Cryptographic Extensions for Embedded Processors

Fabrice LOZACHMEUR

PhD thesis defended in Lorient on January 23, 2024

CNRS, UBS, Lab-STICC, Lorient, France

PhD thesis CIFRE funded by Thales LAS France SAS

Security of Embedded Systems

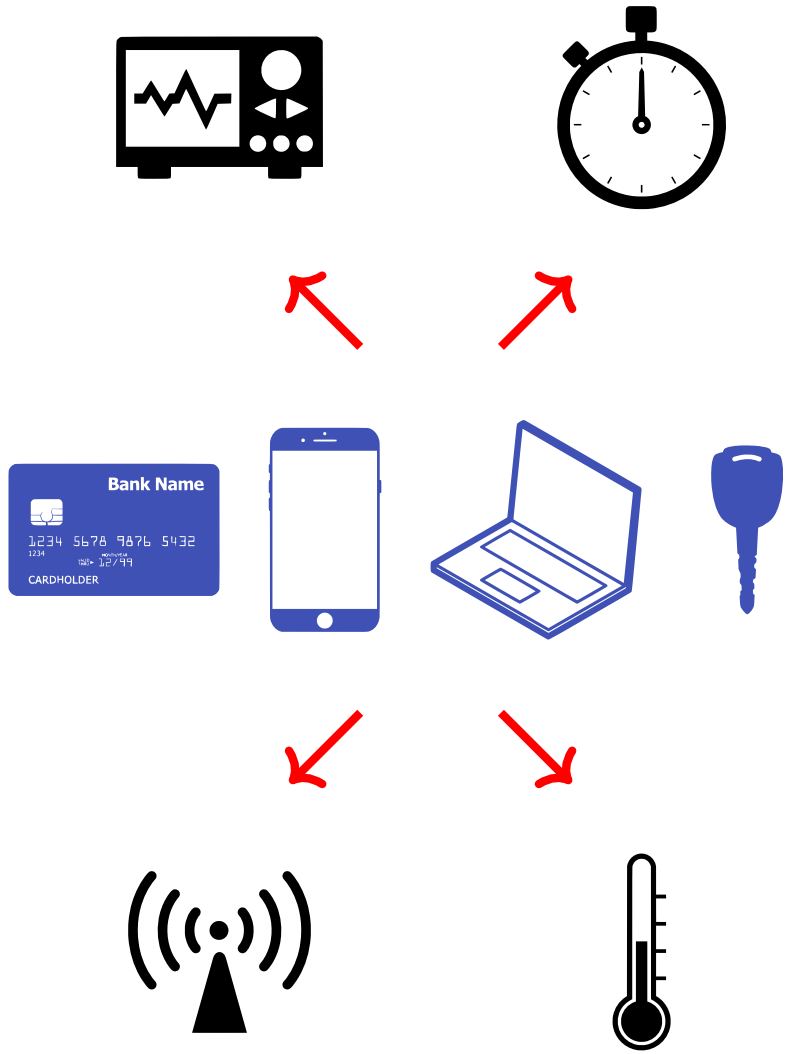
Cryptography:

Embedded systems use mathematically robust ciphers.

Side-channel attack (SCA):

Attack observing physical measurements:

- Execution time
- Power consumption
- Electromagnetic radiation
- Temperature
- ...



Different Implementation Methods

Hardware implementation:

- Efficient for a dedicated task
- Not very flexible

Software implementation:

- High flexibility
- Basic instructions are not designed for cryptography

Instruction set extension (ISE):

- Add new instructions for cryptographic operations
- A trade-off between software and hardware implementations
- Better performance while maintaining flexibility

Thesis Objectives

Develop ISEs to protect against SCA:

- New instructions for masking countermeasure
- Support high security levels
- Flexible masking solution implemented in software and hardware

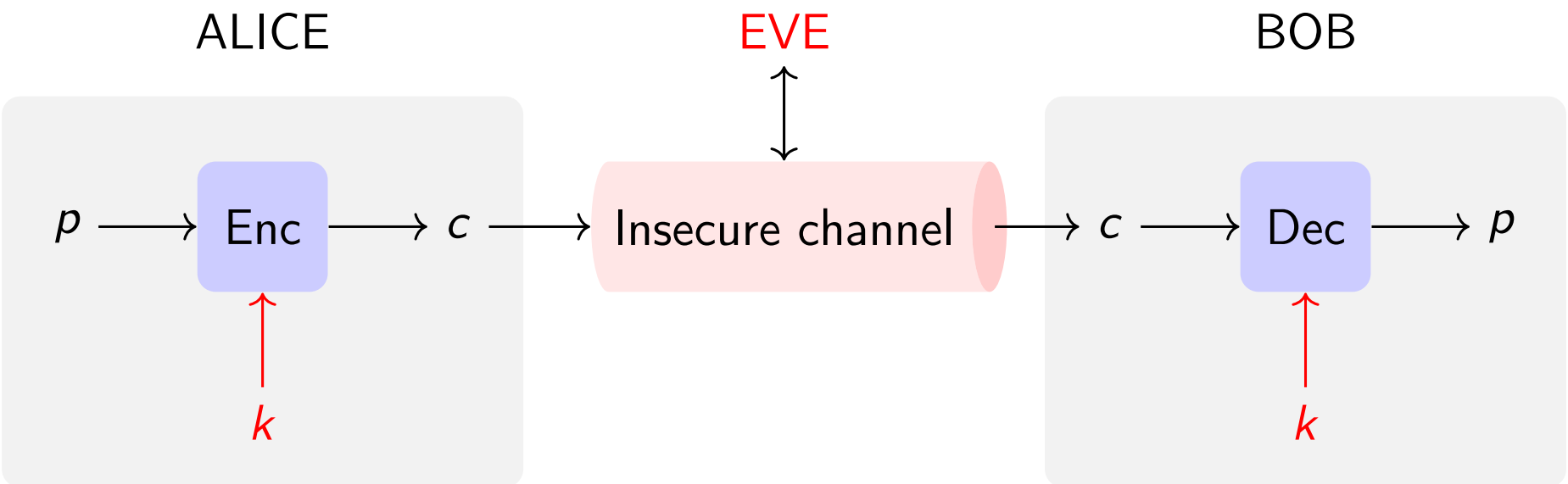
Implementation and evaluation:

- Implement our ISEs on an open source RISC-V processor
- Performance and area results of the FPGA implementation
- Security assessment of our solutions

Table of Contents

Table of Contents

Symmetric Cryptography

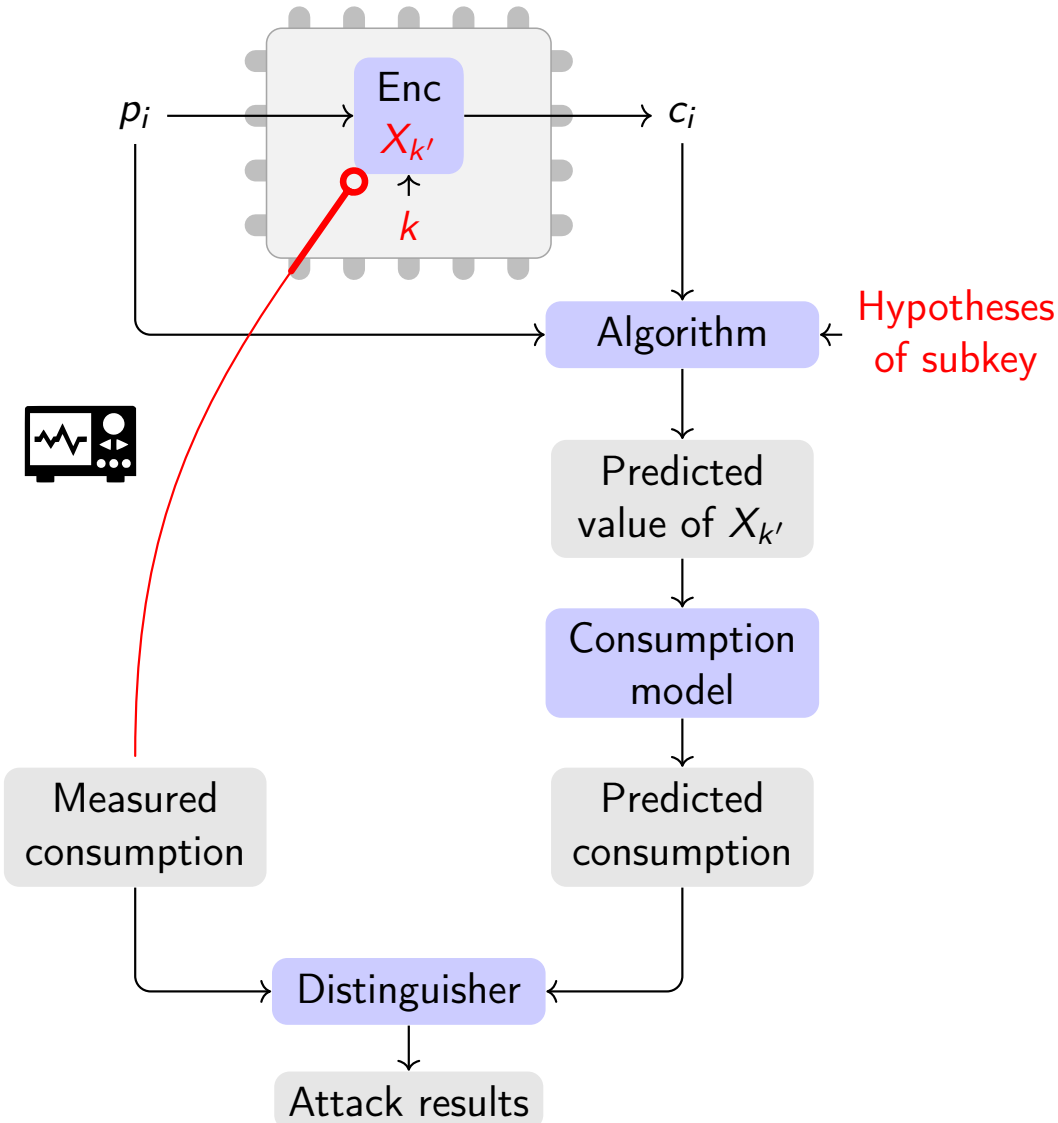


- Enc encrypts a plaintext p into a ciphertext c using a secret key k
- Dec decrypts the ciphertext c using the same secret key k
- Eve can read the ciphertext c without obtaining any secret

