Riding the Waves Towards Generic Single-Cycle Masking in Hardware

CHES 2022

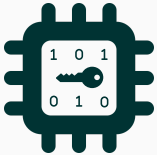
Rishub Nagpal, Barbara Gigerl, Robert Primas and Stefan Mangard

September 21, 2022

IAIK – Graz University of Technology

Motivation

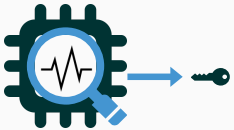
Power analysis attacks pose a threat to real-world crypto implementations.



Motivation

Power analysis attacks pose a threat to real-world crypto implementations.

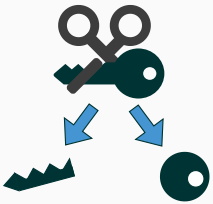
- Masking is a countermeasure where secret data is split into **shares** and processed separately*



Motivation

Power analysis attacks pose a threat to real-world crypto implementations.

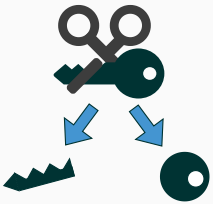
- Masking is a countermeasure where secret data is split into **shares** and processed separately*
- A masked implementation is considered d^{th}-order secure if an attacker needs (at least) $d + 1$ probes to recover secrets.



Motivation

Power analysis attacks pose a threat to real-world crypto implementations.

- Masking is a countermeasure where secret data is split into **shares** and processed separately*
- A masked implementation is considered d^{th}-order secure if an attacker needs (at least) $d + 1$ probes to recover secrets.
- Glitches from combinatorial circuits can reveal cryptographic secrets.



Motivation cont.

However, masking is expensive!

Motivation cont.

However, masking is expensive!

- $ Requires more circuit area, an RNG, and more clock cycles to compute.

Motivation cont.

However, masking is expensive!

- $ Requires more circuit area, an RNG, and more clock cycles to compute.
- For performance-critical applications, such as memory encryption, the extra computation time is a non-starter.

However, masking is expensive!

- $ Requires more circuit area, an RNG, and more clock cycles to compute.
- For performance-critical applications, such as memory encryption, the extra computation time is a non-starter.

Low Latency Masking

Can we implement masked circuits which compute (securely) in a single clock cycle?

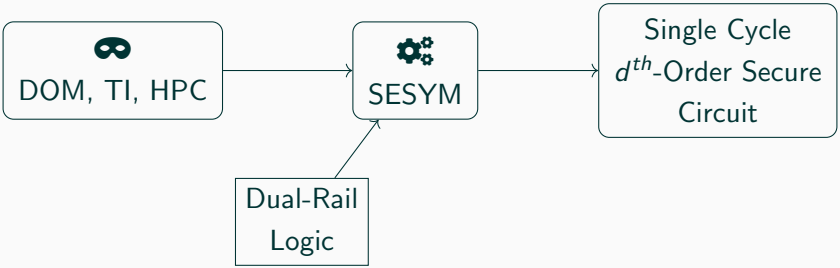
Self-Synchronized Masking

Self-Synchronized Masking (SESYM)

SESYM is a design technique applied over **any** masking scheme to achieve single-cycle d^{th}-order probing security **without** requiring additional randomness.

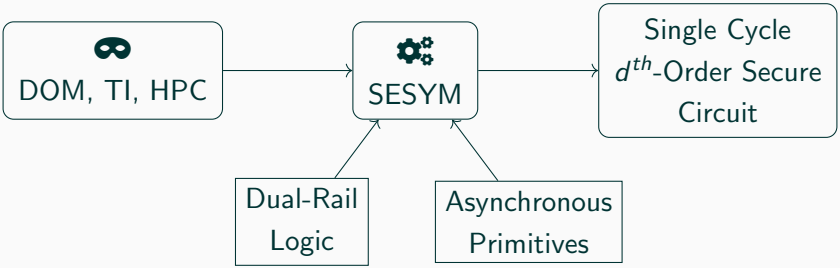
Self-Synchronized Masking (SESYM)

SESYM is a design technique applied over **any** masking scheme to achieve single-cycle d^{th}-order probing security **without** requiring additional randomness.



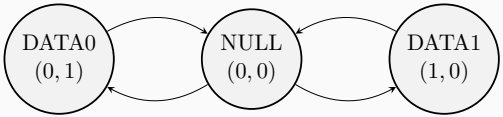
Self-Synchronized Masking (SESYM)

SESYM is a design technique applied over **any** masking scheme to achieve single-cycle d^{th}-order probing security **without** requiring additional randomness.

