KU LEUVEN

Practical Aspects of Theoretical Models

Siemen Dhooghe
COSADE 2023

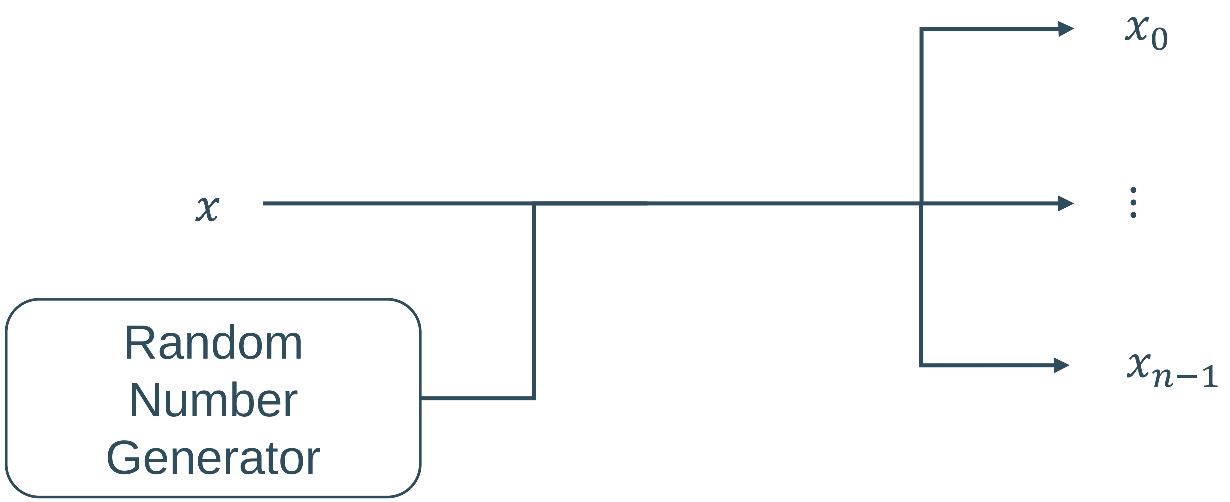
COSIC

In Short: Security Frameworks

- A timeline of security frameworks
  - Threshold implementations
  - Non-interference
- Adversary models and challenges
  - The probing model
  - The random probing model
  - The bounded query probing model
  - The bounded computational probing model

Boolean Masking

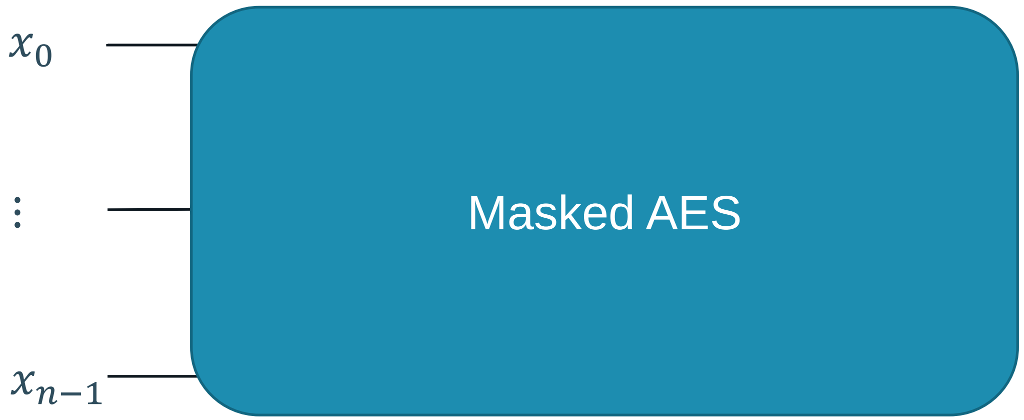
- Boolean masking[1,2] splits a variable $x \in \mathbb{F}_2$ in multiple parts $(x_0, \ldots, x_{n-1})$
  - $x = \sum_{i=0}^{n-1} x_i$
  - Each part is randomly distributed



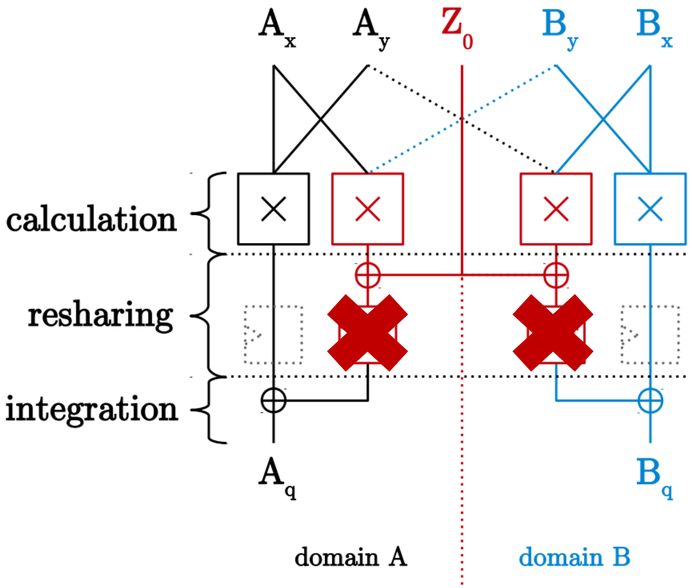
Paper cheat sheet

1. Goubin et al.: DES and differential power analysis (the "duplication" method)
2. Chari et al.: Towards sound approaches to counter-act power-analysis attacks

How not to Implement Masking



How not to Implement Masking



Paper cheat sheet

1. Gross et al.: Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order

Threshold Implementations: Non-Completeness

- Glitches can make implementations insecure as shown by Mangard et al.
- Non-completeness by Nikova et al. requires that the combinatorial logic can not use all shares

Example: Multiplier

$$z_0 = F_0(x_0, x_1, y_0, y_1) = x_0 y_0 \oplus x_0 y_1 \oplus x_1 y_0$$

$$z_1 = F_1(x_1, x_2, y_1, y_2) = x_1 y_1 \oplus x_1 y_2 \oplus x_2 y_1$$

$$z_2 = F_2(x_0, x_2, y_0, y_2) = x_2 y_2 \oplus x_0 y_2 \oplus x_2 y_0$$

Dependencies

z_0		
x_0	x_1	
y_0	y_1	

z_1		
	x_1	x_2
	y_1	y_2

z_2		
x_0		x_2
y_0		y_2

Paper cheat sheet

1. Mangard et al.: Successfully attacking masked AES hardware implementation
2. Nikova et al.: Threshold Implementations Against Side-Channel Attacks and Glitches