Power Analysis Attacks [KJJ99]

Measure power consumption to deduce secret data:

- Target a sensitive variable
- Measure consumption
- Predict consumption
- Statistical comparison



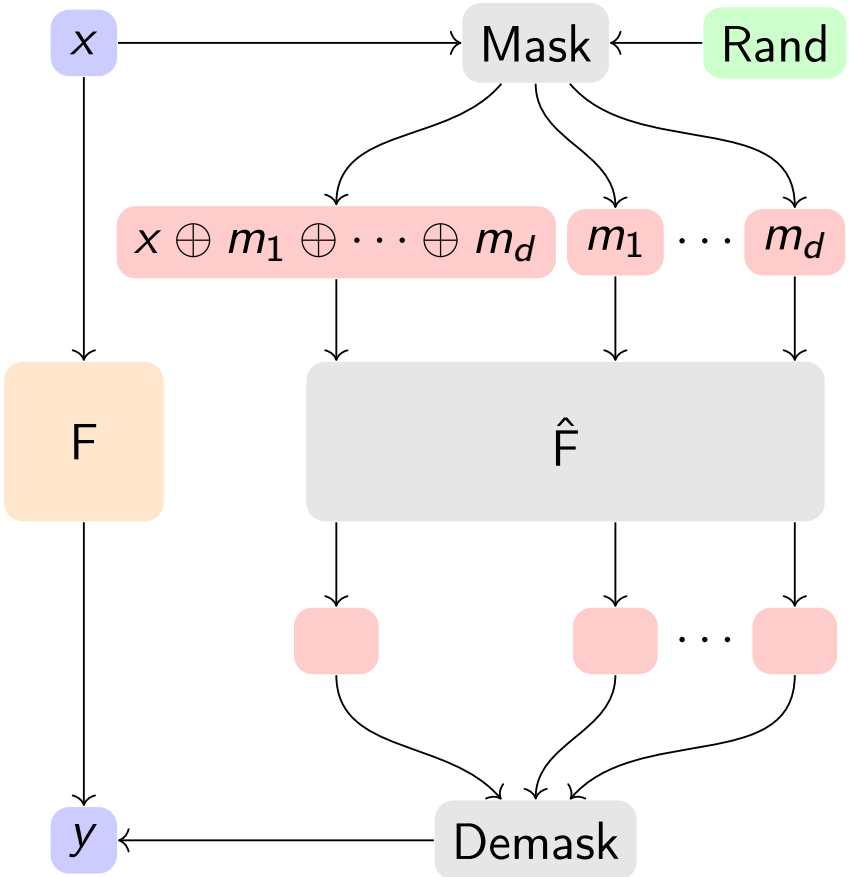
Masking Countermeasure [Cha+99]

Boolean masking of order d:

- Mask the variable x in $d+1$ shares

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

  with $m_1, \cdots, m_d$ random masks
- Apply a d-order masked function $\hat{\mathrm{F}}$
- Demask to get y



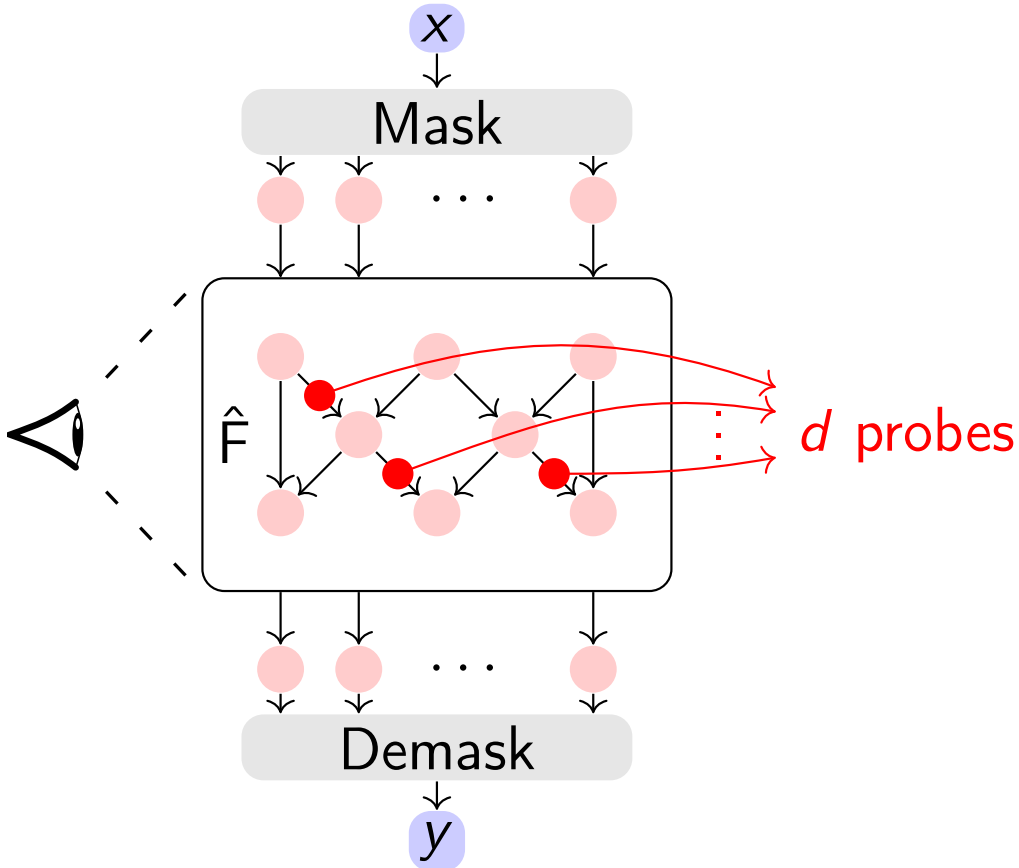
Evaluating the Security of Masked Functions

Security analysis:

- Performing SCA Attacks
- Proving security in a theoretical model

Probing model [ISW03]:

- Probe exact values of d intermediate variable
- Mask and demask functions are not probed



Recombinations of Shares [RP10]

Boolean AND masked at order 1:

- x masked in two shares (x_0, x_1)
- y masked in two shares (y_0, y_1)
- Get xy by demasking (z_0, z_1):

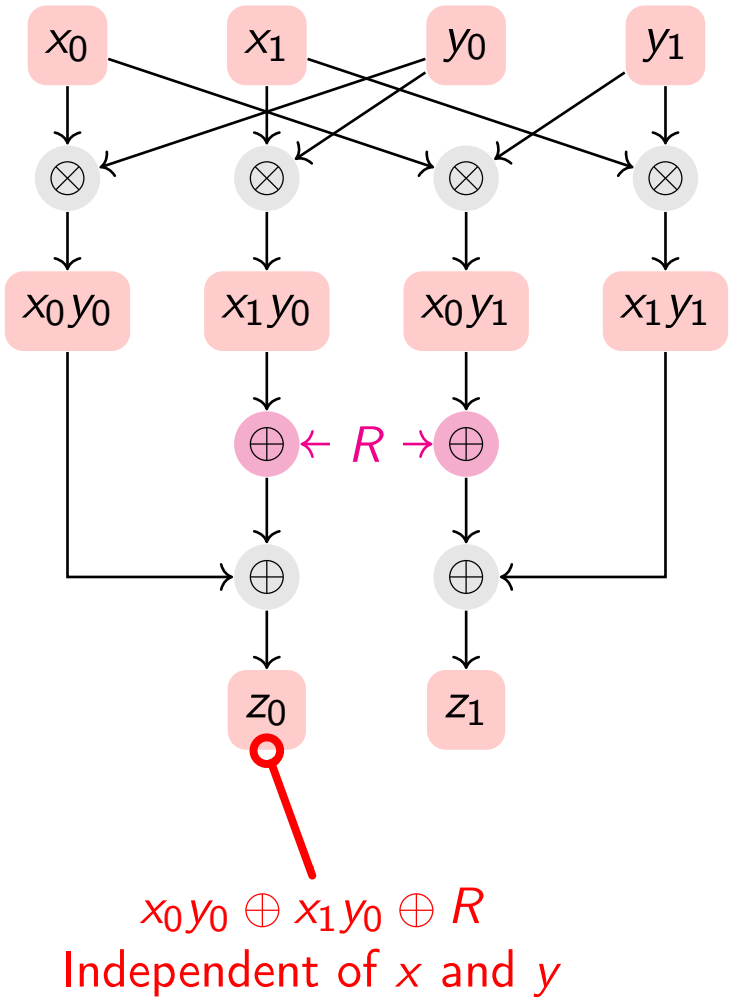
$$z_0 \oplus z_1 = xy$$

Recombinations of shares:

Masked variables can be demasked during calculation.

Prevent recombinations:

Add random values at well-chosen places.



