A RISC-V Instruction Set Extension for Flexible Hardware/Software Protection of Cryptosystems Masked at High Orders

Fabrice Lozachmeur[1] and Arnaud Tisserand[2]

[1]Thales LAS France SAS, Lab-STICC, Université Bretagne Sud, Lorient, France.

[2]CNRS, Lab-STICC, hosted at ENSTA Bretagne, Brest, France.

MWSCAS, August 7, 2023

Table of Contents

Table of Contents

Introduction

Side channel attacks (SCAs) [KJJ99]

Exploit correlations between measured physical values and operations and operands processed in the circuit

Masking countermeasure [Cha+99; GP99]

Randomize all intermediate sensitive variables

Instruction set extensions (ISEs)

New masked instruction to accelerate masking and increase security

Contribution

A RISC-V ISE for flexible hardware/software protection of cryptosystems masked at high orders

Introduction

Side channel attacks (SCAs) [KJJ99]

Exploit correlations between measured physical values and operations and operands processed in the circuit

Masking countermeasure [Cha+99; GP99]

Randomize all intermediate sensitive variables

Instruction set extensions (ISEs)

New masked instruction to accelerate masking and increase security

Contribution

A RISC-V ISE for flexible hardware/software protection of cryptosystems masked at high orders

Introduction

Side channel attacks (SCAs) [KJJ99]

Exploit correlations between measured physical values and operations and operands processed in the circuit

Masking countermeasure [Cha+99; GP99]

Randomize all intermediate sensitive variables

Instruction set extensions (ISEs)

New masked instruction to accelerate masking and increase security

Contribution

A RISC-V ISE for flexible hardware/software protection of cryptosystems masked at high orders

Introduction

Side channel attacks (SCAs) [KJJ99]

Exploit correlations between measured physical values and operations and operands processed in the circuit

Masking countermeasure [Cha+99; GP99]

Randomize all intermediate sensitive variables

Instruction set extensions (ISEs)

New masked instruction to accelerate masking and increase security

Contribution

A RISC-V ISE for flexible hardware/software protection of cryptosystems masked at high orders

Table of Contents

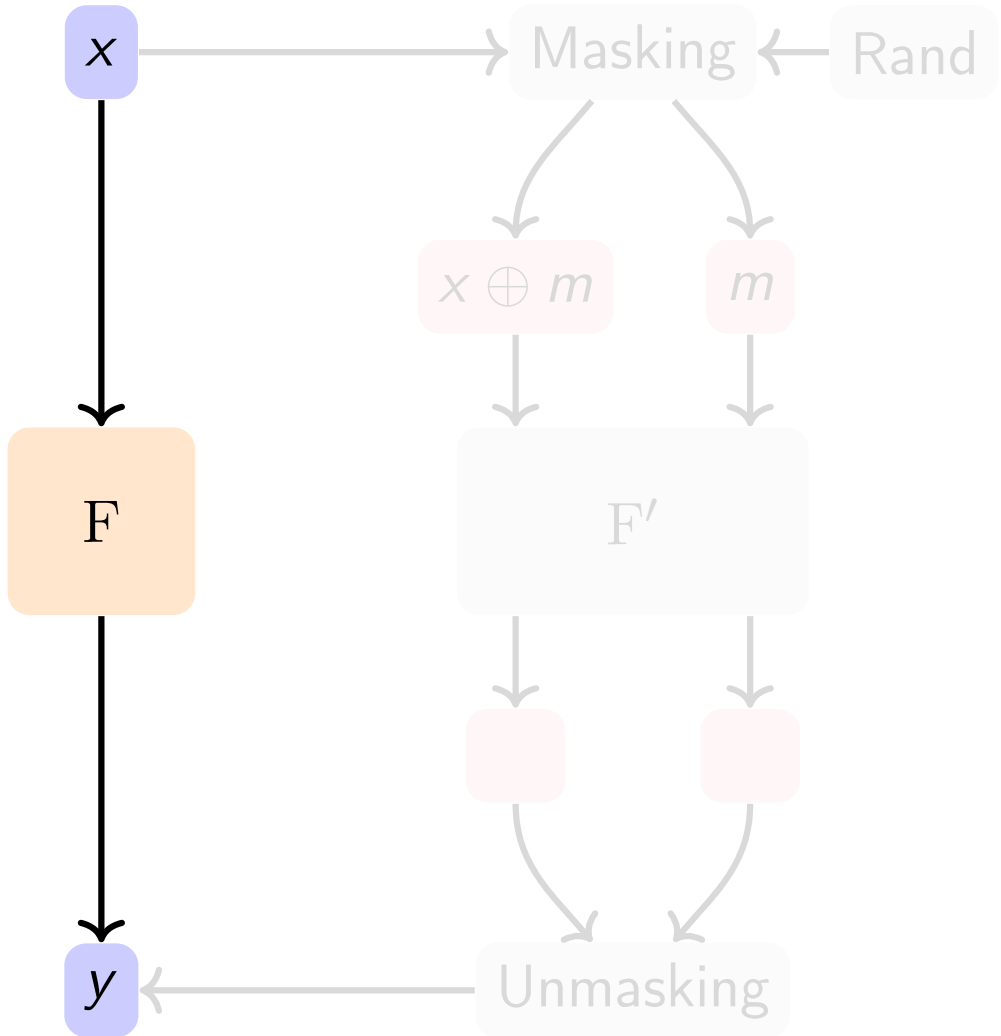
Masking Countermeasure

Masking

- Mask x into $(x \oplus m, m)$ with m a random mask
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



