arm

Introduction to Morello

Richard Grisenthwaite

SVP Chief Architect and Fellow

Richard.Grisenthwaite@arm.com

Essentials of the CHERI architecture in one slide

INDUSTRIAL STRATEGY
UK Research and Innovation

- CPU architecture adds 128-bit “capabilities”
  - Capability contains the address, bounds information, permission information etc
  - Loads/stores using capabilities as addresses are checked to be legal
    - Within address range and matching the supplied permissions
  - Data processing on capabilities has rules to limit operations
    - Bounds cannot be arbitrarily increased, permissions cannot be relaxed etc
- A memory tagging bit is added as metadata that distinguishes a capability from normal data
    - This memory tagging bit prevents “forging” of a capability
    - This functionality gives strong provenance of capabilities
- Capability is used in place of a normal pointer in some or all situations
  - Exactly how when this happens is part of the software usage case
  - Simply replacing all pointers with capabilities gives scope for strong spatial memory protection
      - But clearly is an ABI change and increases cache pressure

arm

Why is Arm interested in the CHERI architecture

INDUSTRIAL STRATEGY
UK Research and Innovation

- Arm has been working with UoCambridge on CHERI for some 4-5 years
- Big step to addressing security based on strong fundamental principles
- Addresses spatial memory safety robustly and some ideas for temporal safety
  - Memory safety issues reported to be involved with ~70% of vulnerabilities (Matt Miller, BlueHat IL, 2019)
- Has scope to be the foundation of a new mechanism for compartmentalisation
  - Potentially far cheaper than using translation tables
- Interesting scope to address temporal safety issues as well as spatial ones….
- Many of the Arm software vendors are similarly interested in the possibilities of CHERI
  - Microsoft, Google and others have expressed strong interest in exploring the concept…
  - … but lots of questions about the real-world performance costs and usage models
  - …understanding the intended usage models is important to refine the architectural features
- But is a novel thing to do with additional costs to the system and software

arm

Performance effects of CHERI ?

INDUSTRIAL STRATEGY
UK Research and Innovation

- Spatial memory safety involves replacing some/all of the pointer with capabilities
  - 128-bit items in place of 64-bit items hits the effective cache size to an unknown degree
  - Are all pointers replaced by capabilities or just some of them (esp for Java/Javascript)
- How are the tags held in memory?
  - 129th bit (similar to ECC) or by carving out a separate area of memory
  - Do I need a tag-cache to hold the tag bits, is it hierarchical, what size is it etc etc
- What is the performance implications for using CHERI for compartmentalisation?
  - Can I measure the improved performance from doing this vs (ab)using the process model
  - If I have more lightweight compartmentalisation, how do I segment my software efficiently
    - What is the performance effects of doing this?
- What is the performance cost of using CHERI for temporal memory safety?
- How do any of these benefits compare for real performance vs today's established ways
  - Is the benefit worth the effort?

arm

The Morello Board

INDUSTRIAL STRATEGY
UK Research and Innovation

- An Industrial Demonstrator of a Capability architecture
- Uses a prototype capability extension to the Arm Architecture
  - Prototype is a “superset” of what could be adopted into the Arm architecture
- Use of a superset of the architecture is very unusual
  - Also unrealistic as a commercial product – there will be some frequency effects
  - However, there are tight timescales so architecture is nearly complete now
- The superset of the architecture will allow a lot of software experimentation
  - Various different mechanisms for compartmentalisation
  - Collection of features for which the justification is unclear
  - Techniques for holding the capability tag bit
- Architecture will have formally proved security properties (with UoC and UoE)
- **Morello Board will be the ONLY physical implementation of this prototype architecture**
  - **Learnings from these experiments will be adopted into a mainstream extension to the Arm architecture**
  - **NO COMMITMENT TO FULL BINARY COMPATIBILITY TO THE PROTOTYPE ARCHITECTURE**
    - **But successful concepts are expected to be carried forward into the architecture and can be reused there**

arm

Morello Board overview

INDUSTRIAL STRATEGY
UK Research and Innovation

- Quad core bespoke high-end CPU with prototype capability extensions
  - Backwards compatibility with v8.2 AArch64-only
  - Based on Neoverse N1 core
    - Multi-issue out-of-order superscalar core with 3 levels of cache
  - Build in 7nm process
  - Targeting clock frequency around 2GHz
- Reasonable performance GPU and Display controller
  - Standard Mali architecture core – not extended with capability
  - Supports Android
- PCIe and CCIx interfaces including to FPGA based accelerators
- FPGA for peripheral expansion
- SBSA compliant system
- 16GB of System Memory (expandable to 32GB – tbc)

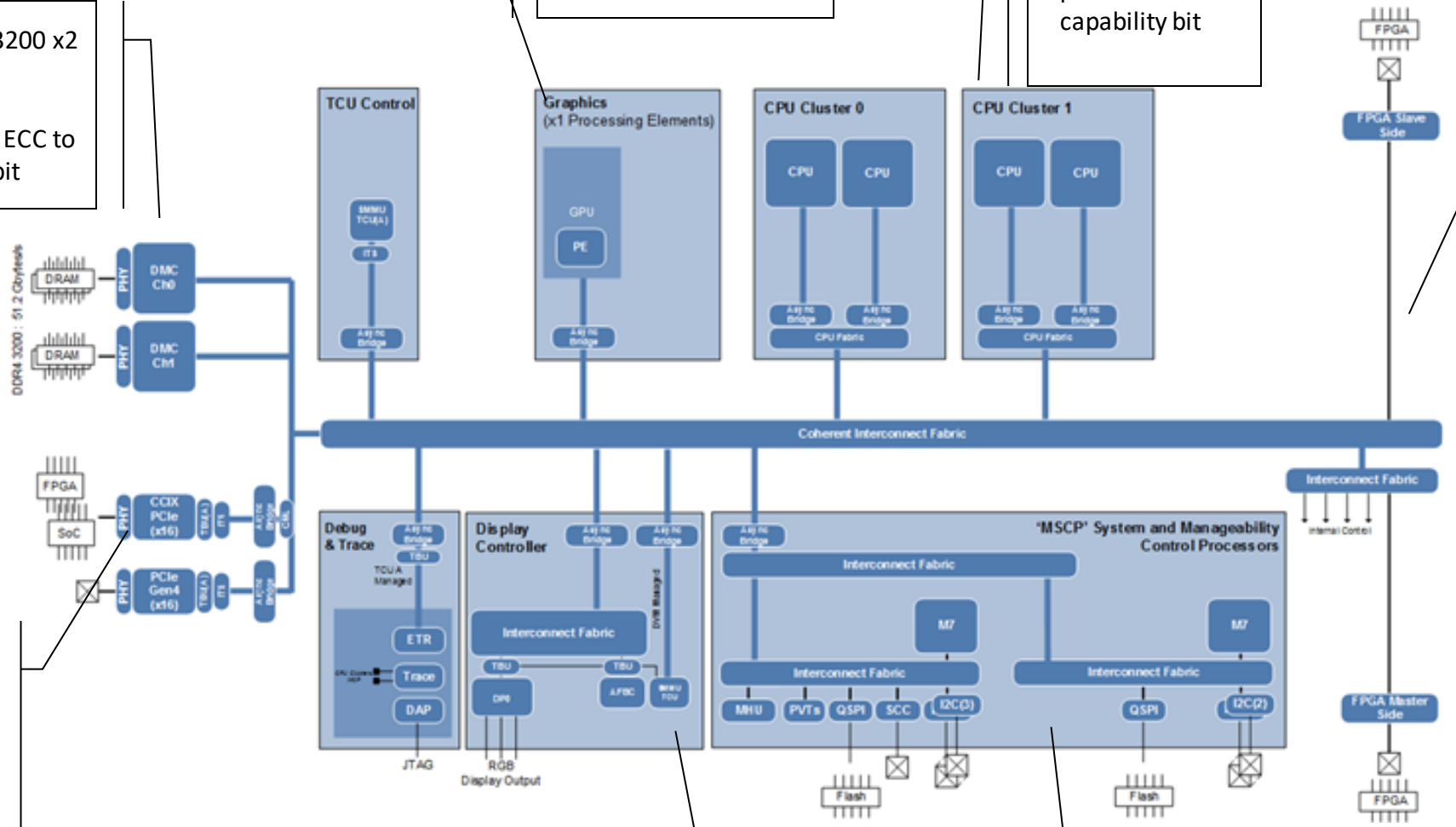
arm

Morello SoC (WIP)

- Mid-range GPU
- Single shader
- 256KByte L2

- Quad Arm core with capabilities
- L1/L2 cache modifications to proliferate capability bit

- SODIMM DDR4 3200 x2 (72pin)
- 51.2 GBytes/s
- Modifications to ECC to store capability bit



- Thin Links to FPGA
- Facilitates a broader set of IO not contained within the SoC itself

- High-end PCIe configuration
- x16 PCIe CCIX enabled
- x16 PCIe IO
- Can't carry capability tags

- Display processor
- Single display output
- Digital 8:8:8 RGB Output
- UXGA60 : 1600 x 1200

- SCP & MCP System control including boot

Software and Tools on Morello Platform

INDUSTRIAL STRATEGY
UK Research and Innovation

- Initial toolchain development is focussed on the LLVM toolchain (including LLDB)
  - GNU tools being developed as a secondary activity

- Initial OS focus is FreeBSD (developed with UoCambridge), Android
  - Secondary focus: Windows PE, Yocto (Linux Distribution for IoT) ,
  - Tertiary focus: Debian, RedHat Fedora, SuSE Tumbleweed,

arm

Timescales

INDUSTRIAL STRATEGY
UK Research and Innovation

- October 2020:
  - Virtual Platform Model of Morello board (behavioural software model)
  - Architecture Specification of the CPU architecture used in the Morello board
    - This will include XML and Pseudo-code to allow formal proofs and other auto-generated collateral
- December 2021
  - Morello boards made available with initial software and toolchains

arm

What do we want to get from this…

INDUSTRIAL STRATEGY
UK Research and Innovation

- Answers to the performance questions for a wide range of different usage models
- Compelling examples of Capabilities offering a security/performance improvements
  - Backed up by “Red-teams” having attacked the system and demonstrated security of the system
  - Compelling in comparison with existing deployed state of the art approaches
- Understanding of how different languages and run-times can use capabilities
  - Not just C and C++, but also Javascript, Java
- Far better understanding of how fine-grained compartmentalisation can be used
- A showcase to encourage other architectures to adopt capabilities
- Experience of what the right SoC hardware is for building capabilities
- An architectural approach with formally proven security properties

=> What to put into the Arm architecture to support Capabilities in the future

arm

arm

Questions?

