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

Fabrice Lozachmeur, Arnaud Tisserand

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

Fabrice Lozachmeur[1] and Arnaud Tisserand[2]

***Abstract*— The paper presents an instruction set extension for the CV32E40P RISC-V processor core to protect cryptosystems against side channel attacks using masking at high orders. A first masking order is fixed in hardware at synthesis time. Then a higher masking order is obtained using software composition over the hardware one. The solution has been implemented, validated and evaluated in FPGA.**

***Index Terms*— hardware security; side-channel attack; countermeasure; processor instruction set extension**

I. INTRODUCTION

Implementations of cryptosystems, even mathematically strong ones, are likely to *leak information* through the *observation of side channels* (e.g., computation time, power consumption, electromagnetic radiation). *Side channel attacks* (SCAs) [1] exploit potential correlations, even tiny ones, between *measured physical values* and *operations and operands* processed in the circuit to recover sensitive data.

In both hardware and software implementations of cryptosystems, *masking* is a widely studied countermeasure against SCAs [2], [3]. It consists in *randomizing* intermediate values, see Sec. II-A. But the amount of randomness, called masking *order*, required against recent SCAs is increasing. For instance, [4] presents some key recovery for an order-5 masked AES implementation in less than 10 traces. Moreover, masking at a high order leads to important overheads in area, computation time and memory/program sizes.

Dedicated hardware masked accelerators are used in some applications such as smart cards, but they are *not flexible* enough to adapt the security level without changing the hardware. Then software implementation is still key for long term security (e.g., security updates in programs and firmwares).

Instruction set extensions (ISEs) have been proposed in many research and commercial processors to accelerate software cryptographic codes in the past. Recently, a few ISEs support a masking protection in hardware but mainly for low orders and for the AES cryptosystem (see Sec. II-E).

In this work, we propose a *flexible hardware and software* solution for efficient and secure implementation of *various cryptosystems* using *masking protections at various high orders*. A first masking level is obtained in hardware, then a higher level is obtained with software combination. The hardware masking is implemented in a new ISE of a RISC-V processor (see Sec. II-D). The circuit designer *fixes* the masking order to be implemented in hardware *at synthesis time* depending on performance/cost constraints. The software masking uses secure *composability* in combination to the hardware one from the masked ISE to reach various higher orders at run time (see Sec. III).

Sec. II recalls related background elements and previous works. Our contribution is detailed in Sec. III. Sec. IV compares our solution to previous ones. A conclusion and future prospects are given in Sec. V.

II. STATE OF THE ART

A. Masking Countermeasure

Masking [2], [3] consists in *randomizing* all the intermediate sensitive variables processed in a cryptographic algorithm in order to break statistical dependencies between secret data and leakage measurements. *Boolean masking* at *order* d represents a sensitive variable x by $d+1$ *shares*, denoted x_i for $0 \leq i \leq d$, such that $x = x_0 \oplus x_1 \oplus \ldots \oplus x_d$, where any subset of d shares is statistically independent from x. In practice, there are d random values for the shares x_1 to x_d and the first share x_0 is x XORed with all the random values. Order-d masking protects against attackers able to use up to d probes (observation of a variable in the algorithm), see [5] for details. Masking and unmasking transformations only require XOR operations and high-quality random numbers, but computing on masked values is more complex.

Software masking at a high order d leads to significant overheads in computation time (which increases as $O(d^2)$, see [5]), executable code size, memory and throughput from a high-quality random number generator (RNG). More, pure software protections are very limited due to leaks in the micro-architecture of the processor. For example, [6] shows that bit-interaction leakages between instructions in ARM M0/M3 cores are devastating for high-order masking. Then providing some masking level in hardware is relevant.

To mask a complex circuit, one common solution is to mask sub-circuits and compose them. However, a simple direct composition of masked sub-circuits is rarely secure [7]. Several compositional properties have been proposed to ensure a high protection. Refresh operations, as proposed in [8], avoid shares recombination problems by inserting new random values during the masked computations. Inserting the minimum number of refresh operations required to ensure a high security is not simple, it also needs a higher RNG throughput. The PINI property, introduced in [9], guarantees direct composability and allows a simple implementation of linear functions (e.g., XOR for addition in GF(2)). Masking other gates (e.g., AND for multiplication in GF(2)) is more

[1]Fabrice Lozachmeur is with Thales LAS France SAS and Lab-STICC UMR6285, Université Bretagne Sud, Lorient, France. fabrice.lozachmeur@univ-ubs.fr

[2]Arnaud Tisserand is with CNRS, Lab-STICC UMR6285, hosted at ENSTA Bretagne, Brest, France. arnaud.tisserand@cnrs.fr

complex. [10] presents a PINI masked AND gate called HPC3 with some protection against leakages due to glitches (a flip-flop is added to filter glitches).

