AFRL-RY-WP-TR-2021-0210

CAPABILITY HARDWARE ENHANCED RISC INSTRUCTIONS (CHERI) INSTRUCTION-SET ARCHITECTURE FORMAL VERIFICATION (CIFV)

Peter G. Neumann
SRI International

Peter Sewell and Robert N. M. Watson
University of Cambridge (UK)

OCTOBER 2021
Final Report

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the USAF Air Force Research Lab (AFRL) Public Affairs Office (PAO) and is available to the general public, including foreign nationals.

Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2021-0210 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signature//

BASHAR W. ANTOON, DR-02
Program Manager
Resilient and Agile Avionics Branch
Spectrum Warfare Division

//Signature//

MICHAEL T. NIZZI, MAJOR, USAF
Chief, Resilient and Agile Avionics Branch
Spectrum Warfare Division

//Signature//

JOHN F. CARR, DR-04
Chief, Spectrum Warfare Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
October 2021	Final	31 January 2018 – 30 September 2021

4. TITLE AND SUBTITLE
CAPABILITY HARDWARE ENHANCED RISC INSTRUCTIONS (CHERI) INSTRUCTION-SET ARCHITECTURE FORMAL VERIFICATION (CIFV)

5a. CONTRACT NUMBER
FA8650-18-C-7809

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Peter G. Neumann (SRI International)
Peter Sewell and Robert N. M. Watson (University of Cambridge (UK))

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
Y1R9

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SRI International
333 Ravenswood Ave.
Menlo Park CA 94025-3493

University of Cambridge (UK)
Cambridge CB2 3AX,
United Kingdom

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH
45433-7320
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency (DARPA/MTO)
675 North Randolph Street
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYWA

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2021-0210

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number AFRL-2021-3576, Clearance Date 18 October 2021. Report contains color.

14. ABSTRACT
This is the final technical report for the project under Contract FA8650-18-C-7809. It summarizes the work done since the inception of the project.

15. SUBJECT TERMS
CHERI hardware, ISA Formal Verification Fundamental Trustworthiness Properties

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Bashar Antoon
Unclassified	Unclassified	Unclassified	SAR	30	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Table of Contents

Section **Page**

List of Figures

Figure **Page**

1.0 Summary

Developed in a sequence of DARPA I2O, AEO, and MTO projects, by SRI International (SRI) and the University of Cambridge (Cambridge), CHERI introduces processor support for fine-grained memory protection and scalable software compartmentalization through a hybrid capability-system model. CHERI instruction set architectures (ISAs) strongly mitigate not only many known vulnerability classes and exploit techniques, but also future vulnerability classes and exploits. CHERI is portable across multiple architectures. It was originally prototyped as an extension to the 64-bit MIPS ISA, and more recently as an extension to the 64-bit RISC-V ISA. Arm, partly supported by the UK project Digital Security by Design (DSbD) program, is currently developing a prototype CHERI extension of the Armv8-A architecture, Morello, together with a prototype processor implementation and development board, to allow industrial evaluation.

The main purpose of CIFV is to provide high assurance that CHERI ISAs fulfill the goal of high trustworthiness in the presence of malicious adversaries, by formalizing critical security properties of the hardware-software interface specifications (as embodied in the ISAs) and verifying them with machine checked mathematical proofs. This is necessary because the CHERI approach is grounded in complex architectural security properties, which must be tightly integrated with already complex production ISAs such as 64-bit MIPS and ARMv8, leaving substantial room for human error. Secondary goals are to lay the necessary foundations for future hardware and software verification efforts, e.g., of the security of hypervisors or OS kernels above CHERI hardware (which will depend on a formal characterization of the ISA), and to support ARMs potential adoption of our modeling and verification technologies.

All work required for CIFV has been completed successfully. We have established mechanized specifications of the CHERI-MIPS ISA and the Morello (CHERI-ARM) ISA, formalized critical security properties of them, and verified that those properties hold with machine checked mathematical proofs. During the course of the project, in work also supported by ECATS, we also developed mechanized specifications of the RISC-V and CHERI-RISC-V ISAs.

In the process, our RISC-V formal specification has been adopted as the official formal model by RISC-V International.

Our proofs for Morello provide the first example of a full-scale commercial ISA with formally proven security properties.

2.0 Introduction

2.1 CIFV Administrative Information

SRI International's CIFV (CHERI Instruction-set architecture Formal Verification) contract FA8650-18C-7809 bears the start date of 31 January 2018. The Principal Investigator is Dr. Peter G. Neumann, Peter.Neumann@SRI.COM, 1-650-859-2375. Dr. Prashanth Mundkur is the SRI co-PI. SRI's postal address is 333 Ravenswood Avenue, Menlo Park, CA 94025-3493.

The SRI contract includes a subcontract to the University of Cambridge. Professor Peter Sewell is Principal Investigator. Dr. Robert N. M. Watson is the Co-Principal Investigator at the University of Cambridge, and Professor Simon Moore is also a key participant at Cambridge.

The contract/administrative/financial contact at SRI is Colleen Conway, Colleen.Conway@SRI.COM, 1-650-859-4199. Her postal address is 333 Ravenswood Avenue, Menlo Park, CA 94025-3493.

2.2 Goals

We recall the original goals from the project proposal (ECU 17-409, 17 March 2017) and Statement of Work (FA8650-18-C-7809 Attachment #1, 28 Sep 2017):

Capability Hardware Enhanced RISC Instructions (CHERI), developed as part of the DARPA I2O CRASH research program by SRI International (SRI) and the University of Cambridge (UK), introduces processor support for fine-grained memory protection and scalable software compartmentalization through a hybrid capability system model. This innovative security approach supports the efficient application of the Principle of Least Privilege to both low-level generated code and higher-level software abstractions. The CHERI instruction-set architecture (ISA) strongly mitigates not only many known vulnerability classes and exploit techniques, but also future vulnerability classes and exploits. Although prototyped against the 64-bit MIPS ISA, the CHERI protection model is portable across multiple architectures. In recent work funded by DARPA AEO and DARPA I2O, we have been transitioning the technology to ARM UK Ltd as an extension to their widely used 64-bit ARMv8 ISA (CHERI-ARM). If successful, the transition project will make CHERI protection available to designers of security-critical mobile and embedded devices, from phones and tablets to network switches and cyber-physical systems, offering unprecedented system trustworthiness.

The purpose of this proposal is to provide high assurance that our CHERI-MIPS ISA and its CHERI-ARM ISA counterpart fulfill the goal of high trustworthiness in the presence of malicious adversaries, by formalizing critical security properties of the hardware-software interface specifications (embodied in the two ISAs) and verifying them with machine-checked mathematical proofs. This is necessary because the CHERI approach is grounded in complex architectural security properties, which must be tightly integrated with already complex production ISAs such as 64-bit MIPS and ARMv8, leaving substantial room for human error. If

successful, this will be the first example of a commercial ISA with formally proven security properties.

Secondary goals are to lay the necessary foundations for future hardware and software verification efforts, e.g., of the security of hypervisors or OS kernels above CHERI, that will depend on a formal characterization of the ISA, and to support ARMs potential adoption of our modeling and verification technologies.

The goal of Capability Hardware Enhanced RISC Instructions (CHERI) Instruction Set Architecture (ISA) Formal Verification (FV) is to (1) provide high assurance that our CHERI-MIPS ISA and its CHERI-ARM ISA counterpart have high trustworthiness in the presence of malicious adversaries, (2) formalize critical security properties of the hardware-software interface specifications (embodied in the two ISAs) and, (3) verify them with machine-checked mathematical proofs.