From CHERI to Morello: Architecture, Software, and Research

Robert N. M. Watson, Simon W. Moore, Peter Sewell, Peter G. Neumann

Hesham Almatary, Jonathan Anderson, Alasdair Armstrong, John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Khilan Gudka, Brett Gutstein, Alexandre Joannou, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Peter Sewell, Stacey Son, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb

University of Cambridge and SRI International

Arm Research Summit – September 2020

Approved for public release; distribution is unlimited. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 ("CTSRD"), with additional support from FA8750-11-C-0249 ("MRC2"), HR0011-18-C-0016 ("ECATS"), and FA8650-18-C-7809 ("CIFV") as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Approved for public release; distribution is unlimited.

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 ("CTSRD"), with additional support from FA8750-11-C-0249 ("MRC2"), HR0011-18-C-0016 ("ECATS"), and FA8650-18-C-7809 ("CIFV") as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

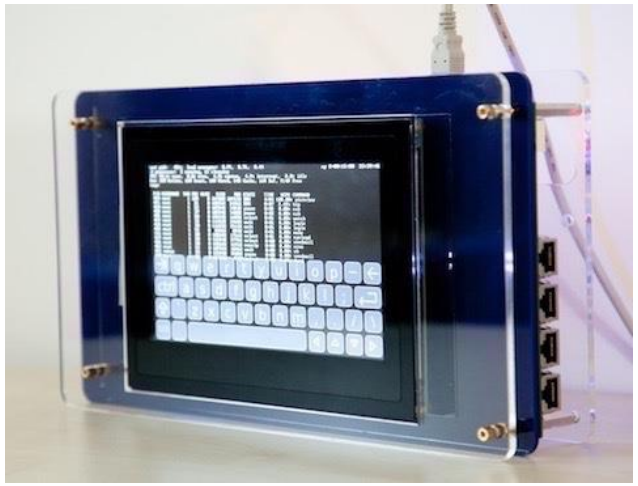
We also acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, and the Gates Cambridge Trust.

Introduction

- We will …
  - Briefly review the CHERI architectural primitives
  - Discuss CHERI's application to ARMv8-A in Morello
  - Describe our Morello-adapted CHERI prototype software stack
  - Identify potential future research directions building on CHERI and Morello
- To learn more about the CHERI architecture and prototypes:

  http://www.cheri-cpu.org/
- Watson, Moore, Sewell, and Neumann. **An Introduction to CHERI**, UCAM-CL-TR-941, September 2019.

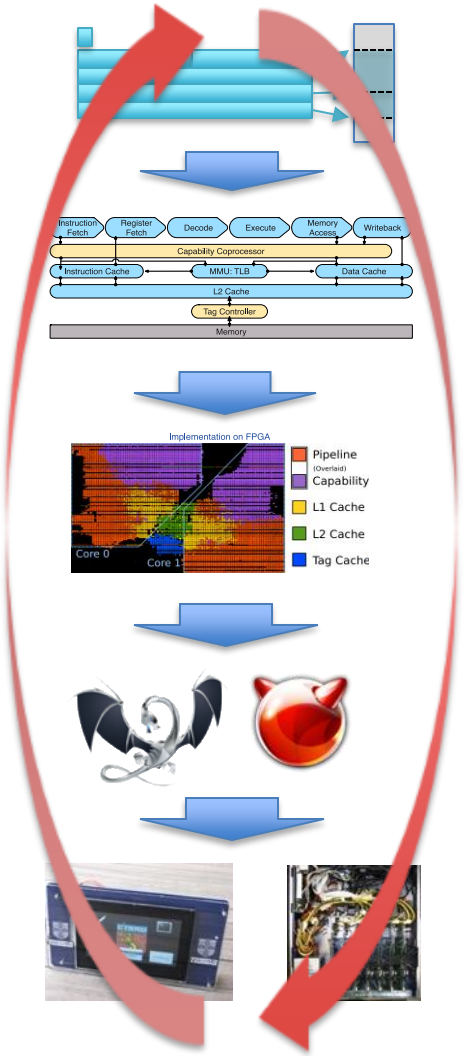
What is CHERI?



An early experimental FPGA-based CHERI tablet prototype running the CheriBSD operating system and applications, Cambridge, 2013

- SRI + Cambridge over 10 years + 3 DARPA programs (~$26M), EPSRC (£7.4M); Innovate (£2.7M); Google / DeepMind / Arm / HPE … (~£1M)
- CHERI is an **architectural protection model**
  - Composes a capability-system model with hardware and software
  - Adds new security primitives to Instruction-Set Architectures (ISAs)
  - Implemented by microarchitectural extensions to the CPU/SoC
  - Enables new security behavior in software
- CHERI mitigates vulnerabilities in **C/C++ Trusted Computing Bases**
  - Hypervisors, operating systems, language runtimes, browsers, ….
  - Fine-grained memory protection, scalable compartmentalization
  - Directly impedes common exploit-chain tools used by attackers
  - Mitigates many vulnerability classes .. even unknown future classes

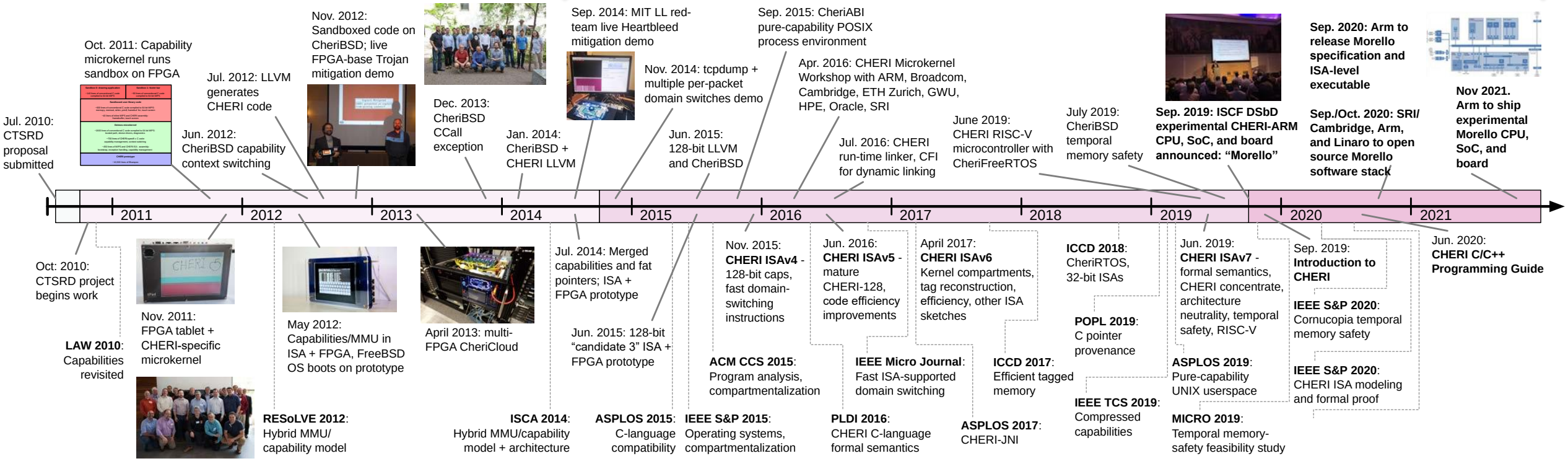
Hardware-software-semantics co-design



- Architectural mitigation for C/C++ TCB vulnerabilities
  - Tagged memory, new hardware capability data type
  - Model hybridizes cleanly with contemporary RISC ISAs, CPU designs, MMU-based OSes, and C/C++-language software
  - New hardware enables incremental software deployment
- Hardware-software-semantics co-design
  - CHERI abstract protection model; concrete ISA instantiations in 64-bit MIPS, 32/64-bit RISC-V, 64-bit ARMv8-A
  - Formal ISA models, Qemu-CHERI, and multiple FPGA prototypes
  - Formal proofs that ISA security properties are met, automatic testing
  - CHERI Clang/LLVM/LLD, CheriBSD, C/C++-language applications
  - Repeated iteration to improve {performance, security, compatibility, ..}

CHERI research and development timeline

SRI/Cambridge collaboration with Arm to develop CHERI adaptation of ARMv8-A + port of the CHERI software stack