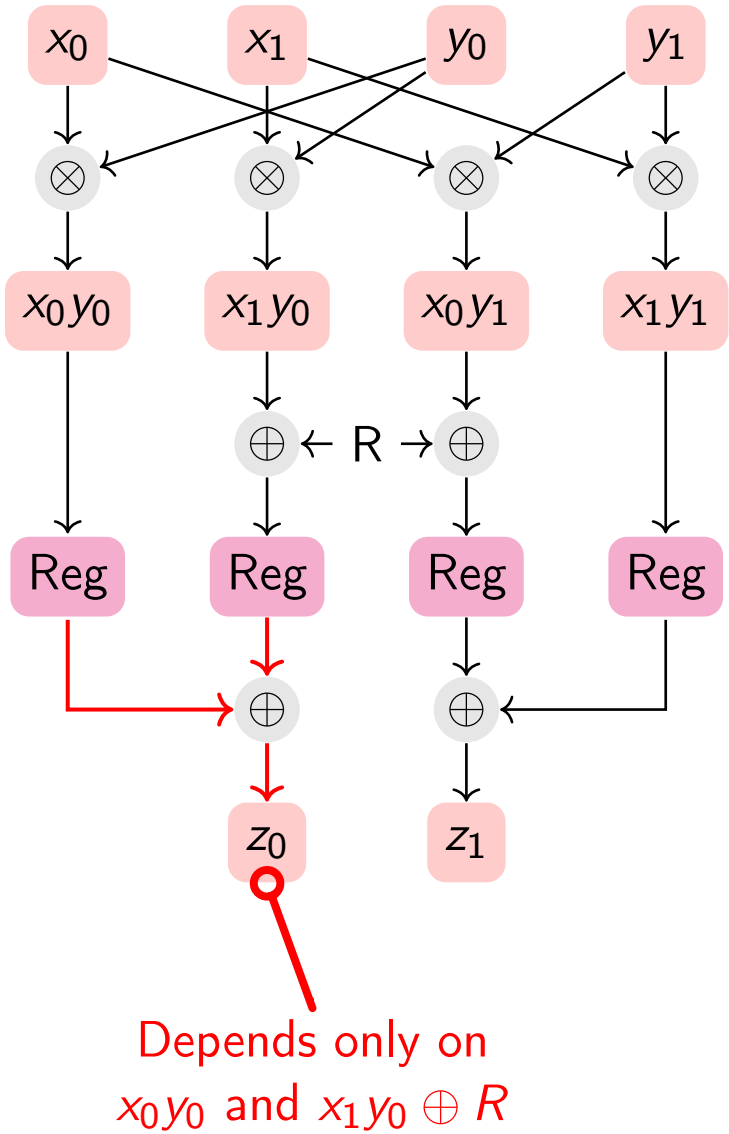
Security against Glitches

Glitch reconbinaisons [MPG05]:

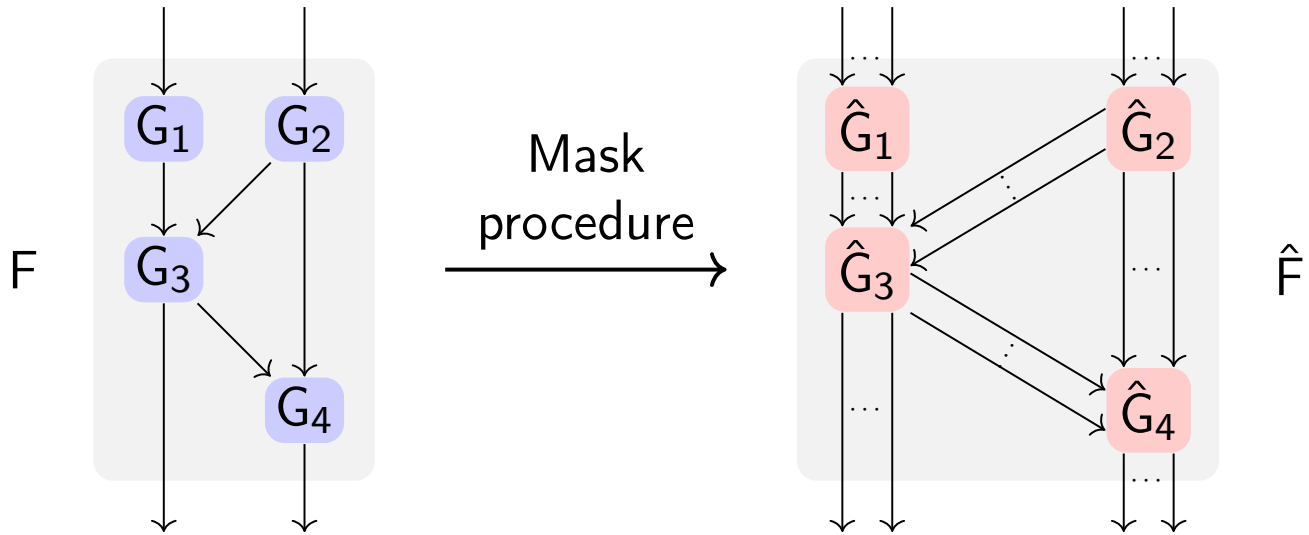
Glitches can reveal information.

Prevent glitch recombinations [GMK16]:

Add registers to stop glitches.



How to Mask a Large Function?



Composition of masked functions [RP10]:

- Divide the function into parts small enough to be easily masked
- Mask each part and compose them
- Composition of secure parts is not always secure

Composability properties:

- Strong Non-Interference (SNI) [Bar+16]
- Probe Isolating Non-Interference (PINI) [CS20]

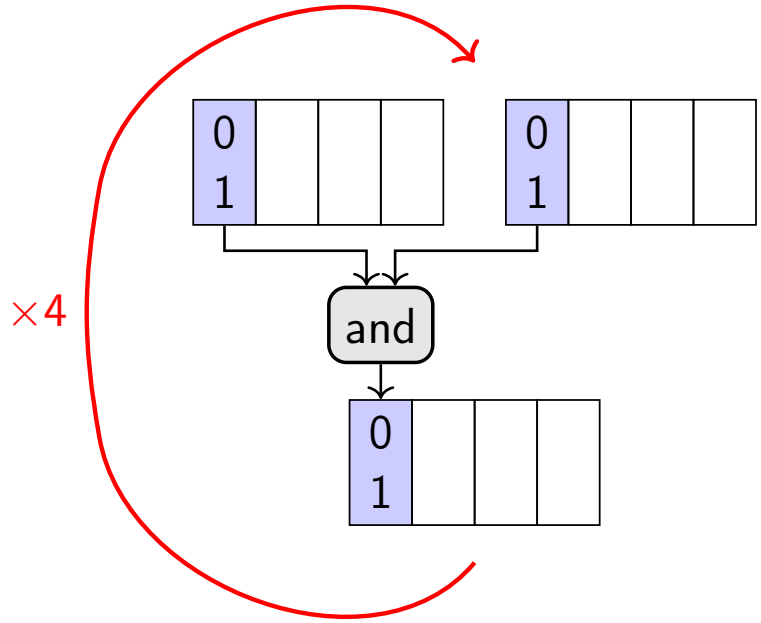
Bit Slicing Implementations

Boolean calculations in processors:

- Calculation on only one bit
- Under utilization of resources

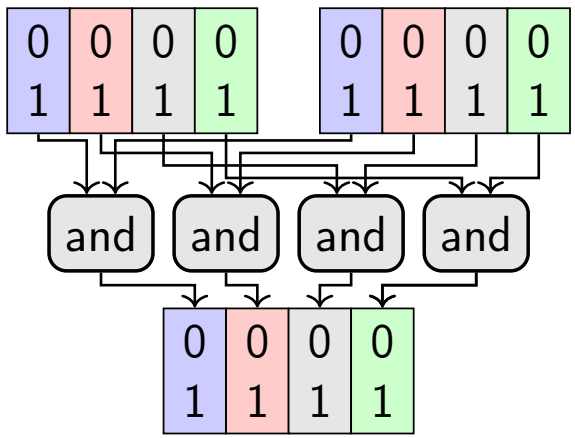
Bit slicing (BS) [Bih97]:

- Parallel calculation on several independent data bits
- High-throughput implementations



USUBA [MD19]:

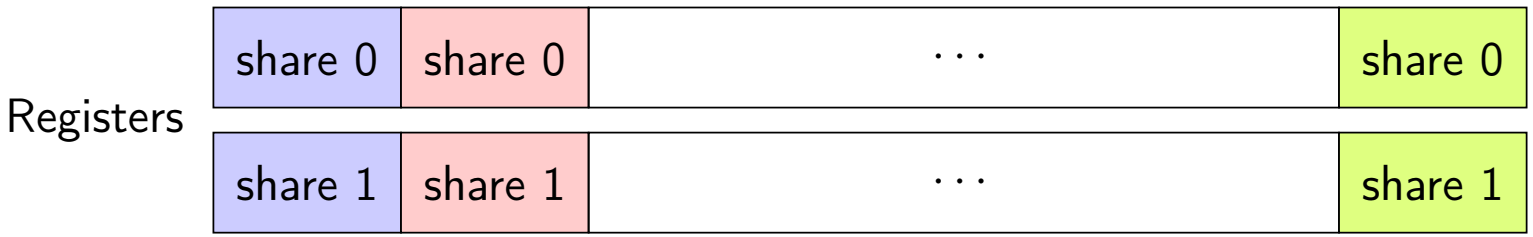
- Language to describe BS implementations
- A compiler allows to synthesize USUBA codes into C codes



How to Mask Bit Slicing Implementations?

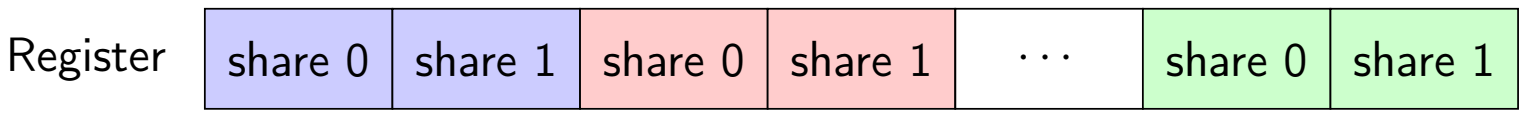
Masking using Tornado [Bel+20]:

- Shares of one bit are placed into different physical registers
- Recombinations of shares can occur by writing a register



Share slicing [JS17]:

- Shares of one bit are placed in the same physical register
- Avoids recombinations of shares



