Scalar Crypto Standardization Status Summary

Status at a glance:

- **Current Definition-of-Done Status**: Ratified
- We finished public review on Oct 17th, 2021
- See here for a summary of public review feedback.
- Scalar Crypto Standardization have been Ratified
- **Next Definition of Done Status**: Ratification
- Definition of Done Status Top Sheet

Scalar Crypto Specification:

- Latest Draft Scalar Crypto Specification (v1.0.0)
  - Contains feedback from public review.

Lightweight instruction set extensions for RV32 and RV64 HARTs. Proposed extensions:

- Extensions fully defined in the Scalar Crypto Specification: Zk, Zkn, Zks, Zkr, Zksne, Zksnd, Zksnh, Zksed, Zksh, Zkt
- Shared with the Bit-Manipulation Specification: Zbxx, Zbkc, Zbkb
- Instruction Group Names Diagram (click thumbnail for full-size image)

Pub

Architecture & Opcode consistency review

- The results of the review can be seen here. Scalar Crypto Arch Review Results.pdf
- After discussion within the task group, the following changes will be made to the spec, with the changes expected to land in 0.9.4 before the end of August:
  - Stop overlapping the aes32 and aes64 opcodes. See riscv/riscv-opcodes#77
  - Clarify the effect of misa.k
  - Some updates to the entropy source and zkt detailed here.
  - Miscellaneous editorial issues.
- Some downstream tasks for our group development partners now we are in public review.

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>Status</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLCT</td>
<td>GCC</td>
<td>Not started</td>
<td>Update extension names: Zkb Zbkb etc.</td>
</tr>
<tr>
<td>PLCT</td>
<td>GCC</td>
<td>Not started</td>
<td>Update aes32/64* encodings if applicable.</td>
</tr>
<tr>
<td>PLCT</td>
<td>LLVM</td>
<td>Done</td>
<td>Update extension names: Zkb Zbkb etc.</td>
</tr>
<tr>
<td>PLCT</td>
<td>LLVM</td>
<td>Done</td>
<td>Update aes32/64* encodings if applicable.</td>
</tr>
<tr>
<td>IIT</td>
<td>riscv-config</td>
<td>Not started</td>
<td>Update extension names: Zkb Zbkb etc.</td>
</tr>
<tr>
<td>IIT</td>
<td>riscv-config</td>
<td>Not started</td>
<td>Update instruction inclusion in different extensions</td>
</tr>
<tr>
<td>IIT</td>
<td>riscv-arch-tests</td>
<td>Not started</td>
<td>Break up K suite into Zk* extensions</td>
</tr>
<tr>
<td>IIT</td>
<td>riscv-arch-tests</td>
<td>Not started</td>
<td>Update instruction inclusion in different extensions</td>
</tr>
</tbody>
</table>

Architecture Tests

- Test plan for the scalar-crypto specific instructions is available.
- Imperas have a complete set of tests, written to the existing test plan, for the scalar crypto instructions and the bitmanip instructions we borrow.
  - These have been merged into the main test suite as of PR#177, with many thanks to Imperas for the contribution.
  - Spike, OVPSim and Sail all agree on the test signatures.
  - They form a base we can use to develop prototype implementations / Spike / SAIL / QEMU very easily and quickly.
- Upstream Spike support for enabling it to work with the K test suite is being added in PR#687.
- IIT Madras are also looking at writing the scalar crypto tests for integration into the official architectural tests repo as well.
  - Agreed SoW for IITM
  - They are re-implementing the tests as part of the blessed coverage and test generation tooling.
Making good progress with the simple test patterns for scalar-crypto specific instructions A/O April 7'th '21

- We then switch over to using the IIT tests when they are finished, since they will be easier to maintain/extend going forward than the Imperas tests.
- YAML config changes for K have been merged in. See here.
- Update from IIT Madras as on 25-Oct
  - Sample cgf file for Kspec ver 1.0.0-rc4 is in https://github.com/anku-anand/riscv-ctg/tree/kspec-rc4

Status from IIT Madras as on 17-Jun:
- All test cases and coverage reports has been generated and presented.
- If there are any changes in future on these that is required in future, IIT Madras will enhance the scripts as per requirements.

Status from IIT Madras as on 20-May:
- Real world test cases as per the test plan has been generated.
- Currently waiting for the fixed toolchain with K extension from PLCT to test the generated test cases. All the test cases are working fine when we run against the patched toolchain
- A PR has been raised with a pull request for this suite to be reviewed and merged in the riscv-arch tests github repo.

Status from IIT Madras as on 12-May:
- Coverage report for all developed cases in CTG/ISAC has been generated and it is reported as 100%
- Currently real world test cases are being developed as per test plan and will be completed and send for review by beginning of next week

Status from IIT Madras as on 05-May:
- Resolved issues in running the rv64ik toolchain after interacting with PLCT and compile the relevant tests generated from CTG and run them on spike
- Currently resolving issues in the running the rv64ibk toolchain. Once this is done, will generate the coverage report of the test cases built till now and share with team.

Status from IIT Madras as on 26-Apr:
- Completed the coverage points specification for all 32-bit and 64-bit instructions
- Generated test cases from the coverage points
- Currently working on trying to install the scalar crypto enabled toolchain.