Years 1-2: Research platform, prototype architecture

Years 2-4: Hybrid C/OS model, compartment model

Years 4-7: Efficiency, CheriABI/C/C++/linker, ARMv8-A

Years 8-10: RISC-V, temporal safety, formal proof

SRI International

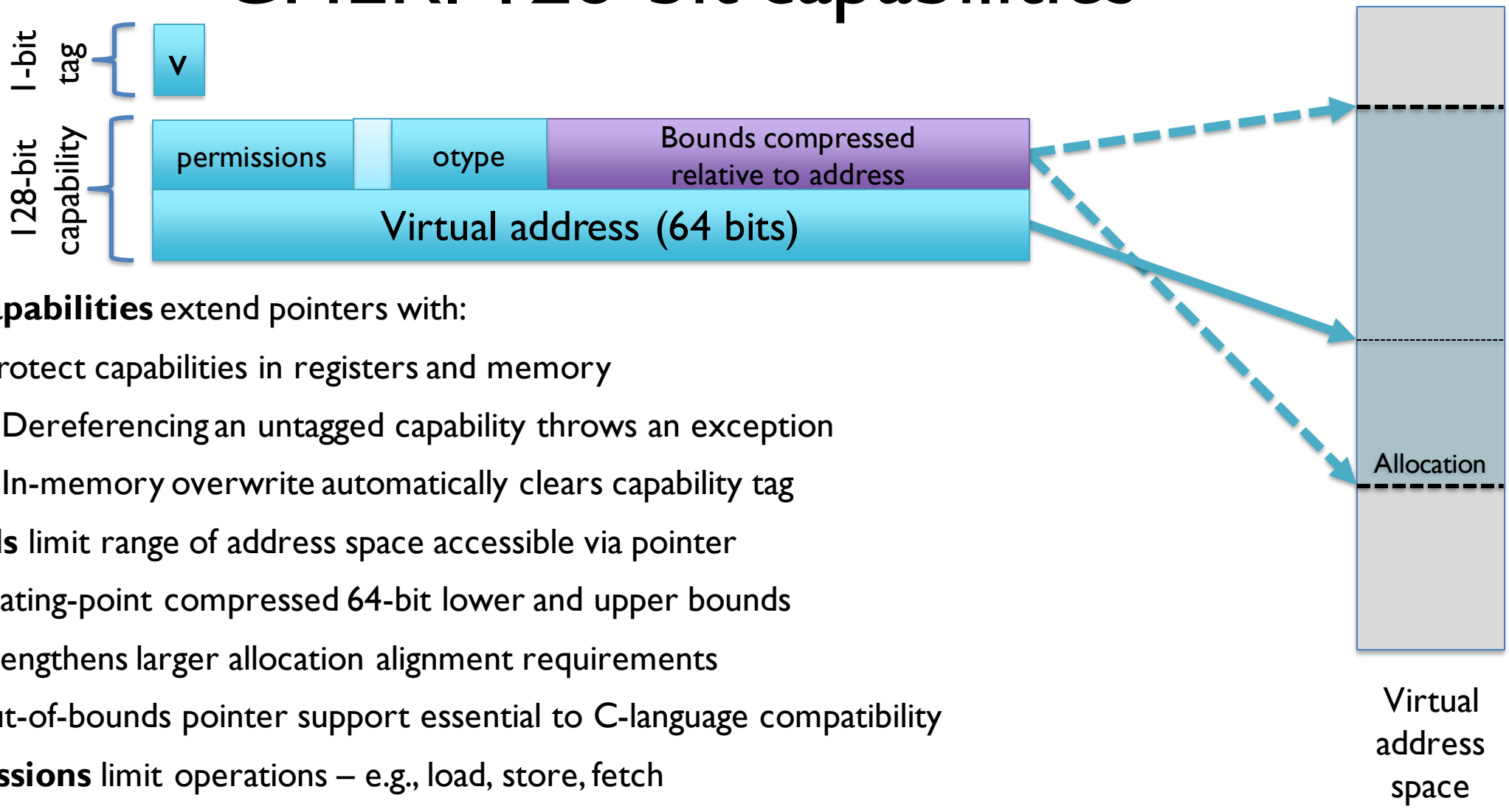
UNIVERSITY OF CAMBRIDGE

CHERI PROTECTION MODEL AND ARCHITECTURE

CHERI design goals and approach

- **De-conflate memory virtualization and protection**
  - Memory Management Units (MMUs) protect by **location (address)**
  - CHERI protects existing **references (pointers)** to code, data, objects
  - Reusing **existing pointer indirection** avoids adding new architectural table lookups
- **Architectural mechanism** that enforces **software policies**
  - **Language-based properties** – e.g., referential, spatial, and temporal integrity (C/C++ compiler, linkers, OS model, runtime, …)
  - **New software abstractions** – e.g., software compartmentalization (confined objects for in-address-space isolation, …)

CHERI 128-bit capabilities

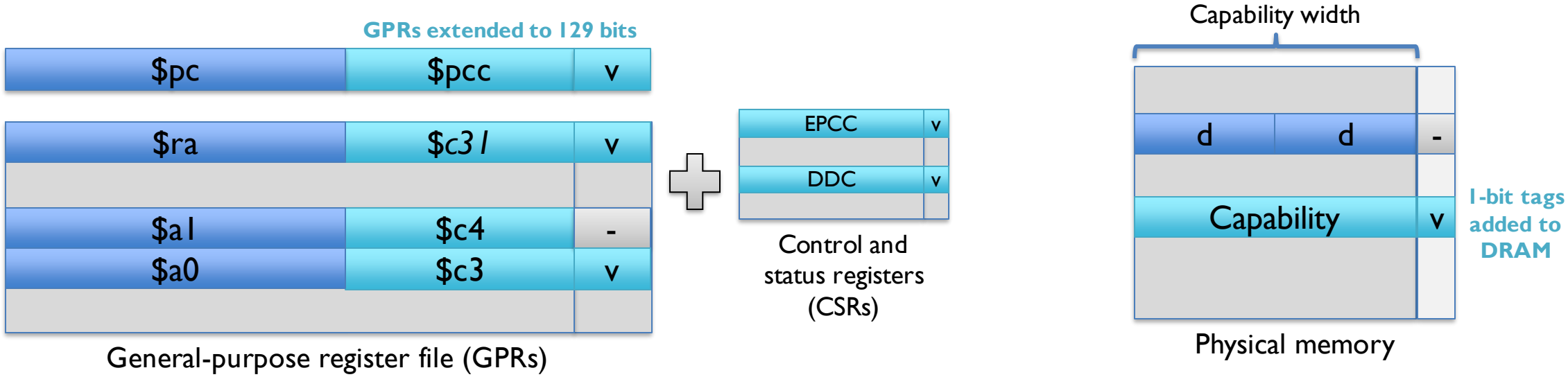


CHERI capabilities extend pointers with:

- **Tags** protect capabilities in registers and memory
  - Dereferencing an untagged capability throws an exception
  - In-memory overwrite automatically clears capability tag
- **Bounds** limit range of address space accessible via pointer
  - Floating-point compressed 64-bit lower and upper bounds
  - Strengthens larger allocation alignment requirements
  - Out-of-bounds pointer support essential to C-language compatibility
- **Permissions** limit operations – e.g., load, store, fetch
- **Sealing**: **immutable**, **non-dereferenceable capabilities** – used for non-monotonic transitions

Merged capability register file + tagged memory

(as found in Morello and CHERI-RISC-V; MIPS used a split register file)



- **64-bit general-purpose registers (GPRs)** are extended with **64 bits of metadata** and a **1-bit validity tag**
- **Program counter (PC)** is extended to be the **program-counter capability ($PCC)**
- **Default data capability ($DDC)** constrains legacy integer-relative ISA load and store instructions
- **Tagged memory** protects capability-sized and -aligned words in DRAM by adding a **1-bit validity tag**
- **Various system mechanisms** are extended (e.g., capability-instruction enable control register, new TLB/PTE permission bits, exception code extensions, saved exception stack pointers and vectors become capabilities, etc.)

CHERI ISA refinement over 10 years

Technical Report

UCAM-CL-TR-927
ISSN 1476-2986

Number 927

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7)

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, Hongyan Xia

June 2019

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
https://www.cl.cam.ac.uk/

Year	Version	Description
2010-2012	ISAv1	RISC capability-system model w/64-bit MIPS Capability registers, tagged memory Guarded manipulation of registers
2012	ISAv2	Extended tagging to capability registers Capability-aware exception handling Boots an MMU-based OS with CHERI support
2014	ISAv3	Fat pointers + capabilities, compiler support Instructions to optimize hybrid code Sealed capabilities, CCall/CReturn
2015	ISAv4	MMU-CHERI integration (TLB permissions) ISA support for compressed 128-bit capabilities HW-accelerated domain switching Multicore instructions: full suite of LL/SC variants
2016	ISAv5	CHERI-128 compressed capability model Improved generated code efficiency Initial in-kernel privilege limitations
2017	ISAv6	Mature kernel privilege limitations Further generated code efficiency Architectural portability: CHERI-x86, CHERI-RISC-V sketches Exception-free domain transition
2019	ISAv7	Architectural performance optimization for C++ applications Microarchitectural side-channel resistance features Architecture-neutral CHERI protection model All instruction pseudocode from a formal model CHERI Concentrate capability compression Improved C-language support, dynamic linking, sentry capabilities Elaborated CHERI-RISC-V ISA 64-bit capabilities for 32-bit architectures Accelerated tag operations for temporal memory safety
Expected 2020Q3/4	ISAv8	MMU temporal memory-safety assist; e.g., capability dirty bit Optimizations for sentry capabilities CHERI-RISC-V privileged support, general maturity Further C-language semantics improvements

Capabilities + RISC

C/C++ and capabilities

Compartmentalization

128-bit, code efficiency

Multicore

In-kernel use

Temporal memory safety

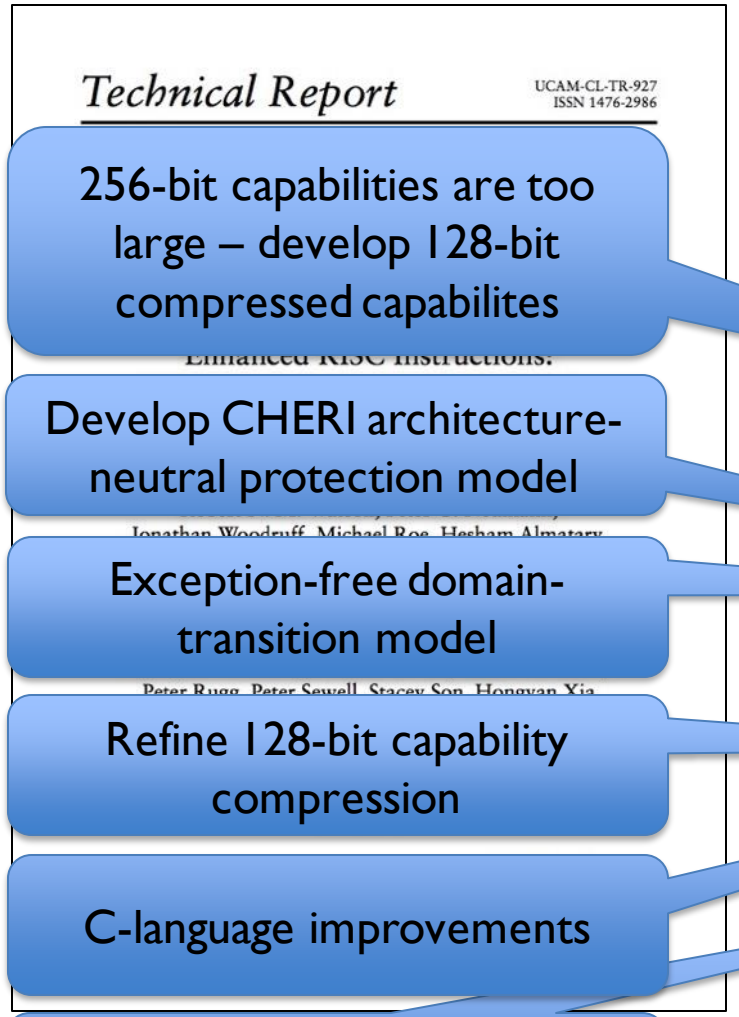
Non-MIPS ISAs: ARMv8-A, ARMv8-M, RISC-V, x86-64

Arm Morello architecture synchronization point

SRI International

UNIVERSITY OF CAMBRIDGE

CHERI ISA refinement over 10 years


Technical Report

UCAM-CL-TR-927
ISSN 1476-2986

Enhanced RISC Instructions:

Jonathan Woodruff, Michael Roe, Hesham Almatary

Peter Rugg, Peter Sewell, Stacey Son, Hongyan Xia

256-bit capabilities are too large – develop 128-bit compressed capabilites

Develop CHERI architecture-neutral protection model

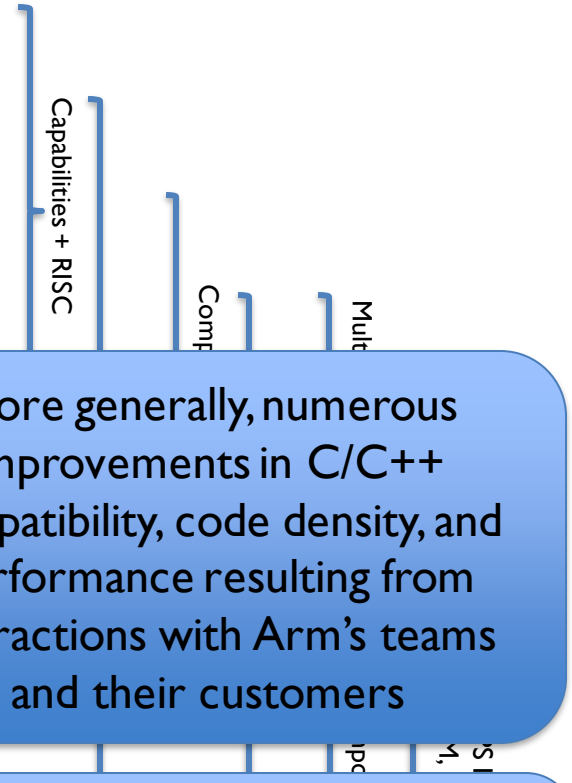
Exception-free domain-transition model

Refine 128-bit capability compression

C-language improvements

Develop and optimize CHERI temporal memory safety

Year	Version	Description
2010-2012	ISAv1	RISC capability-system model w/64-bit MIPS Capability registers, tagged memory Guarded manipulation of registers
2012	ISAv2	Extended tagging to capability registers Capability-aware exception handling Boots an MMU-based OS with CHERI support
2014	ISAv3	Fat pointers + capabilities, compiler support Instructions to optimize hybrid code Sealed capabilities, CCall/CReturn
	ISAv4	MMU-CHERI integration (TLB permissions) ISA support for compressed 128-bit capabilities HW-accelerated domain switching Multicore instructions: full suite of LL/SC variants
2016	ISAv5	**CHERI-128 compressed capability model** Improved generated code efficiency Initial in-kernel privilege limitations
2017	ISAv6	Mature kernel privilege limitations Further generated code efficiency **Architectural portability: CHERI-x86, CHERI-RISC-V sketch** **Exception-free domain transition**
2019	ISAv7	Architectural performance optimization for C++ applications Microarchitectural side-channel resistance features **Architecture-neutral CHERI protection model** All instruction pseudocode from a formal model **CHERI Concentrate capability compression** **Improved C-language support**, dynamic linking, sentry capabilities Elaborated CHERI-RISC-V ISA 64-bit capabilities for 32-bit architectures **Accelerated tag operations for temporal memory safety**
Expe… 2020Q3/4	ISAv8	**MMU temporal memory-safety assist; e.g., capability dirty bit** Optimizations for sentry capabilities CHERI-RISC-V privileged support, general maturity **Further C-language semantics improvements**



More generally, numerous improvements in C/C++ compatibility, code density, and performance resulting from interactions with Arm's teams and their customers

Microarchitectural work such as efficient tag storage in unmodified DRAM using a tag controller / tag cache

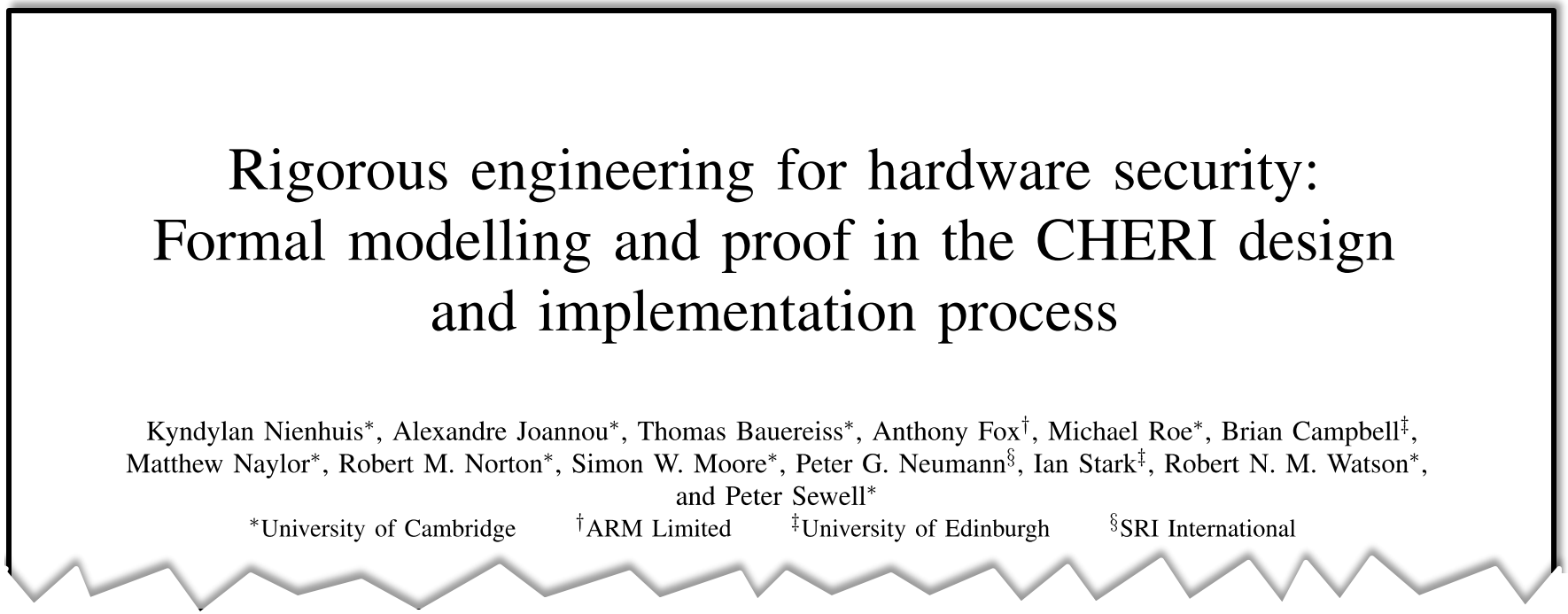
Arm Morello architecture synchronization point

UNIVERSITY OF CAMBRIDGE

High-level CHERI ISAv8 themes

- CHERI ISAv7 released in June 2019; working on ISAv8 release in 2020Q3/4:
  - Several 'experimental' features are no longer considered experimental (e.g., sentry capabilities, baseline temporal memory-safety features)
  - New instructions, MMU behavior to improve to temporal memory-safety performance
  - Performance optimization for sentry capabilities
  - Minor C-language semantics improvements
  - Various CHERI-RISC-V improvements (SCRs, exceptions, alignment, …)
  - Continued general improvements to descriptions, rationales, etc.
- **Aim is to ensure that Arm's Morello and CHERI ISAv8 be synchronized**
  - **… including new temporal memory-safety features**

Recent publications: ISA formal modelling and verification



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

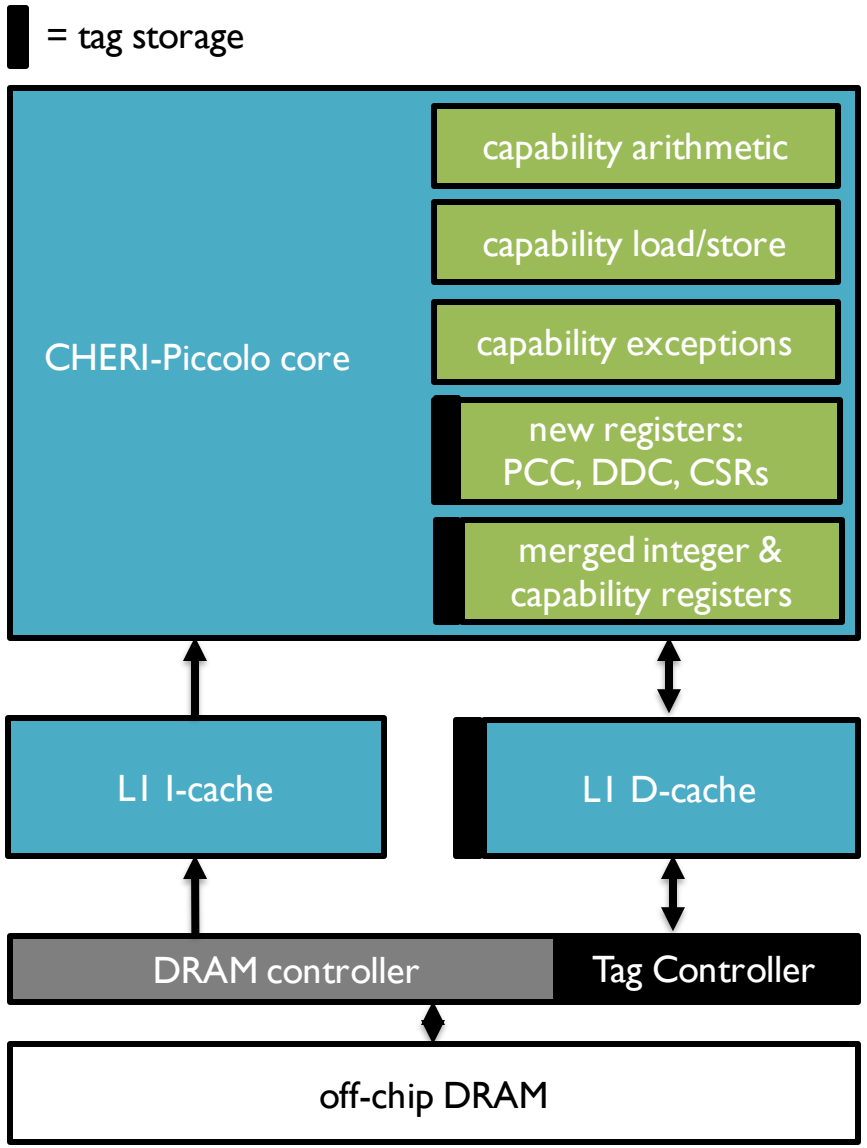
Kyndylan Nienhuis*, Alexandre Joannou*, Thomas Bauereiss*, Anthony Fox†, Michael Roe*, Brian Campbell‡, Matthew Naylor*, Robert M. Norton*, Simon W. Moore*, Peter G. Neumann§, Ian Stark‡, Robert N. M. Watson*, and Peter Sewell*

*University of Cambridge †ARM Limited ‡University of Edinburgh §SRI International

- IEEE Symposium on Security and Privacy ("Oakland"), May 2020
- Formal ISA model for CHERI-MIPS used for rigorous engineering
- Formal modelling of proof of compartmentalization properties

CHERI MICROARCHITECTURE

Example microarchitecture: CHERI-Piccolo microcontroller



Changes to the Piccolo core (RISC-V 3-stage pipeline):

- capability arithmetic
- capability load/store operations with bounds checking
- extended exception model
- PC becomes a capability (PCC)
- default data capability (DDC)
- new control/status registers
- merged integer & capability register file