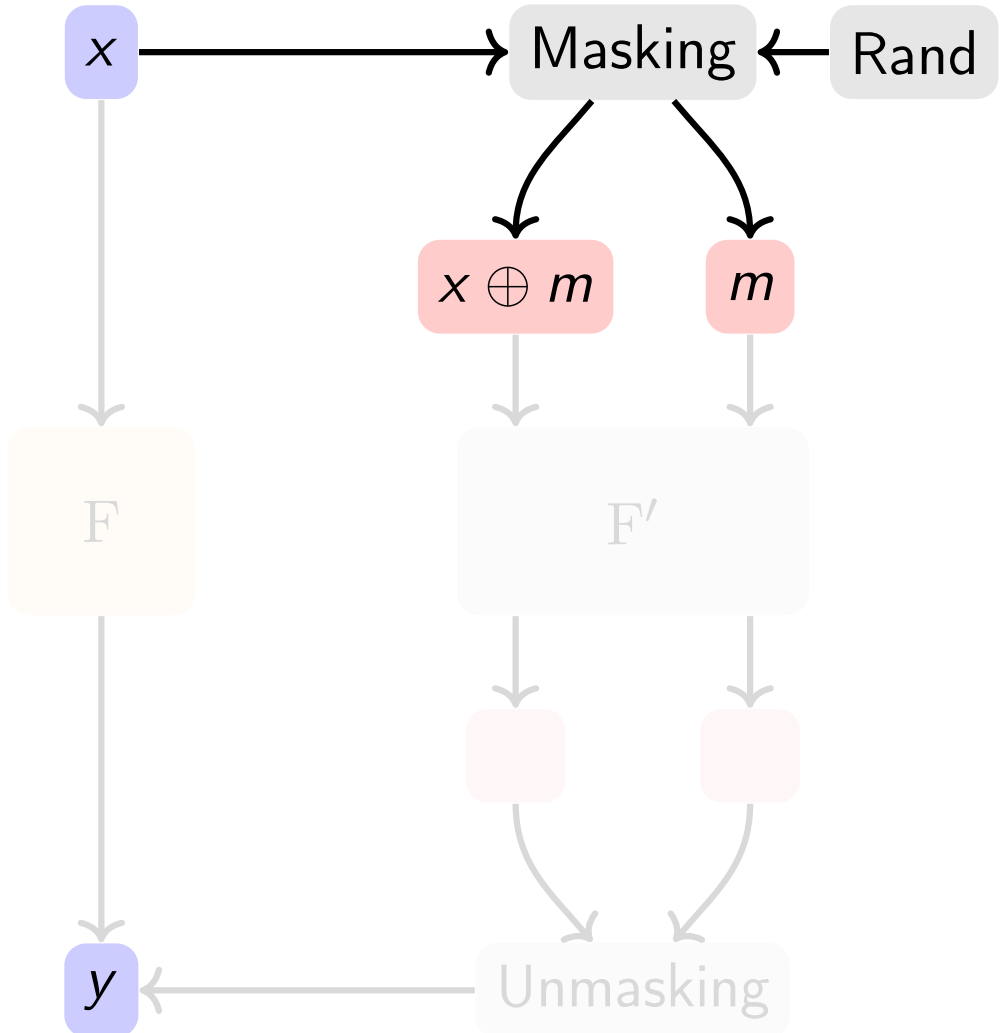
Masking Countermeasure

Masking

- Mask x into $(x \oplus m, m)$ with m a random mask
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



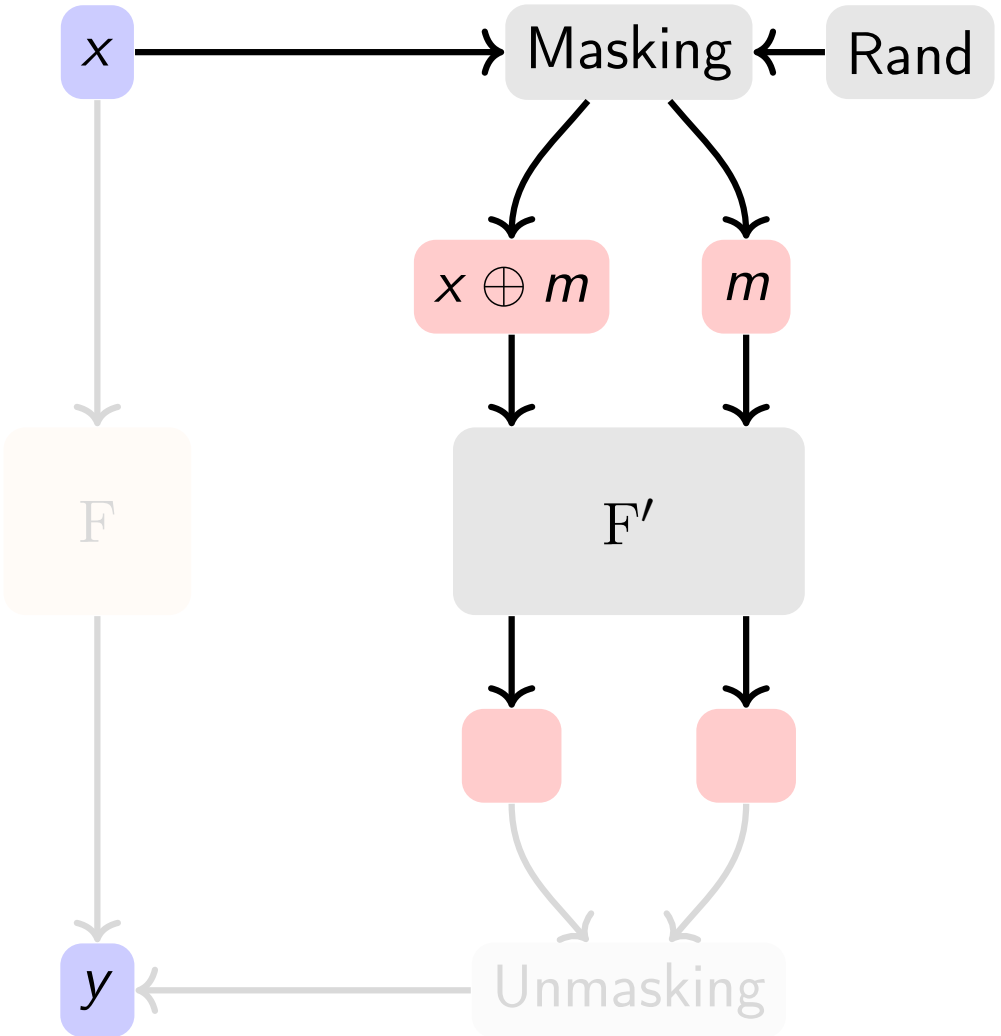
Masking Countermeasure

Masking

- Mask x into $(x \oplus m, m)$ with m a random mask
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



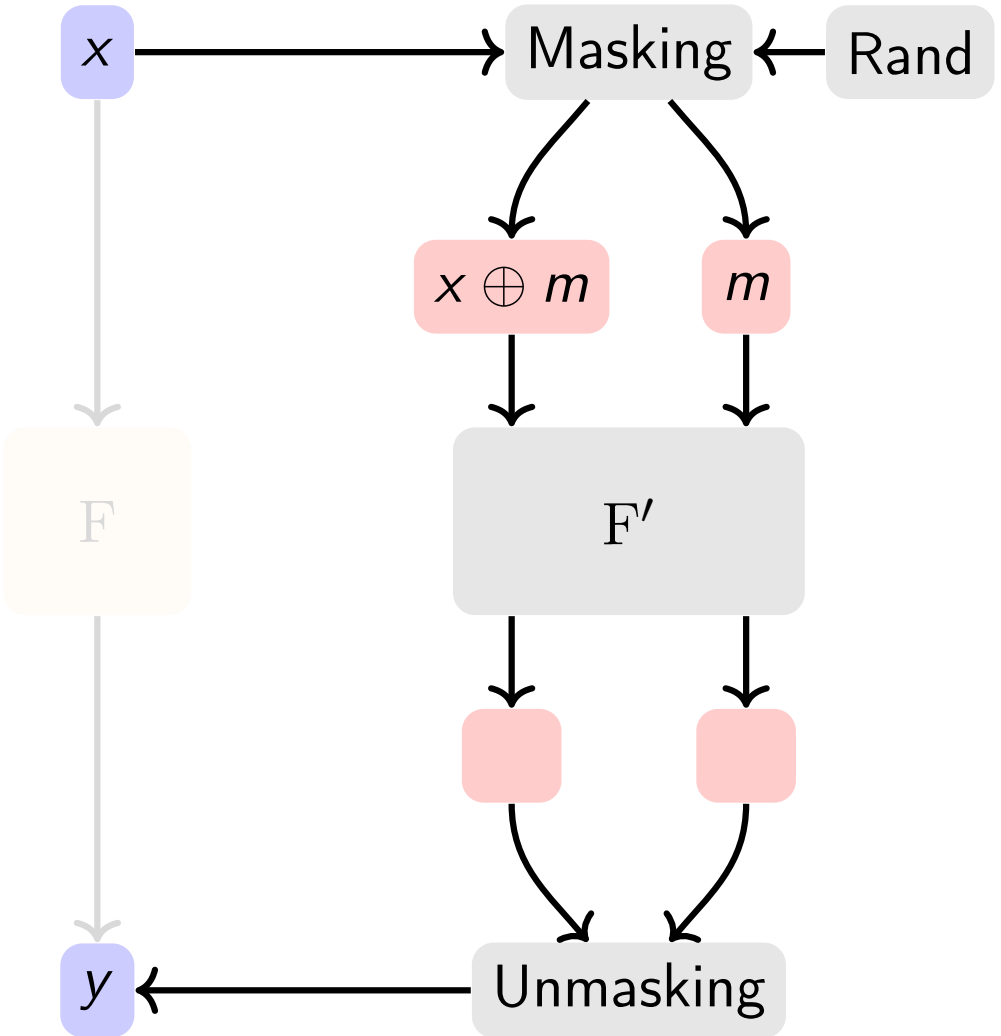
Masking Countermeasure

Masking

- Mask x into $(x \oplus m, m)$ with m a random mask
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



