RISC-V Scalar Crypto

Markku-Juhani O. Saarinen
PQShield Ltd., Oxford, UK
mjos@pqshield.com

Ben Marshall
University of Bristol, UK
ben.marshall@bristol.ac.uk

RWC 2021 – Real World Cryptography Symposium, 13 Jan 2021

RISC-V: Open Source ISA

RISC-V is a *free* and *open* Instruction Set Architecture (ISA), whose development is organized by RISC-V International (`https://riscv.org/`). Industrial and academic support is very broad[1]; Silicon and IP cores are available from multiple vendors, etc.

[1] With the notable exception of Intel® and ARM®, owners of proprietary ISAs !

The K extension (from κρυπτός)

Anyone can create custom (“X”) ISA extensions for RISC-V; but for interoperability (compilers, kernel, ..) we need standard extensions. Krypto is proposed as **“K” / Zk**.

Scalar RV32/RV64 first, Vector later

- **RV32K**: Lightweight 32-bit microcontroller / security controller cryptography. *(Comparable to ARMv7/M, Xtensa. Example: ESP32-C3 MCU has an RV32IMC).*
- **RV64K**: General-purpose 64-bit “application” processors, often running Linux. *(Comparable to AMD64/Intel, ARMv8/A. Example: PolarFire SoC has an RV64GC).*
- **RVVK**: RISC-V Vector architecture. RVV resembles ARM’s new Scalable Vector Extension (SVE) more than SIMD extensions like NEON or Intel’s AVX2/512.

RISC-V is popular in embedded / IoT, which needs crypto, but may not have vector.

In 2019 it was decided to add “scalar” (non-vector) crypto as well; only ISA with it!

Look under the releases tab at: `https://github.com/riscv/riscv-crypto`

RISC-V Cryptographic Extension Proposals
Volume I: Scalar & Entropy Source Instructions

Editor: Ben Marshall
ben.marshall@bristol.ac.uk

Version 0.8.1 (December 18, 2020)

`https://github.com/riscv/riscv-crypto/`
`git:master @ bffeed3bf3d9cc306a9d33753f22a8af4d6b8a90`

Contributors to all versions of the spec in alphabetical order (please contact editors to suggest corrections):
Alexander Zeh, Andy Glew, Barry Spinney, Ben Marshall, Daniel Page, Derek Atkins, Ken Dockser, Markku-Juhani O. Saarinen, Nathan Menhorn, Richard Newell, Claire Wolf

Scalar Crypto is in Opcode and Consistency Review; Stable/Freeze in Q1/2021.

Lots of Quantitive Considerations

Problem: How to keep the crypto extension generally useful?

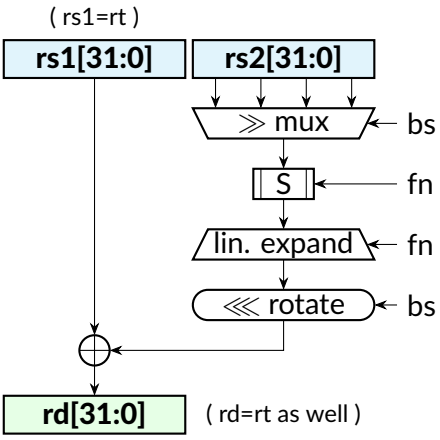
Solution: Require a subset of Bitmanip ("B" extension) instructions.

- Rotations! (Not part of base RV32I/RV64I) – for SHA3, ChaCha, LWC.
- Carryless multiply for GCM (and binary fields; e.g. code-based crypto).
- Specific uses for `grev` (byte & bit reversal), `andn` (SHA3), `shfli`/`unshfli`, etc.

Special instructions only for standard things that need them the most:

- "NIST Suite" (Zkn): AES, SHA2 and "ShangMi Suite" (Zks): SM4, SM3.
- Entropy Source (Zkr) for 800-90B (FIPS 140-3) and AIS-31 (CC) RNGs.
- RISC; Test things in microbenchmarks, compilers, do instruction counts, etc.
  (Design rationale is open, published as academic research – but industry-driven.)

RV32: T-Tables in hardware approach – requires only one SBox.




```
aes32esmi rt, rs2, bs // rt=rd=rs1
aes32esi  rt, rs2, bs // Last rd.
aes32dsmi rt, rs2, bs // Decrypt
aes32dsi  rt, rs2, bs // Last rd.
sm4ks     rt, rs2, bs // SM4 key
sm4ed     rt, rs2, bs // Enc & Dec
```

- “Destructive” `rt=rd=rs1` encoding.
- 16 cycles per rd. + key load (4×`ld`).
  ≈ 5× faster than tables + XORs.
- Constant time ⋙ bitsliced speed.
- 1K..2K NAND2 Gates. (AES = LWC!)
- Same data path can be used for SM4.