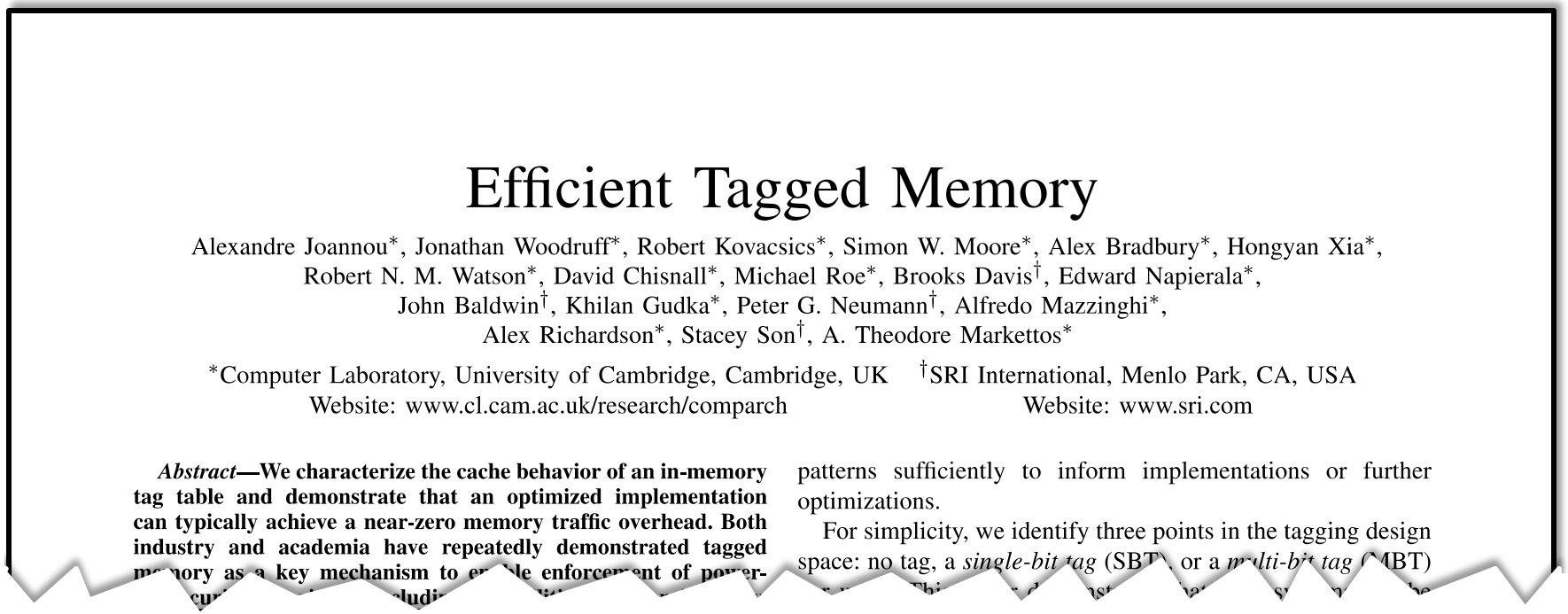
Memory subsystem:

- AXI user-field added to transport tag bits & data width doubled
- caches extended to include tags

DRAM changes:

- New tag controller uses a hierarchical tag table to efficiently store tag bits backed by top of DRAM

Microarchitectural tag storage for off-the-shelf DRAM



Efficient Tagged Memory

Alexandre Joannou*, Jonathan Woodruff*, Robert Kovacsics*, Simon W. Moore*, Alex Bradbury*, Hongyan Xia*,
Robert N. M. Watson*, David Chisnall*, Michael Roe*, Brooks Davis†, Edward Napierala*,
John Baldwin†, Khilan Gudka*, Peter G. Neumann†, Alfredo Mazzinghi*,
Alex Richardson*, Stacey Son†, A. Theodore Markettos*

*Computer Laboratory, University of Cambridge, Cambridge, UK
Website: www.cl.cam.ac.uk/research/comparch

†SRI International, Menlo Park, CA, USA
Website: www.sri.com

***Abstract*—We characterize the cache behavior of an in-memory tag table and demonstrate that an optimized implementation can typically achieve a near-zero memory traffic overhead. Both industry and academia have repeatedly demonstrated tagged**

patterns sufficiently to inform implementations or further optimizations.

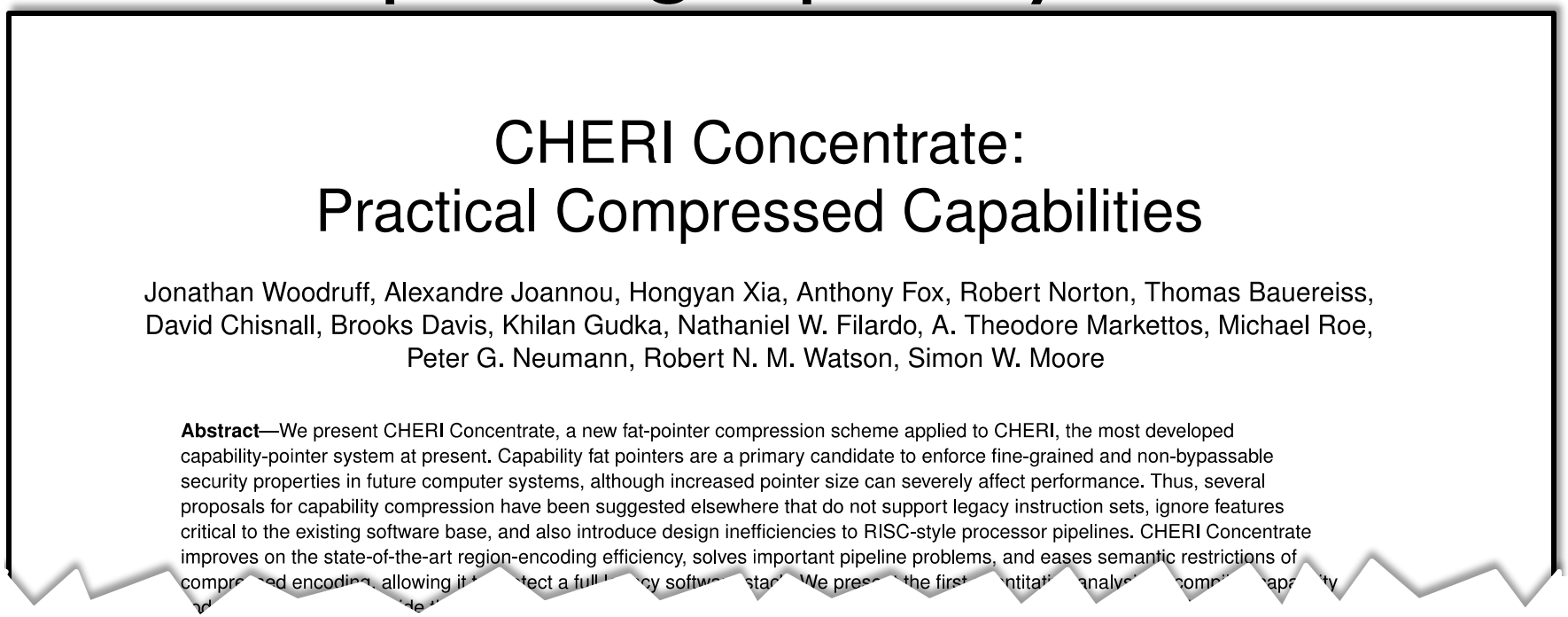
For simplicity, we identify three points in the tagging design space: no tag, a *single-bit tag* (SBT), or a *multi-bit tag* (MBT)

- Published in the IEEE International Conference on Computer Design (ICCD) 2017
- Shift from flat to hierarchal tag table to hold tags in DRAM
  - Exploit inconsistent density of tags in physical memory
  - Reduces DRAM access overhead for a variety of workloads

SRI International

UNIVERSITY OF CAMBRIDGE

Compressing capability bounds



CHERI Concentrate:
Practical Compressed Capabilities

Jonathan Woodruff, Alexandre Joannou, Hongyan Xia, Anthony Fox, Robert Norton, Thomas Bauereiss, David Chisnall, Brooks Davis, Khilan Gudka, Nathaniel W. Filardo, A. Theodore Markettos, Michael Roe, Peter G. Neumann, Robert N. M. Watson, Simon W. Moore

Abstract—We present CHERI Concentrate, a new fat-pointer compression scheme applied to CHERI, the most developed capability-pointer system at present. Capability fat pointers are a primary candidate to enforce fine-grained and non-bypassable security properties in future computer systems, although increased pointer size can severely affect performance. Thus, several proposals for capability compression have been suggested elsewhere that do not support legacy instruction sets, ignore features critical to the existing software base, and also introduce design inefficiencies to RISC-style processor pipelines. CHERI Concentrate improves on the state-of-the-art region-encoding efficiency, solves important pipeline problems, and eases semantic restrictions of

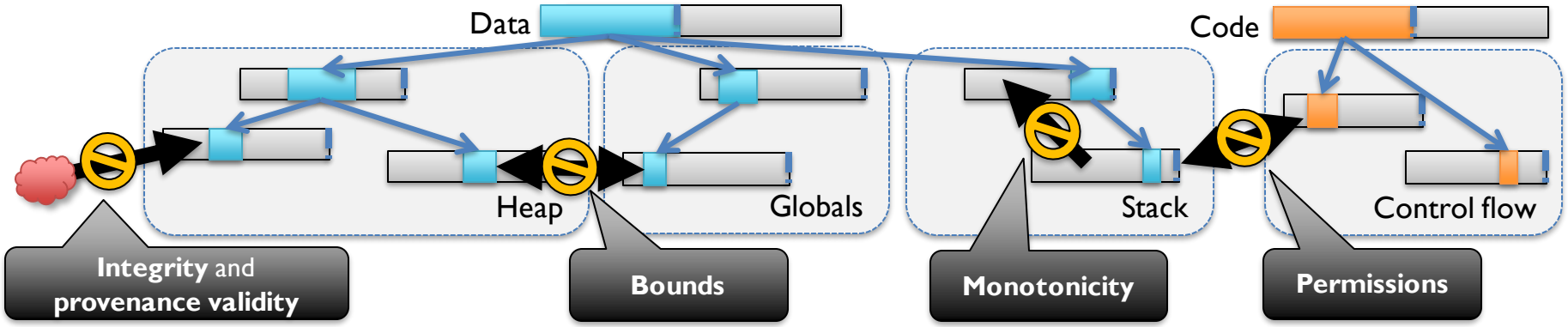
- Published in IEEE Transactions on Computers, April 2019
- Efficient compressed capabilities for 32-bit and 64-bit processors
  - Reduces size of capabilities from 4x machine word size to 2x
  - Large reduction in cache overheads
  - Efficiently fits into a RISC pipeline with negligible impact on clock frequency
  - Maintains all security and software compatibility properties

SRI International

UNIVERSITY OF CAMBRIDGE

SOFTWARE ON CHERI

Implementing pointer protection using capabilities



- **Integrity** and **provenance validity** ensure that valid pointers are derived from other valid pointers via valid transformations; **invalid pointers cannot be used**
  - Valid pointers, once removed, cannot be reintroduced solely unless rederived from other valid pointers
  - E.g., Received network data cannot be interpreted as a code/data pointer – even previously leaked pointers
- **Bounds** prevent pointers from being manipulated to access the wrong object
  - Bounds can be minimized by software – e.g., stack allocator, heap allocator, linker
- **Monotonicity** prevents pointer privilege escalation – e.g., broadening bounds
- **Permissions** limit unintended use of pointers; e.g., W^X for pointers
- These primitives not only allow us to implement **strong spatial and temporal memory protection**, but also higher-level policies such as **scalable software compartmentalization**

What are CHERI's implications for software?