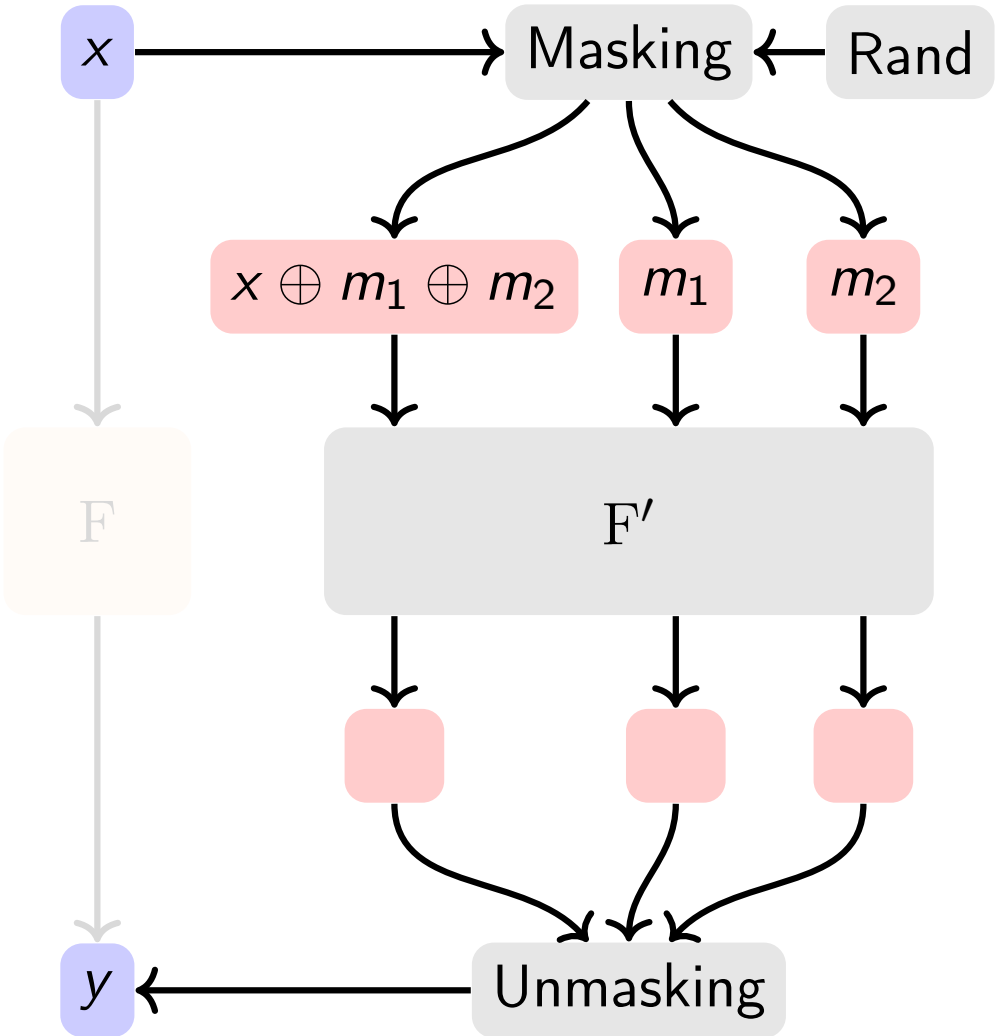
Masking Countermeasure

2-order masking

- Mask x into $(x \oplus m_1 \oplus m_2, m_1, m_2)$ with m_1, m_2 random masks
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



Masking Countermeasure

d-order masking

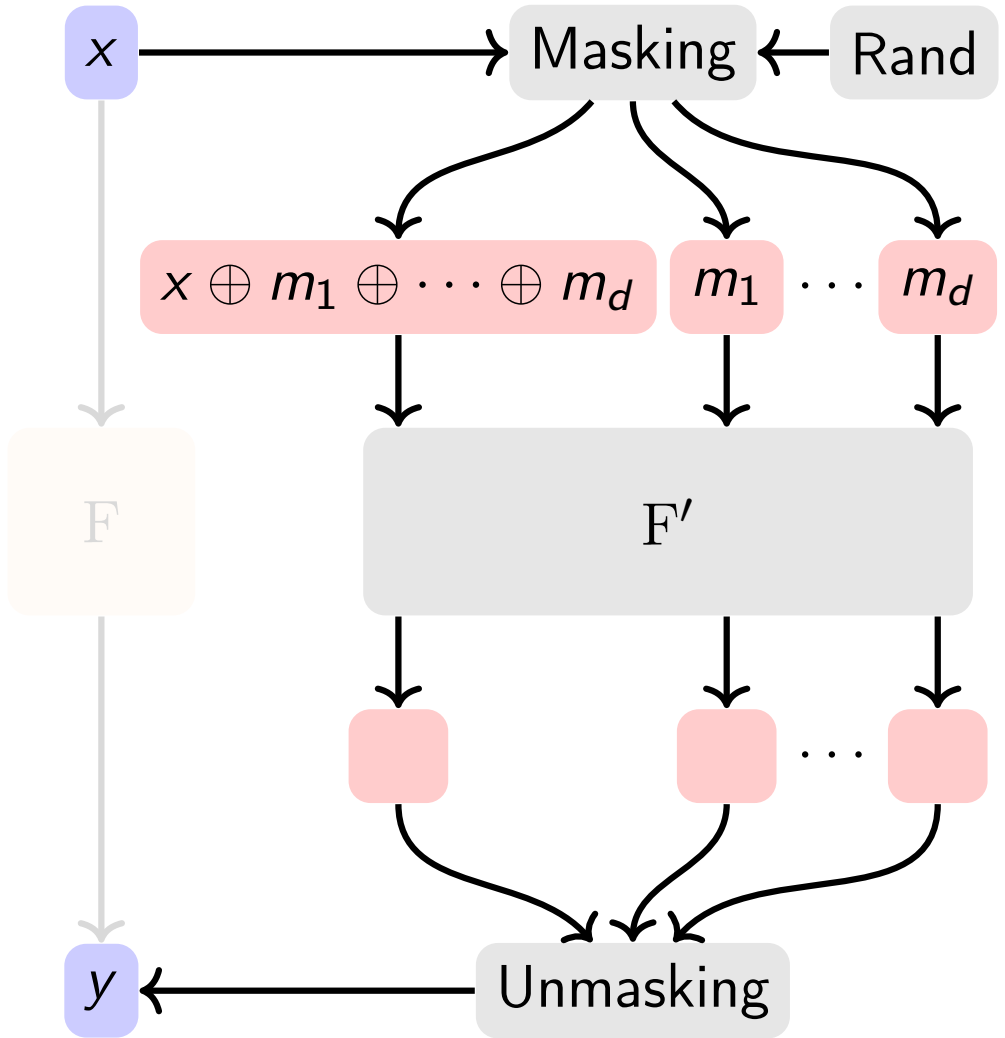
- Mask x into

  $(x \oplus m_1 \oplus \cdots \oplus m_d, m_1, \cdots, m_d)$

  with $m_1, \cdots, m_d$ random masks
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