2.3 Status

All noted above, all work required for CIFV has been completed.

We have established mechanized specifications of the CHERI-MIPS ISA and the Morello (CHERI-ARM) ISA, formalized critical security properties of them, and verified that those properties hold with machine checked mathematical proofs of the entire ISA.

During the course of the project, in work also supported by ECATS, we moreover developed mechanized specifications of the RISC-V and CHERI-RISC-V ISAs. Our RISC-V specification has been adopted as the official formal model by RISC-V International.

2.4 Funding

The papers and software deliverables described here combine work directly funded by CIFV and work supported by other funding from Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts HR0011-18-C-0016 (ECATS), FA8750-10-C-0237 (CTSRD), and FA8750-11-C-0249 (MRC2), and by other UK, EU, and industry funding, notably the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694 (which is funding the development of Morello); UK EPSRC program grant EP/K008528/1 (REMS: Rigorous Engineering for Mainstream Systems); the European Research Council (ERC) Advanced Grant 789108 (ELVER) under the European Unions Horizon 2020 research and innovation program (Sewell); a Gates studentship (Nienhuis); and donations from Arm and Google.

CIFV has specifically funded parts of the development of Sail, including the Isabelle, SMT, and Isla (symbolic evaluation) backends (Armstrong), of the flow from Arm's ASL language to Sail (Bauereiss, Armstrong), of our proofs for CHERI-MIPS above the L3 model (Nienhuis), the proof technology we then developed for general CHERI architectures, prototyped for CHERI-MIPS, CHERI-RISC-V, and early versions of CHERI-ARM and later applied (largely funded by DSbD) to Morello (Bauereiss), and much of the development of the RISC-V and CHERI-RISC-V models, tooling, and documentation (Mundkur).

3.0 Methods, Assumptions, and Procedures

Architecture specifications establish the basic interface between hardware and software. Modern architectures for application-class processors (as used in mobile devices, workstations, and servers) are large and complex, e.g., with the Armv8-A architecture described in an 8000+ page document. To establish high confidence in the CHERI architecture designs, one has to be able to do machine checked mathematical proofs about them. That means the architecture specifications have to be expressed in mechanized formalisms. Moreover, to ensure that they accurately capture the design intent and relate to the CHERI hardware and software implementations (rather than merely being untested *post facto* formal artifacts), it has been essential to integrate formal architecture specifications into the main CHERI design and engineering processes.

We have done this with new languages and tooling for defining formal instruction set architectures. First, we transitioned CHERI-MIPS development from its previous informal architecture specification to one written in the earlier L3 ISA definition language. L3 supports automatic generation of an emulator and of definitions in the Isabelle theorem-prover; we used the former as a test oracle for testing hardware and for software bring-up, and then used the latter as a basic for mechanized proofs of security properties of the CHERI-MIPS ISA design. In parallel, we developed an improved ISA definition language, Sail. We transitioned the CHERI-MIPS ISA model to Sail and we developed Sail definitions of RISC-V and CHERI-RISC-V. The CHERI-MIPS and CHERI-RISC-V Sail definitions are integrated into the documentation for those, and the RISC-V Sail definition has been adopted as its official formal model by RISC-V International. Arm internally use their own ISA definition language, ASL. We developed an automatic translator from ASL to Sail (Sail having been carefully designed to make this possible while also supporting the stronger typing one wants for mechanized proofs in Isabelle, HOL4, or Coq), and used that to produce Sail models for the base Armv8-A architecture and for the Morello extension thereof with CHERI features. From Sail, we can automatically produce theorem-prover definitions for the Isabelle, HOL4, and Coq theorem provers, a C emulator, SMT definitions, SMT-based symbolic evaluation (Isla), automated instruction sequence test generation using Isla, specification coverage measurement, and tooling for concurrency semantics exploration. Our Sail and L3 architecture specifications cover the full sequential ISAs, and are complete enough to boot conventional operating systems. This rich tooling let us support and extend traditional engineering methods both in lightweight ways, by using the Sail models as the principal design artifacts, for documentation, as a golden reference for testing, and for test generation; and for mechanized proof of security properties of CHERI-MIPS and Morello. Fig. 1 shows the main Sail tooling flows we have developed and used. The work thereby spanned multiple research cultures in a hardware/software/semantics co-design process.

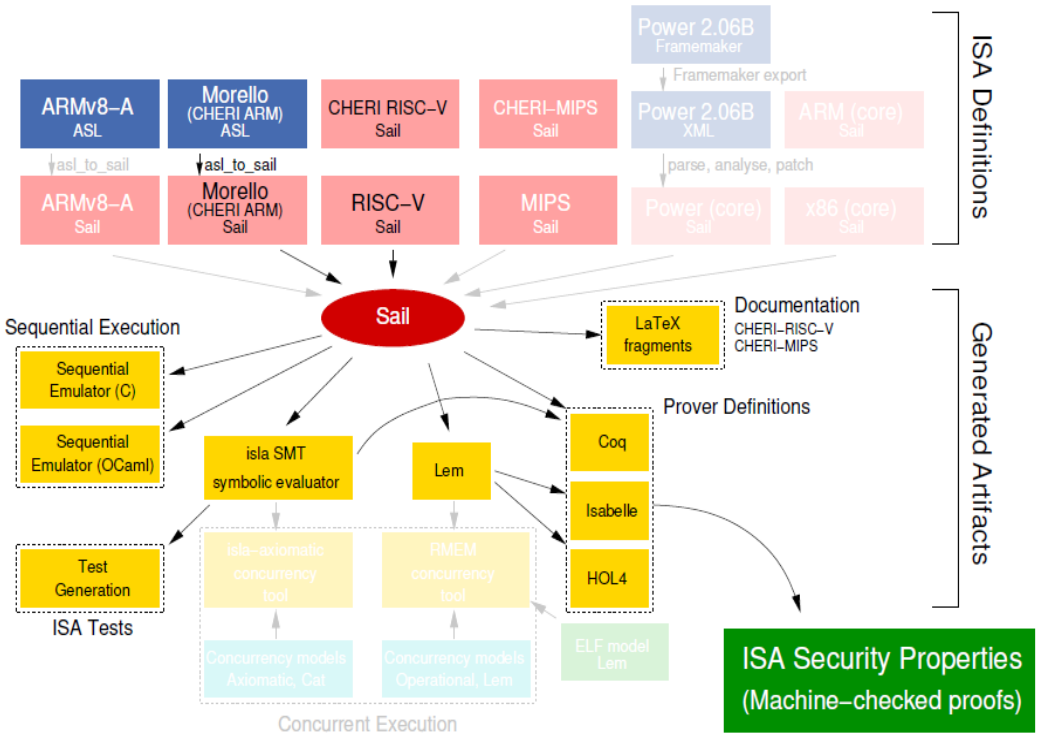

Figure 1. Sail Tool Flow

The only substantial simplifying assumption that we made is that our security property statements and proofs for CHERI-MIPS and Morello cover only sequential execution. The concurrency semantics of modern architectures remains a challenging research problem in itself, especially now for the interaction of systems features (exceptions, instruction fetch, virtual memory, etc.) with concurrency. We used our Sail infrastructure for ISA definition and SMT based symbolic evaluation to integrate the sequential details of instruction semantics with concurrency model evaluation, as a step towards this.