B. Bit Slicing

Bit slicing (BS) was introduced to accelerate software implementations of unprotected cryptosystems in general purpose processors without cryptographic accelerator, see [11] for a historic BS implementation of the DES block cipher. BS transposes k input blocks of l bits into l registers (or slices) of k bits where register j contains the j-th bit of each input block. For AES, blocks are $l = 128$ bits wide. The cryptographic algorithm is then expressed as a circuit composed of elementary boolean gates (e.g. AND, XOR, OR, NOT). All bits in a slice (from various independent blocks) are processed in parallel using the corresponding bitwise logic instruction. The open-source tool Usuba [12] generates BS implementations from a C code.

[13], [14] mask a BS implementation by distributing the shares into different registers. Tornado [15] uses Usuba to generate a BS code which is then masked by inserting refresh operations. As several independent input blocks are processed in parallel for BS, the shares of one bit are distributed over numerous memory words [16].

C. Share Slicing

Other solutions (e.g., [16], [14]) use *share slicing* which places all the shares of a masked bit into one slice of a physical register. For instance, at order-3 (4 shares), there are 8 slices in a 32-bit register. Share-slicing avoids intermediate recombinations of shares during load/store operations [6] and requires less memory words than BS for the shares of one masked bit (this avoids some cache misses).

D. RISC-V Processor(s)

RISC-V is a family of processors specified as open and free instruction set architectures to allow easy implementations and extensions. Several RISC-V cores have been proposed with various characteristics and purposes. They usually come with complete open source software tools.

We use the CV32E40P a 4-stage in-order 32-bit core from the OpenHW Group. Adding ISEs to this core and to its software tools is simple. The new instructions must be integrated in the assembler and the simulator. Intrinsics are be used for programming in C language.

E. Instruction Set Extensions

General purpose processors are rarely efficient for implementing cryptography. In the past, many ISEs have been proposed, on various processors, mainly to speed up unprotected AES (SHA support is more recent).

Cryptographic ISEs with embedded SCA protections are quite recent. The added masked instructions help to reduce the masking overhead by replacing some (possibly long) sequences of instructions by a very few dedicated ones.

[17] presents a masked ISE for the V-scale RISC-V core and order $d \in \{1, 2, 3, 4\}$ fixed at synthesis time. This solution is limited to small orders since the size of register file is multiplied by $d + 1$. It uses domain-oriented masking (DOM) [18], then users have to insert refresh operations.

[19] presents a masked ISE at order-1 for a RISC-V core with a SCA protection during memory accesses.

[20] presents Skiva an ISE for the LEON3 core (SPARC V8 instruction set). Skiva instructions accelerate masking at orders 1 or 3 and BS implementations. It also combines masking and fault detection protections.

[21] presents a masked ISE at order-1 for the SCARV RISC-V core using the DOM representation [18]. The register file is divided in 2 parts (one part for each share). This solution is limited to small orders (order-d would divide the register file into $d + 1$ parts).

[22] presents SME a masked ISE for the SCARV RISC-V core with order $d \in \{1, 2, 3\}$ fixed at synthesis time. It integrates the official RISC-V scalar cryptographic extension for AES [23].

III. PROPOSED SOLUTION

Previous masked ISEs have only been implemented for orders $d \leq 4$. Or the solution was designed for a single small order [19], or there is flexibility at synthesis time [17], [20], [21] but the size of the register file is multiplied by $d + 1$ (even more in some cases) leading to a huge area.

Below, we propose a hardware/software masking solution for much higher orders where a large flexibility is possible at design time *and* run time for various cost/performance trade-offs. A first masking level is performed in hardware in a masked ISE where the hardware order, denoted d_{H}, is fixed at synthesis time. We have FPGA implementations for d_{H} up to 31. Then a higher order can be obtained in software using simple and secure composition over the masked instructions. In practice, our hardware/software solution can reach multiple orders of the form:

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

where s is a software multiplicative factor combined over d_{H}. Finally, our ISE can be used for various cryptosystems.