Masking Countermeasure

d-order masking

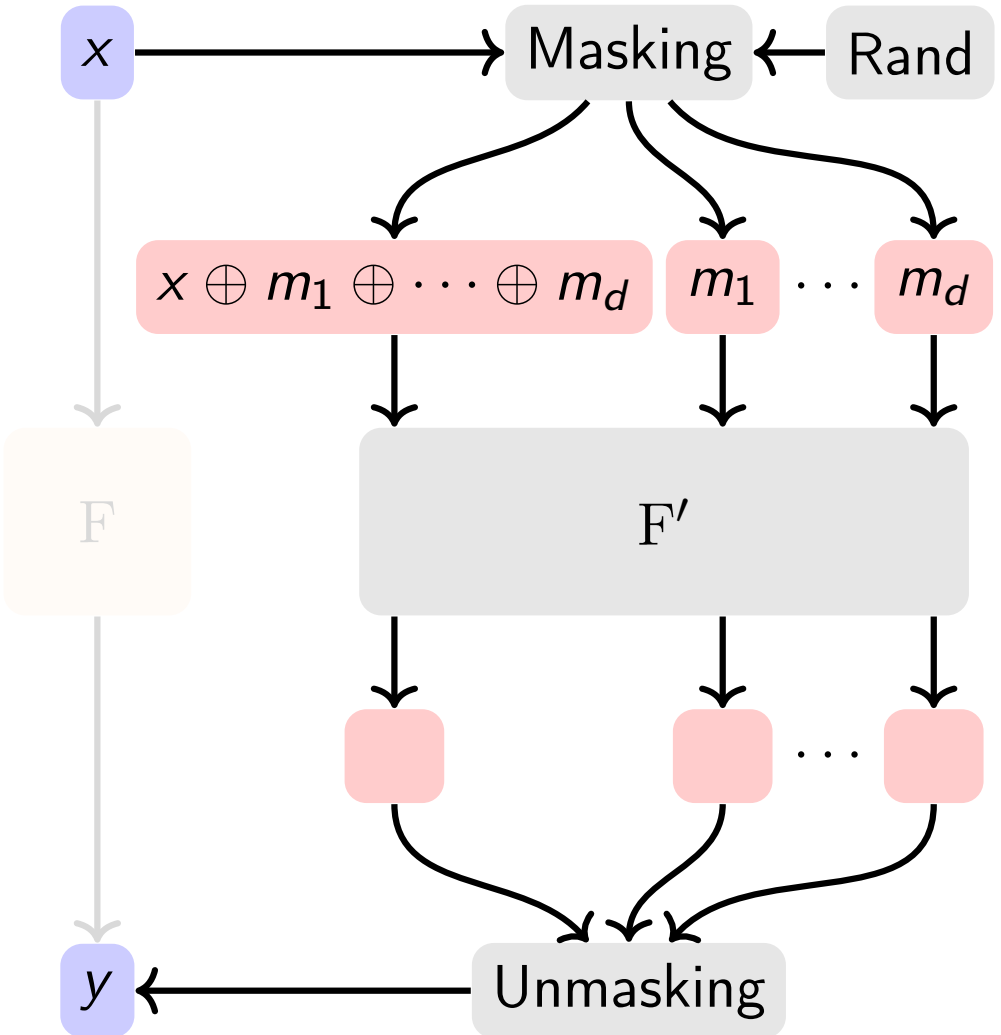
- Mask x into

  $(x \oplus m_1 \oplus \cdots \oplus m_d, m_1, \cdots, m_d)$

  with $m_1, \cdots, m_d$ random masks
- Apply a masked function F'
- Unmask to get y

Software masking

- Computation time increases as $O(d^2)$
- Micro-architectural leakage



Instruction Set Extensions for Masking

Reference	RV	Masking order	Flexbility at design time	Flexbility at run time
[Gro+16]	✓	$\{1, 2, 3, 4\}$	✓	✗
[DGH19]	✓	1	✗	✗
[Gao+21]	✓	1	✗	✗
SKIVA [Kia+21]	✗	$\{1, 3\}$	✗	✓
SME [MP21]	✓	$\{1, 2, 3\}$	✓	✗
Our ISE	✓	$\{1, \cdots, 31\}$	✓	✓

Table of Contents

Our Hardware/Software Masking Solution

A hardware masked ISE

Masked ISE with order d_{H} fixed at synthesis time

Software usage of our hardware masking ISE

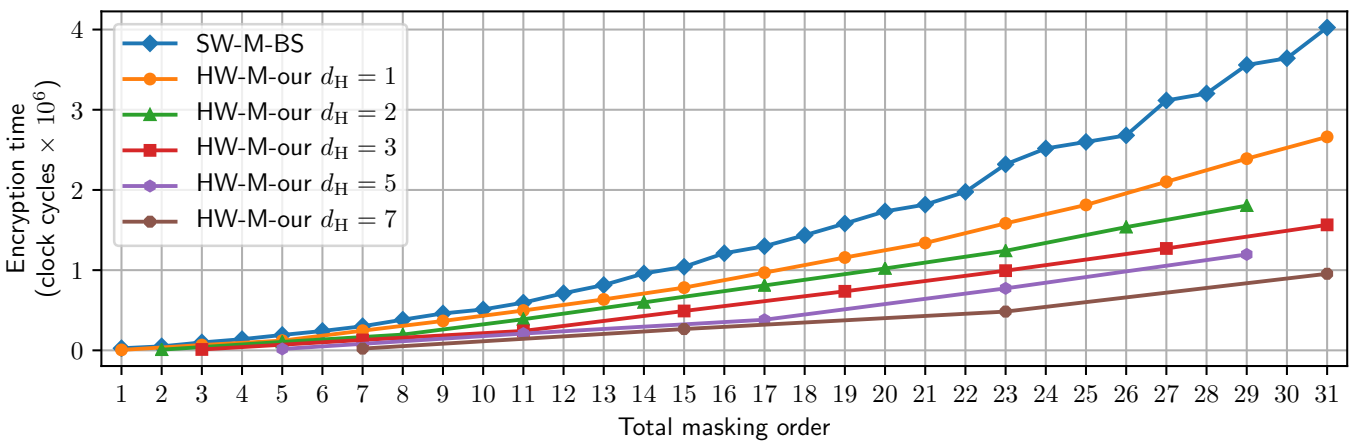
Secure composition over the masked instructions to mask at order:

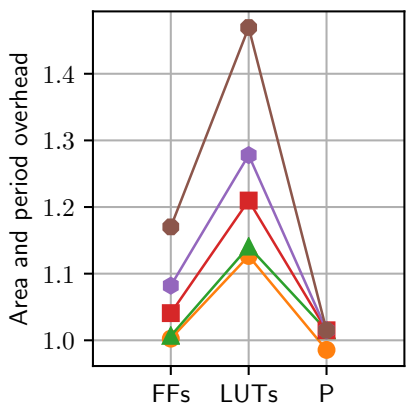
$$d = s(d_{\mathrm{H}} + 1) - 1,$$

where s is a software multiplicative factor fixed at run time

Implementation results

Masked ISE at orders $d_{\mathrm{H}} \in \{1, 2, 3, 5, 7\}$ and various total orders