4.0 Result and Discussion

4.1 CIFV Publications

The following first three papers summarize the main CIFV work in detail. They are the best presentations of the main results of CIFV. We give their abstracts inline below, but have not otherwise duplicated their content in this report.

1. ISA semantics for ARMv8-A, RISC-V, and CHERI-MIPS [1].

   Abstract: Architecture specifications notionally define the fundamental interface between hardware and software: the envelope of allowed behavior for processor implementations, and the basic assumptions for software development and verification. But in practice, they are typically natural-language prose and pseudocode documents, not rigorous or executable artifacts, leaving software and verification on shaky ground.

   In this paper, we present rigorous semantic models for the sequential behavior of large parts of the mainstream ARMv8-A, RISC-V, and MIPS architectures, and the research CHERI-MIPS architecture, that are complete enough to boot operating systems, variously Linux, FreeBSD, or seL4. Our ARMv8-A models are automatically translated from authoritative ARM-internal definitions, and (in one variant) tested against the ARM Architecture Validation Suite.

   We do this using a custom language for ISA semantics, Sail, with a lightweight dependent type system, that supports automatic generation of emulator code in C and OCaml, and automatic generation of proof-assistant definitions for Isabelle, HOL4, and (currently only for MIPS) Coq. We use the former for validation, and to assess specification coverage. To demonstrate the usability of the latter, we prove (in Isabelle) correctness of a purely functional characterization of ARMv8- A address translation. We moreover integrate the RISC-V model into the RMEM tool for (user-mode) relaxed-memory concurrency exploration. We prove (on paper) the soundness of the core Sail type system.

   We thereby take a big step towards making the architectural abstraction actually well-defined, establishing foundations for verification and reasoning.

2. Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process [2].

   Abstract: The root causes of many security vulnerabilities include a pernicious combination of two problems, often regarded as inescapable aspects of computing. First, the protection mechanisms provided by the mainstream processor architecture and C/C++ language abstractions, dating back to the 1970s and before, provide only coarse-grain virtual-memory-based protection. Second, mainstream system engineering relies almost exclusively on test-and-debug methods, with (at best) prose specifications. These methods have historically sufficed commercially for much of the computer

industry, but they fail to prevent large numbers of exploitable bugs, and the security problems that this causes are becoming ever more acute.

In this paper we show how more rigorous engineering methods can be applied to the development of a new security-enhanced processor architecture, with its accompanying hardware implementation and software stack. We use formal models of the complete instruction-set architecture (ISA) at the heart of the design and engineering process, both in lightweight ways that support and improve normal engineering practice as documentation, in emulators used as a test oracle for hardware and for running software, and for test generation and for formal verification. We formalize key intended security properties of the design, and establish that these hold with mechanized proof. This is for the same complete ISA models (complete enough to boot operating systems), without idealization.

We do this for CHERI, an architecture with *hardware capabilities* that supports fine-grained memory protection and scalable secure compartmentalization, while offering a smooth adoption path for existing software. CHERI is a maturing research architecture, developed since 2010, with work now underway on an Arm industrial prototype to explore its possible adoption in mass market commercial processors. The rigorous engineering work described here has been an integral part of its development to date, enabling more rapid and confident experimentation, and boosting confidence in the design.

3. Verified security for the Morello capability-enhanced prototype Arm architecture [3].

Abstract: Memory safety bugs continue to be a major source of security vulnerabilities in our critical infrastructure. The CHERI project has proposed extending conventional architectures with hardware-supported *capabilities* to enable fine-grained memory protection and scalable compartmentalization, allowing historically memory-unsafe C and C++ to be adapted to deterministically mitigate large classes of vulnerabilities, while requiring only minor changes to existing system software sources. Arm is currently designing and building Morello, a CHERI- enabled prototype architecture, processor, SoC, and board, extending the high performance Neoverse N1, to enable industrial evaluation of CHERI and pave the way for potential mass market adoption. However, for such a major new security-oriented architecture feature, it is important to establish high confidence that it does provide the protections it intends to, and that cannot be done with conventional engineering techniques.

In this paper we put the Morello architecture on a solid mathematical footing from the outset. We define the fundamental security property that Morello aims to provide, reachable capability monotonicity, and prove that the architecture definition satisfies it. This proof is mechanized in Isabelle/HOL, and applies to a translation of the official Arm Morello specification into Isabelle. The main challenge is handling the complexity and scale of a production architecture: 62,000 lines of specification, translated to 210,000 lines of Isabelle. We do this by factoring the proof via a narrow abstraction capturing the

essential properties of instruction execution in an arbitrary CHERI ISA, expressed above a monadic intra-instruction semantics. We also develop a model-based test generator, which generates instruction- sequence tests that give good specification coverage, used in early testing of the Morello implementation and in Morello QEMU development. We also use Arm's internal test suite to validate our internal model.

This gives us machine-checked mathematical proofs of whole-ISA security properties of a full-scale industry architecture, at design-time. To the best of our knowledge, this is the first demonstration that that is feasible, and it significantly increases confidence in Morello.

The following two papers build on the CIFV ISA semantics work, developing tooling enabling one to reason about concurrent Armv8-A and RISC-V code, including some systems concurrency aspects of Armv8-A. These are steps towards enabling reasoning about concurrent Morello and CHERI-RISC-V code.

4. ARMv8-A system semantics: instruction fetch in relaxed architectures [4].

   Abstract: Computing relies on *architecture specifications* to decouple hardware and software development. Historically these have been prose documents, with all the problems that entails, but research over the last ten years has developed rigorous and executable-as-test-oracle specifications of mainstream architecture instruction sets and "user-mode" concurrency, clarifying architectures and bringing them into the scope of programming language semantics and verification. However, the *system semantics*, of instruction fetch and cache maintenance, exceptions and interrupts, and address translation, remains obscure, leaving us without a solid foundation for verification of security critical systems software.

   In this paper we establish a robust model for one aspect of system semantics: instruction fetch and cache maintenance for ARMv8-A. Systems code relies on executing instructions that were written by data writes, e.g. in program loading, dynamic linking, JIT compilation, debugging, and OS configuration, but hardware implementations are often highly optimized, e.g. with instruction caches, line fill buffers, out-of-order fetching, branch prediction, and instruction prefetching, which can affect programmer-observable behavior. It is essential, both for programming and verification, to abstract from such micro-architectural details as much as possible, but no more. We explore the key architecture design questions with a series of examples, discussed in detail with senior Arm staff; capture the architectural intent in operational and axiomatic semantic models, extending previous work on "user-mode" concurrency; make these models executable as test oracles for small examples; and experimentally validate them against hardware behavior (finding a bug in one hardware device). We thereby bring these subtle issues into the mathematical domain, clarifying the architecture and enabling future work on system software verification.

5. Isla: Integrating full-scale ISA semantics and axiomatic concurrency models [5].

   Abstract: Architecture specifications such as Armv8-A and RISC-V are the ultimate foundation for software verification and the correctness criteria for hardware verification. They should define the allowed sequential and relaxed-memory concurrency behavior of programs, but hitherto there has been no integration of full scale instruction-set architecture (ISA) semantics with axiomatic concurrency models, either in mathematics or in tools. These ISA semantics can be surprisingly large and intricate, e.g. 100k+ lines for Armv8-A.

   In this paper we present a tool, Isla, for computing the allowed behaviors of concurrent litmus tests with respect to full-scale ISA definitions, in Sail, and arbitrary axiomatic relaxed-memory concurrency models, in the Cat language. It is based on a generic symbolic engine for Sail ISA specifications, which should be valuable also for other verification tasks. We equip the tool with a web inter- face to make it widely accessible, and illustrate and evaluate it for Armv8-A and RISC-V.