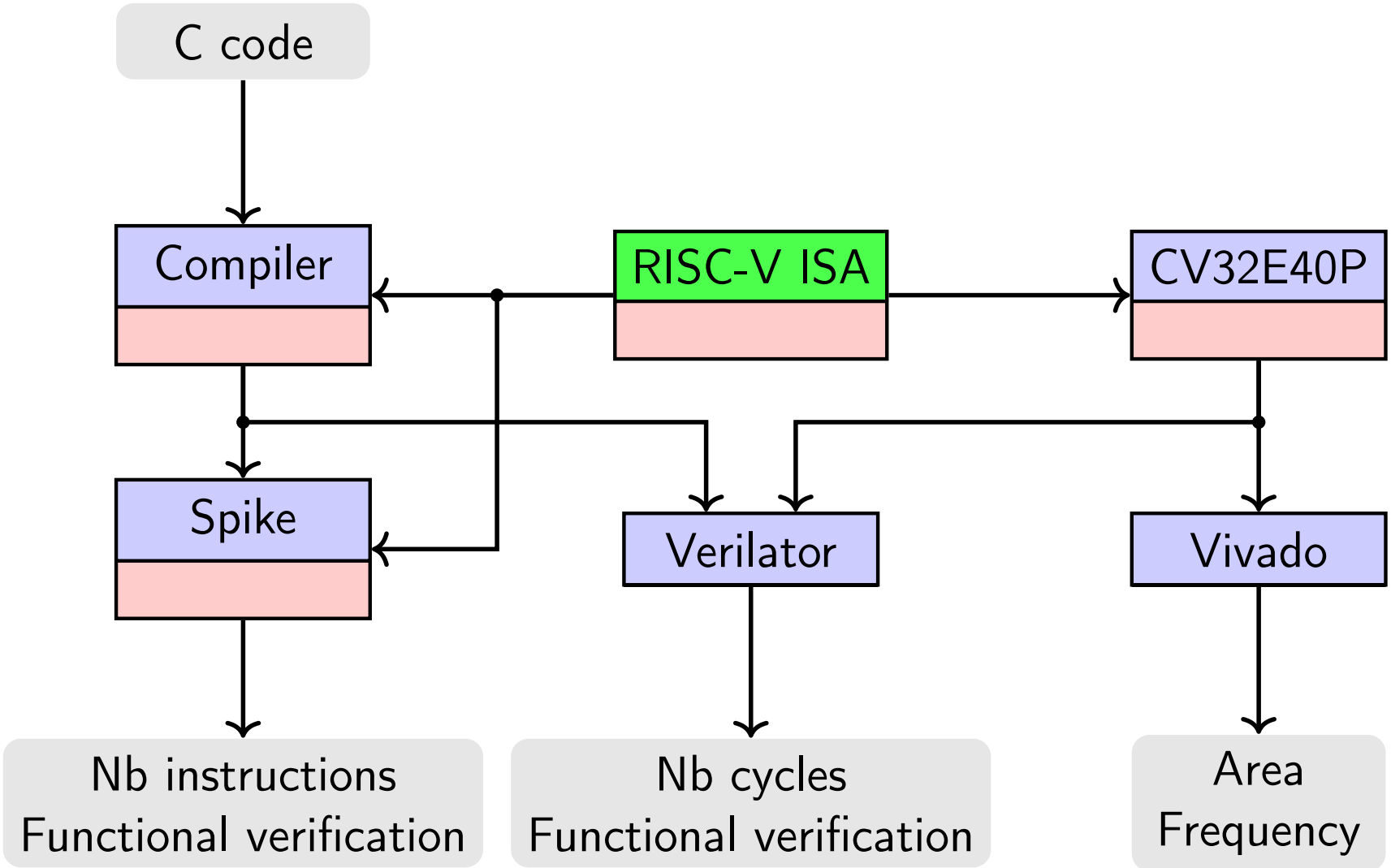
Masked ISE of the State of the Art

- Protection limited to small masking orders
- Not flexible enough to change the masking order at design time and run time

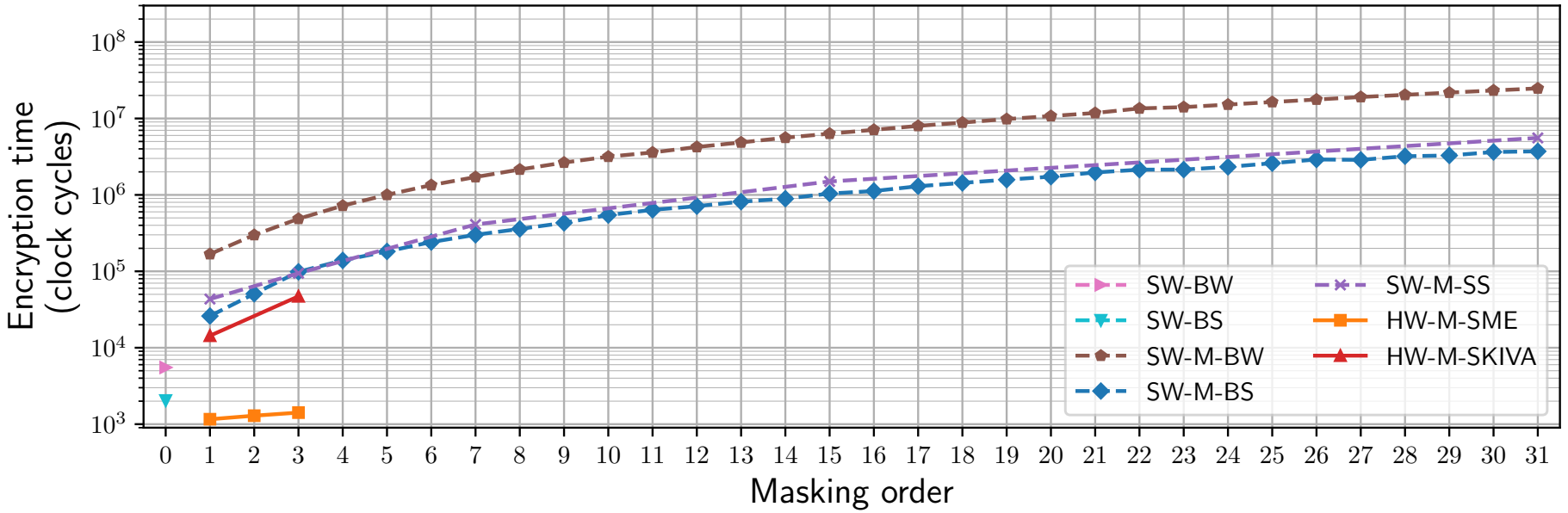
Reference	RISC-V	Masking order	Flexbility at design time	Flexbility at run time
[Gro+16]	✓	$\{1, 2, 3, 4\}$	✓	✗
[DGH19]	✓	1	✗	✗
[Gao+21]	✓	1	✗	✗
SKIVA [Kia+21]	✗	$\{1, 3\}$	✗	✓
SME [MP21]	✓	$\{1, 2, 3\}$	✓	✗

Table of Contents

Experimental Environment



Reimplementation of Works from the State of the Art



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

- SW-BW is unmasked and byte-wise [DR02]
- SW-BS 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 share-slicing [JS17]
- HW-M-SME is masked with SME [MP21]
- HW-M-SKIVA is masked with SKIVA [Kia+21]

Analysis of Resistance against SCA Attacks

Generation of simulated consumption traces:

- Extracts internal state of the register file
- Simulates consumption with Hamming distance

SCA analysis:

- Leakage assessment of order 1
- SCA attacks of order 1

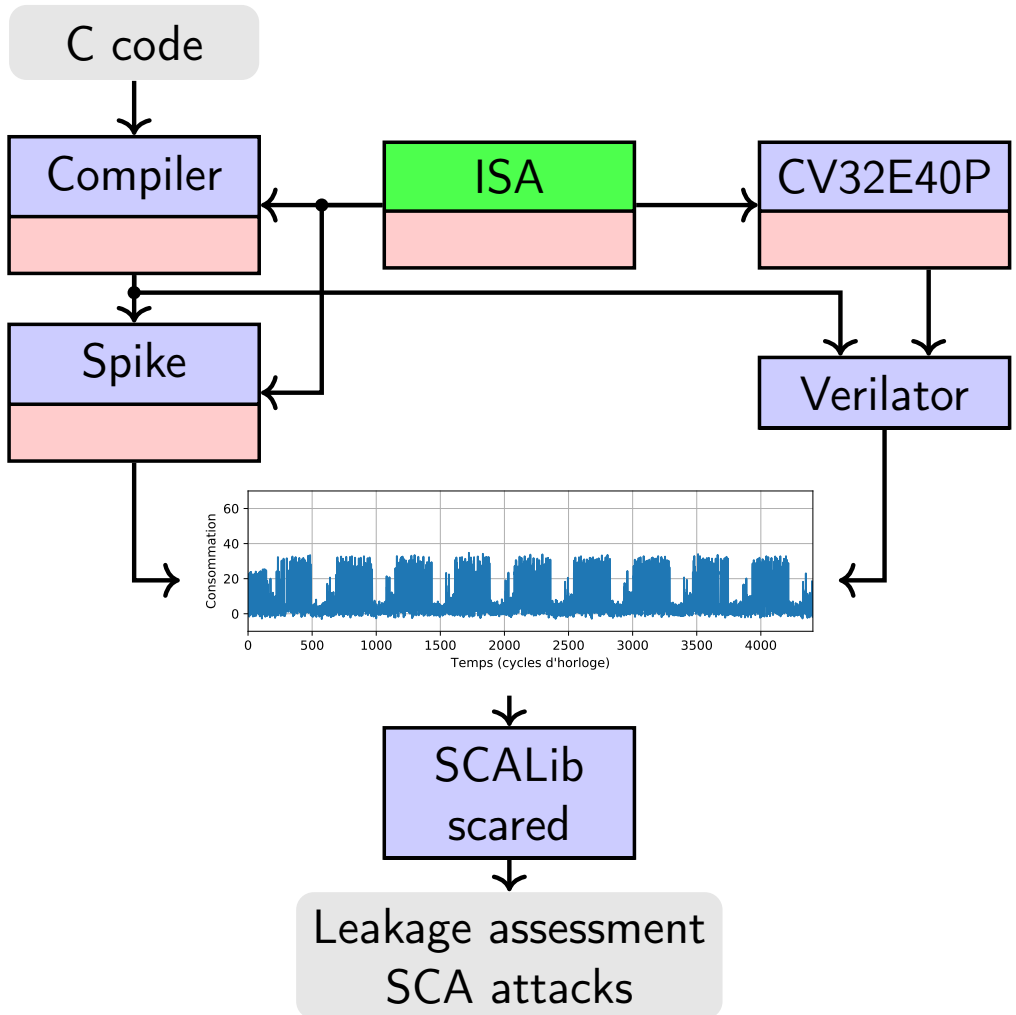


Table of Contents

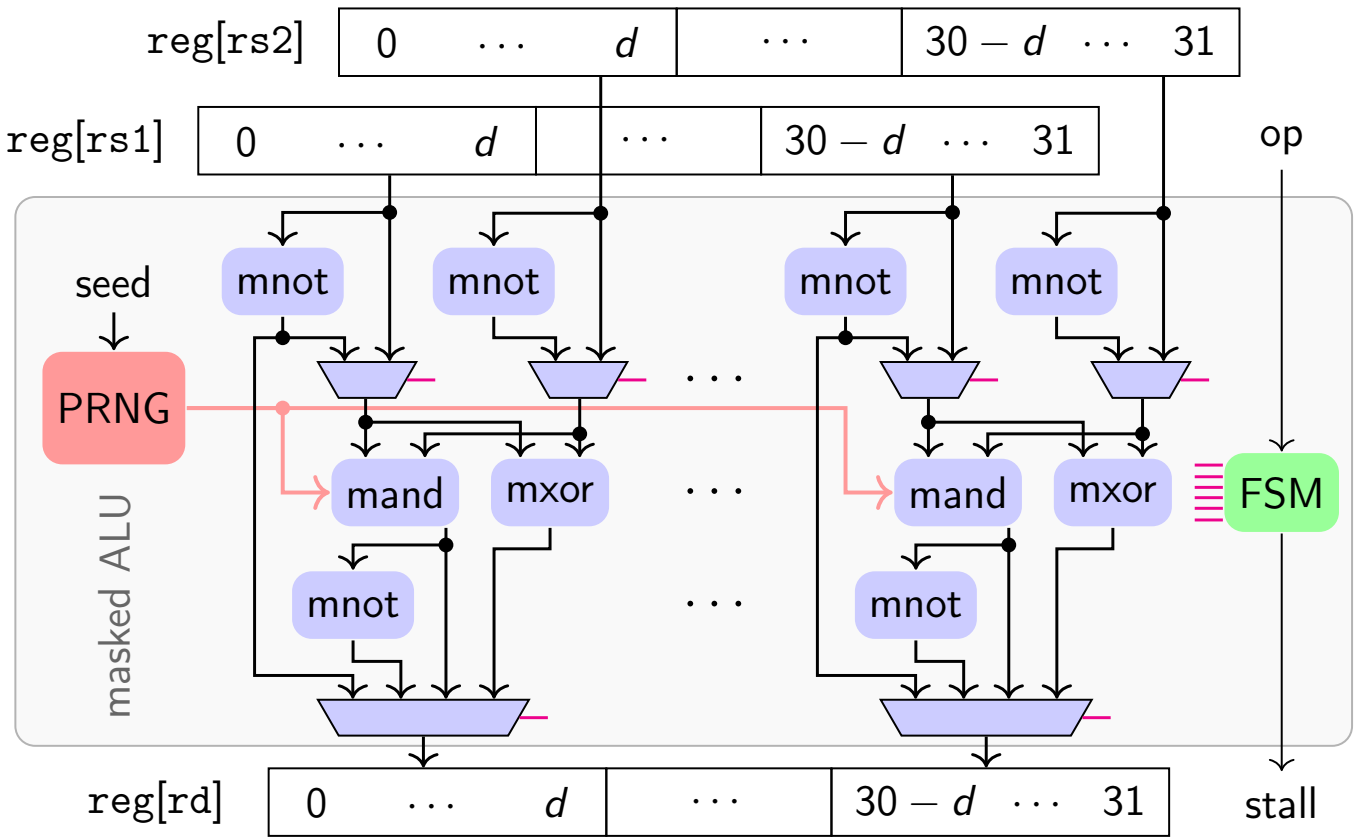
ISE1: Specification of our Extension

- Masked instructions for share slicing implementations
- Masking order $d \in \{1, \cdots, 31\}$ is fixed at synthesis time
- Masked codes are generated using USUBA
- PINI instructions for secure code by direct composition

Instruction	Format	Latency	Random bits
masked AND	`ise1.and rd, rs1, rs2`	2	$32(d-2)$
masked OR	`ise1.or rd, rs1, rs2`	2	$32(d-2)$
masked NOT	`ise1.not rd, rs1, rs2`	1	0
masked XOR	`ise1.xor rd, rs1, rs2`	1	0

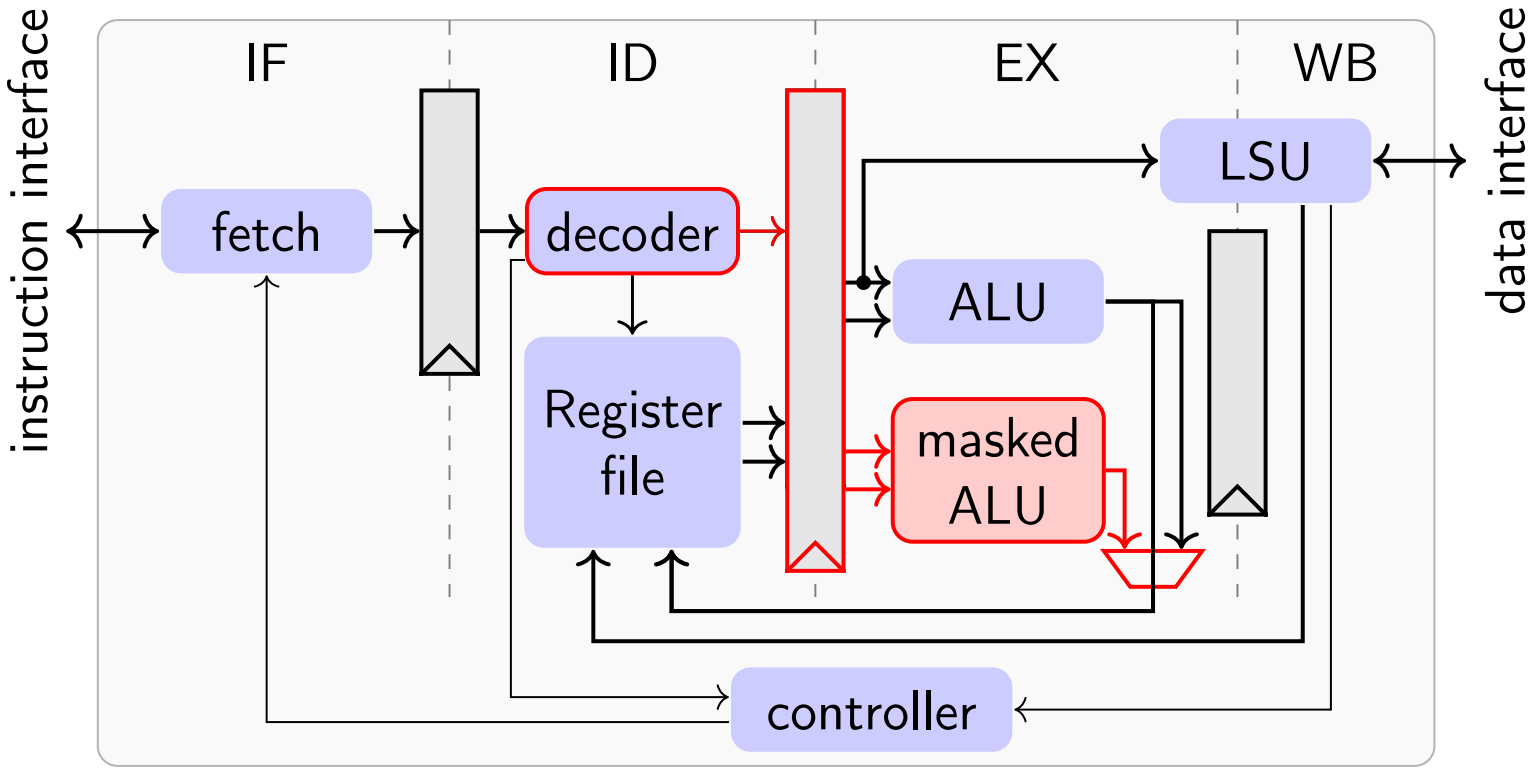
ISE1: Proposed Masked ALU

- Source registers are divided into blocks of $d + 1$ bits
- Masked PINI gates mand, mxor and mnot are applied to each block
- An FSM controls the masked ALU
- A PRNG provides random values

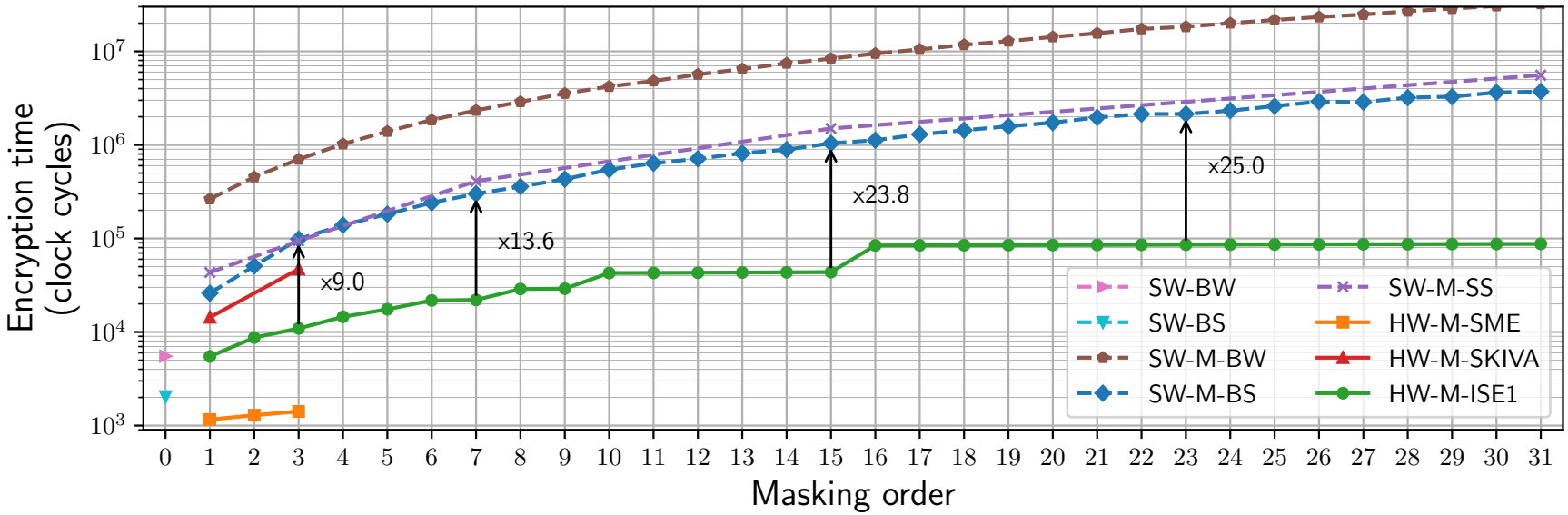


ISE1: Integration into the CV32E40P Core

- CV32E40P is a 32-bit RISC-V processor from the OpenHW Group
- Red parts are added or modified for our masked ISE1