Threshold Implementations: Uniformity

- For n Boolean shares, all sets of $n-1$ shares are uniformly random distributed
  - For example, $(x_0, x_1) \in \mathbb{F}_2$ needs
    - x_0 is a uniform random bit
    - x_1 is a uniform random bit
    - (x_0, x_1) together are not uniform because $x_0 + x_1 = x$
- More context on the previous example

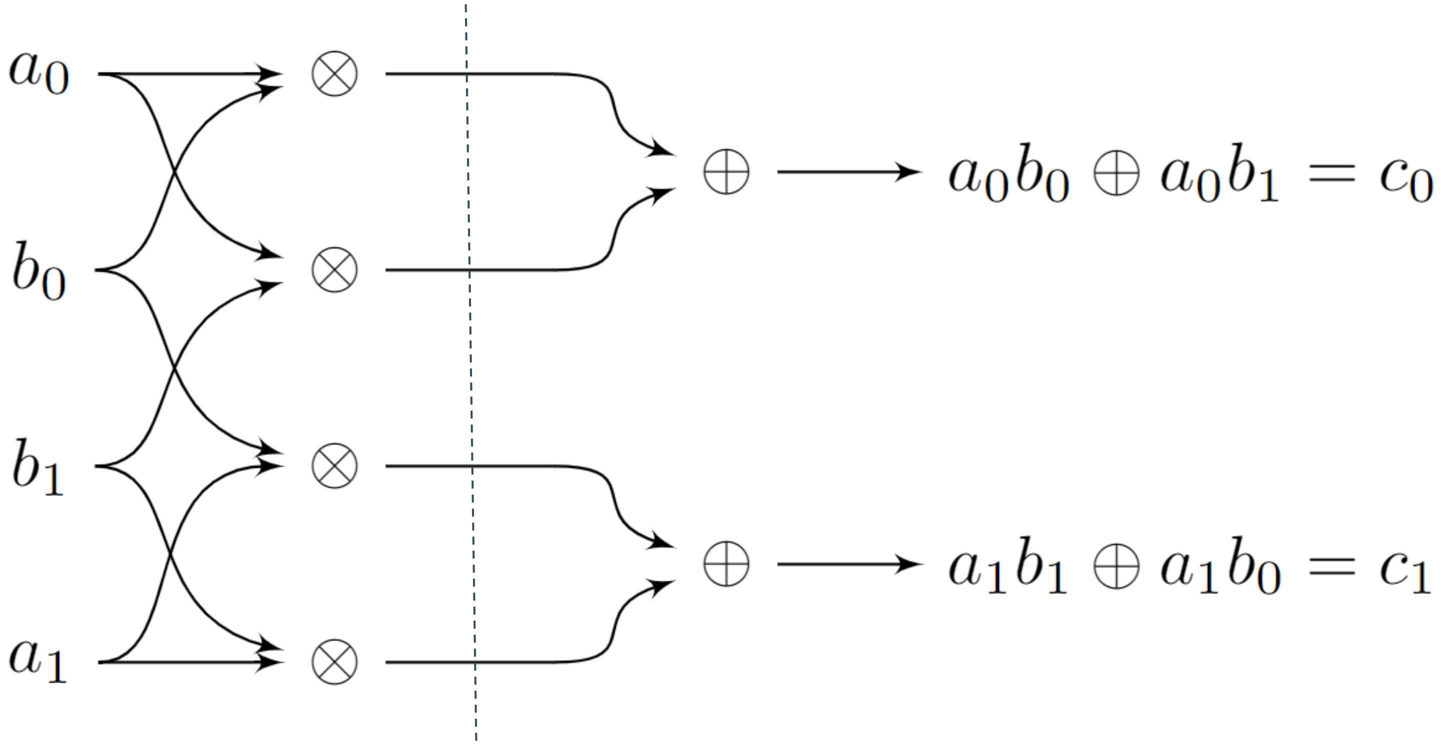
$$z_0 = F_0(x_0, x_1, y_0, y_1) = x_0 y_0 \oplus x_0 y_1 \oplus x_1 y_0$$