   By using full-scale and authoritative ISA semantics, this lets one evaluate litmus tests using arbitrary user instructions with high confidence. Moreover, because these ISA specifications give detailed and validated definitions of the sequential aspects of *systems* functionality, as used by hypervisors and operating systems, e.g. instruction fetch, exceptions, and address translation, our tool pro- vides a basis for developing concurrency semantics for these. We demonstrate this for the Armv8-A instruction-fetch model and self-modifying code examples of Simner et al.

The following five technical reports describe the CHERI project as a whole, give an introduction to it (including the CIFV work), give a preliminary version of Publication 2, and describe *capability essential IP* ideas essential to the creation of a contemporary CHERI capability system in architecture and microarchitecture which Arm and Cambridge have made available for use without restriction.

6. Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7) [6].

   Abstract: This technical report describes CHERI ISAv7, the seventh version of the Capability Hardware Enhanced RISC Instructions (CHERI) Instruction Set Architecture (ISA) being developed by SRI International and the University of Cambridge. This design captures nine years of research, development, experimentation, refinement, formal analysis, and validation through hardware and software implementation. CHERI ISAv7 is a substantial enhancement to prior ISA versions. We differentiate an architecture-neutral protection model vs. architecture-specific instantiations in 64-bit MIPS, 64-bit RISC-V, and x86-64. We have defined a new CHERI Concentrate compression model. CHERI-RISC-V is more substantially elaborated. A new compartment-ID register assists in resisting micro-

architectural side-channel attacks. Experimental features include linear capabilities, capability coloring, temporal memory safety, and 64-bit capabilities for 32-bit architectures.

CHERI is a *hybrid capability-system architecture* that adds new capability-system primitives to commodity 64-bit RISC ISAs, enabling software to efficiently implement *fine-grained memory protection* and *scalable software compartmentalization*. Design goals include incremental adoptability within current ISAs and software stacks, low performance overhead for memory protection, significant performance improvements for software compartmentalization, formal grounding, and programmer-friendly underpinnings. We have focused on providing strong, non-probabilistic, efficient architectural foundations for the principles of *least privilege* and *intentional use* in the execution of software at multiple levels of abstraction, preventing and mitigating vulnerabilities.

The CHERI system architecture purposefully addresses known performance and robustness gaps in commodity ISAs that hinder the adoption of more secure programming models centered around the principle of least privilege. To this end, CHERI blends traditional paged virtual memory with an in address-space capability model that includes capability registers, capability instructions, and tagged memory. CHERI builds on the C-language fat-pointer literature: its capabilities can describe fine-grained regions of memory, and can be substituted for data or code pointers in generated code, protecting data and also improving control- flow robustness. Strong capability integrity and monotonicity properties allow the CHERI model to express a variety of protection properties, from enforcing valid C-language pointer provenance and bounds checking to implementing the isolation and controlled communication structures required for software compartmentalization.

CHERI's hybrid capability-system approach, inspired by the Capsicum security model, allows incremental adoption of capability-oriented design: software implementations that are more robust and resilient can be deployed where they are most needed, while leaving less critical software largely unmodified, but nevertheless suitably constrained to be incapable of having adverse effects. Potential deployment scenarios include low-level software Trusted Computing Bases (TCBs) such as separation kernels, hypervisors, and operating-system kernels, as well as user space TCBs such as language runtimes and web browsers. We also see potential early-use scenarios around particularly high-risk software libraries (such as data compression, protocol parsing, and image processing), which are concentrations of both complex and historically vulnerability-prone code exposed to untrustworthy data sources, while leaving containing applications unchanged.

7. Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 8) [7].

Abstract: This technical report describes CHERI ISAv8, the eighth version of the CHERI architecture being developed by SRI International and the University of Cambridge. This design captures ten years of research, development, experimentation, refinement, formal analysis, and validation through hardware and software implementation.

CHERI introduces an architecture-neutral capability-based protection model, which has been instantiated in various commodity base architectures to give CHERI-MIPS, CHERI-RISC-V, Arm's prototype Morello architecture, and (sketched) CHERI-x86-64. It enables software to efficiently implement fine- grained memory protection and scalable software compartmentalization, by providing strong, non-probabilistic, efficient mechanisms to support the principles of least privilege and intentional use in the execution of software at multiple levels of abstraction, preventing and mitigating vulnerabilities. Design goals include incremental adoptability from current ISAs and software stacks, low performance overhead for memory protection, significant performance improvements for software compartmentalization, formal grounding, and programmer-friendly underpinnings.

CHERI blends traditional paged virtual memory with an in-address-space capability model that includes capability values in registers, capability instructions, and tagged memory to enforce capability integrity. This hybrid approach, inspired by the Capsicum security model, addresses the performance and robustness issues that arise when trying to express more secure programming models, minimizing privilege, above conventional architectures that provide only MMU-based protection. CHERI builds on the C-language fat-pointer literature: its capabilities can describe fine-grained regions of memory, and can be substituted for data or code pointers in generated code, protecting data and improving control-flow robustness. Strong capability integrity and monotonicity properties allow CHERI to express a variety of protection idioms, from enforcing valid C-language pointer provenance and bounds checking to implementing the isolation and controlled communication structures required for software compartmentalization.

CHERI's hybrid approach allows incremental adoption of capability-oriented design: critical components can be ported and recompiled to use capabilities throughout, providing fine-grain memory protection, or be largely unmodified but encapsulated in ways that permit only controlled interaction. Potential early deployment scenarios include low-level software Trusted Computing Bases (TCBs) such as separation kernels, hypervisors, and operating-system kernels, user space TCBs such as language runtimes and web browsers, and particularly high-risk software libraries such as data compression, protocol parsing, and image processing (which are concentrations of both complex and historically vulnerability prone code exposed to untrustworthy data sources).

CHERI ISAv8 is a substantial enhancement to prior ISA versions. Capability compression is now part of the abstract model. Both 32-bit and 64-bit architectural address sizes are supported. Various previously experimental features, such as sentry capabilities and CHERI-RISC-V, are now considered mature. We have defined a number of new temporal memory-safety acceleration features including MMU assistance for a load-side-barrier revocation model. We have added a chapter on practical CHERI microarchitecture. CHERI ISAv8 is synchronized with Arm Morello.

8. An introduction to CHERI [8].

   Abstract: CHERI (Capability Hardware Enhanced RISC Instructions) extends conventional processor Instruction-Set Architectures (ISAs) with *architectural capabilities* to enable fine-grained memory protection and highly scalable soft- ware compartmentalization. CHERI's hybrid capability-system approach allows architectural capabilities to be integrated cleanly with contemporary RISC architectures and microarchitectures, as well as with MMU-based C/C++-language software stacks.

   CHERI's capabilities are unforgeable tokens of authority, which can be used to implement both explicit pointers (those declared in the language) and implied pointers (those used by the runtime and generated code) in C and C++. When used for C/C++ memory protection, CHERI directly mitigates a broad range of known vulnerability types and exploit techniques. Support for more scalable software compartmentalization facilitates software mitigation techniques such as sandboxing, which also defend against future (currently unknown) vulnerability classes and exploit techniques.

