SCARV: a side-channel hardened RISC-V platform

https://www.scarv.org https://github.com/scarv

Ben Marshall, **Daniel Page**, Thinh Pham, and James Webb
Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol BS8 1UB, United Kingdom.
csdsp@bristol.ac.uk

02/12/22

CERBERUS Security Laboratories

University of BRISTOL

THALES

- Recap:

$$\begin{array}{lcllll}
\text{SCARV} & \simeq & \text{RISC-V} & + & \text{cryptography} & \\
 & \simeq & \text{RISC-V} & + & \text{cryptographic engineering} & \\
 & \simeq & \text{RISC-V} & + & \text{implementation} & + \ \text{implementation attacks}
\end{array}$$

where

$$\begin{array}{lcl}
\text{WP-A} & \simeq & \text{a side-channel resistant RISC-V implementation} \\
\text{WP-B} & \simeq & \text{RISC-V support for next-generation cryptography} \\
\text{WP-C} & \simeq & \text{a democratised side-channel evaluation lab.}
\end{array}$$

- Generic activities:

  1. **publications**:
     - “*Towards Micro-Architectural Leakage Simulators: Reverse Engineering Micro-Architectural Leakage Features is Practical*” [10] (EUROCRYPT’22).
     - “*HYDRA: a multi-core RISC-V with cryptographically useful modes of operation*” [14] (CARRV’22).
     - “*RISC-V Instruction Set Extensions for Lightweight Symmetric Cryptography*” [5, 6] (NIST LWC workshop, TCHES’23.1).
  2. **events**:
     - “*MIRACLE: MIcRo-ArChitectural Leakage Evaluation*” [13] (TCHES’22.1): best paper award.
     - *Constructive Side-Channel Analysis and Secure Design (COSADE’22)*: invited talk.
     - *Smart Card Research and Advanced Application Conference (CARDIS’22)*: invited talk.
     - *Topics in hArdware SEcurity and RISC-V (TASER’22)* workshop (Sept.’22; $\sim$ 160 registrations, $>$ 100 participants).

https://ches.iacr.org/2022/taser

- Specific activities: ISEs for NIST LWC candidates.

Definition

NIST define [15, Section 1]

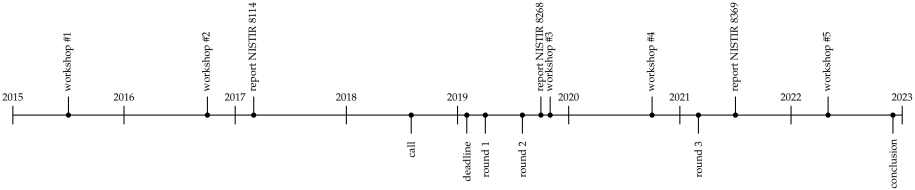
$$\text{lightweight cryptography} \;\simeq\; \text{“tailored for resource-constrained devices”}$$

e.g.,

- efficient on constrained hard/software platforms (vs. existing standards),
- efficient for short messages,
- amenable to countermeasures against implementation attacks,
- ...

with “efficient” read as low-latency, low-footprint, low-power, etc.

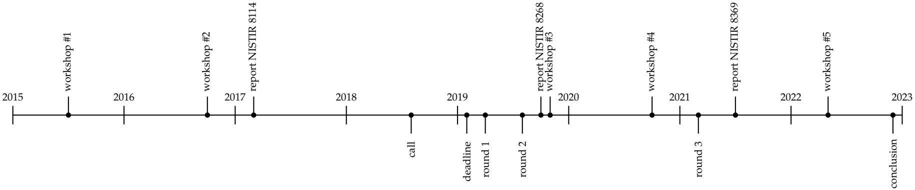
- Specific activities: ISEs for NIST LWC candidates.



  - 57 submissions,
  - 56 selected as round 1 candidates,
  - 32 selected as round 2 candidates,
  - 10 selected as round 3 candidates, i.e., finalists:

Name	Specification	AEAD	Hash	Component(s)
Grain128-AEAD	[12]	✓		Stream cipher
GIFT-COFB	[1]	✓		Block cipher
Romulus	[11]	✓	✓	(Tweakable) Block cipher
Ascon	[8]	✓	✓	Permutation
Elephant	[4]	✓		Permutation
PHOTON-Beetle	[2]	✓	✓	Permutation
Schwaemm and Esch	[3]	✓	✓	Permutation
Xoodyak	[7]	✓	✓	Permutation
ISAP	[9]	✓		Permutation
TinyJAMBU	[16]	✓		(Keyed) Permutation

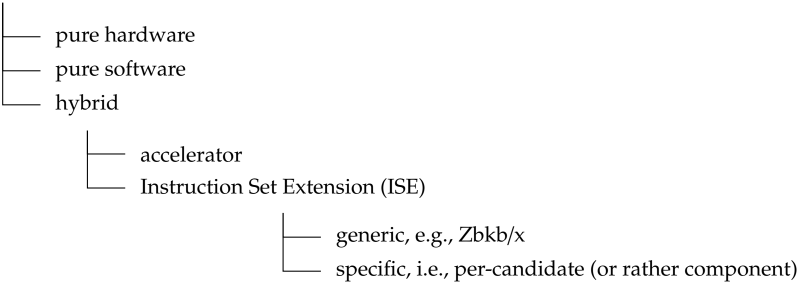
- Specific activities: ISEs for NIST LWC candidates.