$$z_1 = F_1(x_1, x_2, y_1, y_2) = x_1 y_1 \oplus x_1 y_2 \oplus x_2 y_1$$

$$z_2 = F_2(x_0, x_2, y_0, y_2) = x_2 y_2 \oplus x_0 y_2 \oplus x_2 y_0$$

Threshold Implementations: Uniformity

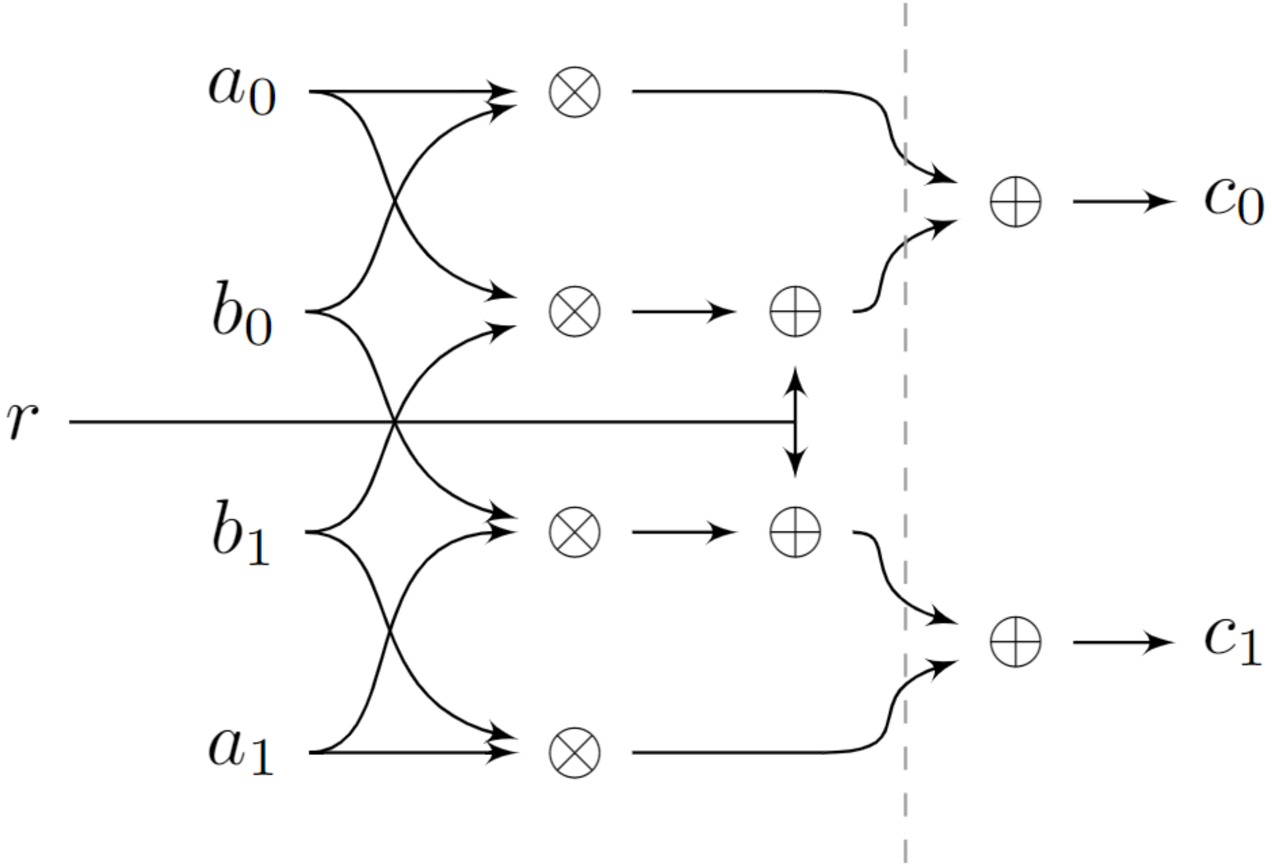
- A uniform shared input has to be mapped to a uniform shared output
  - Your shared function has to be balanced/a permutation