   We have developed, evaluated, and demonstrated this approach through hardware-software prototypes, including multiple CPU prototypes, and a full software stack. This stack includes an adapted version of the Clang/LLVM compiler suite with support for capability-based C/C++, and a full UNIX-style OS (CheriBSD, based on FreeBSD) implementing spatial, referential, and (currently for users pace) non-stack temporal memory safety. Formal modeling and verification allow us to make strong claims about the security properties of CHERI-enabled architectures.

   This report is a high-level introduction to CHERI. The report describes our architectural approach, CHERI's key micro-architectural implications, our approach to formal modeling and proof, the CHERI software model, our software-stack prototypes, further reading, and potential areas of future research.

9. Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process [9].

   Abstract: The root causes of many security vulnerabilities include a pernicious combination of two problems, often regarded as inescapable aspects of computing. First, the protection mechanisms provided by the mainstream processor architecture and C/C++ language abstractions, dating back to the 1970s and before, provide only

coarse-grain virtual-memory-based protection. Second, mainstream system engineering relies almost exclusively on test-and-debug methods, with (at best) prose specifications. These methods have historically sufficed commercially for much of the computer industry, but they fail to prevent large numbers of exploitable bugs, and the security problems that this causes are becoming ever more acute.

In this paper we show how more rigorous engineering methods can be applied to the development of a new security-enhanced processor architecture, with its ac companying hardware implementation and software stack. We use formal models of the complete instruction-set architecture (ISA) at the heart of the design and engineering process, both in lightweight ways that support and improve normal engineering practice – as documentation, in emulators used as a test oracle for hardware and for running software, and for test generation and for formal verification. We formalize key intended security properties of the design, and establish that these hold with mechanized proof. This is for the same complete ISA models (complete enough to boot operating systems), without idealization.

We do this for CHERI, an architecture with *hardware capabilities* that sup- ports fine-grained memory protection and scalable secure compartmentalization, while offering a smooth adoption path for existing software. CHERI is a maturing research architecture, developed since 2010, with work now underway to explore its possible adoption in mass-market commercial processors. The rigorous engineering work described here has been an integral part of its development to date, enabling more rapid and confident experimentation, and boosting confidence in the design.

10. DSbD CHERI and Morello Capability Essential IP (Version 1) [10].

Abstract: The CHERI protection model extends contemporary Instruction Set Architectures (ISAs) with support for architectural capabilities. The UKRI Digital Security by Design (DSbD) programme is supporting the creation of Arms prototype Morello processor, System-on-Chip (SoC), and board. Morello experimentally incorporates the CHERI protection model, developed at the University of Cambridge and SRI International, into the ARMv8-A architecture. This document declares a set of capability essential IP ideas essential to the creation of a contemporary CHERI capability system in architecture and microarchitecture. Arm and Cambridge agree that they have made this IP available for use without restriction. This document also identifies a set of CHERI background documents that may be of value as prior art.

4.2 CIFV Software Deliverables

These deliverables, as a tarball of specifications, proofs, and tools, were delivered along with this DRAFT final report due at the end of July 2021. The contract called for delivery of 'software', but (beyond Sail and the Sail CHERI-MIPS specification) left open exactly what these should be. We have interpreted this deliverable to include the following elements.

They comprise formal instruction-set architecture (ISA) models for CHERI-MIPS (in L3 and Sail), RISC-V, CHERI-RISC-V, and Armv8-A; proofs for CHERI-MIPS; and tools. All are made available under open-source licenses, typically BSD two-clause except for the Armv8-a model, which (by agreement with Arm) is available under a BSD Clear license. They are in public github repositories; the delivered material is snapshots of those.

1. L3 CHERI-MIPS ISA model (`l3mips`)
   `https://github.com/acjf3/l3mips`
2. L3 CHERI-MIPS proofs (`l3-cheri-mips-proofs`)
   `https://github.com/CTSRD-CHERI/l3-cheri-mips-proofs`
3. Sail ISA description language implementation (`sail`)
   `https://github.com/rems-project/sail`
4. Sail CHERI-MIPS ISA model (`sail-cheri-mips`)
   `https://github.com/CTSRD-CHERI/sail-cheri-mips`
5. Sail CHERI-MIPS proofs (`sail-cheri-mips-proofs`)
   `https://github.com/CTSRD-CHERI/sail-cheri-mips-proofs`
6. Sail RISC-V ISA model (`sail-riscv`)
   `https://github.com/rems-project/sail-riscv`
7. Sail CHERI-RISCV ISA model (`sail-cheri-riscv`)
   `https://github.com/CTSRD-CHERI/sail-cheri-riscv`
8. Sail Armv8.5 model (`sail-arm`)
   `https://github.com/rems-project/sail-arm`
9. Tooling to convert from Arm-internal ASL specifications to Sail (asl to sail)
   `https://github.com/rems-project/sail-arm`

As anticipated in the CIFV proposal, the Morello CHERI Arm ISA specification and our proofs above that incorporate some Arm confidential material and are not included in these deliverables, though we expect to be able to make non-confidential versions available in due course.

The above `sail-riscv` Sail RISC-V ISA model has been adopted by RISC-V International as its official formal model, and the repository will move to the RISC-V International organization shortly.

4.3 CIFV Tasks

We recall the original Tasks from the Statement of Work and describe how these have been satisfied by the publications and deliverables listed above.

4.3.1 Task 1: Migrate CHERI-MIPS ISA L3 model to Sail

"The contractor shall research and develop the CHERI-MIPS Sail model, migrating the CHERI-MIPS ISA specification from L3 to the Sail modeling language. The contractor shall port the existing CHERI-MIPS ISA specification from L3 to Sail, and validate the new model against the existing L3 model and against the contractor's QEMU and FPGA implementations."

The model of MIPS and CHERI-MIPS written in the L3 modeling language (Deliverable 1) were migrated to a Sail CHERI-MIPS model (Deliverable 4). The C emulators generated from Sail boot FreeBSD and CheriBSD and pass the CHERI-MIPS test suite. The model was maintained to track the development of the CHERI-MIPS architecture as it evolved during this project, used as a reference for testing hardware and software, and used to generate documentation that is integrated into the CHERI-MIPS sections of the CHERI architecture specification, in Publications 6 and 7. All this is described in Publications 1 and 2.

4.3.2 Task 2: Develop Sail models of ARMv8-A, CHERI-ARM, RISC-V, and (with ECATS) CHERI-RISC-V

"The contractor shall research and develop ARM Sail models of ARMv8-A and of CHERI-ARM, with a modeling flow from ARM's proprietary ASL ISA definition into Sail ISA formal descriptions. The contractor shall develop new automated translation tooling to convert ARM ASL definitions into the Sail ISA modeling language, and shall use this to develop Sail models, both of the base ARMv8-A architecture and of the CHERI-ARM architecture currently being designed. The contractor shall do this incrementally, first addressing the fragment of ASL needed for selected instructions, then scaling up to cover all user-mode aspects without floating point, vector instructions, or exceptions, and then extending to cover exceptions and other system-mode aspects as feasible. The contractor shall continue to develop Sail as necessary to support this."

4.3.2.1 Armv8-A and Morello

During the course of CIFV, the CHERI-ARM work within Arm, on an extension of the Armv8-A ISA to incorporate CHERI features, migrated from an Arm-internal effort to the Digital Security by Design (DSbD) program, funded by the UK Government and industry (£70m and £117m respectively). As the main part of this, Arm are developing the Morello ISA, extending Armv8.2-A, a hardware implementation based on the high-performance Neoverse N1 processor, a system-on-chip (SoC), a development board, and system software. An important part of the Digital Security by Design program is formal verification of security properties of the Morello ISA, enabled by the technologies we developed within CIFV and previously.