- Efficient fine-grained **architectural memory protection** enforces:

  Provenance validity: Q: Where do pointers come from?

  Integrity: Q: How do pointers move in practice?

  Bounds, permissions: Q: What rights should pointers carry?

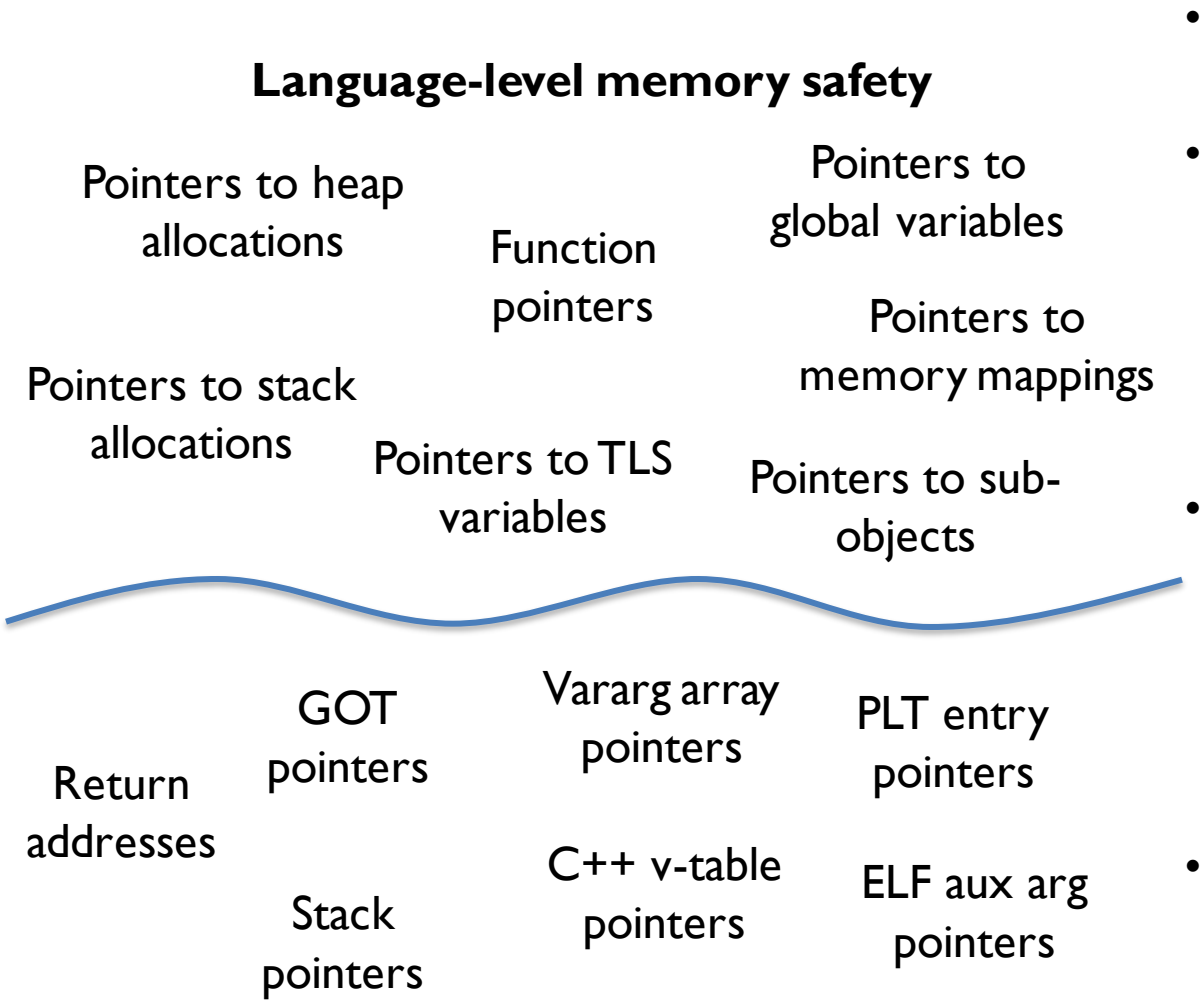
  Monotonicity: Q: Can real software play by these rules?

- Scalable fine-grained **software compartmentalization**

  Q: Can we construct **isolation** and **controlled communication** using integrity, provenance, bounds, permissions, and monotonicity?

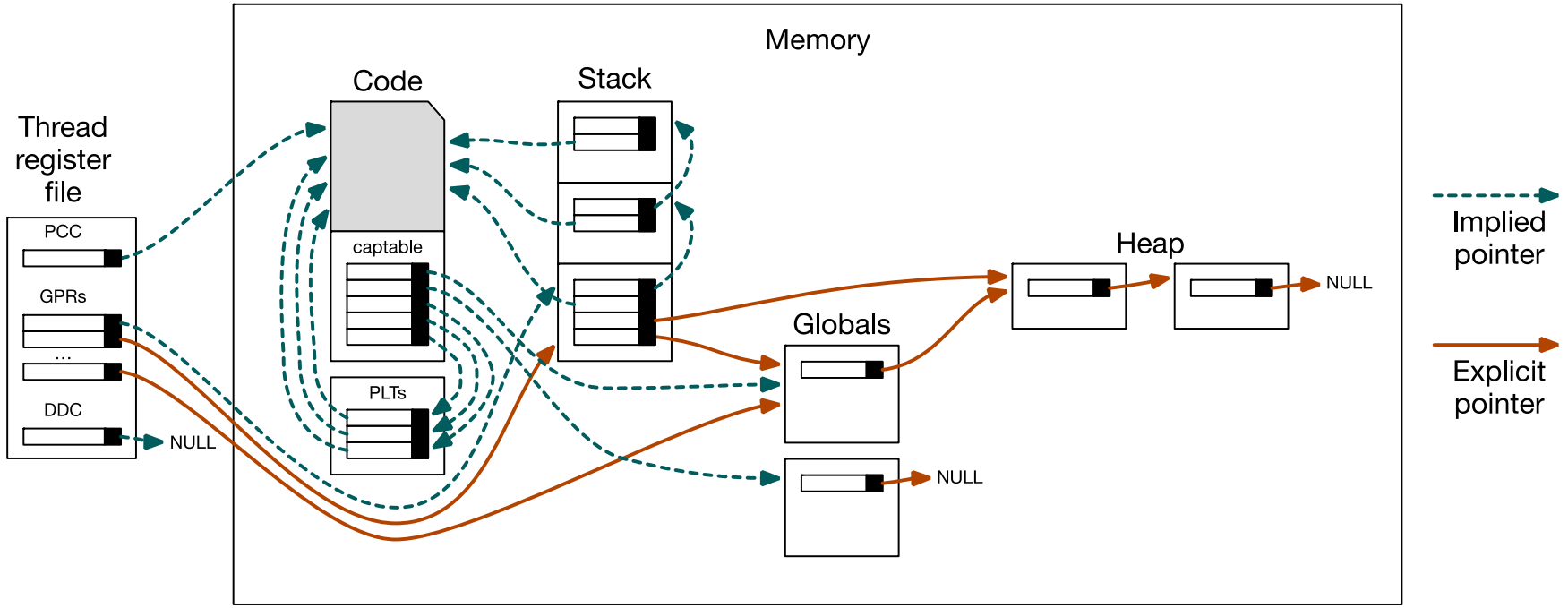
  Q: Can **sealed capabilities**, **controlled non-monotonicity**, and **capability-based sharing** enable safe, efficient compartmentalization?

Memory protection for the language and the language runtime



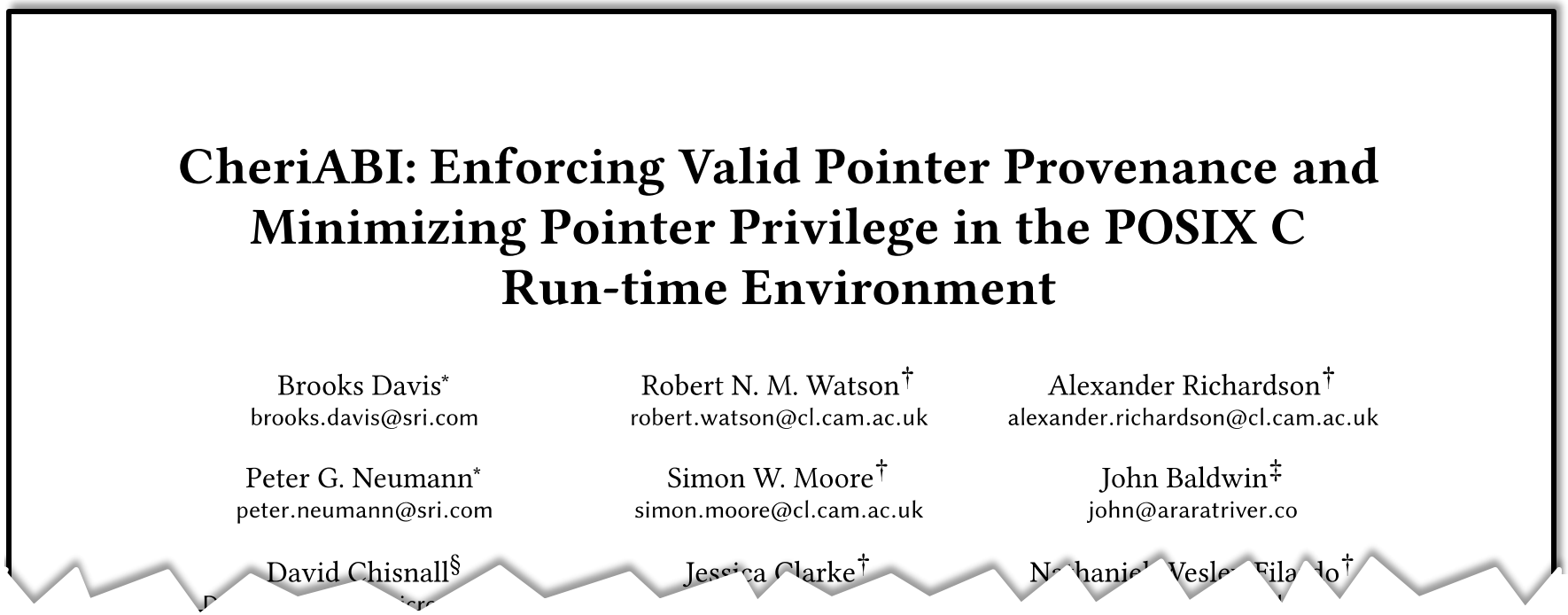
- Capabilities are refined by the kernel, run-time linker, compiler-generated code, heap allocator, …
- Protection mechanisms:
  - Referential memory safety
  - Spatial memory safety + privilege minimization
  - Temporal memory safety
- Applied **automatically** at two levels:
  - **Language-level pointers** point explicitly at stack and heap allocations, global variables, …
  - **Sub-language pointers** used to implement control flow, linkage, etc.
- Sub-language protection mitigates bugs in the language runtime and generated code, as well as attacks that cannot be mitigated by higher-level memory safety
  - (e.g., union type confusion)