a_0	b_0	b_1	c_0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Not Balanced

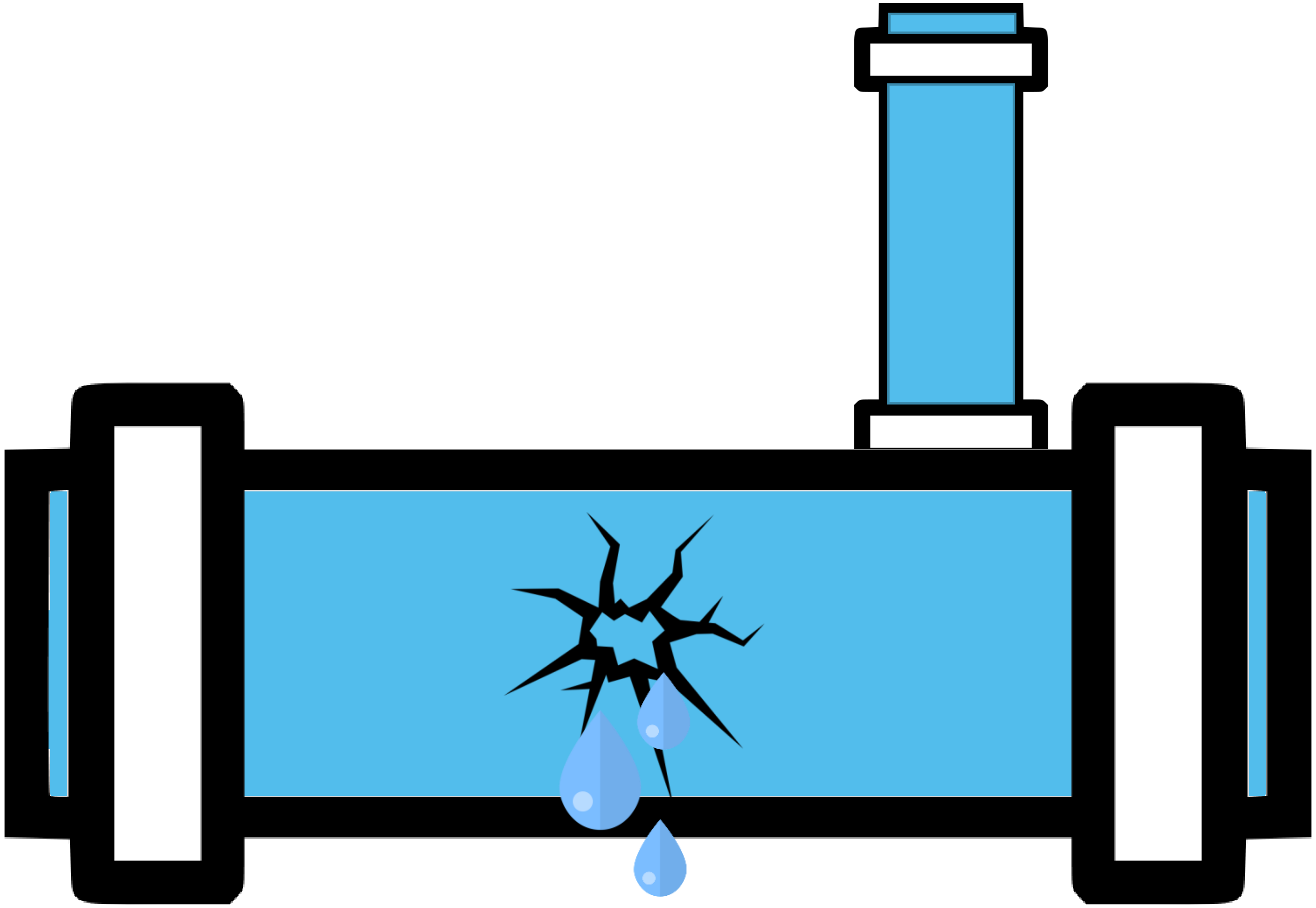
Threshold Implementations: Uniformity



a_0	b_0	b_1	r	c_0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

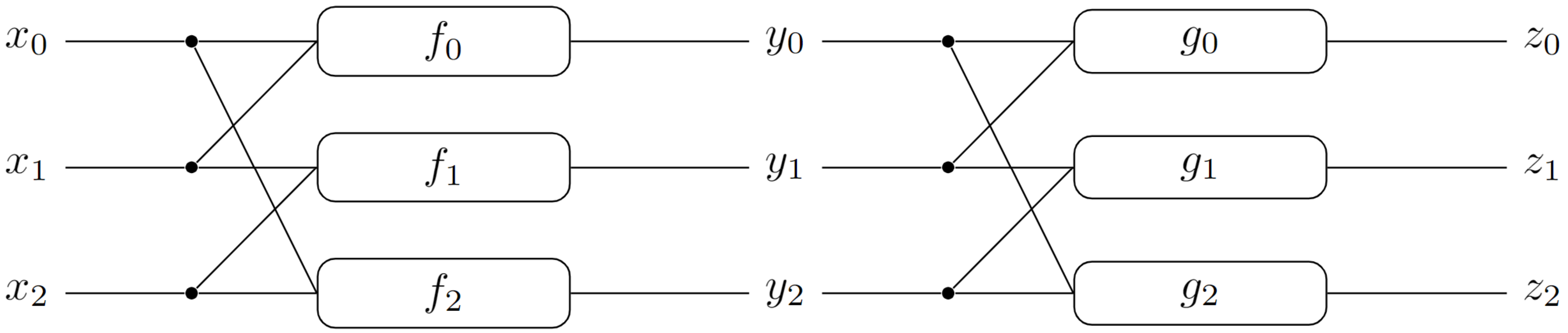
Balanced

Threshold Implementations: Uniformity



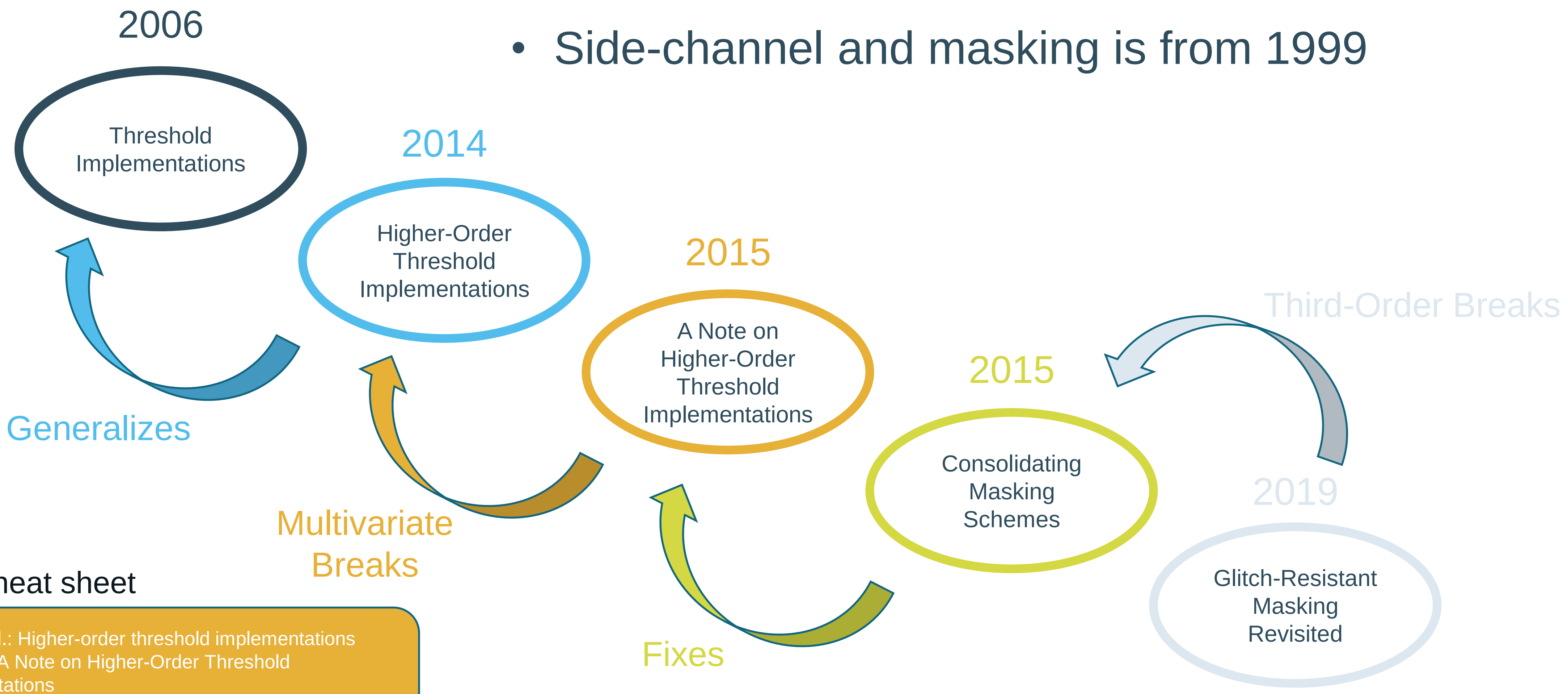
Higher-Order Attacks and Masking

- A first-order attack essentially views one masked function
- In a higher-order attack, the adversary views multiple functions
  - In a univariate attack: only functions in one cycle
  - In a multivariate attack: functions across multiple cycles



Higher-Order Threshold Implementations

- Side-channel and masking is from 1999

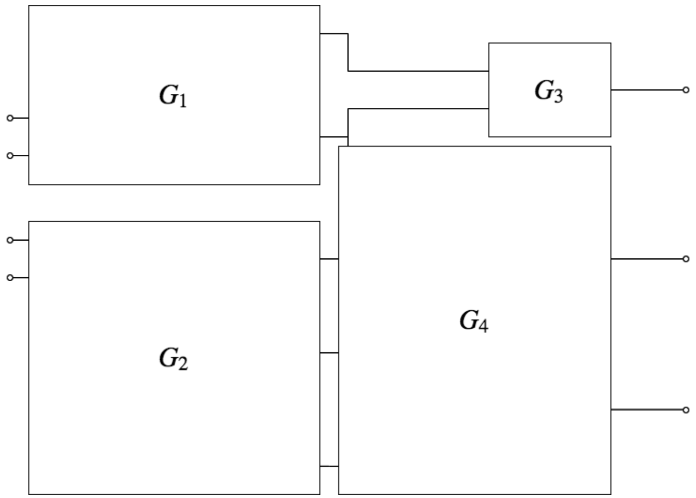


Paper cheat sheet

1. Bilgin et al.: Higher-order threshold implementations
2. Reparaz: A Note on Higher-Order Threshold Implementations
3. Reparaz et al.: Consolidating Masking Schemes
4. Moos et al.: Glitch-Resistant Masking Revisited