ISE1: Evaluation of Performances

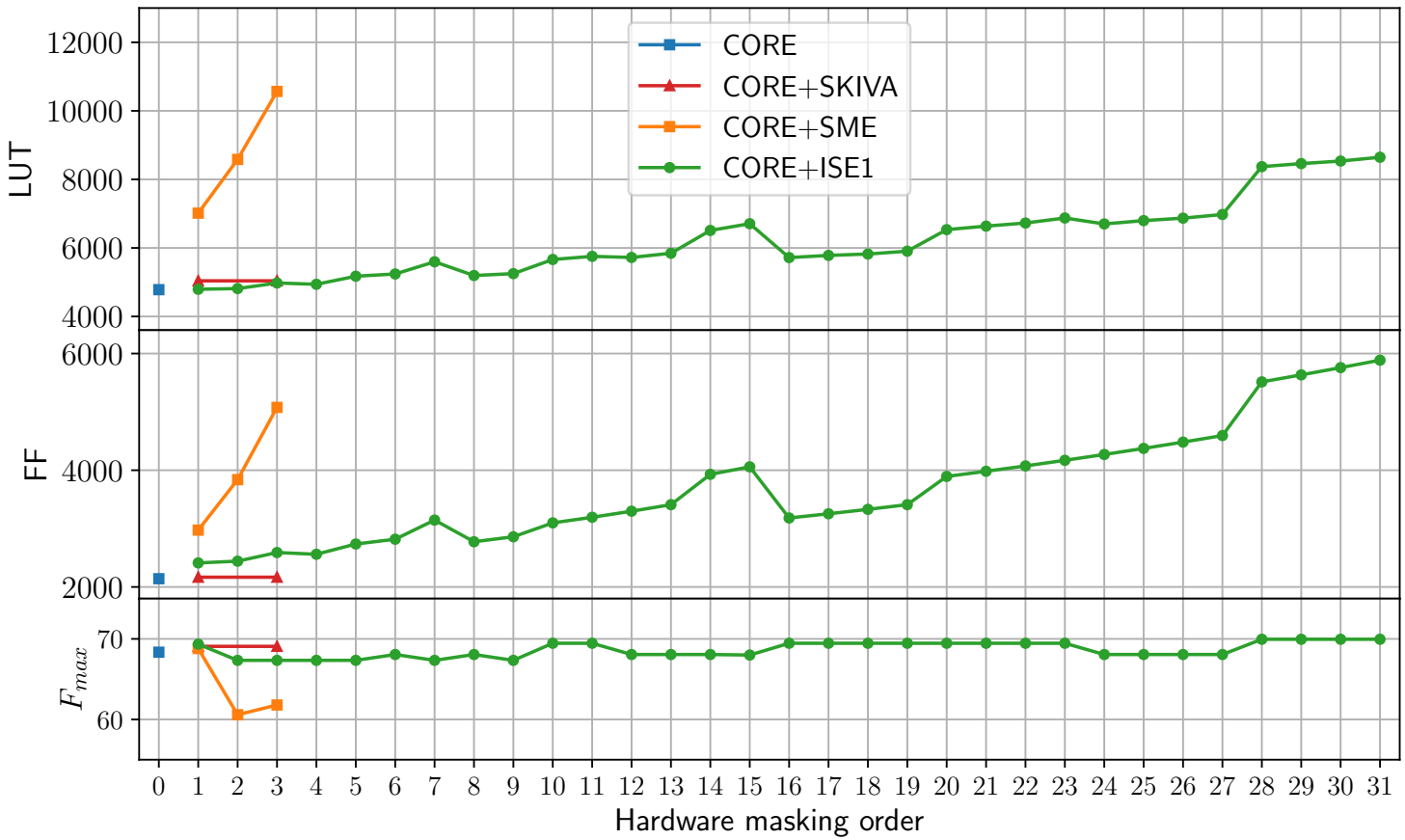


Encryption times in **log scale** for one AES block:

- SW-BW is unmasked and byte-wise [DR02]
- SW-BS 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 share-slicing [JS17]
- HW-M-SME is masked with SME [MP21]
- HW-M-SKIVA is masked with SKIVA [Kia+21]
- HW-M-ISE1 is masked with our masked ISE1

Implementation on FPGA

Area/frequency results on a Digilent Arty A7 FPGA board of the CV32E40P with Skiva, SME and our masked ISE1.



ISE2: Specification of our Extension

Constraints of ISE1:

- ISE1 encrypts several independent blocks in parallel
- Limits usable encryption modes

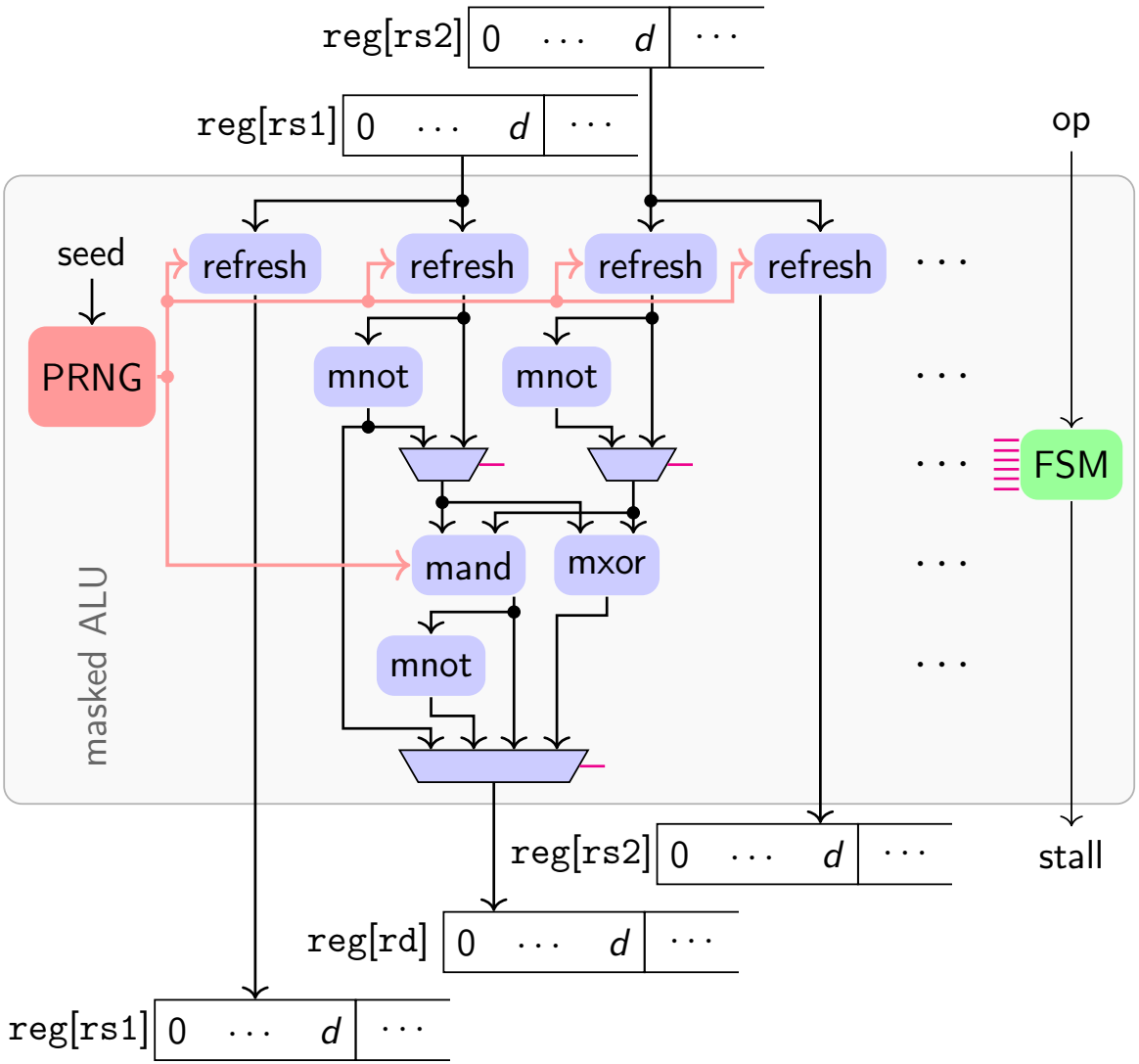
Our masked ISE2:

- Extends ISE1 to reduce the number of independent blocks encrypted in parallel
- Allows to mask one block at a time at orders 1, 3, 7, 15
- Larger and slower than ISE1

Instruction	Format	Latency	Random bits
masked SLL	`ise2.sll rd, rs1, rs2`	1	0
masked SRL	`ise2.srl rd, rs1, rs2`	1	0
masked SLLI	`ise2.slli rd, rs1, imm`	1	0
masked SRLI	`ise2.srli rd, rs1, imm`	1	0