We therefore adapted our work accordingly, targeting the Morello ISA rather than the earlier Arm-internal CHERI-ARM prototype. The scale-up that this entailed was enabled by combining CIFV and other funding, especially from DSbD.

We have done this for the complete sequential ISA specifications of Armv8-A and Morello, covering the sequential behavior of all user and systems aspects (including floating-point, vector instructions, exceptions, and much more).

Our automated translation tooling from ASL to Sail is provided as Deliverable 9. An early version was described in Publication 1, and it was exercised first on the base Armv8-A architecture to produce Sail Armv8.3-A and Armv8.5-A models, the latter of which forms Deliverable 8. More recently, it has been used to produce Sail models of the Morello ISA, as described in Publication 3.

As anticipated in the CIFV proposal, the machine-readable Sail Morello CHERI Arm ISA specification and our proofs above that incorporate some Arm confidential material and are not included in these deliverables, though we expect to be able to make non-confidential versions available in due course. They are described in Publication 3.

For the concurrent aspects of Armv8-A behavior, we have established models for Armv8-A instruction fetch and cache maintenance, validated by experiment and by detailed discussion with Arm; this is reported in Publication 4. We have moreover developed tooling to integrate the above sequential ISA models for Armv8-A, and our sequential ISA model for RISC-V, with concurrency models for user-mode and system semantics, as our Isla symbolic evaluation engine for Sail; this is reported in Publications 5 and 6.

4.3.2.2 RISC and CHERI-RISC-V

In conjunction with the DARPA ECATS project, developing CHERI-RISC-V, we further extended our scope to provide formal models of the base RISC-V architecture and of CHERI-RISC-V. The former has been adopted as the official formal model of the architecture by RISC-V International (formerly known as the RISC-V Foundation); the latter has been (and continues to be) the principal design artifact during ISA design, and included in the documentation (Publications 6 and 7).

The Sail RISC-V model covers the user-mode and privileged-mode sections of the RV32G and RV64G architectural specifications. This includes coverage of the extensions that specify com- pressed instructions and single and double floating point instructions. The model specifies assembly language formats of the instructions, the corresponding encoders and decoders, and the instruction semantics. Fig. 2 shows the structure of the model. It is also integrated with the operational and axiomatic models of the RISC-V RVWMO (Weak Memory Ordering) relaxed memory model, which is the default memory consistency model for the architecture.

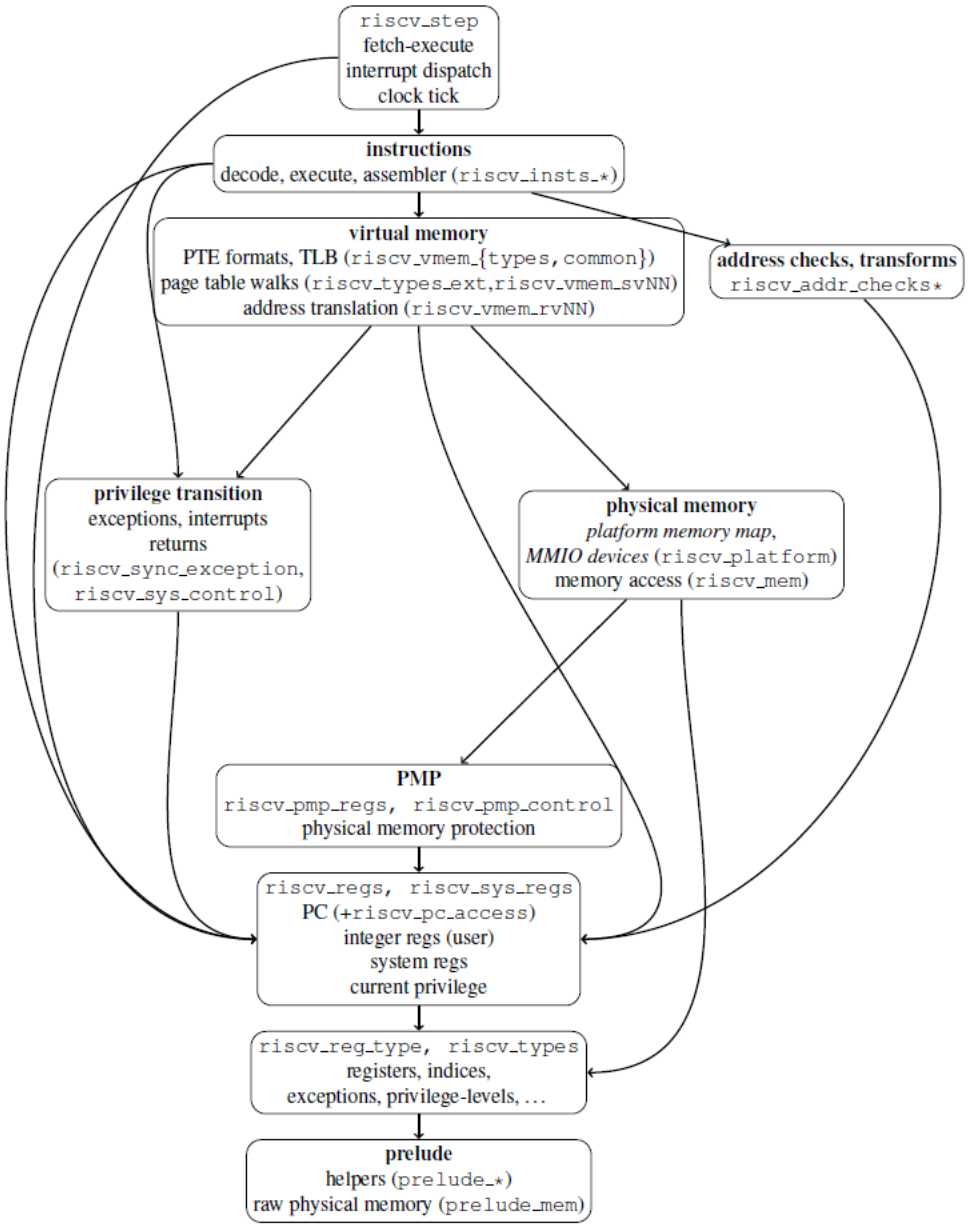

Figure 2. Structure of the Sail RISC-V model, taken from its online documentation

Sequential execution From the model, Sail generates OCaml and C emulators that can execute RISC-V ELF files, and both emulators provide platform support sufficient to boot Linux, FreeBSD and seL4. The OCaml emulator can generate its own platform device-tree description, while the C emulator currently requires a consistent description to be manually provided. The emulators implement ELF loading and platform devices, define the physical memory map, and can be configured to select implementation-specific ISA choices.

Concurrent execution The model integrates the operational model of the default RVWMO memory consistency model of the RISC-V concurrency architecture using the RMEM tool[1]. It is also integrated with the RISC-V axiomatic concurrency model as part of our `isla axiomatic` tool[2], as described in Publication 5.

Concurrent testing As part of our concurrency architecture work, we have produced and released a library of approximately 7000 *litmus tests*[3] The operational and axiomatic RISC-V concurrency models are in sync for these tests, and they moreover agree with the corresponding ARM architected behavior for the tests in common. We have also run these tests on RISC-V hardware, on a SiFive RISC-V FU540 multicore proto board (Freedom Unleashed), kindly on loan from Imperas. To date, we see only sequentially consistent behavior there.

