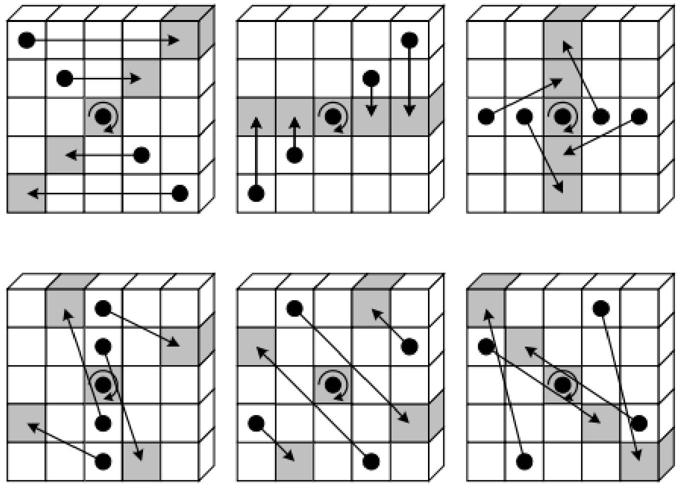


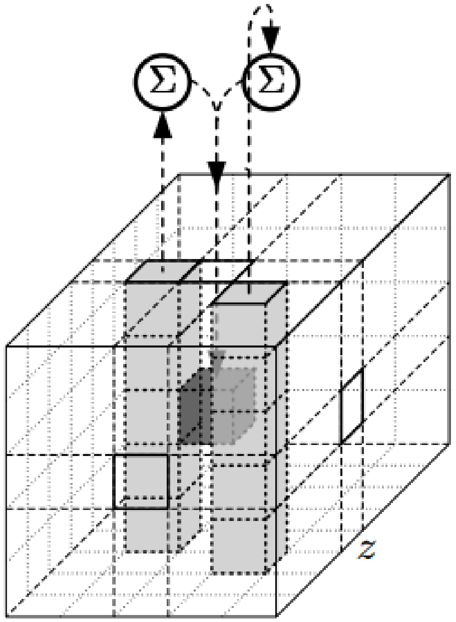
ι

Bertoni et al. The Keccak Reference

Linear Layer: Shuffling and Mixing





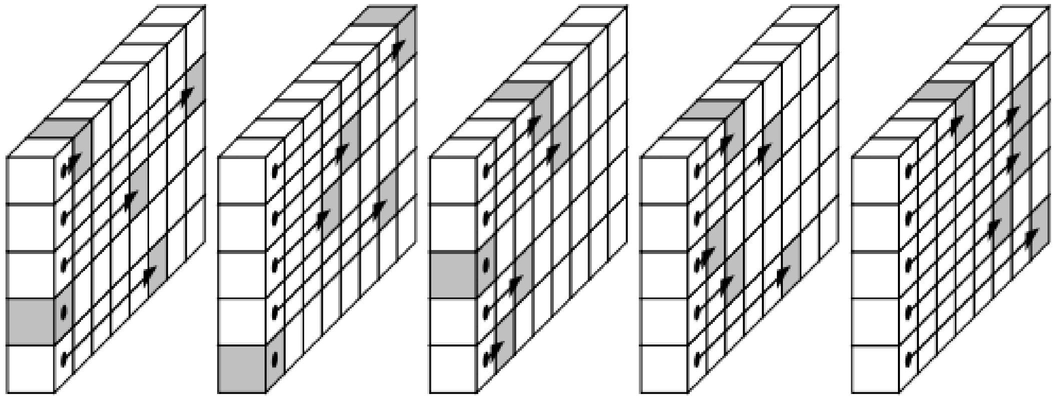


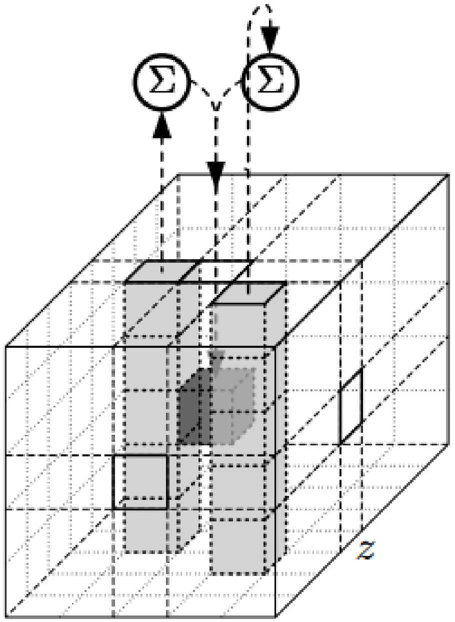
round constant

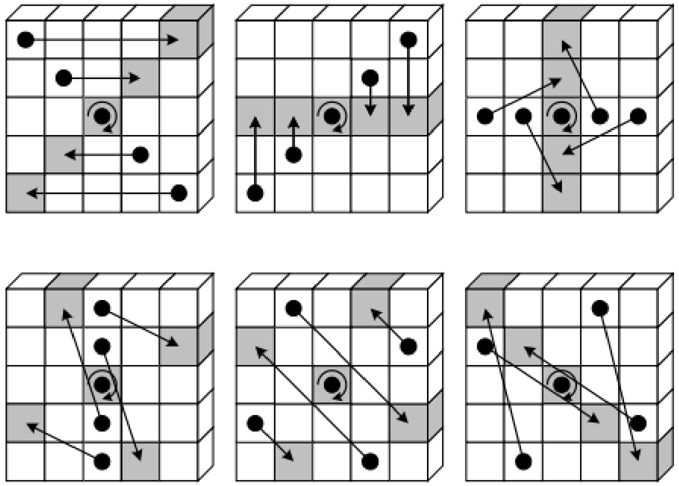
Bertoni et al. The Keccak Reference

Linear Layer: Shuffling and Mixing