Non-Interference (NI)

- A framework by Barthe et al. from 2015 providing compositional security
  - Some different frameworks include PINI by Cassiers et al. and IOS by Goudarzi et al.
- The circuit is now divided in gadgets
  - Each gadget is proven (S)NI
  - Ensures security for the whole circuit



Paper cheat sheet

1. Barthe et al.: Strong non-interference and type-directed higher-order masking
2. Cassiers et al.: Trivially and efficiently composing masked gadgets with probe isolating non-interference
3. Goudarzi et al.: Probing security through input-output separation and revisited quasilinear masking

Composable Security

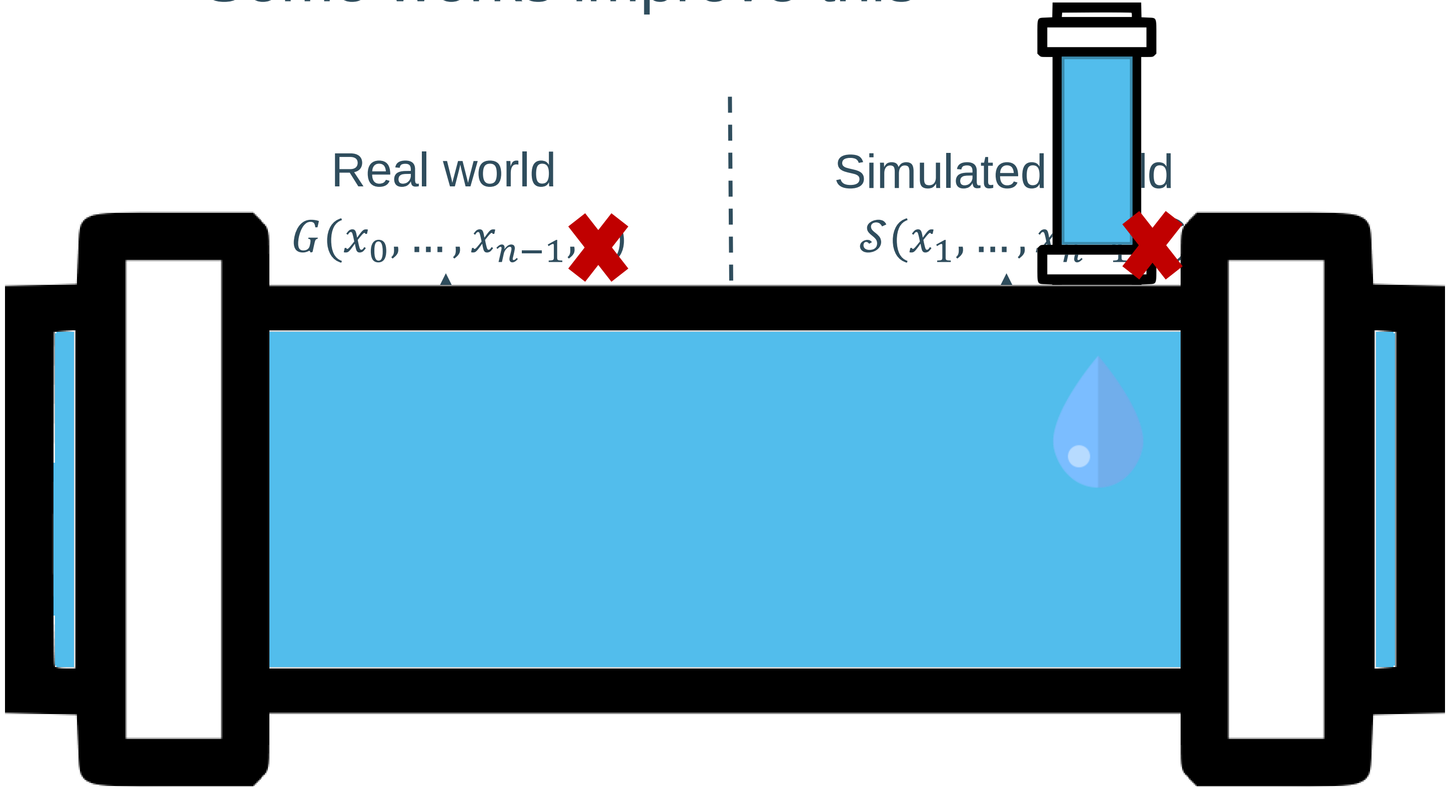
- Allows for higher-order secure circuits
- Allows for the easy verification of circuits
  - MaskVerif by Barthe et al.
  - Ironmask by Belaid et al.
  - SILVER by Knichel et al.
- Allows for the automatization of masked circuits
  - Via replacing AND/XORs using ISW-like approaches
  - By transforming functions directly, e.g. Knichel et al.

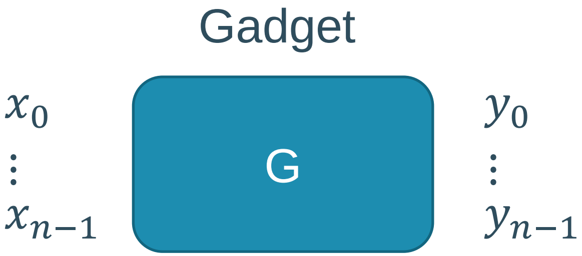
Paper cheat sheet

1. Barthe et al.: maskVerif: Automated verification of higherorder masking in presence of physical defaults
2. Belaid et al.: Ironmask: Versatile verification of masking security
3. Knichel et al.: SILVER - statistical independence and leakage verification
4. Knichel et al.: Generic Hardware Private CircuitsTowards Automated Generation of Composable Secure Gadgets

The Other Side of Composable Security

- Simulation-based security always needs randomness
  - Some works improve this[1,2]





Paper cheat sheet

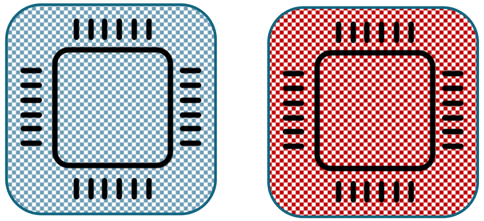
1. Faust et al.: Amortizing Randomness Complexity in Private Circuits
2. Feldtkeller et al.: Randomness Optimization for Gadget Compositions in Higher-Order Masking