A. Instruction Set Extension and Unit for Masking

Our ISE is composed of masked instructions adapted to the share slicing representation introduced in Sec. II-C. Tab. I presents our 4 masked instructions: AND, OR, NOT, XOR.

The original arithmetic and logic unit (ALU) of the CV32E40P cannot be used for masking since operations may lead to shares recombination as explained in [6].

TABLE I
SUMMARY OF THE 4 MASKED INSTRUCTIONS IN OUR ISE.

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

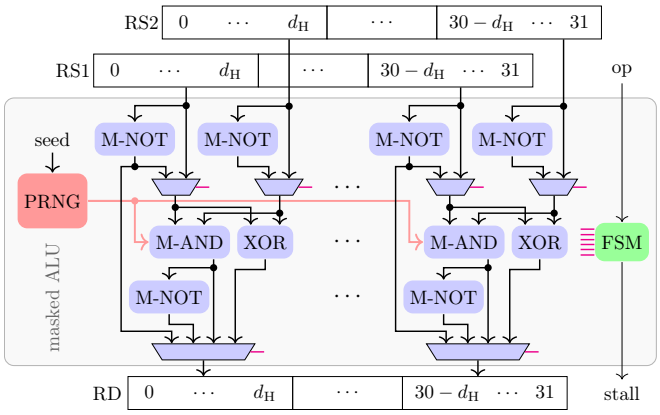

Fig. 1. Architecture of our masked ALU.

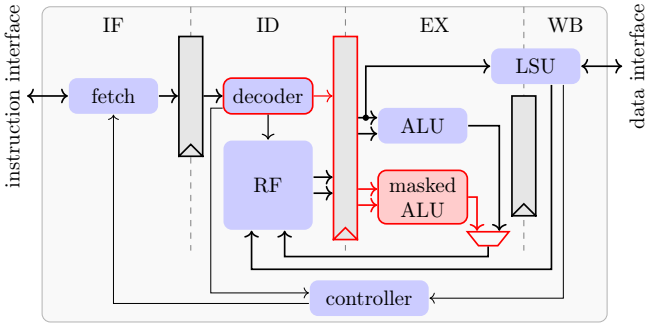

Fig. 2. Simplified schematic of the modified core with our protection. Red parts are added or modified for the masking ISE.

We designed the masked ALU described in Fig. 1. The source registers RS1 and RS2 (from the register file) are divided into several blocks where each block contains the $d_{\mathrm{H}}+1$ bits of a share. For instance, at order 3, there are 4 shares per variable and 8 blocks in a 32-bit register. The result of the masked ALU is written into the register file.

We use PINI gates, see [24], to maintain security by direct composition. The masked XOR is a standard XOR since it is linear in GF(2). For AND, OR and NOT gates, we used secure solutions from the literature. The masked AND uses the HPC3 gate from [10], it requires one clock cycle since a flip-flop is inserted to avoid glitch leakage. The masked NOT inverts one bit of the $d_{\mathrm{H}}+1$ bits in a share. The masked OR uses the masked AND where its operands and result pass through masked NOT gates according to de Morgan's laws.

A FSM controls the internal multiplexers of the masked ALU depending on the decoded operation. It also controls the pipeline for multi-cycle instructions.

A pseudo-random number generator (PRNG) provides random values for the shares and the refresh operations. It uses a single round of the Keccak permutation specified in [25]. The permutation parameters are the smallest to ensure the state size is large enough to provide enough random bits at each clock cycle. The PRNG is regularly re-seeded using a hardware true RNG (TRNG) outside of the core (not discussed in this paper, see [26] for instance).

B. Integration into the CV32E40P Core

As the CV32E40P core was designed to be modified and extended, the integration of our masked ISE was simple. The core was modified as depicted in Fig. 2. The modified instruction decoder deals with the instructions from Tab. I. To avoid shares recombination problems, we designed a masking datapath confined to the masked ALU. This requires to extended the pipeline register ID/EX with 2 new 32-bit registers only accessible from the masked ALU. The masked ALU presented above is added to the core. According to the decoded instruction, either the masking datapath or the normal datapath is activated.