**Compilers / Toolchains**

Imperas maintain pre-built toolchains for various in-progress RISC-V extensions here. See the "rvk-*" branches for scalar crypto.

**GCC and Binutils**

- Experimental / development toolchain available in the riscv-crypto repository.
- This cannot be up-streamed, but can be used for development work for now.
- **Intrinsics proposal from Markku**
- PLCT lab have developed complete Binutils and GCC patches.
  - The pull requests into the main RISC-V repos can be found here:
    - Binutils pull request: https://github.com/riscv/riscv-binutils-gdb/pull/254
    - GCC pull request: https://github.com/riscv/riscv-gcc/pull/250 (merged)
- The PLCT lab continuous integration server can be found here:
  - https://ci.rvperf.org/view/Krypto-Scalar/
  - This is a good place to start for re-producing the PLCT builds of GCC/Binutils
  - Some small changes will be needed as we move to v1.0 of scalar crypto around encodings.

**LLVM**

- Work will be done by PLTC lab under the group contributor model.
  - Github repository
  - Continuous integration status
  - Slides from PLCT Update Weds 10'th Feb
  - As of 21'st April ’21, LLVM work is mostly complete, waiting on PLCT lab for an update about merging things upstream.
  - Some small changes will be needed as we move to v1.0 of scalar crypto around encodings.
  - As of 24'st Jan ’22, LLVM upstream is able to compile the assembly code of Zk* and Zbk*

**Simulators**

Though all listed under "simulators", these are actually a collection of formal model / virtual machine / architectural simulators / DV simulators etc.

**SAIL**

- Currently working on getting support merged in upstream in PR#99
  - Support for all scalar-crypto dedicated instructions and the entropy source is present.
  - No support for Bitmanip yet.

**Spike**

- Upstream support has been merged in.
- Instructions are up to date with v1.0.0

**riscvOVPSimPlus**

- Imperas Commercial Simulator
Freeware version
Supports:
- Crypto-scalar v0.7.2, v0.8.1, 0.9.0, 0.9.2, 1.0.0-rc1, + Bitmanip subsets
- Bitmanip 0.90, 0.91, 0.92, 0.93-draft, 0.93, 0.94, 1.0.0
- Functional coverage collection.

QEMU
- Work will be done by PLTC lab under the group contributor model.
  - Github repository
  - Continuous integration status
  - Waiting to hear back from PLCT on status.

Proof-of-Concept implementations

Hardware

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Base Architecture</th>
<th>Level of Implementation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand-alone functional units</td>
<td>RV32/64</td>
<td>Yosys Synthesis</td>
<td>Stand-alone functional-unit style implementations of the dedicated scalar crypto instructions. Free to use as &quot;drop-ins&quot; for prototyping.</td>
</tr>
<tr>
<td>scav-cpu</td>
<td>RV32</td>
<td>Behavioural RTL simulation / Yosys Synthesis / FPGA</td>
<td>Completely Public/Open Source. Useful as a public baseline. Commercial implementations should aim to be better than this!</td>
</tr>
<tr>
<td>PQShield security core</td>
<td>RV32</td>
<td>(assumed) Behavioural RTL simulation. Running on FPGA.</td>
<td>Closed / commercial source - PQShield.</td>
</tr>
<tr>
<td>Minidice TRNG</td>
<td>N/A</td>
<td>FPGA Implementation</td>
<td>Closed / commercial source - PQShield. Complete implementation of the RISC-V entropy source.</td>
</tr>
<tr>
<td>Romain Dolbeau / VexRISC-V</td>
<td>RV32</td>
<td>Running on FPGA.</td>
<td>Uses VexRiscv core as a base. Completely independent implementation from scratch, outside the Crypto TG.</td>
</tr>
<tr>
<td>IQonIC Works RV32IC_P5</td>
<td>RV32</td>
<td>In development</td>
<td>&quot;Implemented Zkn (..), along with selectable Zb* and Zkb. We also have an optional custom extension that does AES block encrypt/decrypt, and a bus-based AES/cipher-mode accelerator. Work in progress benchmarking them on FPGA to compare relative performance in accelerating crypto library functions.&quot;</td>
</tr>
<tr>
<td>croyde-riscv</td>
<td>RV64</td>
<td>Behavioural RTL simulation / Yosys Synthesis / FPGA</td>
<td>3-stage RV64 micro-controller. Free/Open source. Something commercial implementations should better. Implements everything except ZK3.</td>
</tr>
</tbody>
</table>

⚠️ We still need RV64 implementations. ⚠️
Barry Spinney has offered to do advanced node synthesis runs for open source implementations.
- I (Ben) intend to take him up on this when I get time. No idea when that will be.

Software

<table>
<thead>
<tr>
<th>Project/Maintainer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romain Dolbeau</td>
<td>Independent implementations of various important ciphers + modes of operation.</td>
</tr>
<tr>
<td>rvkrypto-fips / Markku</td>
<td>&quot;FIPS 140-3 and higher-level algorithm Tests for RISC-V Crypto Extension&quot;</td>
</tr>
<tr>
<td>riscv-crypto benchmarks</td>
<td>Initial benchmarks used to develop the scalar crypto extension.</td>
</tr>
</tbody>
</table>

ABI Extensions
- None required