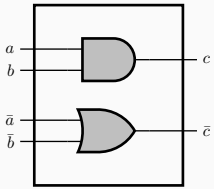


Dual-Rail Logic

- Use two wires to encode a bit
- Dual-rail logic is evaluated in two steps:
  1. Precharge - Drive all wires from DATA to NULL
  2. Evaluate - Compute NULL to DATA



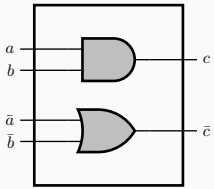
Wave Dynamic Differential Logic (**WDDL**) is a dual-rail logic style based on standard cells.



WDDL AND gate - $\boxed{\odot}$

Wave Dynamic Differential Logic (**WDDL**) is a dual-rail logic style based on standard cells.
Features:

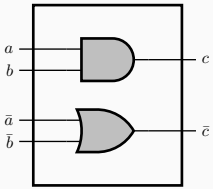
- WDDL gates do not compute intermediate results.
  Ex. NULL $\boxed{\oplus}$ DATA = NULL



WDDL AND gate - $\boxed{\odot}$

Wave Dynamic Differential Logic (**WDDL**) is a dual-rail logic style based on standard cells.
Features:

- WDDL gates do not compute intermediate results.
  Ex. NULL $\boxed{\oplus}$ DATA = NULL
- WDDL gates are positive and monotonic.

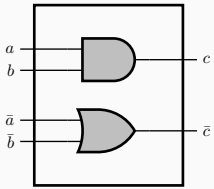


WDDL AND gate - $\boxed{\odot}$

Wave Dynamic Differential Logic (**WDDL**) is a dual-rail logic style based on standard cells.
Features:

- WDDL gates do not compute intermediate results.
  Ex. NULL $\boxed{\oplus}$ DATA = NULL
- WDDL gates are positive and monotonic.

→ WDDL gates do not glitch [TV04].

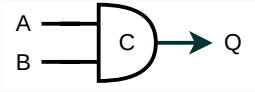


WDDL AND gate - $\boxed{\odot}$

Asynchronous Primitive – Muller C-element [Mul56]

- The C-element is a logic gate which can determine if a set of signals are synchronized.

- The C-element is a logic gate which can determine if a set of signals are synchronized.



(a) Symbol

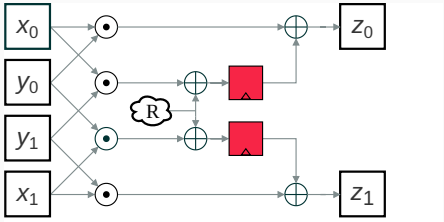
A	B	Q
0	0	0
0	1	-
1	0	-
1	1	1

(b) Truth Table

Designing with SESYM

Applying SESYM on a 1^{st}-order DOM AND gate

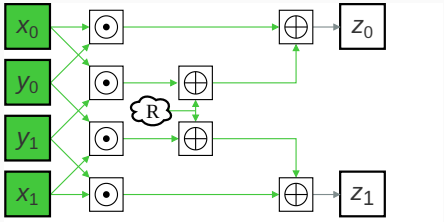
The 1^{st}-order DOM AND gate requires two clock cycles to compute due to the registers. Can we remove them?



Applying SESYM on a 1^{st}-order DOM AND gate

The 1^{st}-order DOM AND gate requires two clock cycles to compute due to the registers. Can we remove them?

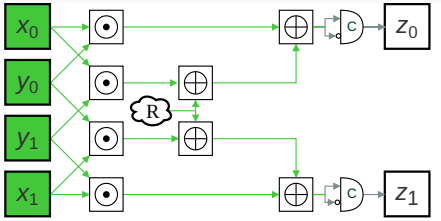
1. Convert inputs and gates to dual-rail.