Specification coverage measurement in testing The Sail-generated C emulator can measure specification branch coverage of any executed tests, displaying the results as per-file tables and as html-annotated versions of the model source.

Use as test oracle in tandem verification For tandem verification of random instruction streams we support the protocols used in TESTRIG[4] to directly inject instructions into the C emulator and produce trace information. This has been used for cross testing against the Spike simulator and the various implementations of RISC-V (Piccolo, Flute, Tooba) written in Bluespec SystemVerilog by Bluespec Inc., and against the CHERI extensions of these developed in the ECATS project.

Use in test generation Our OCaml backend can produce QuickCheck-style random generators for types in Sail specifications, which we have used to produce random instructions sequences for testing. The generation of individual types can be overridden by the developer to, for example, remove implementation-specific instructions or introduce register biasing.

Generating theorem-prover definitions From the model, Sail generates idiomatic theorem prover definitions for multiple tools. At present we support Isabelle, HOL4 and Coq, and provide snapshots of the generated theorem prover definitions. Our theorem-prover translation can target multiple monads for different purposes. The first is a state monad with nondeterminism and exceptions, suitable for reasoning in a sequential setting, assuming that state-changing expressions are executed without interruptions and with exclusive access to the state. For reasoning about con- currency, where instructions execute out-of-order, speculatively, and non-atomically, we provide a free monad over an effect datatype of memory actions. This monad is also used as part of our aforementioned concurrency support via the RMEM tool.

[1]Available at http://www.cl.cam.ac.uk/users/pes20/rmem

[2]Available at https://isla-axiomatic.cl.cam.ac.uk/

[3]Available at https://github.com/litmus-tests/litmus-tests-riscv.

[4]Available at https://github.com/CTSRD-CHERI/TestRIG.

Model validation The model was used to boot the seL4, Linux and FreeBSD operating systems, and validated against the Spike simulator commonly used in RISC-V development. Continuous integration runs the tests from the `riscv-tests` test suite, and the C and OCaml emulators have been incorporated into the official `riscv-compliance` suite developed by the compliance team of RISC-V International. The RISC-V model behavior was also validated on concurrency litmus tests using RMEM.

4.3.3 Task 3: Develop Sail verification flow to theorem provers

"The contractor shall research and develop the Sail verification flow, from Sail to theorem provers. The contractor shall develop automated translations from Sail into theorem provers Isabelle, HOL4, and/or Coq."

We have developed Sail to automatically produce, from a Sail ISA definition, versions of that definition in all these of these major interactive theorem-proving tools, Isabelle, HOL4, and Coq (the first two via Lem). This tooling is part of the Sail implementation, which is available as Deliverable 3. Snapshots of the generated theorem-prover models are included in Deliverables 4, 6, 7, and 8, respectively for CHERI-MIPS, RISC-V, CHERI-RISCV, and Armv8.5, with Isabelle and Coq for all and additionally HOL4 for RISC-V. This work is described in Publications 1 and 3.

We also developed an experimental backend for Sail for (non-interactive) SMT theorem- proving tools (such as Z3) and the Z3-based Isla symbolic evaluation engine mentioned above, described in Publication 5. These enabled quick checking of critical properties about key aspects of the specification, e.g., capability encoding, as described in Publication 3.

4.3.4 Task 4: Research, develop, and verify, to the extent possible, security properties for CHERI-MIPS, CHERI-ARM, and CHERI-RISC-V ISAs

"The contractor shall research, develop, and verify security properties for CHERI-MIPS and CHERI-ARM. The contractor shall formally state and verify key security properties of CHERI-MIPS and CHERI-ARM. The contractor shall assess whether the formal statements of security properties capture the informal design intent of CHERI-MIPS and CHERI-ARM by a careful review process."

For CHERI-MIPS, we first formally stated and verified key security properties above the Isabelle export of our earlier CHERI-MIPS model in the L3 language; this is reported in Deliverable 2 and Publication 2. We then developed streamlined proof technology, including an abstract characterization of CHERI, aiming to make it scalable to CHERI-ARM, above our later CHERI MIPS model in Sail; this is available in Deliverable 5. We used that technology to give statements and proofs of key security properties for the entire Morello ISA specification, with a flow summarized in Fig. 3; this is reported in Publication 3. The technology is to a significant extent architecture-neutral, with properties that capture the essential aspects of any CHERI architecture. We confirmed this with partial proof experiments for CHERI-RISC-V, but prioritized Morello

proofs above that, so these have not been completed to date. These publications also describe how the formal properties relate to the original design intent, which they have significantly clarified.

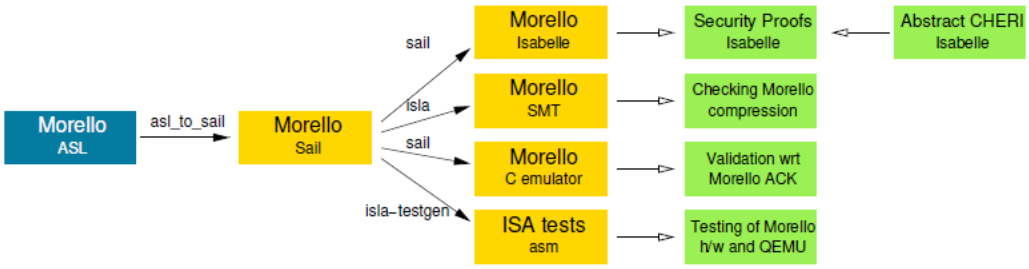

Figure 3. From Morello ASL source (blue) to auto-generated artifacts (yellow) and verification outcomes (green)

4.3.5 Task 5: Transitioning models, tools, and techniques

"The contractor shall transition the models, tools, and techniques. The contractor shall continue engagement with the ARM development effort and reusability of the specifications, security properties, tools, and methodology by ARM, its partners, and the broader academic community. The SRI-Cambridge CHERI-MIPS design and technology transition effort is entirely open source, and the formal analysis of CHERI-MIPS will also be entirely open, to maximize usefulness to adopters and to facilitate peer review and scientific reproducibility."

All our artifacts are open-source, with the exception described above of the current version of the Morello specification and proofs. This maximizes reusability by all parties, industrial and academic.

We have engaged in detail with Arm throughout the development of Morello, and continue to do so:

- Using our automated flow from ASL to Sail, we have worked from weekly drops of the Arm-internal ASL specification.
- In related work (funded by DSbD, not CIFV) we have developed an automated test generator that generates useful ISA instruction-sequence tests for Morello directly from the Sail version of the ISA specification, and measured the specification coverage of our tests and the Arm-internal Architecture Compliance Kit (ACK) tests. These tests have been used for Arm's Morello hardware design, pre-tape-out, and for test-driven-development of the University of Cambridge adaption of QEMU to Morello.
- Using the Sail-generated C emulator for Morello, we have validated the Sail version of the Morello ISA specification against those ACK tests, increasing confidence in our ASL-to-Sail flow and Sail tooling.

- We have provided feedback to Arm on a variety of issues in the specification, in their executable models, and in their tests, found by the above proof and testing work.

By providing theorem-prover definitions in the two main interactive theorem-provers in use today (Isabelle and Coq), especially for Armv8-A, RISC-V, CHERI-RISC-V, and CHERI-MIPS, we create opportunities for others to do formal work and proof about any or all of these.