RUB





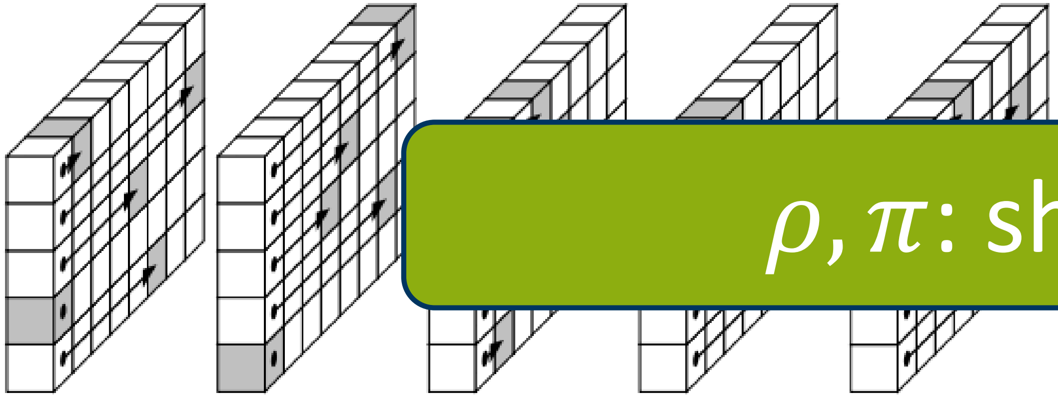


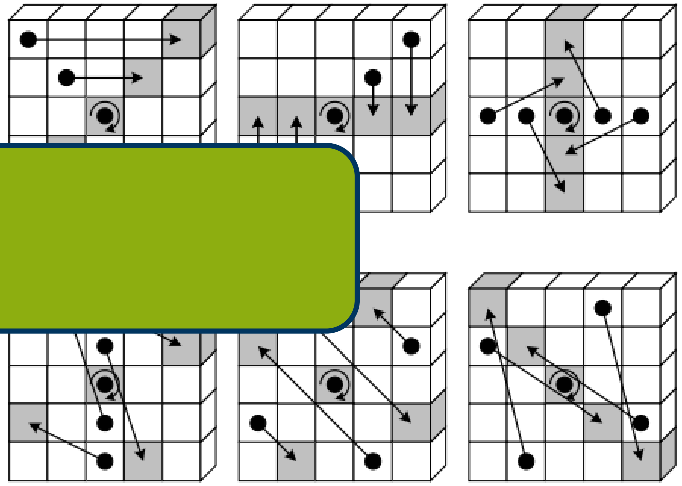
~~round constant~~

Bertoni et al. The Keccak Reference

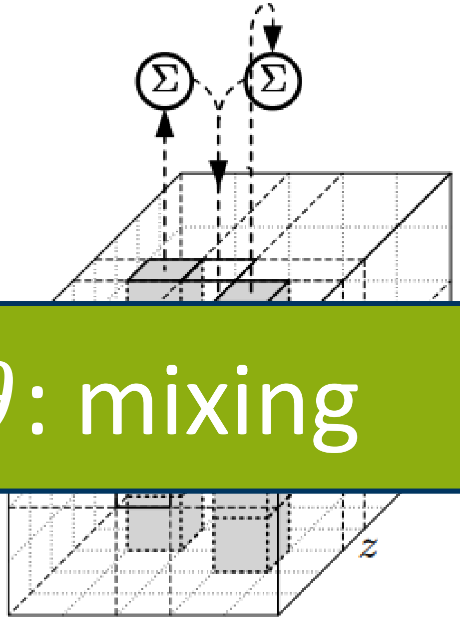
Linear Layer: Shuffling and Mixing

RUB





ρ, π: shuffling



θ: mixing

Bertoni et al. The Keccak Reference

Simulation Part II

RUB

How to simulate entropy of masked Keccak-*f*[200]?