C. FPGA Implementation Results

We implemented, on a Digilent Arty A7 FPGA board using Vivado 2019.2 from AMD-Xilinx, the modified core with our masked ISE for all hardware masking orders d_{H} from 1 up to 31 (d_{H} is fixed at synthesis time). An extract of the corresponding results is presented in Fig. 3 (on next page) for a few hardware orders (labeled HW-M-our where M stands for masked). The encryption time per AES block is obtained for all total (hardware and software) orders between 1 and 31. Fig. 3 includes software results for a masked bit-sliced version (SW-M-BS) on the CV32E40P without ISE.

The area overheads are limited. For instance with $d_{\mathrm{H}} = 7$, less than 48% more LUTs and 30% less flip-flops are required (many previous solutions require a register file d_{H} times larger). Other area results for $1 \leq d_{\mathrm{H}} \leq 31$ corroborate this trend in Fig. 5. The period obtained for all implemented d_{H} is close to the original one (the critical path is not in our ISE). Finally, left side of Fig. 3 illustrates the flexibility at design time and run time of our solution.

D. Software Usage of our Hardware Masking ISE

We use Usuba [12] to generate a BS implementation in C. The elementary operations are replaced by intrinsics for our masked instructions. The computation starts by translating the input blocks into share-slicing representation using shifts, logic operations and values from the PRNG. After all the computations in the masked domain, the output block is translated back to the standard representation using shifts and logic operations. We coded all these translations in C functions. The final C code is compiled.

The easy and secure composability in the software combination above the hardware masking comes from the PINI property. The total order obtained in hardware and software is given by Eq. 1 and leads to a high flexibility at design time and run time. For instance in an area constrained application, an order-1 hardware masking can be implemented in the circuit and used in software for various s leading to a total order in $\{1, 3, 5, 7, 9, 11, \ldots\}$. In a high security application, a larger order-5 circuit can be implemented and then used in software at a total order in $\{5, 11, 17, 23, 29\}$.

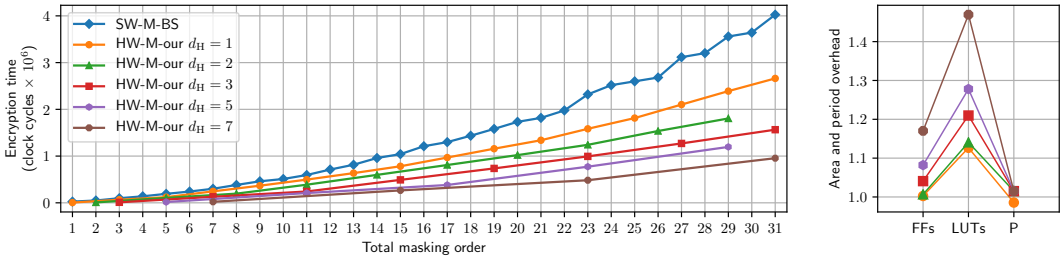

Fig. 3. Artix XC7A35 FPGA implementation results for our masked ISE at hardware masking orders $d_H \in \{1, 2, 3, 5, 7\}$ and various total orders (with software masking over the hardware one).

IV. COMPARISON TO PREVIOUS SOLUTIONS

As the cores and FPGAs used in previous works are different from ours, direct comparisons are not relevant. We have implemented several state of the art solutions and our solutions, all for AES, in the CV32E40P without ISE for software solutions (labeled SW-...) and with an ISE for hardware solutions (labeled HW-...). The second field in labels is M or U for masked or unmasked.

SW-U-BW is unmasked and byte-wise (each state byte is placed in a register). SW-U-BS is unmasked and bit-sliced generated by Usuba [12]. SW-M-BW is masked and byte-wise from [7]. SW-M-BS is masked and bit-sliced generated by Tornado [15]. SW-M-SS is masked and uses share-slicing from [16]. To compare with hardware solutions, we have also implemented two recent hardware masked ISEs in the CV32E40P: SME from [22] (HW-M-SME) and Skiva from [20] (HW-M-SKIVA). Our hardware solutions are labeled HW-M-our and only contains hardware masking ($d = d_H$ for d in $\{1, 2, 3, \dots 31\}$) for the evaluations below.