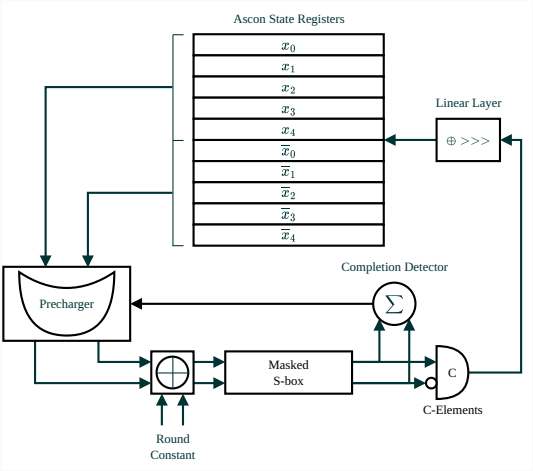
Applying SESYM on a 1^{st}-order DOM AND gate

The 1^{st}-order DOM AND gate requires two clock cycles to compute due to the registers. Can we remove them?

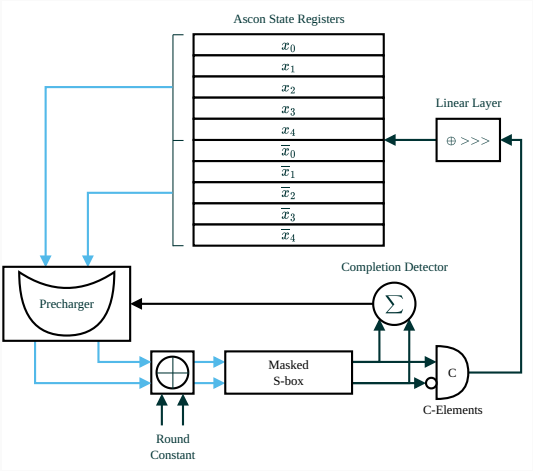
1. Convert inputs and gates to dual-rail.
2. Insert C-elements to latch the computed result and convert back to single-rail.



