RISC-V Scalar Cryptography Public Review Summary

Table of Contents

1. Introduction

This document summarises the feedback received during the public review period for the Scalar Cryptography extensions, and any outcomes from them.

The public review period for Scalar Cryptography ran from September 2'nd 2021 to October 17'th, 2021.

Feedback was received via two main sources:

- The RISC-V isa-dev mailing list.
- Github issues in our repository. Any which related to public review were labelled as such.

2. isa-dev Feedback

The initial announcement to the list of the public review period can be found here.

The following gives a short overview of all threads related to Scalar Crypto at the time of writing.

2.1. AES Key expansion

- Thread available here.
- A question about how to use the AES Key expansion instructions. An answer was provided and no follow up questions were received. We assume the answer was sufficient.

2.2. aes_rnum_to_rcon function

- Thread available here.
- A typo in the specification Sail code was pointed out, where a function call was using the wrong name, but the intended function was still obvious. This was fixed in this commit.

2.3. Comments on Scalar Crypto v1.0.0-rc2

- Thread available here.
- Pointed out lots of editorial issues and was otherwise very positive about the specification as a whole.
- Raised some much broader questions about how the Sail model code can be best included in specifications, which triggered some further discussion.

2.4. Zkr comments

- Email link on isa-dev
- These comments raised some concerns about how the Zkr extension interacts with the Hypervisor. Some concerns were about terminology, and some were functional concerns about the kinds of traps raised and when.
- These were addressed in Pull Request #134.

3. Github issues feedback

All Github issues raised in response to public review can be found here. The following is a short summary of their content and any conclusions.

3.1. Zkt: explicitely consider timing dependences across several instructions

- Issue thread
- Asks for clarification on how the `Zkt` extension handles execution latency differences arrising from *sequences* of instructions, rather than individual instructions.
- Recommendation: Add clarifying words to note that inter-instruction optimisations are *out of scope* for `Zkt`, and that it applies only to intra-instruction latency.
- This recommendation was merged in PR#137.

3.2. Reconsider destructive encoding form for AES instructions

- Issue thread
- This is a proposal to re-consider the encodings for the 32-bit AES and 32/64-bit SM4 instructions. Specifically, to change them back to having a destructive form where `rd=rs1`, thus saving encoding space.
- There has been substantial past discussion about this within the task group and on Github (links in the issue thread). At various times in the last two years, we (the task group) have changed the encodings from non-destructive to destructive and back again, based on updated information from various people about the percieved scarcity of opcode space.
- Our last feedback on this issue was from the architectural review committee, who told us that the opcode space saving was not worth introducing the destructive encoding form to the architecture at this time. Hence our decision to enter public review with the non-destructive encodings.
- Given this would be a major, change to the frozen specification, we (the task group) believe it is for the architecture review committee to consider 1) whether revisiting the discussion is wise given the amount of past discussion and 2) whether making the change is worth it.

NOTE	This issue is distinct from one described later about optimising the AES32 encodings to better fit with some choices made by the Bitmanip task group.

3.3. Encoding section minor beautification

- Issue thread
- Notes some opportunities to improve the typography of instruction encoding diagrams. This will likely be done when the spec is merged into the main architecture manual.

3.4. name consistency of brev8/rev8 and xperm8/xperm4

- Issue thread
- Asked about the inconsistency of mnemonic use between the normative specification and the appendix.
- The inconsistency was fixed in this commit.
- Another person also noted further down in the thread that some mnemonic names were inconsistent with the wider architecture. The TGs reply was to say these names were chosen for us during architecture review, and that any changes now would need to be very strongly motivated due to the frozen status of the specification.

3.5. Scalar Cryptography v1.0.0-rc2. Incorrect ZIP/UNZIP insns encoding

- Issue thread
- To date this is the only actual *bug* which has been discovered during public review. The encodings for the `zip` and `unzip` instructions were found to be incorrect, and were fixed in RC4 of the specification.
- This issue caused more trouble than it needed to, due to confusion about Zip and Unzip being "swapped", which turned out not to be the case.

3.6. Does SMx algorithmes require clmuls?

- Issue thread
- A question about why `SM*` instructions appeared with carry-less multiply instructions in some extensions. Answer points out that SM4 is commonly used in the "GCM" mode of operation, which uses carry-less multiply for efficient implementations.

3.7. rv32 sha512sigDX wrong order

- Issue thread
- It was pointed out that the code examples for some SHA512 instructions were incorrect.
- This was fixed in this commit.

3.8. Minor suggestion to the code modification for aes64ksi

- Issue thread
- A small recommendation for improving the clarity of Sail code for the `aes64ks1` instruction.
- Suggestion implemented in this commit.

3.9. Minor typographical error while referring to Zkt instructions

- Issue thread
- Tiny editorial correction.

3.10. Zero and sign extension of pack.h vs pack.w

- Issue thread
- A suggestion for clarifying the zero and sign-extension of the `pack*` instructions. Also an issue for the Bitmanip TG. No action taken yet.

3.11. Scalar Cryptography v1.0.0-rc1. Opcodes changing of aes32* instructions.

- Issue thread
- A question about optimising the encodings (particularly of the `aes32`) instructions to better fit with an as yet un-standardised choice by the Bitmanip task group for easily detecting ternary instructions.
- See also:
  - Ternary instructions encoding policy on the isa-dev list.
  - Ternary instructions must die? issue raised against the P extension but referencing choices in the Cryptography TG.
- TG response has been to say that we aren't against more optimal encodings, but that the decision is down to the architecture review committee, who set a high bar for changes at this stage.

NOTE	This issue is distinct from one described earlier about changing the AES32 encodings to use a destructive form.

3.12. Scalar Cryptography v1.0.0-rc1 style issues

- Issue thread
- Miscellaneous editorial issues. Fixed with other editorial issues.

4. Conclusions

The specification and extensions seem to have been well received. Many improvements have been made to the clarity of the specification, and several fixes applied to example code. One major bug was identified in the encodings, which was promptly fixed.

During the course of the public review, it became apparent that there was confusion about the scale

of possible changes which can be made once the specification is frozen. As an early extension to go through this process, we have sometimes had to "be the first" to encounter such issues. While these are important discussions, they are much more general than the Scalar Cryptography extension in particular, so this document doesn’t address them.

We, the Cryptography Task Group, would like to sincerely thank everyone who participated in the public review process for their time and hard work in improving our specification.