Fig. 4 presents timing comparisons for all these solutions using the same experimental setup. Our ISE (HW-M-our) outperforms masked bit-slicing solutions in software: a speedup of 9 at order 3, 13.8 at order 7, 23.6 at order 15 and 27.0 at order 23. Other masking solutions in software, SW-M-BW and SW-M-SS, are slower. Our results confirm that bit-slicing speeds up computations: SW-U-BS is faster than SW-U-BW. Regarding hardware solutions, our ISE is slightly faster than Skiva but slower than SME (since SME is optimized for AES).

Fig. 5 presents area results for all hardware solutions: HW-M-SKIVA, HW-M-SME and ours with d_H in $\{1, 2, 3, \dots 31\}$. Our solutions for order-1 and 3 have an area close to Skiva (but we are faster). Our solutions for order-1, 2 and 3 are much smaller than SME (but we are slower). It seems that SME would not fit into small FPGAs for high orders while our can. Our solution has a limited impact on area even for high hardware orders. The impact on frequencies of our solution is very small (the critical path is not in the ISE).

Tab. II shows that our implementation results of Skiva and SME are consistent with the results presented in the related papers. Our extension is faster than Skiva by a factor between 2 and 3. We are slower than SME (dedicated for AES), but SME will not scale up to high orders in area.

A strong interest in our solution is its hardware/software flexibility to reach much higher masking orders than previous solutions with a small silicon cost.

TABLE II
OVERHEAD COMPARISON OF HARDWARE MASKING ISEs FROM STATE OF THE ART AND OUR HARDWARE ONES.

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

V. CONCLUSION

We proposed a flexible masking solution in hardware and software. A masked instruction set extension was added to the CV32E40P RISC-V core where a first masking order is fixed at synthesis time. Our masked hardware solution offers much higher orders than previous ones in a limited silicon area. Our extension can then be used in software to reach higher orders using secure composability over the hardware masking. Our solution offers flexibility at design time, to meet various performance and budget constraints, and at run time to increase security over time.

We plan to add new masked instructions to reduce the number of independent blocks required in share-slicing solutions. Other future works include security evaluation using physical attacks and the optimization of the masked instruction set extension for AES and post-quantum cryptography.

ACKNOWLEDGMENT

This work was done in the LATERAL common laboratory between THALES and Lab-STICC. The authors are grateful to Nicolas Tranchant for his valuable feedback.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annual Cryptology Conference (CRYPTO)*. Springer, Aug. 1999, pp. 388–397.

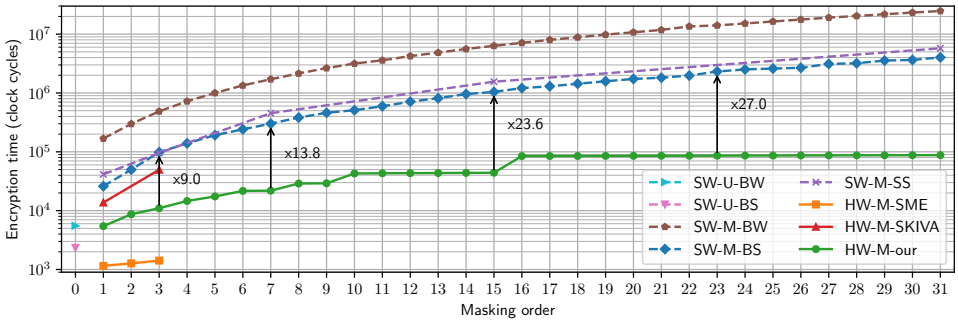

Fig. 4. Encryption times (log scale) for one AES block and various software and hardware masking solutions. For our solutions, the implemented hardware orders are $d_{\mathrm{H}} \in \{1, 2, 3, \ldots, 30, 31\}$ (there is no software masking over the hardware one to compare with Skiva and SME ISEs).

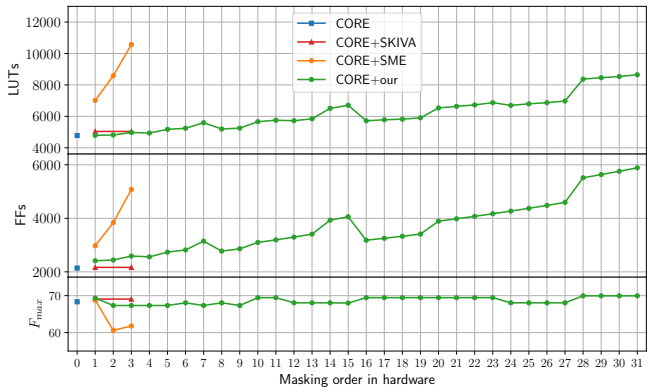

Fig. 5. Area/frequency results corresponding to timing results in Fig. 4.