The Randomness Cost

- An example AES masking from De Cnudde et al.
  - Uses an unrolled PRINCE to generate randomness

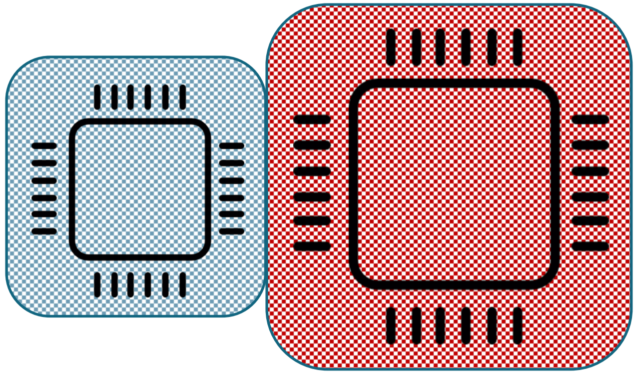
First-Order

Masking Randomness



Second-Order

Masking Randomness

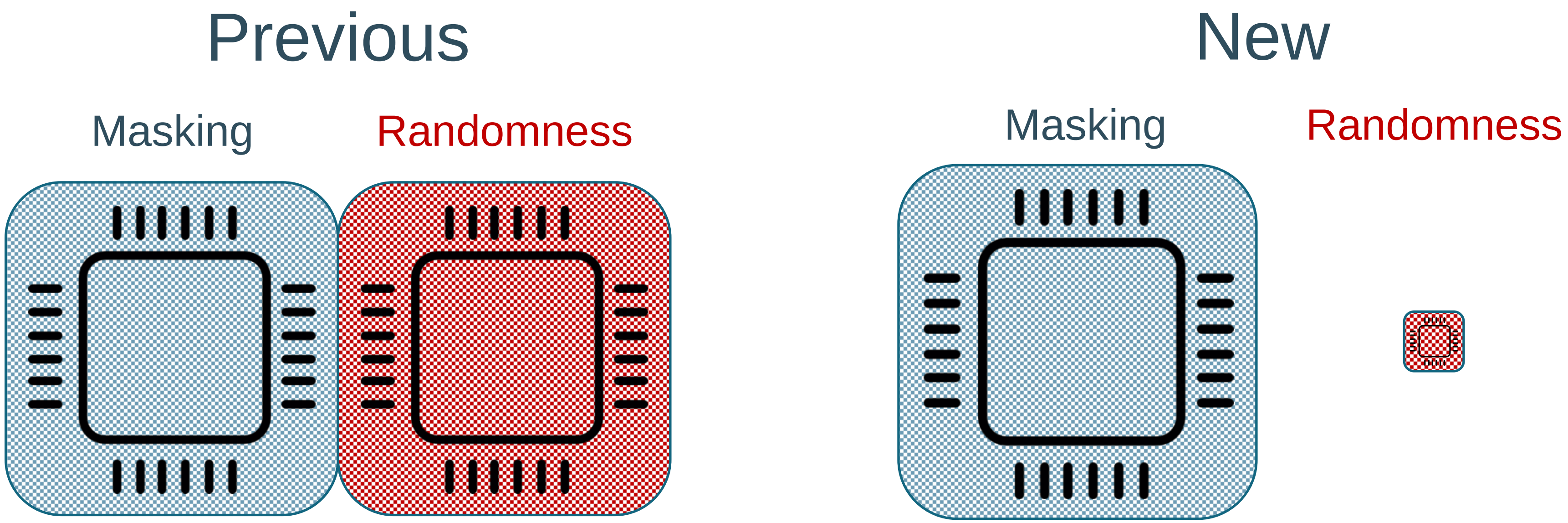


Paper cheat sheet

1. De Cnudde et al.: Masking AES with d + 1 Shares in Hardware

The Randomness Cost

- First-order low-randomness AES maskings[1,2]



Paper cheat sheet

1. Askeland et al.: Guarding the First Order: The Rise of AES Maskings
2. Shahmirzadi et al.: Re-Consolidating First-Order Masking Schemes Nullifying Fresh Randomness

The Randomness Cost

- First-order low-randomness AES maskings[1,2]
- Second-order low-randomness maskings of lightweight ciphers[3]
- A new framework for higher-order threshold implementations based on cryptanalysis[4]
  - Based on a bounded number of probing queries
  - Low-randomness second-order AES maskings[5,6]
  - Low-randomness second-order lightweight ciphers[7,8]
- However, the maskings are handmade
  - No automatic verification or automatization of the whole circuit

Paper cheat sheet

1. Askeland et al.: Guarding the First Order: The Rise of AES Maskings
2. Shahmirzadi et al.: Re-Consolidating First-Order Masking Schemes Nullifying Fresh Randomness
3. Shahmirzadi et al: Second-Order SCA Security with almost no Fresh Randomness
4. Beyne et al.: Cryptanalysis of Masked Ciphers: A not so Random Idea
5. Beyne et al.: A Low-Randomness Second-Order Masked AES
6. Dhooghe et al.: Second-Order Low-Randomness $d + 1$ Hardware Sharing of the AES
7. Beyne et al.: Cryptanalysis of Efficient Masked Ciphers: Applications to Low Latency
8. Shahmirzadi et al.: Low-Latency and Low-Randomness Second-Order Masked Cubic Functions