Exhaustive Testing:
2^{600} states - impossible

Simulation Part II

RUB

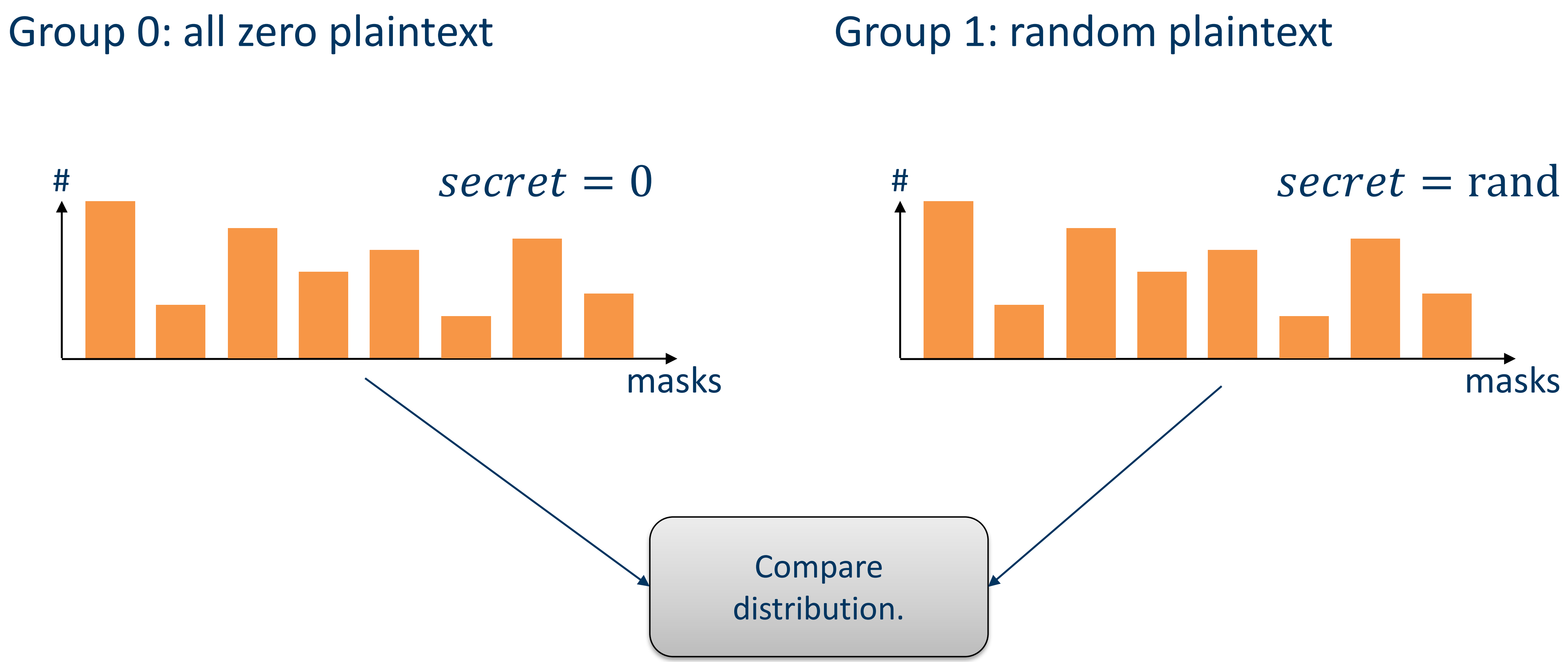
How to simulate entropy of masked Keccak-*f*[200]?