Our Hardware/Software Masking Solution

A hardware masked ISE

Masked ISE with order d_{H} fixed at synthesis time

Software usage of our hardware masking ISE

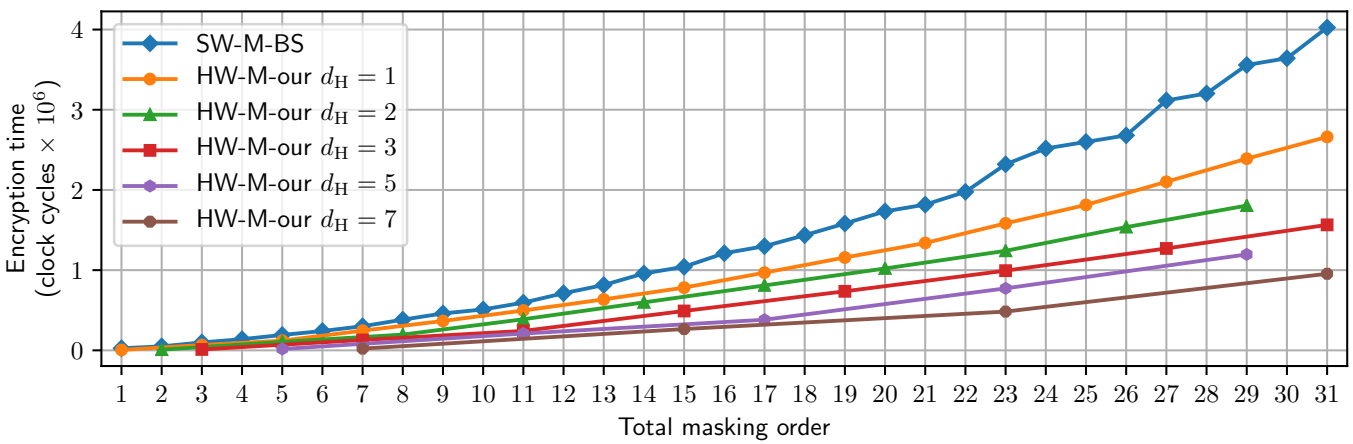
Secure composition over the masked instructions to mask at order:

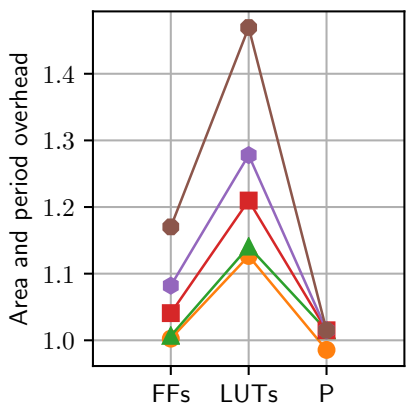
$$d = s(d_{\mathrm{H}} + 1) - 1,$$

where s is a software multiplicative factor fixed at run time

Implementation results

Masked ISE at orders $d_{\mathrm{H}} \in \{1, 2, 3, 5, 7\}$ and various total orders





Our Hardware/Software Masking Solution

A hardware masked ISE

Masked ISE with order d_{H} fixed at synthesis time

Software usage of our hardware masking ISE

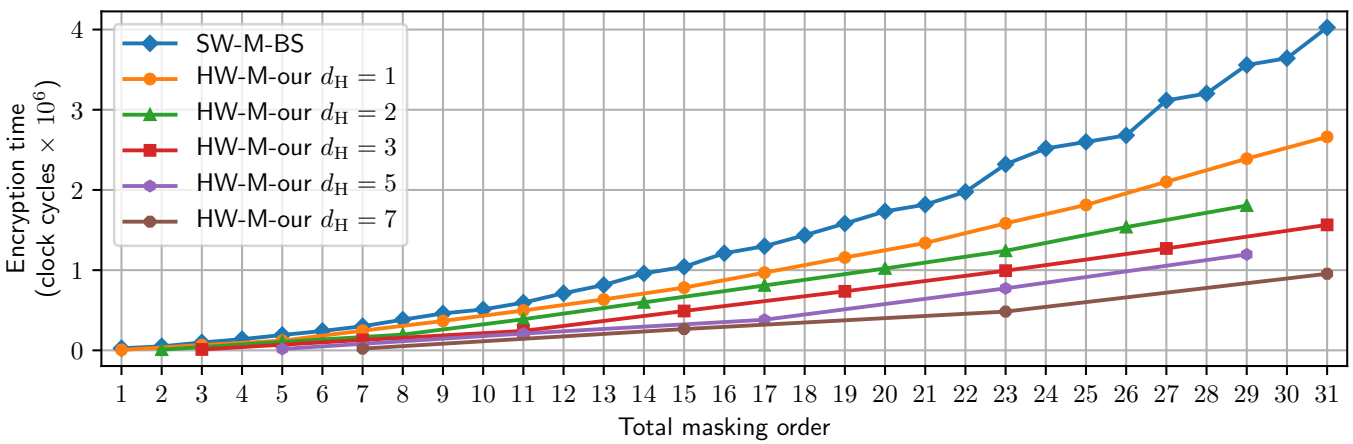
Secure composition over the masked instructions to mask at order:

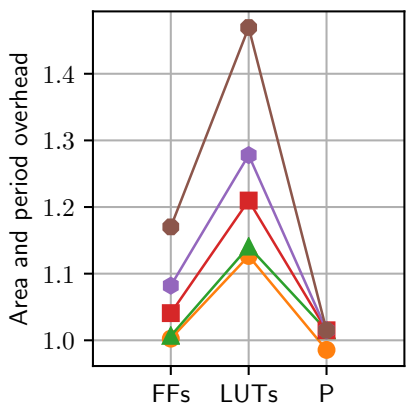
$$d = s(d_{\mathrm{H}} + 1) - 1,$$

where s is a software multiplicative factor fixed at run time

Implementation results

Masked ISE at orders $d_{\mathrm{H}} \in \{1, 2, 3, 5, 7\}$ and various total orders