- 57 submissions,
- 56 selected as round 1 candidates,
- 32 selected as round 2 candidates,
- 10 selected as round 3 candidates, i.e., finalists:

Name	Specification	AEAD	Hash	Component(s)
Grain128-AEAD	[12]	✓		L/NFSRs
GIFT-COFB	[1]	✓		GIFT-128
Romulus	[11]	✓	✓	Skinny-128-384+
Ascon	[8]	✓	✓	Ascon-p
Elephant	[4]	✓		Spongent-$\pi[n]$ or Keccak-$f[m]$
PHOTON-Beetle	[2]	✓	✓	$\texttt{PHOTON}_{256}$
Schwaemm and Esch	[3]	✓	✓	Sparkle (inc. Alzette ARX-box)
Xoodyak	[7]	✓	✓	Xoodoo
ISAP	[9]	✓		Ascon-p or Keccak-$f[m]$
TinyJAMBU	[16]	✓		P_n (inc. LFSR)

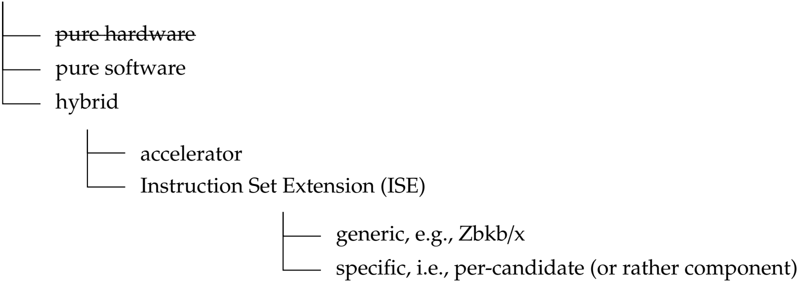
- Specific activities: ISEs for NIST LWC candidates.
  - options:



- pure hardware
- pure software
- hybrid
  - accelerator
  - Instruction Set Extension (ISE)
    - generic, e.g., Zbkb/x
    - specific, i.e., per-candidate (or rather component)

  - scope:
    - consider RV32 *and* RV64; focus on Rocket-based hardware
    - ignore hash API
    - ignore implementation attacks, bar data-independent latency
    - use “partial” component (e.g., only encryption) where possible
  - criteria:
    - strictly 3-address (i.e., 2-input, 1-output) instructions
    - disallow additional state (e.g., CSRs)

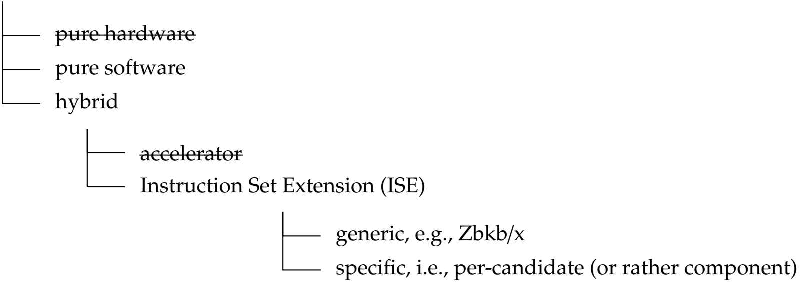
- Specific activities: ISEs for NIST LWC candidates.
  - options:



- ~~pure hardware~~
- pure software
- hybrid
  - accelerator
  - Instruction Set Extension (ISE)
    - generic, e.g., Zbkb/x
    - specific, i.e., per-candidate (or rather component)

  - scope:
    - consider RV32 *and* RV64; focus on Rocket-based hardware
    - ignore hash API
    - ignore implementation attacks, bar data-independent latency
    - use “partial” component (e.g., only encryption) where possible
  - criteria:
    - strictly 3-address (i.e., 2-input, 1-output) instructions
    - disallow additional state (e.g., CSRs)

- Specific activities: ISEs for NIST LWC candidates.
  - options:



~~pure hardware~~

pure software

hybrid

~~accelerator~~

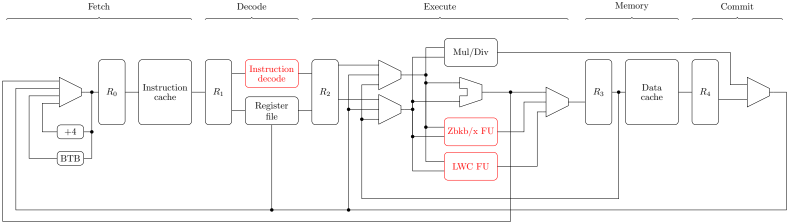
Instruction Set Extension (ISE)

generic, e.g., Zbkb/x

specific, i.e., per-candidate (or rather component)

  - scope:
    - consider RV32 *and* RV64; focus on Rocket-based hardware
    - ignore hash API
    - ignore implementation attacks, bar data-independent latency
    - use “partial” component (e.g., only encryption) where possible
  - criteria:
    - strictly 3-address (i.e., 2-input, 1-output) instructions
    - disallow additional state (e.g., CSRs)

- Specific activities: ISEs for NIST LWC candidates.
  - strategy: for each candidate
    - analysis, then on-paper ISE design
    - software implementation using stock GCC-based tool-chain plus `.insn`
    - simulate using (patched) Spike
    - hardware implementation using Rocket



  - results (see [5, 6] for detail):
    - ISEs reduce software latency + footprint, at cost of some hardware overhead
    - ISEs reduce "gap" between hardware- and software-oriented candidates
    - Zbkb/x already makes significant impact
    - only ISE-assisted Schwaemm improves on ISE-assisted AES-GCM re. latency
    - ...

https://github.com/scarv/lwise

Questions?

References

[1] S. Banik et al. GIFT-COFB. Submission to NIST (version 1.1). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/gift-cofb-spec-final.pdf. 2021 (see pp. 5, 6).

[2] Z. Bao et al. PHOTON-Beetle. Submission to NIST. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/photon-beetle-spec-final.pdf. 2021 (see pp. 5, 6).

[3] C. Beierle et al. *Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the* SPARKLE *Permutation Family*. Submission to NIST (version 1.2). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/sparkle-spec-final.pdf. 2021 (see pp. 5, 6).

[4] T. Beyne et al. *Elephant*. Submission to NIST (version 2.0). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/elephant-spec-final.pdf. 2021 (see pp. 5, 6).

[5] H. Cheng et al. "RISC-V Instruction Set Extensions for Lightweight Symmetric Cryptography". In: *5th NIST Lightweight Cryptography Workshop*. 2022 (see pp. 3, 10).

[6] H. Cheng et al. "RISC-V Instruction Set Extensions for Lightweight Symmetric Cryptography". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)* 2023.1 (2022), pp. 193–237. URL: https://doi.org/10.46586/tches.v2023.i1.193-237 (see pp. 3, 10).

[7] J. Daemen et al. *Xoodyak, a lightweight cryptographic scheme*. Submission to NIST (version 2.0). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/xoodyak-spec-final.pdf. 2021 (see pp. 5, 6).

[8] C. Dobraunig et al. *Ascon*. Submission to NIST (version 1.2). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf. 2021 (see pp. 5, 6).

[9] C. Dobraunig et al. *ISAP*. Submission to NIST (version 2.0). https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/isap-spec-final.pdf. 2021 (see pp. 5, 6).

References

[10] S. Gao, E. Oswald, and D. Page. "Towards Micro-Architectural Leakage Simulators: Reverse Engineering Micro-Architectural Leakage Features is Practical". In: *Theory and Application of Cryptographic Techniques (EUROCRYPT)*. Ed. by O. Dunkelman and S. Dziembowski. LNCS 13277. `https://doi.org/10.1007/978-3-031-07082-2_11`. Springer-Verlag, 2022, pp. 284–311 (see p. 3).

[11] C. Guo et al. Romulus. Submission to NIST (version 1.3). `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/romulus-spec-final.pdf`. 2021 (see pp. 5, 6).

[12] M. Hell et al. `Grain-128AEADv2`. Submission to NIST (version 2.0). `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/grain-128aead-spec-final.pdf`. 2021 (see pp. 5, 6).

[13] B. Marshall, D. Page, and J. Webb. "MIRACLE: MIcRo-ArChitectural Leakage Evaluation". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)* 2022.1 (2022). `https://doi.org/10.46586/tches.v2022.i1.175-220`, pp. 175–220 (see p. 3).

[14] B. Marshall et al. "HYDRA: a multi-core RISC-V with cryptographically useful modes of operation". In: *6th Workshop on Computer Architecture Research with RISC-V (CARRV)*. 2022 (see p. 3).

[15] *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process.* `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf`. 2018 (see p. 4).

[16] H. Wu and T. Huang. *TinyJAMBU.* Submission to NIST (version 2.0). `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/tinyjambu-spec-final.pdf`. 2021 (see pp. 5, 6).