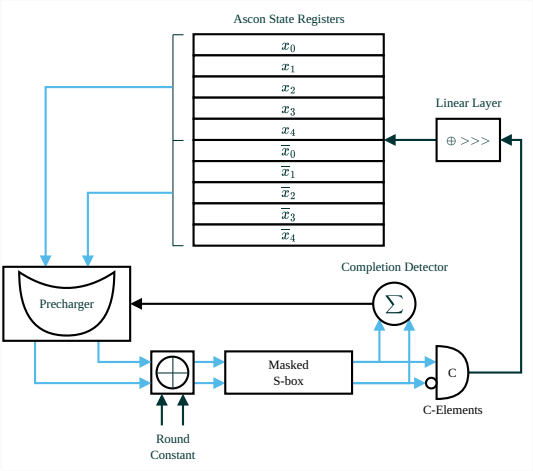
Applying SESYM to the Ascon Permutation



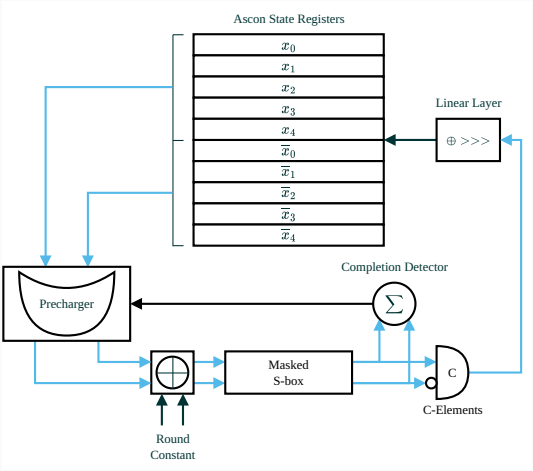
Applying SESYM to the Ascon Permutation



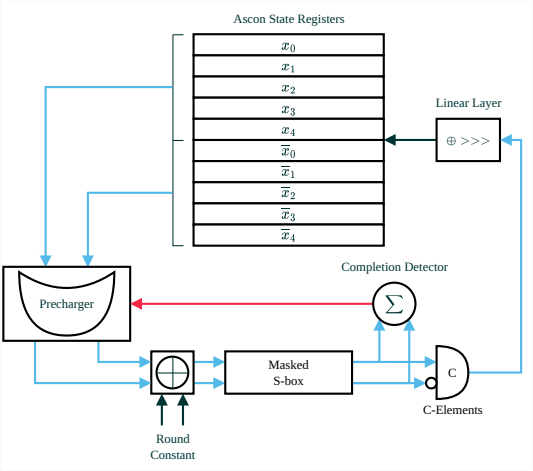
Applying SESYM to the Ascon Permutation



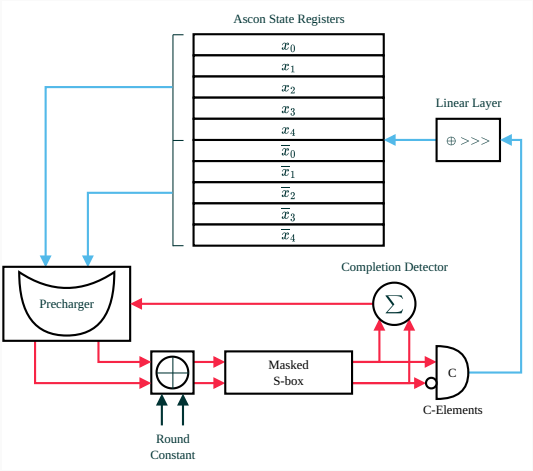
Applying SESYM to the Ascon Permutation



Applying SESYM to the Ascon Permutation



Applying SESYM to the Ascon Permutation



Implementations

Ascon Implementation Results

Protection Order	Area [kGE]	Cycle/round [cycle]	Randomness [bits/cycle]	Max Clock Freq. MHz
This Work, UMC65nm				
1	50.40	1	320	408.3
2	102.39	1	960	377.1
3	172.05	1	1 920	358.4
4	257.13	1	3 200	334.2
5	357.65	1	4 800	312.9
GLM, UMC90nm [GIB18]				
1	42.59	1	2 048	260.0
2	90.78	1	4 608	-
3	153.76	1	8 192	-
4	238.15	1	12 800	-
5	339.67	1	18 432	-

Ascon Implementation Results

Protection Order	Area [kGE]	Cycle/round [cycle]	Randomness [bits/cycle]	Max Clock Freq. MHz
This Work, UMC65nm				
1	50.40	1	320	408.3
2	102.39	1	960	377.1
3	172.05	1	1 920	358.4
4	257.13	1	3 200	334.2
5	357.65	1	4 800	312.9
GLM, UMC90nm [GIB18]				
1	42.59	1	2 048	260.0
2	90.78	1	4 608	-
3	153.76	1	8 192	-
4	238.15	1	12 800	-
5	339.67	1	18 432	-

Implementation	Method	Area [kGE]	Latency [cycles]	Randomness [bits/cycle]
[Sas+20]	LMDPL	3.48	1	36
This work	**SESYM-BP**	**3.98**	**1**	**34**
This work	**SESYM-Canright**	**7.59**	**1**	**18**
[GIB18]	GLM	60.73	1	2 048

Implementation	Method	Area [kGE]	Latency [cycles]	Randomness [bits/cycle]
[Sas+20]	LMDPL	3.48	1	36
This work	**SESYM-BP**	**3.98**	**1**	**34**
This work	**SESYM-Canright**	**7.59**	**1**	**18**
[GIB18]	GLM	60.73	1	2 048

Implementation	Method	Area [kGE]	Latency [cycles]	Randomness [bits/cycle]
This work	**SESYM-BP**	**9.34**	**1**	**102**
This work	**SESYM-Canright**	**14.78**	**1**	**51**
[GIB18]	GLM	57.11	2	4 446
[Cnu+16]	$(d+1)$-share TI	3.66	6	54
[GMK17]	DOM	4.50	8	54

Implementation	Method	Area [kGE]	Latency [cycles]	Randomness [bits/cycle]
This work	**SESYM-BP**	**9.34**	**1**	**102**
This work	**SESYM-Canright**	**14.78**	**1**	**51**
[GIB18]	GLM	57.11	2	4 446
[Cnu+16]	$(d+1)$-share TI	3.66	6	54
[GMK17]	DOM	4.50	8	54

AES-128 Implementations

Impl.	Protection Order	Area [kGE]	Cycle/round [cycle]	Randomness [bits/cycle]	Max Clock Freq. [MHz]
This work	**1**	**104.86**	**1**	**680**	**192.3**
This work	**2**	**203.90**	**1**	**2040**	**169.2**
[Sas+20]	1	157.50	1	976	400

Security Evaluations

Coco [Gig+21] formally verifies the security of a circuit against power analysis attacks while taking glitches and transitions into account.

Coco [Gig+21] formally verifies the security of a circuit against power analysis attacks while taking glitches and transitions into account.

➔ We modeled the WDDL gates based on the glitch-free assumption given in [TV04].

Coco [Gig+21] formally verifies the security of a circuit against power analysis attacks while taking glitches and transitions into account.