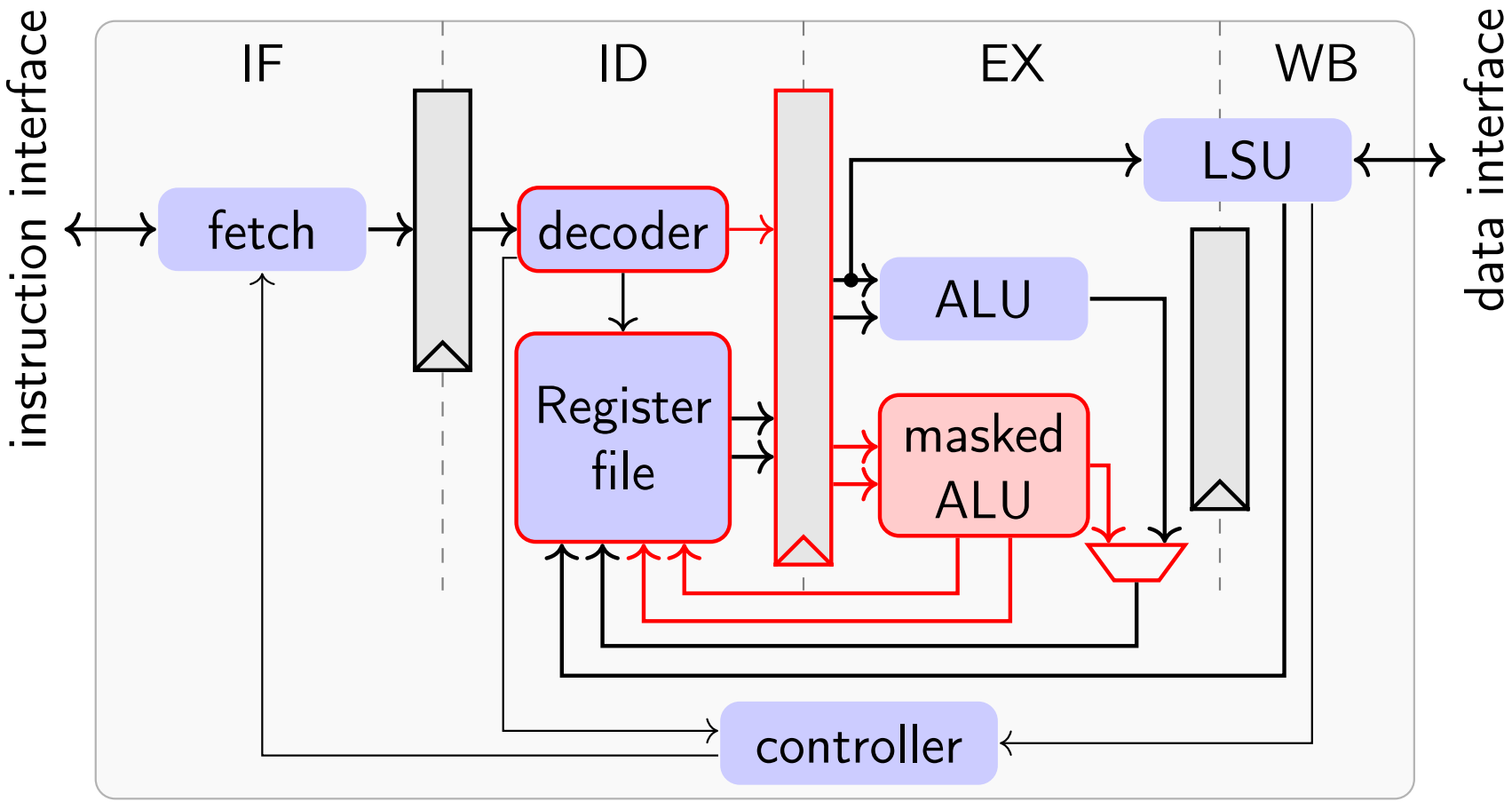
ISE3: Specification of our Extension

- Modifies ISE1 to increase security
- Source registers are refreshed at each use
- Protection against an attacker probing $d/2$ bits per instruction

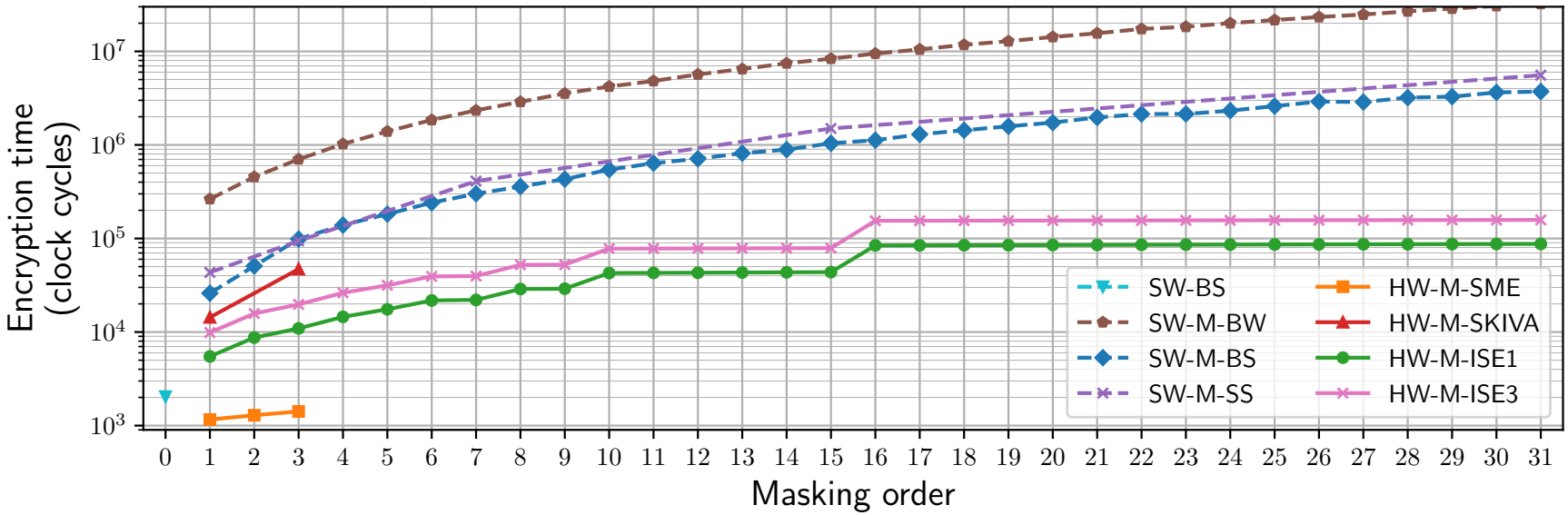


ISE3: Integration into the CV32E40P Core

Red parts are added or modified for our masked ISE3.



ISE3: Evaluation of Performances

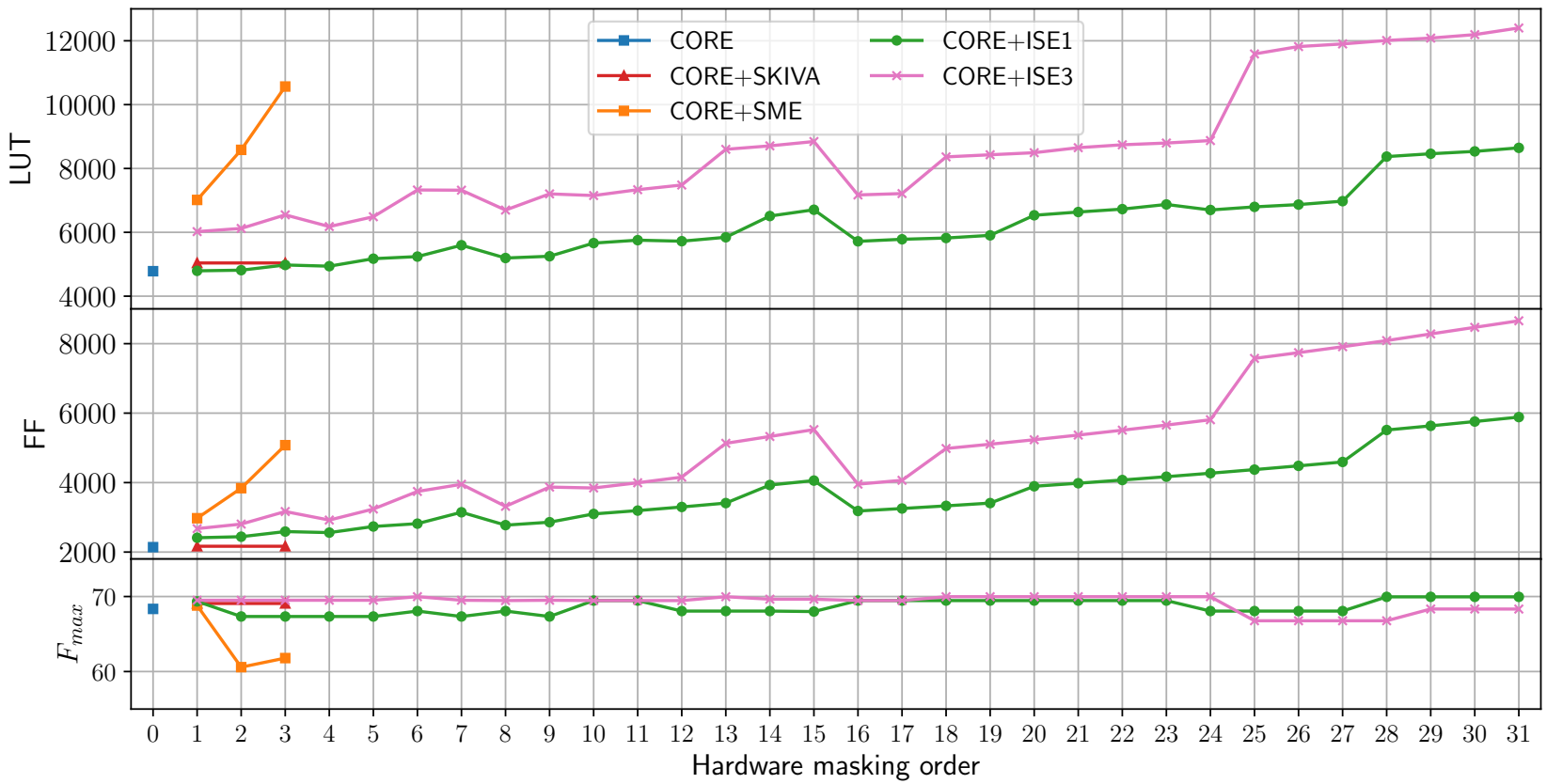


Encryption times in **log scale** for one AES block:

- SW-BS 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 share-slicing [JS17]
- HW-M-SME is masked with SME [MP21]
- HW-M-SKIVA is masked with SKIVA [Kia+21]
- HW-M-ISE1 is masked with our masked ISE1
- HW-M-ISE3 is masked with our masked ISE3

ISE3: Implementation on FPGA

Area/frequency results on a Digilent Arty A7 FPGA board of the CV32E40P with Skiva, SME, our masked ISE1 and our masked ISE3.