Exhaustive Testing:
2^{600} states - impossible

Sampling:
„fixed vs. random“
without power model

Simulation Part II

RUB



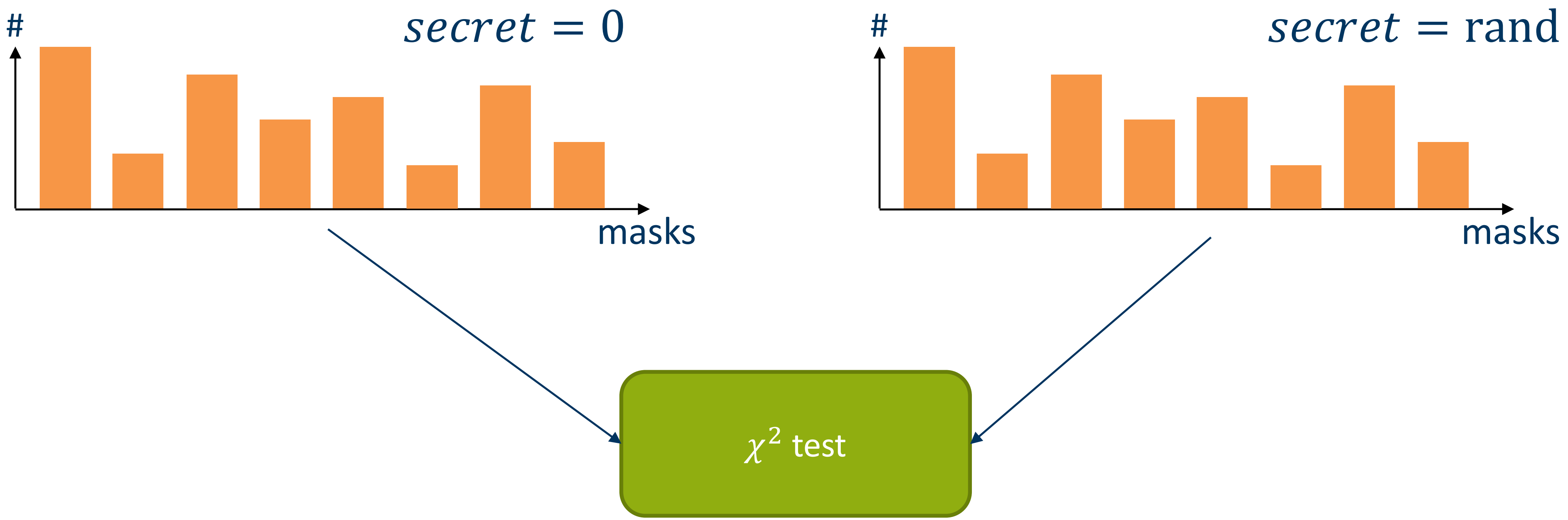
De Meyer, Bilgin, Reparaz. Consolidating Security Notions in Hardware Masking.