CHERI-based pure-capability process memory



- Capabilities are substituted for integer addresses throughout the address space
- Bounds and permissions are minimized by software including the kernel, run-time linker, memory allocator, and compiler-generated code
- Hardware permits fetch, load, and store only through granted capabilities
- Tags ensure integrity and provenance validity of all pointers

Pure-capability userspace



CheriABI: Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Run-time Environment

Brooks Davis*
brooks.davis@sri.com

Robert N. M. Watson†
robert.watson@cl.cam.ac.uk

Alexander Richardson†
alexander.richardson@cl.cam.ac.uk

Peter G. Neumann*
peter.neumann@sri.com

Simon W. Moore†
simon.moore@cl.cam.ac.uk

John Baldwin‡
john@araratriver.co

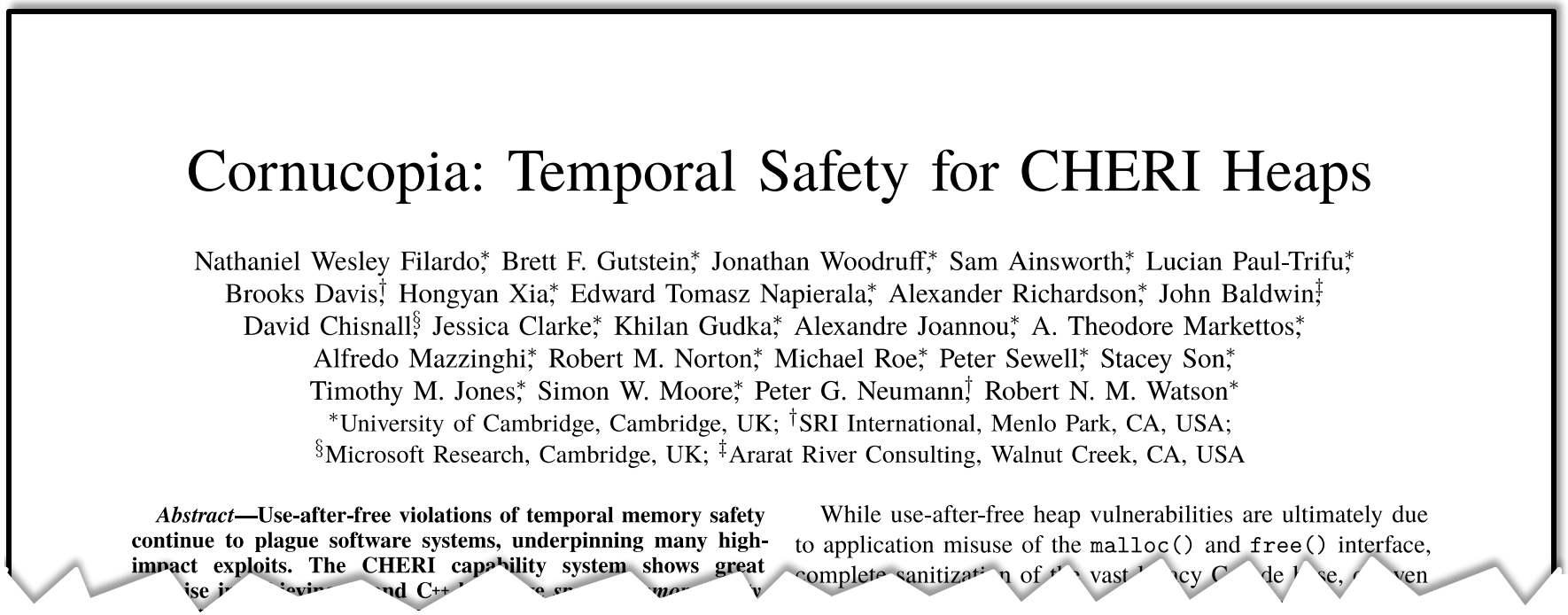
David Chisnall§

Jessica Clarke†

Nathaniel Vesle Filardo†

- Received best paper award at ASPLOS, April 2019
- Complete pure-capability UNIX OS userspace with spatial memory safety
  - Usable for daily development tasks
  - Almost vast majority of FreeBSD tests pass
  - Management interfaces (e.g. ioctl), debugging, etc., work
  - Large, real-world applications have been ported: PostgreSQL and WebKit

Heap temporal memory safety



Cornucopia: Temporal Safety for CHERI Heaps

Nathaniel Wesley Filardo[*], Brett F. Gutstein[*], Jonathan Woodruff[*], Sam Ainsworth[*], Lucian Paul-Trifu[*], Brooks Davis[†], Hongyan Xia[*], Edward Tomasz Napierala[*], Alexander Richardson[*], John Baldwin[‡], David Chisnall[§], Jessica Clarke[*], Khilan Gudka[*], Alexandre Joannou[*], A. Theodore Markettos[*], Alfredo Mazzinghi[*], Robert M. Norton[*], Michael Roe[*], Peter Sewell[*], Stacey Son[*], Timothy M. Jones[*], Simon W. Moore[*], Peter G. Neumann[†], Robert N. M. Watson[*]
[*]University of Cambridge, Cambridge, UK; [†]SRI International, Menlo Park, CA, USA;
[§]Microsoft Research, Cambridge, UK; [‡]Ararat River Consulting, Walnut Creek, CA, USA

***Abstract*—Use-after-free violations of temporal memory safety continue to plague software systems, underpinning many high-impact exploits. The CHERI capability system shows great**

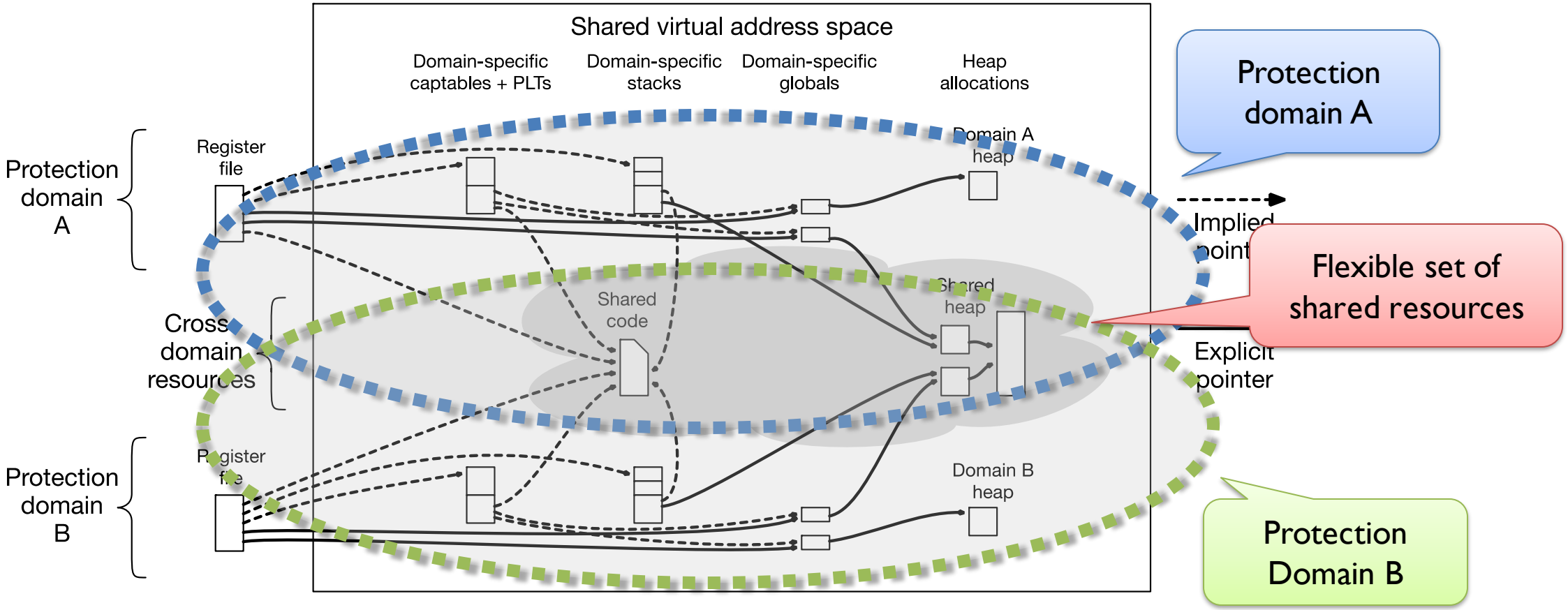
While use-after-free heap vulnerabilities are ultimately due to application misuse of the `malloc()` and `free()` interface,

- IEEE Symposium on Security and Privacy ("Oakland"), May 2020
- Hardware and software support for deterministic temporal memory safety for C/C++-language heaps using capability revocation
- Hardware enables fast tag searching using MMU-assisted tracking of tagged values, tag controller and cache

SRI International

UNIVERSITY OF CAMBRIDGE

CHERI-based compartmentalization



- Isolated compartments can be created using closed graphs of capabilities, combined with a constrained non-monotonic domain-transition mechanism

Opportunities and challenges

- CHERI dramatically improves compartmentalization scalability
  - More compartments
  - More frequent domain transitions
  - Faster shared memory between compartments
- Many potential use cases – e.g., sandbox processing of each image in a web browser, processing each message in a mail application
- Unlike memory protection, software compartmentalization also requires careful software refactoring to support strong encapsulation, and affects software operational model