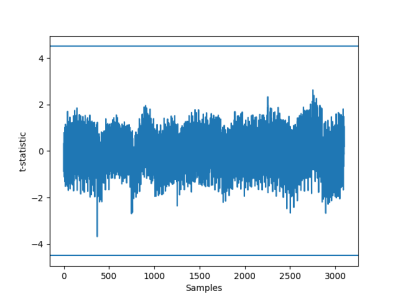
➔ We modeled the WDDL gates based on the glitch-free assumption given in [TV04].

Coco successfully verified:

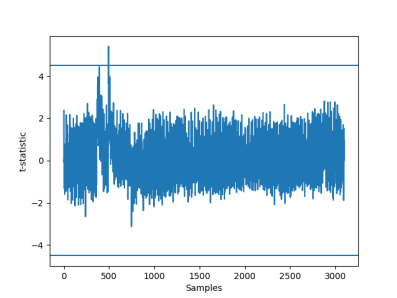
- ✔ 1^{st}-order & 2^{nd}-order Ascon S-box.
- ✔ 1^{st}-order & 2^{nd}-order AES-BP S-box.

We implemented our designs onto a CW305 (Artix-7 FPGA) and performed physical side-channel evaluations.

1^{st}-order ASCON Impl. – 10 Million traces.

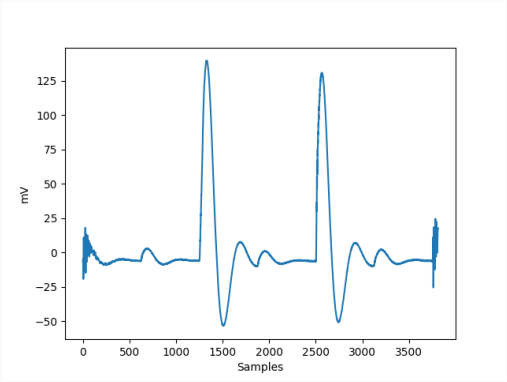


1^{st}-order t-test

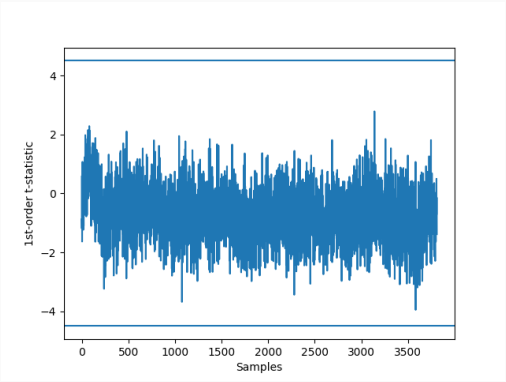


2^{nd}-order t-test

2^{nd}-order AES-Canright – 100 Million traces i

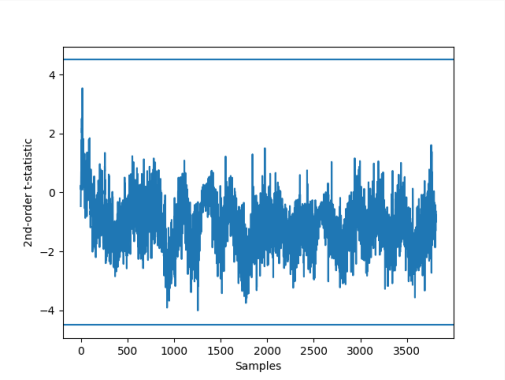


Average Power trace.