HW-SW: Mixed Masking Solution

Hardware masking:

Our masked ISE1 whose order d_{H} is fixed at the synthesis time.

Software masking:

Secure composition over the masked instructions to mask at order:

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

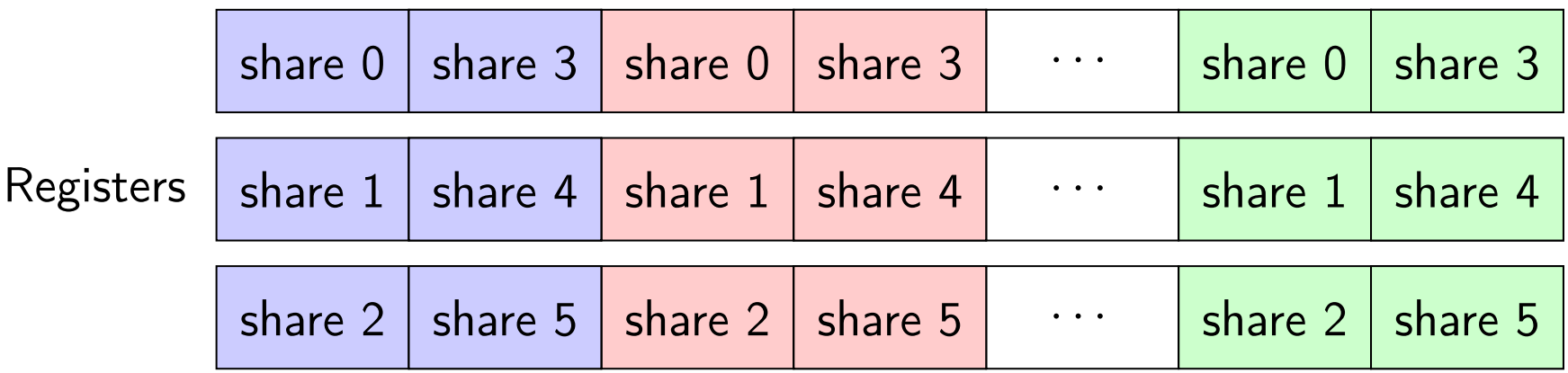
where d_{S} is the software masking order fixed at compilation time.

Flexibility of our solution:

- Allows to adapt the level of security over time
- Allows different cost/performance trade-offs

HW-SW: Our Masked Representation

- Mask at order d_{S} by placing the shares in different registers
- Each share is masked at order d_{H} using share slicing representation
- Total masking order is $d = (d_{\mathrm{S}} + 1)(d_{\mathrm{H}} + 1) - 1$



HW-SW: Our Masking Scheme

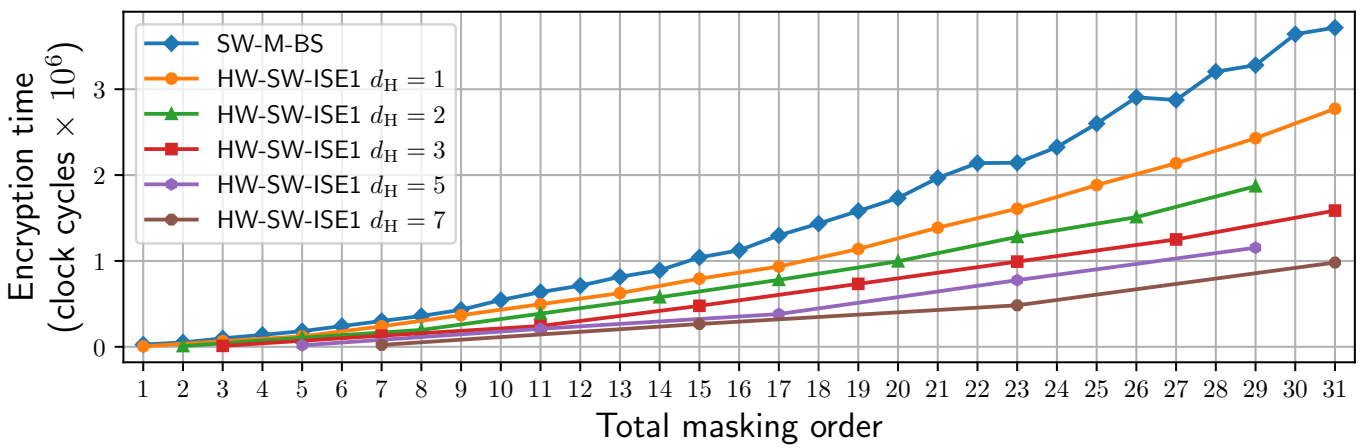
- AND gate masked in software at order d_S and PINI
- Replace elementary operations by instructions of ISE1 masked at order d_H
- Resulting AND gate is PINI and masked at total order:

$$d = (d_S + 1)(d_H + 1) - 1$$

Require: $x_i \in \mathrm{GF}(2)^{d_S+1}$ et $y_i \in \mathrm{GF}(2)^{d_S+1}$
Ensure: $z_i \in \mathrm{GF}(2)^{d_S+1}$
  for $i = 0$ à d_S **do**
    $u_{i,i} \leftarrow \mathrm{ise1.and}(x_i, y_i)$
    for $j = i + 1$ à d_S **do**
      $r_{i,j} \xleftarrow{\$} \mathrm{GF}(32)$
      $a_{i,j} \leftarrow \mathrm{ise1.xor}(y_j, r_{i,j})$
      $b_{i,j} \leftarrow \mathrm{ise1.and}(x_i, a_{i,j})$
      $c_{i,j} \leftarrow \mathrm{ise1.not}(x_i)$
      $d_{i,j} \leftarrow \mathrm{ise1.and}(c_{i,j}, r_{i,j})$
      $u_{j,i} \leftarrow \mathrm{ise1.xor}(b_{i,j}, d_{i,j})$
      $a_{j,i} \leftarrow \mathrm{ise1.xor}(y_i, r_{i,j})$
      $b_{j,i} \leftarrow \mathrm{ise1.and}(x_j, a_{j,i})$
      $c_{j,i} \leftarrow \mathrm{ise1.not}(x_j)$
      $d_{j,i} \leftarrow \mathrm{ise1.and}(c_{j,i}, r_{i,j})$
      $u_{j,i} \leftarrow \mathrm{ise1.xor}(b_{j,i}, d_{j,i})$
    end for
  end for
  for $i = 0$ à d_S **do**
    $v_{i,0} \leftarrow u_{i,j}$
    for $j = 1$ à d_S **do**
      $v_{i,j} \leftarrow \mathrm{ise1.xor}(v_{i,j-1}, u_{i,j})$
    end for
    $z_i \leftarrow v_{i,d_S}$
  end for

HW-SW: Implementation Results of our Solution

Encryption times in log scale for one AES block and area/period overheads for our ISE masked at orders $d_{\mathrm{H}} \in \{1, 2, 3, 5, 7\}$ and various total orders.



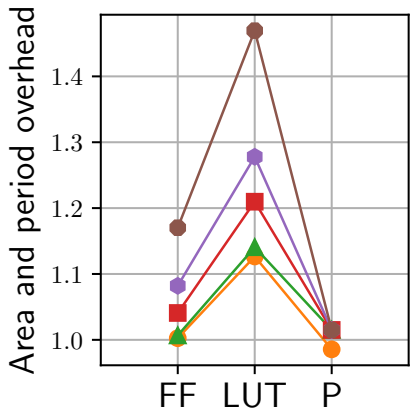


Table of Contents

Conclusion and Future Prospects

Our contributions:

- ISEs for masking countermeasure
- High order masking
- Good speeds up with a limited silicon cost
- Flexibility at synthesis time and compile time
- Can be used on various cryptosystems

Future works:

- Implement other cryptosystems with our solutions
- Security evaluation using physical attacks
- Masked ISE optimized for post-quantum cryptography

End of Presentation

Thank you for your attention

Do you have any questions?

Bibliography I

[Bar+16] Gilles Barthe et al. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: Proc. Conference on Computer and Communications Security (CCS). ACM, Oct. 2016, pp. 116–129. DOI: 10.1145/2976749.2978427.

[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.

[Bih97] Eli Biham. "A fast new DES implementation in software". In: Proc. Fast Software Encryption (FSE). Springer, Jan. 1997, pp. 260–272. DOI: 10.1007/BFb0052352.

[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.

[DR02] Joan Daemen and Vincent Rijmen. The Design of Rijndael. 1st ed. Springer, 2002. ISBN: 978-3-540-42580-9. DOI: 10.1007/978-3-662-04722-4.

[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.

Bibliography II

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. IACR Cryptology ePrint Archive. Nov. 2016. URL: https://eprint.iacr.org/2016/486.

[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.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: Proc. Annual Cryptology Conference (CRYPTO). Springer, Aug. 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27.

[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.

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

Bibliography III

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. "Side-Channel Leakage of Masked CMOS Gates". In: Proc. Cryptographers' Track RSA Conference (CT-RSA). Springer, Feb. 2005, pp. 351–365. DOI: 978-3-540-30574-3_24.

[RP10] Matthieu Rivain and Emmanuel Prouff. "Provably Secure Higher-Order Masking of AES". In: Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Springer, Aug. 2010, pp. 413–427. DOI: 10.1007/978-3-642-15031-9_28.