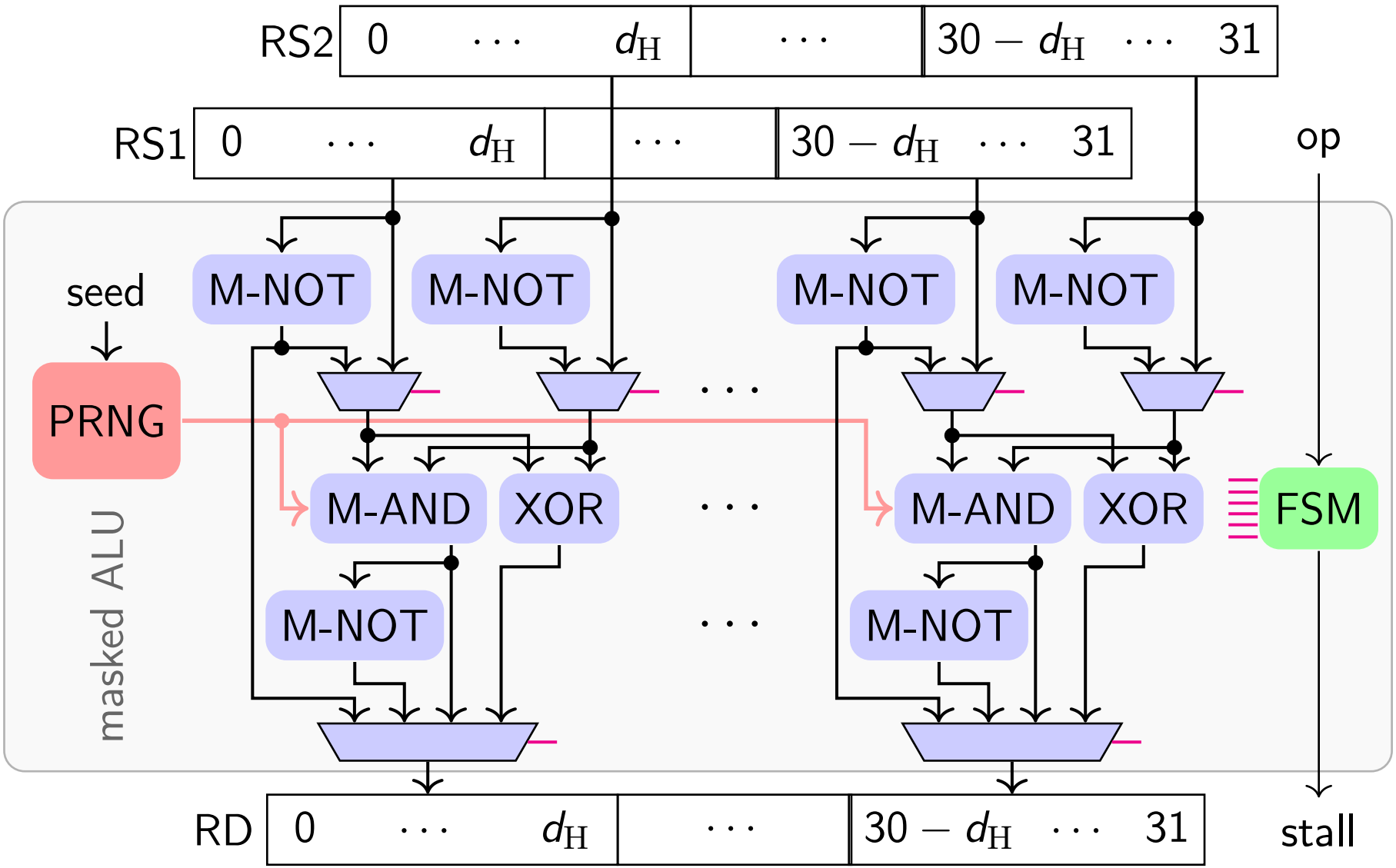
Our masked ISE

Masked Instructions

- Use share slicing representation (see [JS17])
- Verify PINI composability property (see [CS20])
- Easy masking of bit slicing implementations using USUBA (see [MD19])

Instruction	Format	Latency	Random bits
masked AND	`m.and rd, rs1, rs2`	2	$32(d_{\mathrm{H}}-2)$
masked OR	`m.or rd, rs1, rs2`	2	$32(d_{\mathrm{H}}-2)$
masked NOT	`m.not rd, rs1, rs2`	1	0
masked XOR	`m.xor rd, rs1, rs2`	1	0

Unit for Masking



Integration into the CV32E40P Core

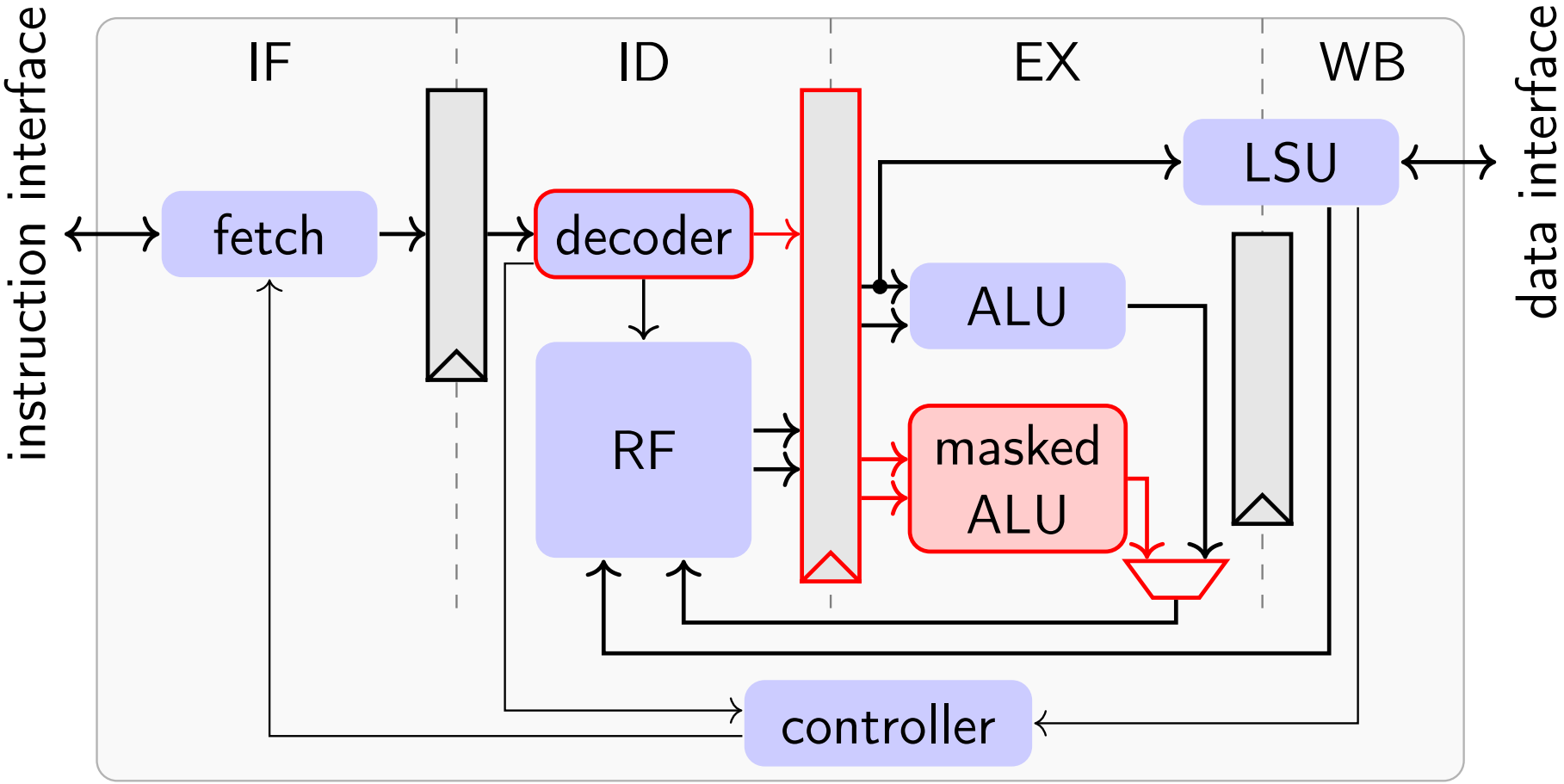
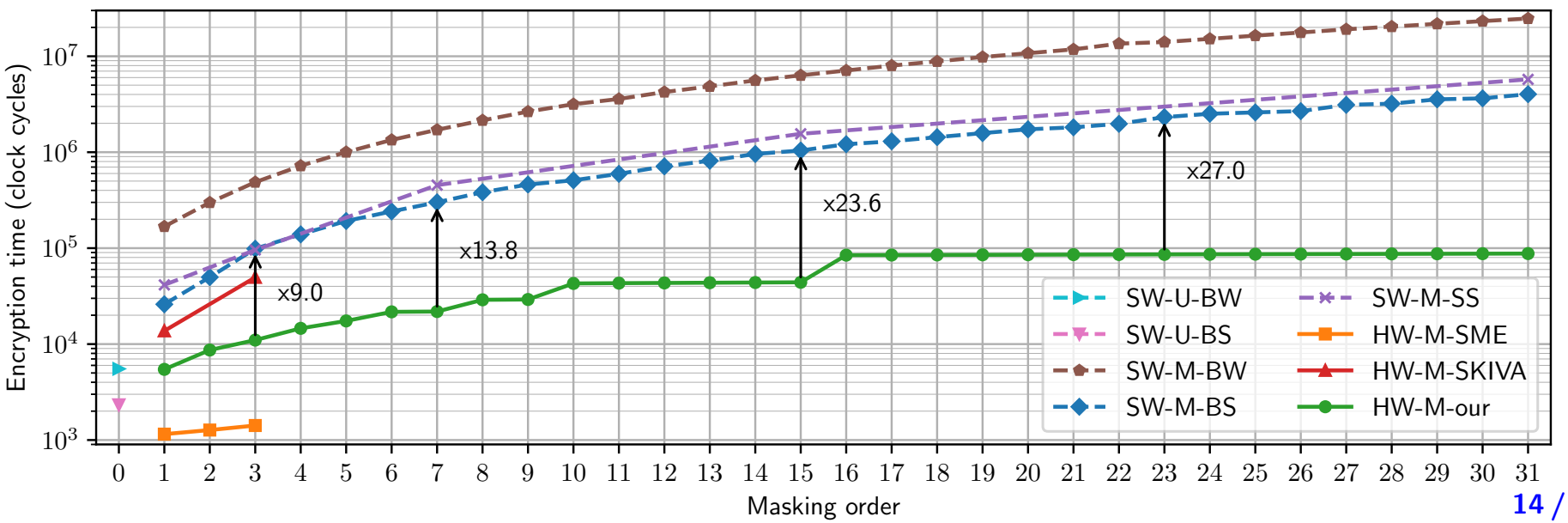