Simulation Part II

RUB

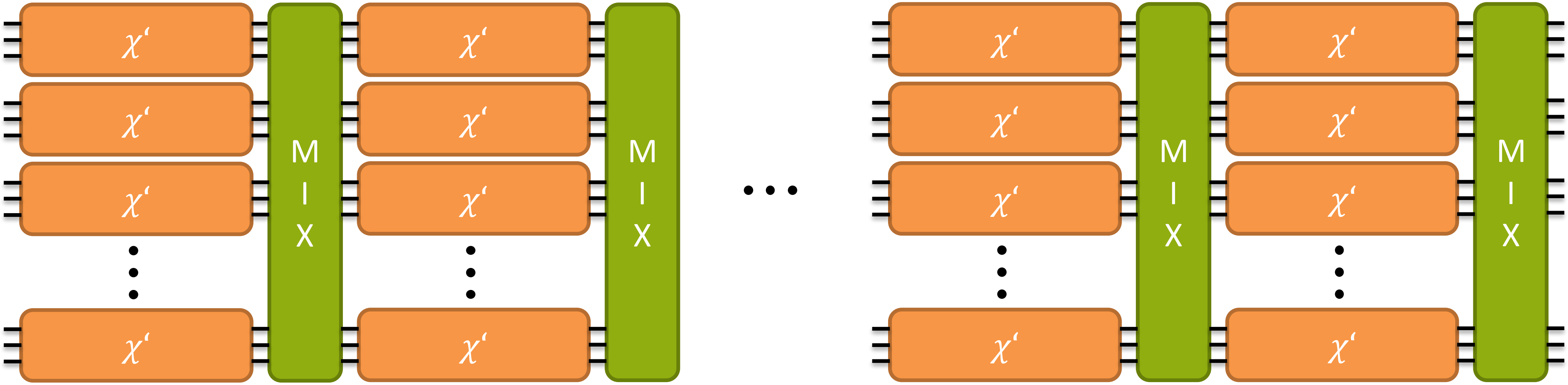
Group 0: all zero plaintext

Group 1: random plaintext

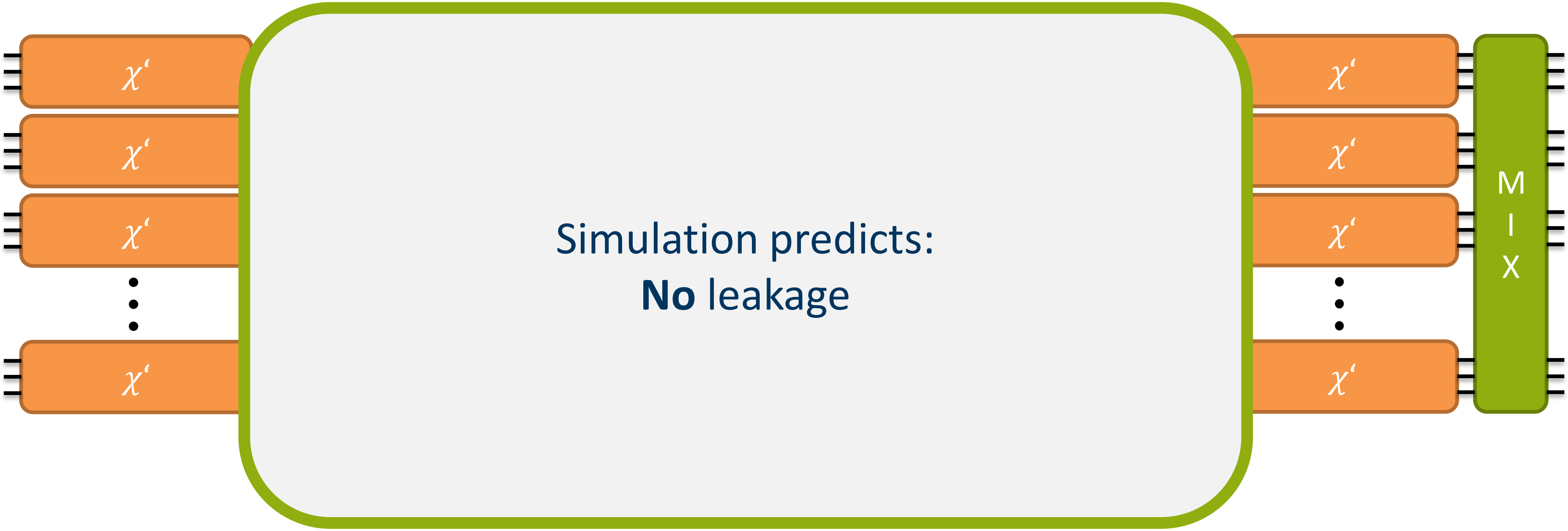


De Meyer, Bilgin, Reparaz. Consolidating Security Notions in Hardware Masking.