[2] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Proc. Annual Cryptology Conference (CRYPTO)*. Springer, Aug. 1999, pp. 398–412.

[3] L. Goubin and J. 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.

[4] O. Bronchain and F.-X. Standaert, "Breaking masked implementations with many shares on 32-bit software platforms," *Transactions on CHES*, pp. 202–234, July 2021.

[5] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Proc. Annual Cryptology Conference (CRYPTO)*. Springer, Aug. 2003, pp. 463–481.

[6] S. Gao, B. Marshall, D. Page, and E. Oswald, "Share-slicing: Friend or foe?" *Transactions on CHES*, pp. 152–174, Nov. 2019.

[7] J.-S. Coron, E. Prouff, M. Rivain, and T. Roche, "Higher-order side channel security and mask refreshing," in *Proc. Fast Software Encryption (FSE)*. Springer, Mar. 2014, pp. 410–424.

[8] M. Rivain and E. 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.

[9] G. Cassiers and F.-X. Standaert, "Trivially and efficiently composing masked gadgets with probe isolating non-interference," *Transactions on Information Forensics and Security (TIFS)*, pp. 2542–2555, Feb. 2020.

[10] D. Knichel and A. Moradi, "Low-latency hardware private circuits," in *Proc. Conference on Computer and Communications Security (CCS)*. ACM, Nov. 2022, pp. 1799–1812.

[11] E. Biham, "A fast new DES implementation in software," in *Proc. Fast Software Encryption (FSE)*. Springer, Jan. 1997, pp. 260–272.

[12] D. Mercadier and P.-E. 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.

[13] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F.-X. Standaert, "On the cost of lazy engineering for masked software implementations," in *Proc. International Conference on Smart Card Research and Advanced Applications (CARDIS)*. Springer, Nov. 2014, pp. 64–81.

[14] D. Goudarzi and M. 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.

[15] S. Belaïd, P.-E. Dagand, D. Mercadier, M. Rivain, and R. Wintersdorff, "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.

[16] A. Journault and F.-X. 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.

[17] H. Gross, M. Jelinek, S. Mangard, T. Unterluggauer, and M. Werner, "Concealing secrets in embedded processors designs," in *Proc. International Conference on Smart Card Research and Advanced Applications (CARDIS)*. Springer, Nov. 2016, pp. 89–104.

[18] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," IACR Cryptology ePrint Archive, Nov. 2016. [Online]. Available: https://eprint.iacr.org/2016/486

[19] E. De Mulder, S. Gummalla, and M. Hutter, "Protecting RISC-V against side-channel attacks," in *Proc. Design Automation Conference (DAC)*. ACM, June 2019, pp. 1–4.

[20] P. Kiaei, D. Mercadier, P.-E. Dagand, K. Heydemann, and P. Schaumont, "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.

[21] S. Gao, J. Großschädl, B. Marshall, D. Page, T. Pham, and F. Regazzoni, "An instruction set extension to support software-based masking," *Transactions on CHES*, pp. 283–325, Aug. 2021.

[22] B. Marshall and D. Page, "SME: Scalable masking extensions," IACR Cryptology ePrint Archive, Oct. 2021. [Online]. Available: https://eprint.iacr.org/2021/1416

[23] B. Marshall, G. R. Newell, D. Page, M.-J. O. Saarinen, and C. Wolf, "The design of scalar AES instruction set extensions for RISC-V," *Transactions on CHES*, pp. 109–136, Dec. 2020.

[24] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: From trivial composition to full verification," *IEEE Transactions on Computers*, pp. 1677–1690, Sept. 2020.

[25] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Sponge-based pseudo-random number generators," in *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, Aug. 2010, pp. 33–47.

[26] B. Yang, V. Rožic, M. Grujic, N. Mentens, and I. Verbauwhede, "ES-TRNG: A high-throughput, low-area true random number generator based on edge sampling," *Transactions on CHES*, pp. 267–292, Aug. 2018.