Table of Contents

Comparison to Previous Solutions

Encryption times in **log scale** for one AES block:

- SW-U-BW is unmasked and byte-wise
- SW-U-SW is unmasked and bit-sliced [MD19]
- SW-M-BW is masked and byte-wise [Cor+14]
- SW-M-BS is masked and bit-sliced [Bel+20]
- SW-M-SS is masked and uses share-slicing [JS17]
- HW-M-SKIVA is masked with SKIVA [Kia+21]
- HW-M-SME is masked with SME [MP21]
- HW-M-our is masked with our ISE



Comparison to Previous Solutions

Area/frequency results on a Digilent Arty A7 FPGA board of the CV32E40P with our and the various state-of-the-art ISEs

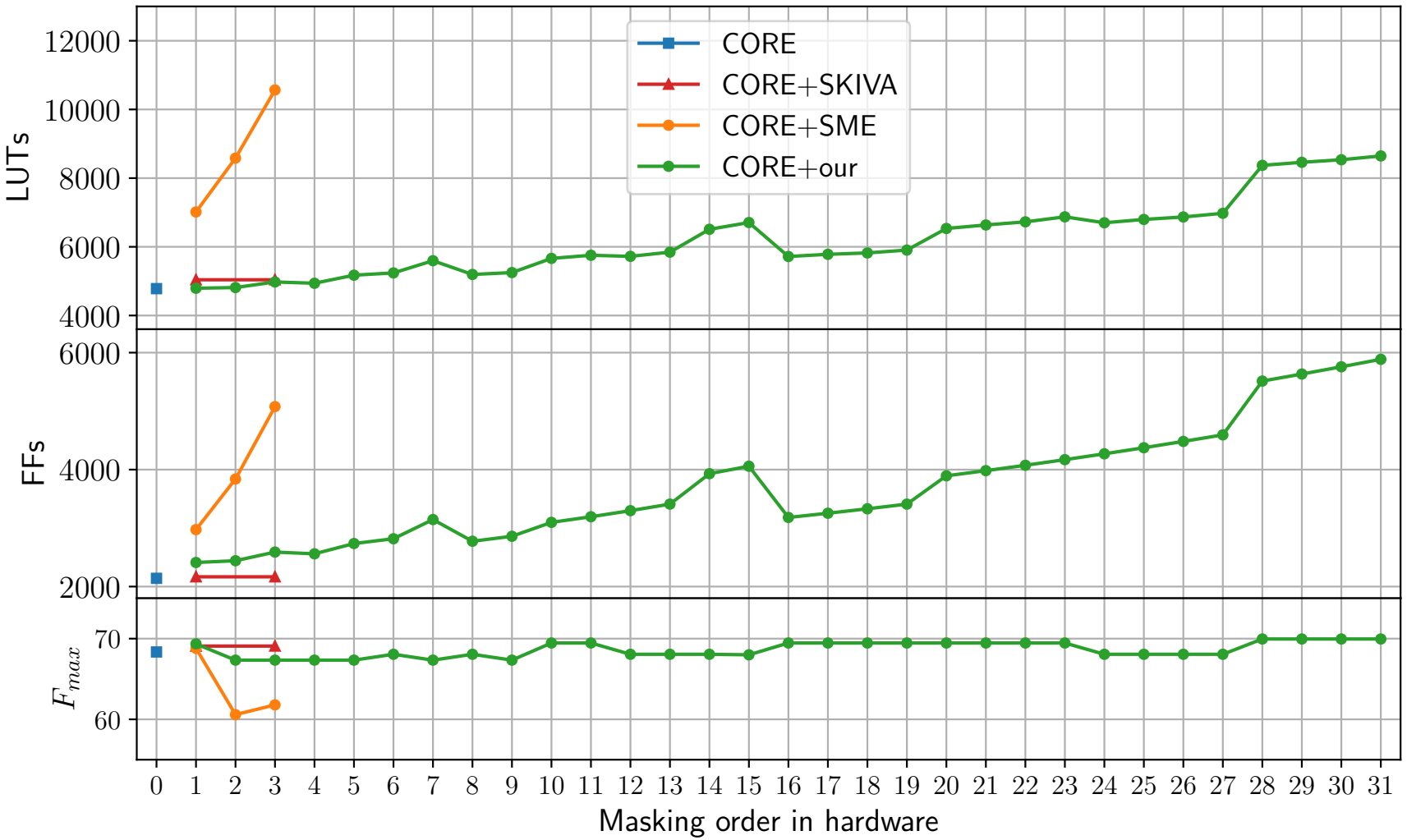


Table of Contents

Conclusion and Future Prospects

Our hardware/software masking solution

- Flexibility at design time and run time
- Speeds up masking with a small silicon cost
- Allow higher order masking
- Apply to various cryptosystems

Future Works

- Security evaluation using physical attacks
- Masked ISE optimized for AES and post-quantum cryptography

Conclusion and Future Prospects

Our hardware/software masking solution

- Flexibility at design time and run time
- Speeds up masking with a small silicon cost
- Allow higher order masking
- Apply to various cryptosystems

Future Works

- Security evaluation using physical attacks
- Masked ISE optimized for AES and post-quantum cryptography

End of Presentation

Thank you for your attention

Do you have any questions?

Bibliography I

[Bel+20] Sonia Belaïd et al. "Tornado: Automatic Generation of Probing-Secure Masked Bitsliced Implementations". In: Proc. Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT). Springer, May 2020, pp. 311–341. DOI: 10.1007/978-3-030-45727-3_11.

[Cha+99] Suresh Chari et al. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: Proc. Annual Cryptology Conference (CRYPTO). Springer, Aug. 1999, pp. 398–412. DOI: 10.1007/3-540-48405-1_26.

[Cor+14] Jean-Sébastien Coron et al. "Higher-Order Side Channel Security and Mask Refreshing". In: Proc. Fast Software Encryption (FSE). Springer, Mar. 2014, pp. 410–424. DOI: 10.1007/978-3-662-43933-3_21.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. "Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference". In: Transactions on Information Forensics and Security (TIFS) (Feb. 2020), pp. 2542–2555. DOI: 10.1109/TIFS.2020.2971153.

[DGH19] Elke De Mulder, Samatha Gummalla, and Michael Hutter. "Protecting RISC-V against Side-Channel Attacks". In: Proc. Design Automation Conference (DAC). ACM, June 2019, pp. 1–4. DOI: 10.1145/3316781.3323485.

[Gao+21] Si Gao et al. "An Instruction Set Extension to Support Software-Based Masking". In: Transactions on CHES (Aug. 2021), pp. 283–325. DOI: 10.46586/tches.v2021.i4.283-325.

[GP99] Louis Goubin and Jacques Patarin. "DES and Differential Power Analysis The Duplication Method". In: Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Springer, Aug. 1999, pp. 158–172. DOI: 10.1007/3-540-48059-5_15.