Next Design: Mix Only

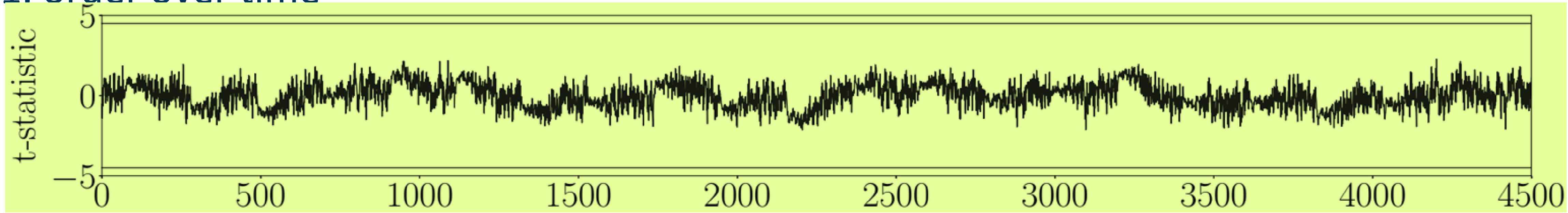


Next Design: Mix Only

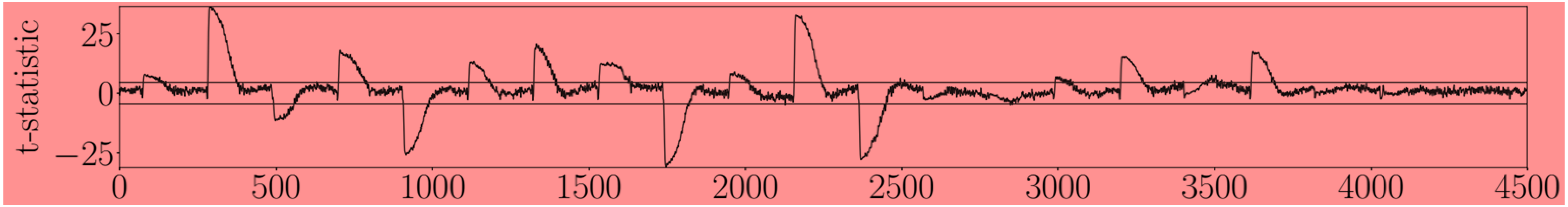


18 Rounds of Mixing: χ', θ

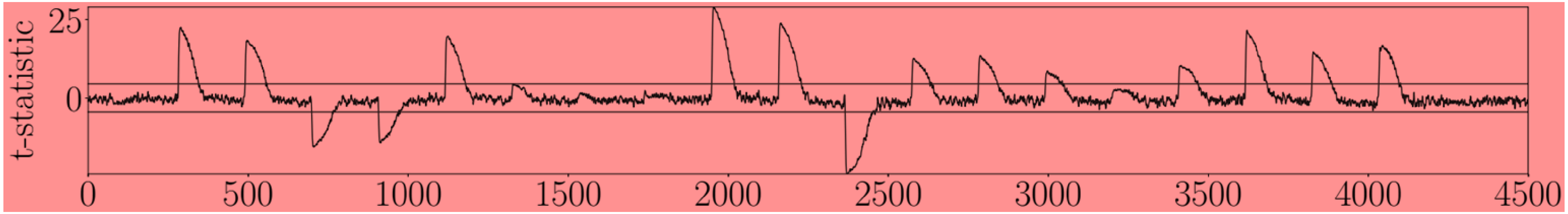
1. order over time



2. order over time

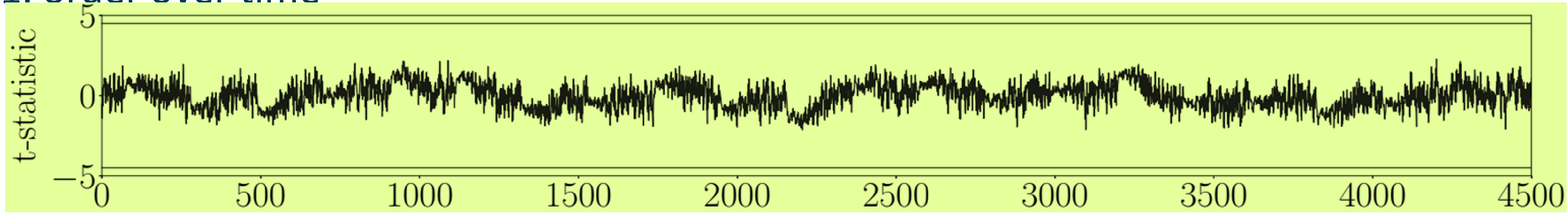


3. order over time

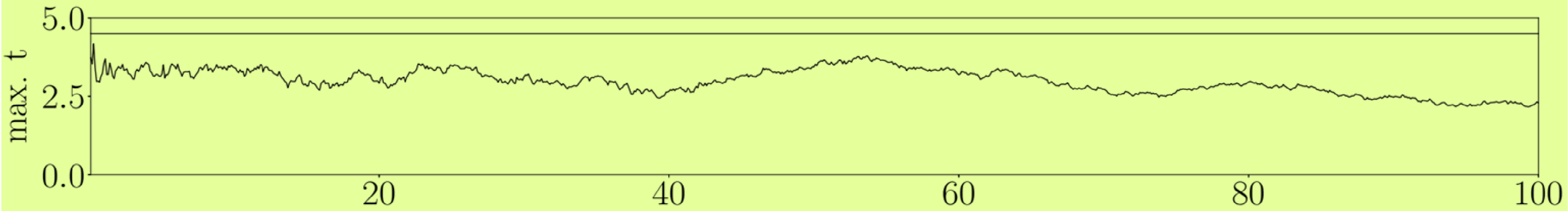


18 Rounds of Mixing: χ', θ

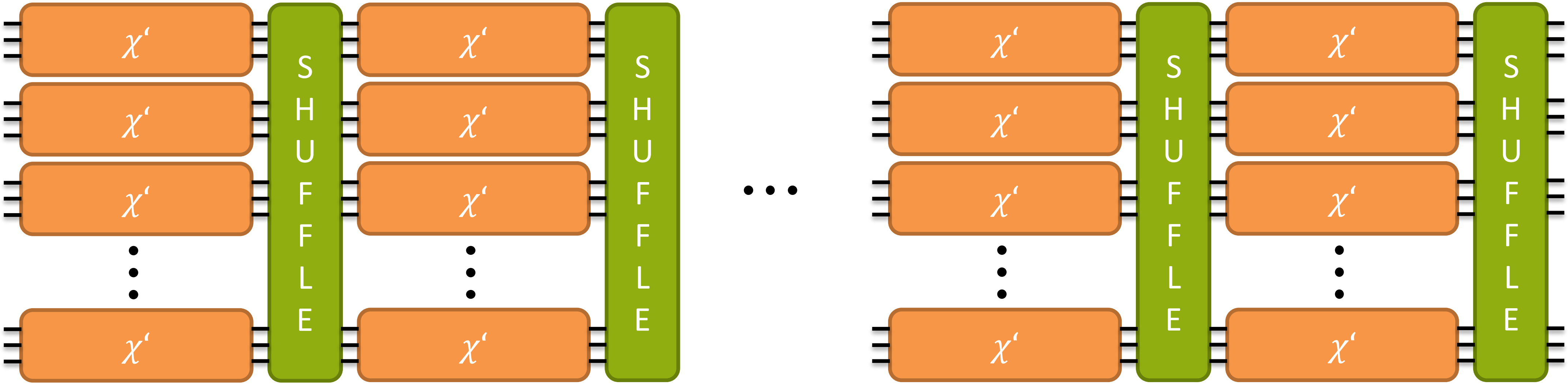
1. order over time



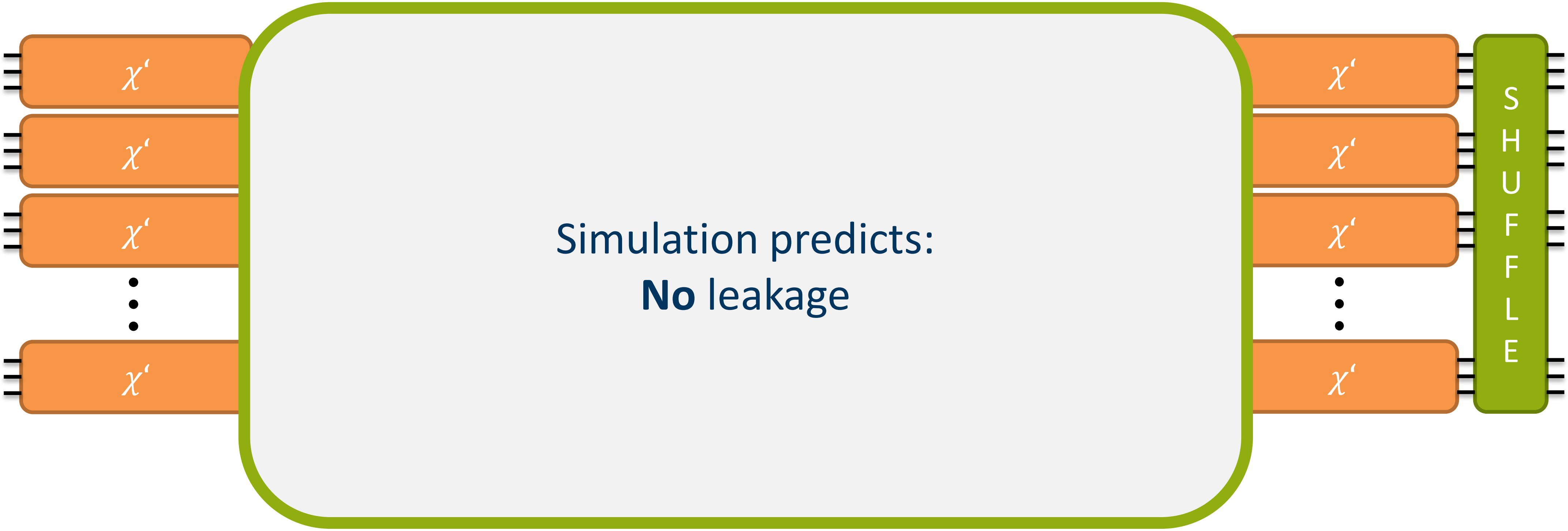
1. order over traces



Next Design: Shuffle Only



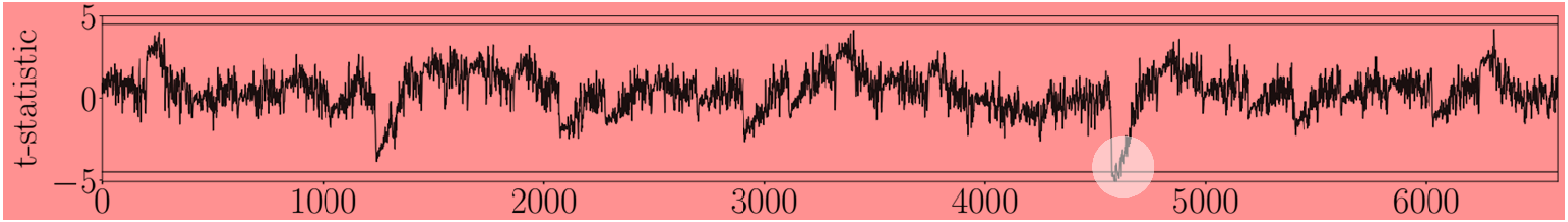
Next Design: Shuffle Only



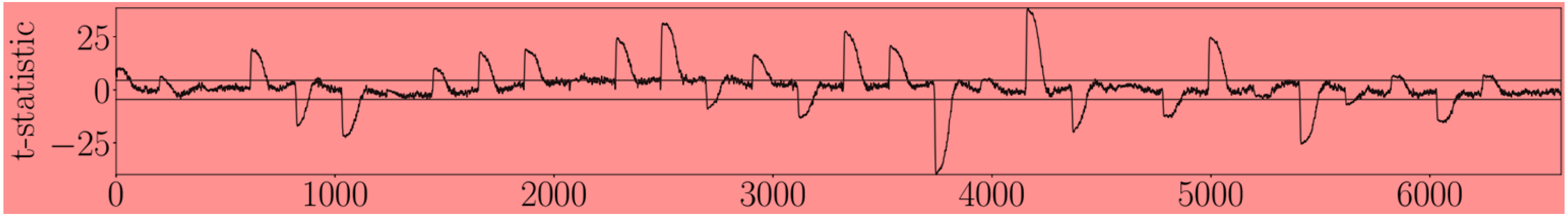
18 Rounds of Shuffling: χ', ρ, π

RUB

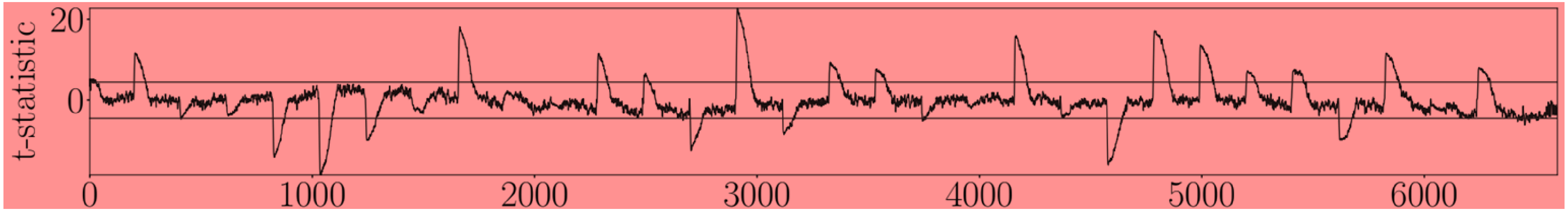