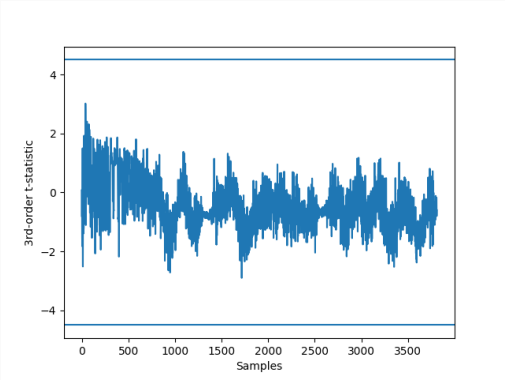


1^{st} order t-test

2^{nd}-order AES-Canright – 100 Million traces ii

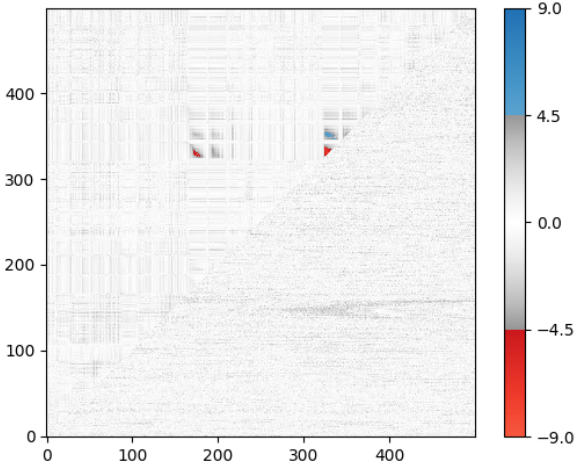


2^{nd} order t-test



3^{rd} order t-test

2^{nd}-order AES-Canright – Bivariate, 10 Million traces



Main Contributions

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.
  - → SESYM combines ordinary masking with asynchronous circuit design (Dual-Rail + C-elements). The overall design is still mostly synchronous.

Main Contributions

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.
  - → SESYM combines ordinary masking with asynchronous circuit design (Dual-Rail + C-elements). The overall design is still mostly synchronous.
  - → No balancing requirements for the dual-rail encoding!

Main Contributions

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.
  - → SESYM combines ordinary masking with asynchronous circuit design (Dual-Rail + C-elements). The overall design is still mostly synchronous.
  - → No balancing requirements for the dual-rail encoding!
- ✔ SESYM enables designers to focus on performance goals, rather than on security requirements. For example, a designer can easily unroll SESYM designs.

Main Contributions

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.
  - → SESYM combines ordinary masking with asynchronous circuit design (Dual-Rail + C-elements). The overall design is still mostly synchronous.
  - → No balancing requirements for the dual-rail encoding!
- ✔ SESYM enables designers to focus on performance goals, rather than on security requirements. For example, a designer can easily unroll SESYM designs.
- ✔ Implemented and verified the first practical higher-order masked single-cycle implementation of AES in the literature.

Main Contributions

- ✔ Introduced **Self-Synchronized Masking** - a design technique for implementing d^{th}-order masked circuits which evaluate in a single clock cycle.
  - → SESYM combines ordinary masking with asynchronous circuit design (Dual-Rail + C-elements). The overall design is still mostly synchronous.
  - → No balancing requirements for the dual-rail encoding!
- ✔ SESYM enables designers to focus on performance goals, rather than on security requirements. For example, a designer can easily unroll SESYM designs.
- ✔ Implemented and verified the first practical higher-order masked single-cycle implementation of AES in the literature.

Thank You!

References

[Cnu+16] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen. Masking AES With d+1 Shares in Hardware. In: Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016. Ed. by B. Bilgin, S. Nikova, and V. Rijmen. ACM, 2016, p. 43. DOI: 10.1145/2996366.2996428. URL: https://doi.org/10.1145/2996366.2996428.

[GIB18] H. Groß, R. Iusupov, and R. Bloem. Generic Low-Latency Masking in Hardware. In: IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018.2 (2018), pp. 1–21. DOI: 10.13154/tches.v2018.i2.1-21. URL: https://doi.org/10.13154/tches.v2018.i2.1-21.

[Gig+21] B. Gigerl, V. Hadzic, R. Primas, S. Mangard, and R. Bloem. Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs. In: 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021. Ed. by M. Bailey and R. Greenstadt. USENIX Association, 2021, pp. 1469–1468. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/gigerl.

[GMK17] H. Groß, S. Mangard, and T. Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In: Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings. Ed. by H. Handschuh. Vol. 10159. Lecture Notes in Computer Science. Springer, 2017, pp. 95–112. DOI: 10.1007/978-3-319-52153-4_6. URL: https://doi.org/10.1007/978-3-319-52153-4_6.

[Mul56] D. E. Muller. A Theory of Asynchronous Circuits. In: Report 75, University of Illinois (1956).

[Sas+20] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson. Low-Latency Hardware Masking with Application to AES. In: IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020.2 (2020), pp. 300–326. DOI: 10.13154/tches.v2020.i2.300-326. URL: https://doi.org/10.13154/tches.v2020.i2.300-326.

[TV04] K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: 2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France. IEEE Computer Society, 2004, pp. 246–251. DOI: 10.1109/DATE.2004.1268856. URL: https://doi.org/10.1109/DATE.2004.1268856.