5.0 Conclusions and Suggested Research Directions

Previous work on semantics and verification above mainstream processor architecture specifications has, with very few exceptions, had to work above highly idealized and simplified models. For example, projects such as the seL4 verified hypervisor and CompCert varied compiler rest above handwritten specifications of small fragments of the full ISAs. In contrast, CIFV has established full-scale (sequential) models, complete enough to boot operating systems, both for mainstream architectures (Armv8-A, RISC-V) and for the CHERI extensions thereof (Morello, CHERI-RISC-V, CHERI-MIPS). These are well-validated (e.g., against the Arm-internal Architecture Compliance Kit tests), they are available as open-source definitions in Sail and for various theorem provers, including Isabelle, Coq, and in some cases HOL4, and they support automated test generation. Above these, we have shown that one can establish machine-checked proofs of key security properties of complete ISA definitions, for Morello and CHERI-MIPS, and that one can do so at design time, feeding back into the Arm and UCam/SRI design teams during their work, rather than be post facto exercises.

The ability to do this at full scale creates many interesting directions for future work, including (among many other topics):

- Proof of further CHERI security properties. Our main results for Morello and CHERI-MIPS capture reachable capability monotonicity, essentially that software cannot forge capabilities. This is a property of arbitrary code running above the architecture. There are many other important properties that one would like to establish, especially properties that combine such architectural guarantees with properties of specific known code, e.g., of context-switching kernels.
- Demonstration that this work can be the foundation for more general software-facing verification activities, such as around secure hypervisors.
- Proofs of correctness for implementations of temporal safety above CHERI architectures.
- Demonstration in the long term that this methodology can be used to not just prove properties of an ISA once, but maintain them as the ISA is extended (as it inevitably will be mainstream ISAs continually add features). To some extent we did this during the Morello design process, but long-term deployment needs robustness of the proof technology under larger changes to the ISA.
- Use of these specifications as correctness criteria for hardware verification.
- Extension of the models to the full concurrent semantics of Armv8-A and RISC-V, including instruction fetch, instruction and data cache maintenance, virtual memory, interrupts, and so on, and beyond to SoC semantics, to eventually provide a basis for full-system verification.

- Further exploration of how one can use the models in lightweight ways, not just full correct- ness verification, e.g., in test generation, coverage-directed fuzzing, translation validation, and so on.

6.0 References

[1] Alasdair Armstrong, Thomas Bauereiss, Brian Campbell, Alastair Reid, Kathryn E. Gray, Robert M. Norton, Prashanth Mundkur, Mark Wassell, Jon French, Christopher Pulte, Shaked Flur, Ian Stark, Neel Krish- naswami, and Peter Sewell, "ISA semantics for ARMv8-A, RISC-V, and CHERI-MIPS," In *Proceedings of the 46th ACM SIGPLAN Symposium on Principles of Programming Languages*, January 2019. Proc. ACM Program. Lang. 3, POPL, Article 71. http://www.cl.cam.ac.uk/users/pes20/sail/sail-popl2019.pdf.

[2] Kyndylan Nienhuis, Alexandre Joannou, Thomas Bauereiss, Anthony Fox, Michael Roe, Brian Campbell, Matthew Naylor, Robert M. Nor- ton, Simon W. Moore, Peter G. Neumann, Ian Stark, Robert N. M. Watson, and Peter Sewell, "Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process," In *Proceedings of the 41st IEEE Symposium on Security and Privacy (SP)*, pages 1007–1024, May 2020. https://www.cl.cam.ac.uk/users/pes20/cheri-formal.pdf.

[3] Thomas Bauereiss, Brian Campbell, Thomas Sewell, Alasdair Armstrong, Lawrence Esswood, Ian Stark, Graeme Barnes, Robert N.M. Watson, and Peter Sewell, "Verified security for the Morello capability-enhanced prototype Arm architecture," Submitted for publication 2021-07. Submitted version shared with MTO and I2O. https://www.cl.cam.ac.uk/users/pes20/Stuff/morello-proofs-darpa.pdf.

[4] Ben Simner, Shaked Flur, Christopher Pulte, Alasdair Armstrong, Jean Pichon-Pharabod, Luc Maranget, and Peter Sewell, "ARMv8-A system semantics: instruction fetch in relaxed architectures," In *Proceedings of the 29th European Symposium on Programming*, April 2020. http://www.cl.cam.ac.uk/~pes20/iflat/top-extended.pdf.

[5] Alasdair Armstrong, Brian Campbell, Ben Simner, Christopher Pulte, and Peter Sewell, "Isla: Integrating full-scale ISA semantics and axiomatic concurrency models," In *In Proc. 33rd International Conference on Computer-Aided Verification*, July 2021, https://www.cl.cam.ac.uk/~pes20/isla/isla-cav2021.pdf.

[6] Robert N. M. Watson, Peter G. Neumann, Jonathan Woodruff, Michael Roe, Hesham Almatary, Jonathan Anderson, John Baldwin, David Chisnall, Brooks Davis, Nathaniel Wesley Filardo, Alexandre Joannou, Ben Laurie, A. Theodore Markettos, Simon W. Moore, Steven J. Murdoch, Kyndylan Nienhuis, Robert Norton, Alex Richardson, Peter Rugg, Peter Sewell, Stacey Son, and Hongyan Xia, "Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Ver- sion 7)," Technical Report UCAM-CL-TR-927, University of Cambridge, Computer Laboratory, June 2019. 496pp. https://www.cl.cam.ac.uk/ techreports/UCAM-CL-TR-927.pdf.

[7] Robert N. M. Watson, Peter G. Neumann, Jonathan Woodruff, Michael Roe, Hes- ham Almatary, Jonathan Anderson, John Baldwin, Graeme Barnes, David Chisnall, Jessica Clarke, Brooks Davis, Lee Eisen, Nathaniel Wesley Filardo, Richard Grisenthwaite, Alexan- dre Joannou, Ben Laurie, A. Theodore Markettos, Simon W. Moore, Steven J. Murdoch, Kyn- dylan Nienhuis, Robert Norton, Alexander Richardson, Peter Rugg, Peter Sewell, Stacey Son, and Hongyan Xia, "Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 8)," Technical Report UCAM-CL-TR-951, University of Cambridge, Computer Laboratory, October 2020.
https://www.cl.cam.ac.uk/techreports/ UCAM-CL-TR-951.pdf.

[8] Robert N. M. Watson, Simon W. Moore, Peter Sewell, and Peter Neumann, "An introduction to CHERI," Technical Report UCAM-CL-TR-941, University of Cambridge, Computer Laboratory, September 2019.
https://www.cl.cam.ac.uk/techreports/ UCAM-CL-TR-941.pdf.

[9] Kyndylan Nienhuis, Alexandre Joannou, Anthony Fox, Michael Roe, Thomas Bauereiss, Brian Campbell, Matthew Naylor, Robert M. Norton, Simon W. Moore, Peter G. Neumann, Ian Stark, Robert N. M. Watson, and Peter Sewell, "Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process," Technical Report UCAM-CL-TR-940, University of Cambridge, Computer Laboratory, September 2019.
https://www.cl.cam.ac.uk/techreports/ UCAM-CL-TR-940.pdf.

[10] Robert N. M. Watson, Jonathan Woodruff, Alexandre Joannou, Simon W. Moore, Peter Sewell, and Arm Limited, "DSbD CHERI and Morello Capability Essential IP (Version 1)," Technical Report UCAM-CL-TR-953, University of Cambridge, Computer Laboratory, December 2020.
https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-953. pdf.