1. order over time



2. order over time



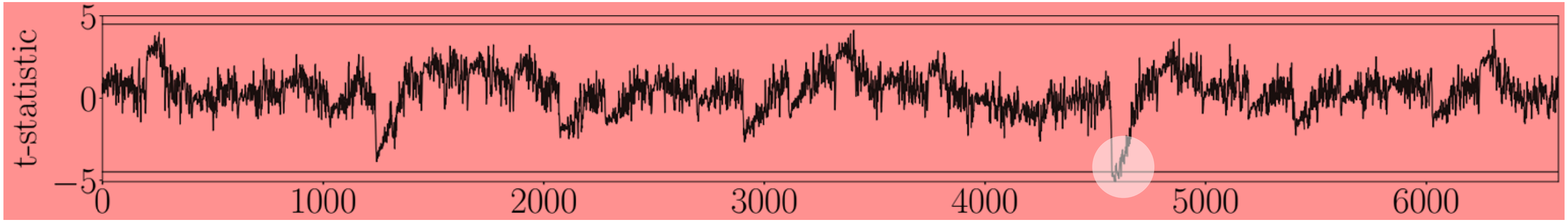
3. order over time



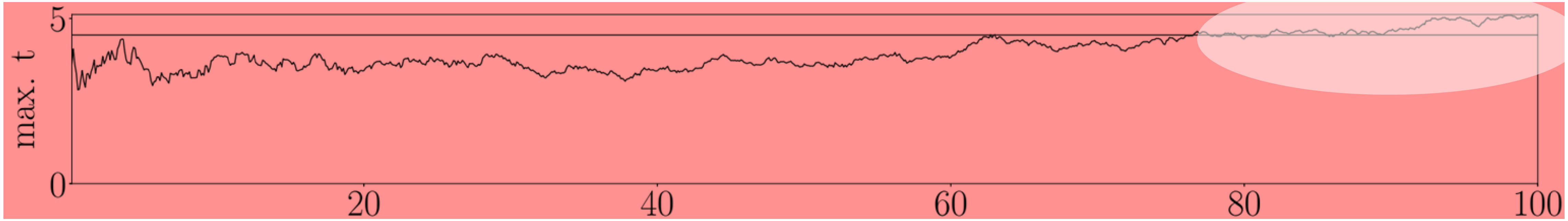
18 Rounds of Shuffling: χ', ρ, π

RUB

1. order over time



1. order over traces



Summary of Results

RUB

Practical Measurements

Active Layers	Detectable Leakage?
Sbox χ'	Yes!
Mix χ', θ	No.
Shuffle χ', ρ, π	**Yes.**
Shuffle and Mix χ', ρ, π, θ	No.

Simulations

Active Layers	Detectable Leakage?
Sbox χ'	Yes!
Mix χ', θ	No.
Shuffle χ', ρ, π	**No.**
Shuffle and Mix χ', ρ, π, θ	No.

Summary of Results

RUB

Practical Measurements

Active Layers	Detectable Leakage?
Sbox χ'	Yes!
Mix χ', θ	No.
Shuffle χ', ρ, π	**Yes.**
Shuffle and Mix χ', ρ, π, θ	No.

Simulations

Active Layers	Detectable Leakage?
Sbox χ'	Yes!
Mix χ', θ	No.
Shuffle χ', ρ, π	**No.**
Shuffle and Mix χ', ρ, π, θ	No.

Conclusion

RUB

Takeaways:

- Use Shuffle **and** Mix for entropy diffusion
- Combine simulations with practical evaluations

Caveats:

- Uniformity is essential in decomposed S-boxes:

Future Work:

- Evaluation of exploitable leakage
- Diffusion in other ciphers (e.g. ASCON)
- Quality criteria for RNG

Thanks! Any questions?

felix.wegener@rub.de

Felix Wegener, Christian Baiker, Amir Moradi

Ruhr University Bochum, Horst Görtz Institute for IT-Security, Germany

SPONSORED BY THE

Federal Ministry of Education and Research

Grant. Nr. 16KIS0666 SYSKIT_HW

Horst Görtz Institute for IT-Security