RV64: Use wider registers to process 128-bit block over two instructions.

Six instructions per AES round:

- AddRoundKey: 2× `ld`, 2× `xor`
- 2× `aes64esm` or `aes64dsm` / decrypt.
  Final round: 2× `aes64es` or `aes64ds`.

Engineering options:

- 1 SBox Instance (Small, 8-cycles)
- 8 SBox Instances (Fast, 1-cycle)
  ≈ 8K GE for single-cycle variant
  ≈ 6K GE for 2-cycle variant.

```
// Encrypt Round
aes64es     rd, rs1, rs2
aes64esm    rd, rs1, rs2

// Decrypt Round
aes64ds     rd, rs1, rs2
aes64dsm    rd, rs1, rs2

// Key Schedule
aes64ks1i   rd, rs1, rcon
aes64ks2    rd, rs1, rs2
aes64imix   rd, rs1
```

(For more on RISC-V AES, see our CHES 2021 paper: `https://ia.cr/2020/930`)

Lightweight SHA2 Support

Directly map onto “sigma” and “sum” functions of FIPS 180-4.

Number of instructions:
RV32: 4 / SHA256, 6 / SHA512
RV64: 4 / SHA256, 4 / SHA512

Very Cheap:
Just XORs & shifts / rotations.

2× Performance, 0.5× code size.

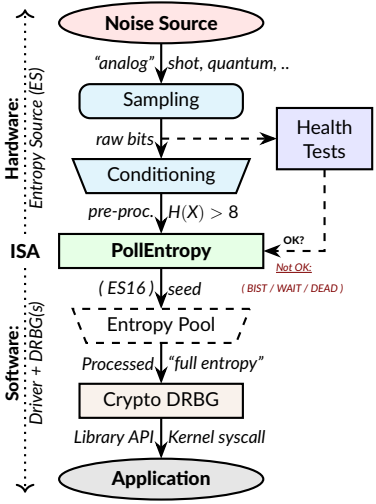
SM3 *has a similar 32-bit data path to SHA2-256. Instructions* `sm3p0` *and* `sm3p1` *offer some speed-up.*

```
sha256sig0  rd, rs1 // RV32 & RV64
sha256sig1  rd, rs1 // (both)
sha256sum0  rd, rs1
sha256sum1  rd, rs1

sha512sig0  rd, rs1 // RV64
sha512sig1  rd, rs1 // (only)
sha512sum0  rd, rs1
sha512sum1  rd, rs1

sha512sig0h rd, rs1, rs2 // RV32
sha512sig0l rd, rs1, rs2 // (only)
sha512sig1h rd, rs1, rs2
sha512sig1l rd, rs1, rs2
sha512sum0r rd, rs1, rs2
sha512sum1r rd, rs1, rs2
```

(On-Chip) Entropy Source Interface



- ISA defines an (AIS-31 PTG.2 or SP 800-90B) **"Entropy Source"**, not a "black box RNG."
- We already have AES and hash instructions; these can be used to generate DRBG output – once initialized with a sufficient entropy.
- We've consulted with certification labs, also BSI and NIST directly about requirements.

```
pollentropy rd  // Poll randomness
getnoise rd     // Testing interface
```

Polls atomically some entropy (`ES16`) or a status: `BIST` / `WAIT` / `DEAD`. Use of raw (certification) test override `getnoise` disables `pollentropy`.

10 minutes is up. But how do I ..

.. do RSA and ECC arithmetic? I do it with Redundant Binary Representation (RBR). RISC-V still has no carry flag – but RBR is constant time, allows easier vectorization.

.. do Camellia, Aria, Magma, Kuznyechik? Instruction `xperm.b` helps with constant time S-Boxes. It's also useful for masking-based side-channel countermeasures.

.. do constant time? Use crypto extensions, avoid table lookups, branches. Krypto defines encoding rules for constant-time multiply. A compiler flag should activate it.

.. show "non-invasive" security? With an oscilloscope or leakage model (emulator). These depend on physical RISC-V *implementation*, not within the scope of ISA.

.. do post-quantum cryptography? I've tested all NIST PQC finalists on RISC-V. No big surprises about performance or suitability; they're fine. Good SHA3 helps. (However, RISC-V offers unique options for hardware security and acceleration.)

Thank You! Q&A To follow.

https://github.com/riscv/riscv-crypto

B. Marshall, G. R. Newell, D. Page, M.-J. Saarinen, C. Wolf:
"The design of scalar AES Instruction Set Extensions for RISC-V." – ia.cr/2020/930
(T)CHES 2021: https://doi.org/10.46586/tches.v2021.i1.109-136

M.-J. Saarinen, G. R. Newell, B. Marshall:
"Building a Modern TRNG: An Entropy Source Interface for RISC-V." – ia.cr/2020/866
Proc. ASHES '20: https://doi.org/10.1145/3411504.3421212