Recap

1999-2006

Masking on Hardware

2007-2015

Higher-Order Masking

2015-2023

Composable Security

2020-2023

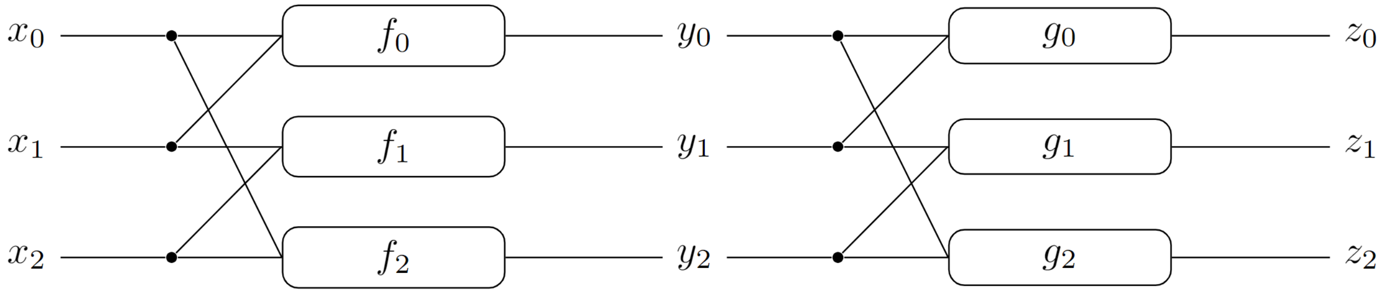
Randomness Complexity

Theory vs. Practice

- There is a difference between side-channel on paper and in practice
  - In theory, side-channel is too strong
    - Region-probing security and horizontal attacks are important on paper but might not lead to attacks in practice
    - In the robust probing model, every glitch is possible
  - In theory, side-channel is too weak
    - A two-share first-order masking is less secure than a three-share first-order masking
    - A probing secure masking can leak in practice
- There are a lot of practical techniques not properly studied yet
  - Noise makers
  - Dual rail methods
  - Non-crypto RNG's

The Probing Model

- Made in 2003 by Ishai et al. to capture probing attacks[1]
- The adversary gets to see a threshold number of intermediate variables
  - There is no noise involved
  - The number of probes determines the order of the attack
  - Can be extended to capture physical effects such as glitches or transitions[2]

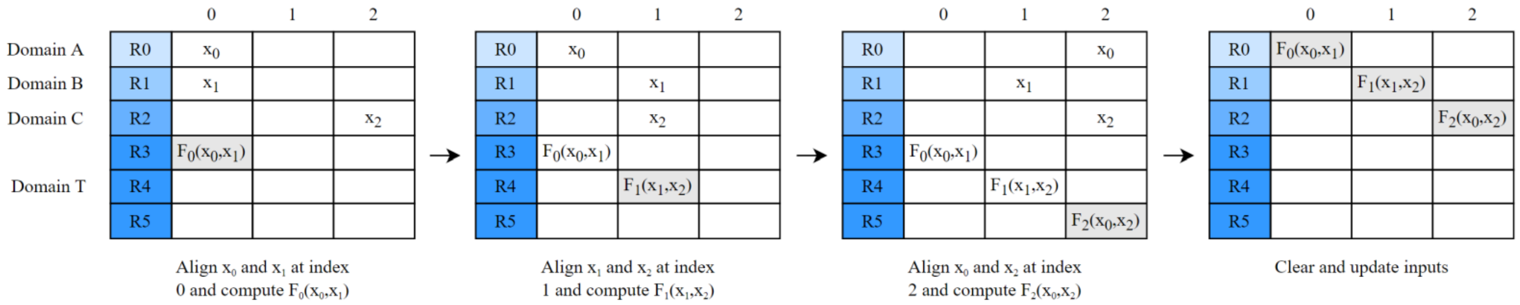


Paper cheat sheet

1. Ishai et al.: Private Circuits: Securing Hardware against Probing Attacks
2. Faust et al.: Composable Masking Schemes in the Presence of Physical Defaults and the Robust Probing Model

Applications of the Probing Model

- The first easy-to-use security model
  - Allows for the making of higher-order masking schemes
  - Allows for verification tools
- The standard model that is often extended to capture leakage effects
  - Example: robust probing[1], software masking[2,3]

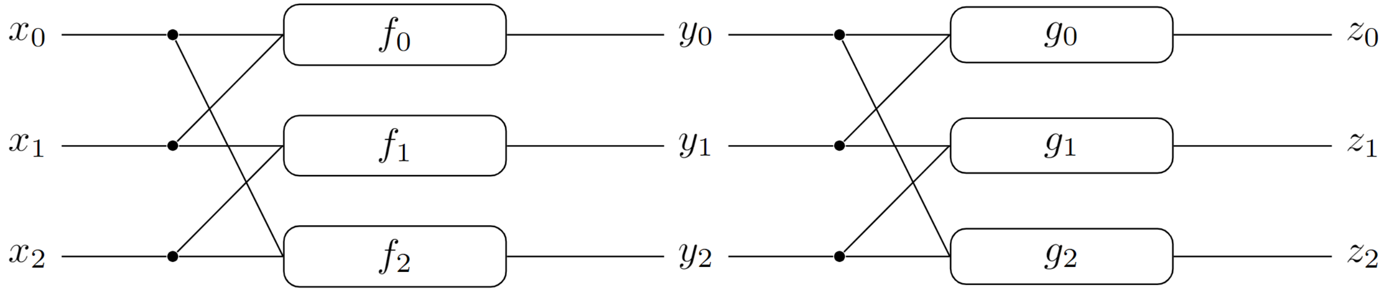


Paper cheat sheet

1. Faust et al.: Composable Masking Schemes in the Presence of Physical Defaults and the Robust Probing Model
2. Zeitschner et al.: PROLEAD_SW - Probing-Based Software Leakage Detection for ARM Binaries
3. Gaspoz et al.: Threshold Implementations in Software: Micro-architectural Leakages in Algorithms

The Random Probing Model

- Originally a virtual model by Duc et al. for a reduction to noisy leakage[1]
- The adversary probes every variable, but the probe can also return nothing
  - When probing x_0, you have an ε-probability to get x_0
  - The location is not random, the location of each probe is known