[GR17] Dahmun Goudarzi and Matthieu Rivain. "How Fast Can Higher-Order Masking Be in Software?" In: Proc. Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT). Springer, Apr. 2017, pp. 567–597. DOI: 10.1007/978-3-319-56620-7_20.

Bibliography II

[Gro+16] Hannes Gross et al. "Concealing Secrets in Embedded Processors Designs". In: Proc. International Conference on Smart Card Research and Advanced Applications (CARDIS). Springer, Nov. 2016, pp. 89–104. DOI: 10.1007/978-3-319-54669-8_6.

[JS17] Anthony Journault and François-Xavier Standaert. "Very High Order Masking: Efficient Implementation and Security Evaluation". In: Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Springer, Sept. 2017, pp. 623–643. DOI: 10.1007/978-3-319-66787-4_30.

[Kia+21] Pantea Kiaei et al. "Custom Instruction Support for Modular Defense Against Side-Channel and Fault Attacks". In: Proc. International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE). Springer, Apr. 2021, pp. 221–253. DOI: 10.1007/978-3-030-68773-1_11.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: Proc. Annual Cryptology Conference (CRYPTO). Springer, Aug. 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.

[MD19] Darius Mercadier and Pierre-Evariste Dagand. "Usuba: High-Throughput and Constant-Time Ciphers, by Construction". In: Proc. Conference on Programming Language Design and Implementation (PLDI). ACM, June 2019, pp. 157–173. DOI: 10.1145/3314221.3314636.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks: Revealing the Secrets of Smart Cards. 1st ed. Springer, 2007. ISBN: 978-0-387-38162-6. DOI: 10.1007/978-0-387-38162-6.

[MP21] Ben Marshall and Dan Page. SME: Scalable Masking Extensions. IACR Cryptology ePrint Archive. Oct. 2021. URL: https://eprint.iacr.org/2021/1416.

[QS01] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards". In: Proc. International Conference on Research in Smart Cards (E-smart). Springer, Sept. 2001, pp. 200–210. DOI: 10.1007/3-540-45418-7_17.

Attack by Side Channel Observation

Available data

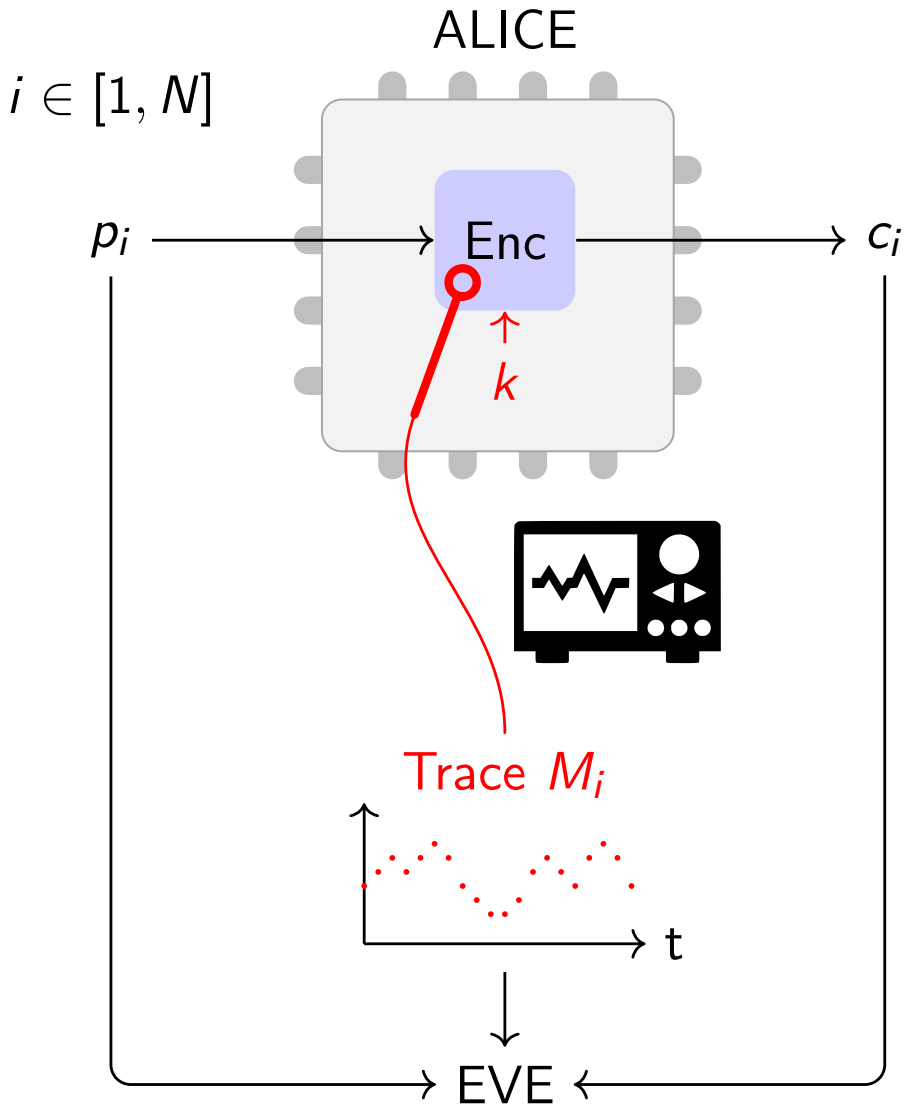
- Plaintexts and/or ciphertexts
- Physical measurements during each encryption

Measured physical quantities

- Power consumption [KJJ99]
- Electromagnetic radiation [QS01]
- ...

Good book

Power Analysis Attacks: Revealing the Secrets of Smart Cards [MOP07].



Bit Slicing

	Bloc 0	Bloc 1	$\cdots$	Bloc 31
R_0	b_0^0	b_0^1	$\cdots$	b_0^{31}
R_1	b_1^0	b_1^1	$\cdots$	b_1^{31}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
R_{127}	b_{127}^0	b_{127}^1	$\cdots$	b_{127}^{31}

Principle

- Transposes k input blocks of l bits into l registers of k bits
- Express algorithms in terms of elementary boolean gates (e.g. AND, XOR, OR, NOT)

Advantages

- High throughput
- Constant time implementation

Masked Bit Slicing

First solution

Shares of one bit are placed into different registers [JS17; Bel+20]

Second solution: share slicing

Shares of one bit are placed into one slice of a physical register [JS17; GR17]

- Avoids intermediate recombinations of shares
- Requires less memory words

Masked Bit Slicing

First solution

Shares of one bit are placed into different registers [JS17; Bel+20]

Second solution: share slicing

Shares of one bit are placed into one slice of a physical register [JS17; GR17]

- Avoids intermediate recombinations of shares
- Requires less memory words

Overhead Comparison of Hardware Masking ISEs

d_H	ISE	Time		Area & period overhead		
		Cycles	Ov.	FFs	LUTs	P
1	SME	1142	n.a.	1.5	1.6	1.6
	Our SME	1152	0.5	1.4	1.5	1.0
	Skiva	2816	4.0	n.a.	n.a.	n.a.
	Our Skiva	13730	4.8	1.0	1.1	1.0
	Our	5452	2.3	1.1	1.0	1.0
2	SME	1333	n.a.	1.9	1.9	1.7
	Our SME	1271	0.5	1.8	1.8	1.1
	Our	8673	3.7	1.1	1.0	1.0
3	SME	1524	n.a.	2.6	2.2	1.7
	Our SME	1417	0.6	2.4	2.2	1.1
	Skiva	9264	13.2	n.a.	n.a.	n.a.
	Our Skiva	24787	17.0	1.0	1.1	1.0
	Our	11010	4.7	1.2	1.0	1.0