Paper cheat sheet

1. Duc et al.: Unifying Leakage Models: from Probing Attacks to Noisy Leakage

Applications of the Random Probing Model

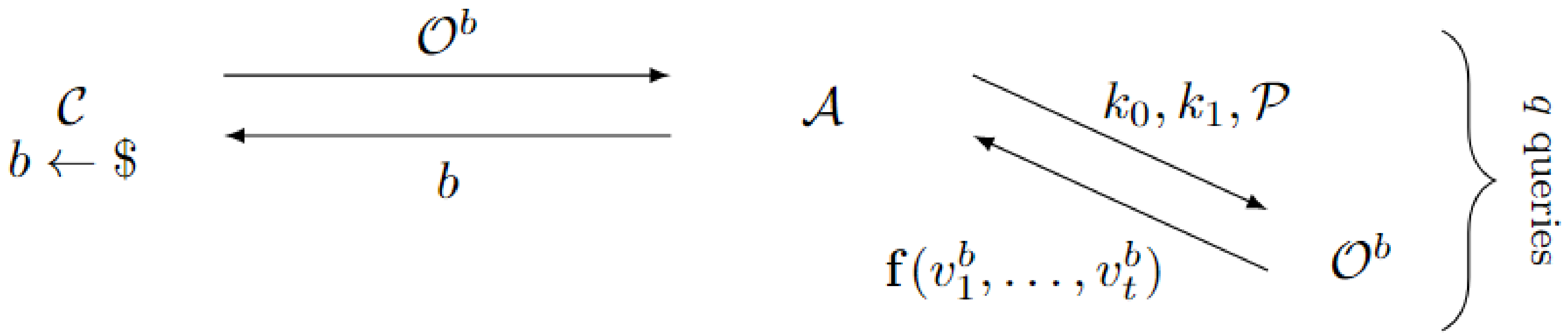
- Often cited because it captures horizontal attacks
  - Unclear whether this is important in hardware
- The random probing model captures linear noise amplifications
  - Scheme 1 with security $4\varepsilon^2 + 6\varepsilon^3$ is first-order secure
  - Scheme 2 with security $2\varepsilon^2 + 12\varepsilon^3$ is also first-order secure, but twice as secure against second-order attacks vs. scheme 1
    - However, it is less secure against a third-order attack
- We can better compare masking methods
  - Using fault countermeasures such as duplication lowers the security

Challenges in the Random Probing Model

- Verify and compare the random probing security of maskings of different S-boxes
- Verify different masking schemes
  - See the effect of randomness reuse
- Verify and compare the security of different masking methods
  - $td + 1$ shares vs. $d + 1$ shares

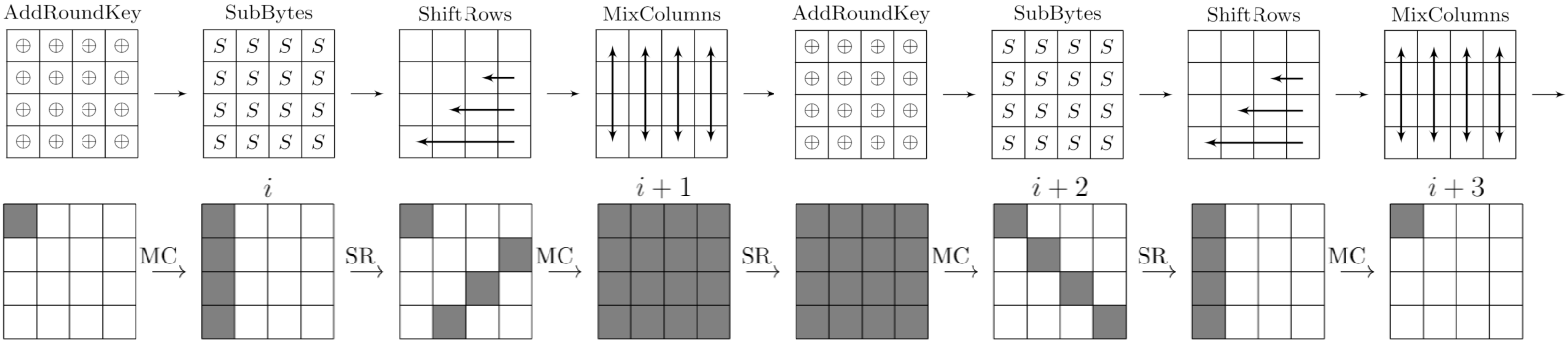
The Bounded-Query Probing Model

- The same as the probing model
  - The adversary only gets a limited number of queries (traces)
  - The adversary still has unlimited computational power and memory
  - (You can exchange the probing model by a random probing model or other)



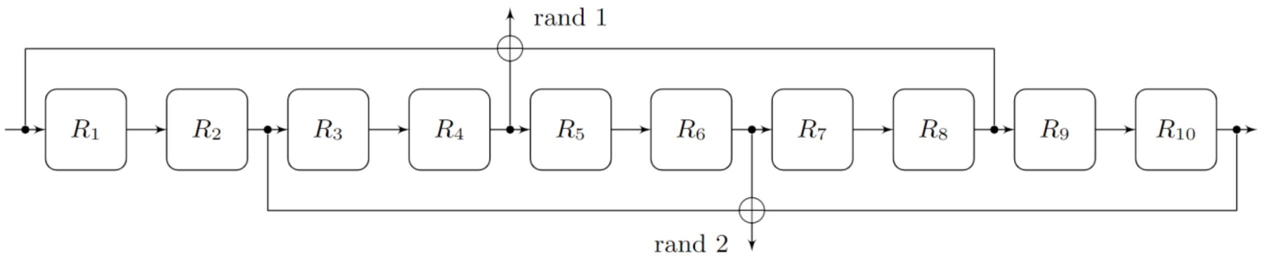
Applications of the Bounded Query Model

- We can make use of cryptanalytic properties of maskings
  - We can reduce randomness of maskings



Challenges in the Bounded Query Model

- The design of maskings with cryptanalytic properties
- We can re-use randomness between cipher calls
  - We can investigate modes of operations where randomness is re-used
- We can investigate the security of maskings including the random number generator
  - Allowing non-cryptographic RNGs



The Bounded Computational Probing Model

- The same as the probing model
  - The adversary has a limited number of queries
  - The adversary has limited computational power and memory
- Does not allow for security proofs
  - Instead, we argue against typical attacks such as DPA against a single S-box

Challenges in the Computational Model

- Provide better bounds compared to the bounded query model
  - Offset the importance of leakage in later rounds
- We can investigate the effect of re-keying primitives (e.g. Zynq UltraScale+[1])
  - We can compare the security of masking with the security of re-keying
- We can include extra diffusion in maskings to thwart key-retrieval attacks
- Etc.…

Paper cheat sheet

1. Hettwer et al.: Side-Channel Analysis of the Xilinx Zynq UltraScale+ Encryption Engine

Conclusions

- A lot of challenging open problems
  - In the random probing model
  - In the bounded query model
- Some consensus about a computational model
- What about those under-studied practical techniques?
  - Noise makers
  - Dual rail methods

Thank you!