Operational models for CHERI compartmentalization

- An **architectural protection model** enabling new software behavior
- As with virtual memory, multiple **operational models** can be supported
  - E.g., with an MMU: Microkernels, processes, virtual machines, etc.
  - How are compartments created/destroyed? Function calls vs. message passing? Signaling, debugging, …?
- We have explored two viable CHERI-based models to date:

Isolated dynamic libraries	Efficient but simple sandboxing in processes
UNIX co-processes	Multiple processes share an address space

- Improved performance and new paradigms using CHERI primitives
- Both will be available in CheriBSD/Morello

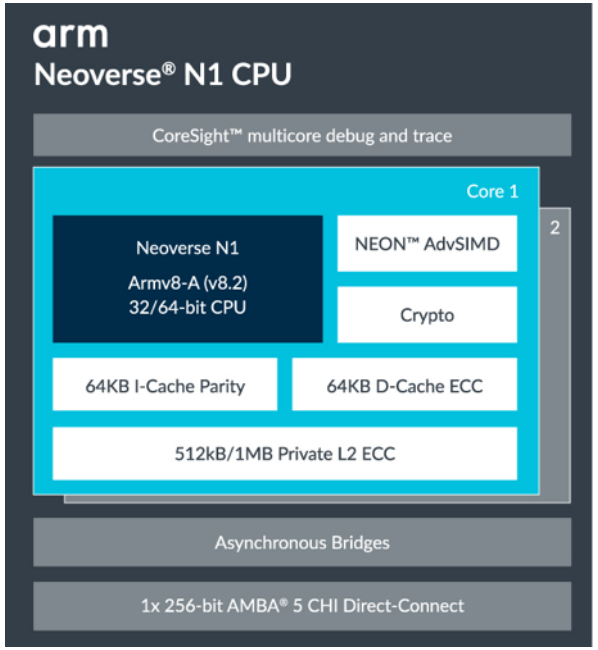
CHERI TRANSITION TO MORELLO

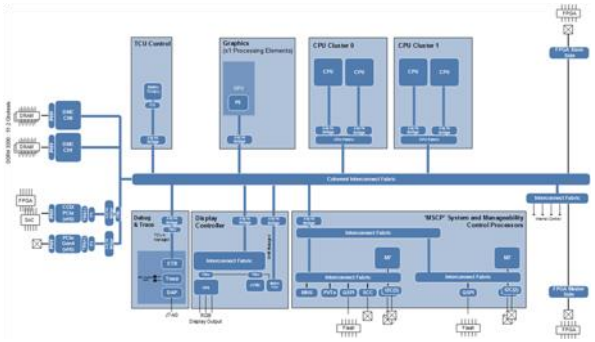
CHERI-ARM research since 2014

- Since 2014, in collaboration with Arm, we have been pursuing joint research to experimentally incorporate CHERI into ARMv8-A:
  - Develop CHERI as an architecture-neutral and portable protection model implemented in multiple concrete architectures
  - Refine and extend the CHERI architecture – e.g., capability compression, tagging µarch, domain transition, and temporal safety
  - Apply concept of architecture neutrality to the CHERI-enabled software stack, including compiler, OS, and applications
  - Expand software: large-scale application experiments, OS use, debuggers, …
  - Extend work in formal modeling and proofs to an industrial-scale architecture
- Solve arising practical {hardware, software, …} problems as part of the research
- Build evidence, demonstrations, SW templates to support potential CHERI adoption

ISCF: Digital Security by Design (UKRI)

- 5-year **Digital Security by Design** UKRI program: £70M UK gov. funding, £117M UK industrial match, to create CHERI-ARM demonstrator SoC + board with proven ISA
- Leap supply-chain gap that makes adopting new architecture difficult – in particular, validation of concepts in microarchitecture, architecture, and software "at scale"
- Support industrial and academic R&D (EPSRC, ESRC, InnovateUK)
- Baseline CPU selected; reuses existing SoC/board designs
- Ongoing collaboration reviewing and distilling {essential, desirable, experimental} CHERI features for use in SoC
- Science designed allowed: Support multiple architectural design choices for software-based evaluation once fabricated
- 2020 emulation models; 2021 "Morello" board delivery





SRI International

UNIVERSITY OF CAMBRIDGE

Why is Arm interested in the CHERI architecture

INDUSTRIAL STRATEGY
UK Research and Innovation

- Arm has been working with UoCambridge on CHERI for some 4-5 years
- Big step to addressing security based on strong fundamental principles
- Addresses spatial memory safety robustly and some ideas for temporal safety
  - Memory safety issues reported to be involved with ~70% of vulnerabilities (Matt Miller, BlueHat IL, 2019)
- Has scope to be the foundation of a new mechanism for compartmentalisation
  - Potentially far cheaper than using translation tables
- Interesting scope to address temporal safety issues as well as spatial ones….
- Many of the Arm software vendors are similarly interested in the possibilities of CHERI
  - Microsoft, Google and others have expressed strong interest in exploring the concept…
  - … but lots of questions about the real-world performance costs and usage models
  - …understanding the intended usage models is important to refine the architectural features
- But is a novel thing to do with additional costs to the system and software
  - Adding a 129th tag bit has a lot of impacts to the memory system
  - it is an ABI change, so non-trivial costs for compatibility for some uses

arm

The Morello Board

INDUSTRIAL STRATEGY
UK Research and Innovation

- An Industrial Demonstrator of a Capability architecture
- Uses a prototype capability extension to the Arm Architecture
  - Prototype is a “superset” of what could be adopted into the Arm architecture
- Use of a superset of the architecture is very unusual
  - Also unrealistic as a commercial product – there will be some frequency effects
  - However, there are tight timescales so architecture is nearly complete now
- The superset of the architecture will allow a lot of software experimentation
  - Various different mechanisms for compartmentalisation
  - Collection of features for which the justification is unclear
  - Techniques for holding the capability tag bit
- Architecture will have formally proved security properties (with UoC and UoE)
- **Morello Board will be the ONLY physical implementation of this prototype architecture**
  - **Learnings from these experiments will be adopted into a mainstream extension to the Arm architecture**
  - **NO COMMITMENT TO FULL BINARY COMPATIBILITY TO THE PROTOTYPE ARCHITECTURE**
    - **But successful concepts are expected to be carried forward into the architecture and can be reused there**

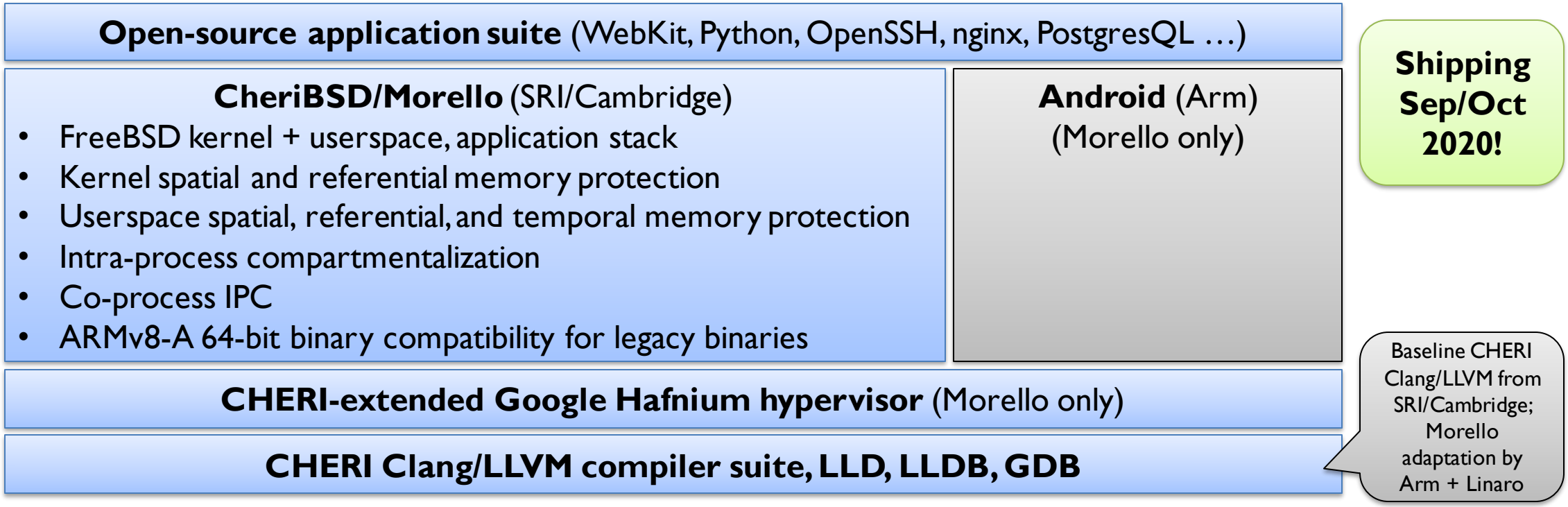
arm

Porting the CHERI software stack to Morello

- Validate and demonstrate the Morello architecture
- Evaluate the Morello implementation (especially performance)
- Provide reference software semantics (spatial and temporal safety, compartmentalization, POSIX integration, OS kernel use, …)
- Act as a template and prototyping platform for industrial demonstration (e.g., for Morello Consortium partners)
- Provide a platform for future research (e.g., 9x 3-to-4-year EPSRC research projects at UK universities starting July-October 2020)

DARPA prototype software stack on Morello

- **Complete open-source CHERI-enabled software stack** from bare metal up: compilers, toolchain, debuggers, operating systems, applications



- Rich CHERI feature use, but fundamentally incremental/hybridized deployment
- We also have an as-yet unpublished, in-progress "CheriOS" clean-slate microkernel OS, and a CHERI adaptation of the open-source FreeRTOS embedded OS

EPSRC Competition

- £10M Research funding
  - £7M from ISCF/DSbD
  - £3m from DCMS

- The EPSRC call covered 3 areas:
  - Capability enabled hardware proof and software verification
  - Impact on system software and libraries
  - Future implications of capability enabled Hardware
- Projects starting July-Oct

Selected Projects

AppControl: Enforcing Application Behaviour through Type-Based Constraints
Dr Wim Vanderbauwhede (University of Glasgow)

CapableVMs – Capable Virtual Machines
Dr Laurence Tratt (King's College London) & Dr Jeremy Singer (University of Glasgow)

CAPcelerate: Capabilities for Heterogeneous Accelerators
Dr Timothy Jones (University of Cambridge)

CapC: Capability C semantics, tools and reasoning
Dr Mark Batty (University of Kent)

CAP-TEE: Capability Architectures for Trusted Execution
Dr David Oswald (University of Birmingham)

CHaOS: CHERI for Hypervisors and Operating Systems
Dr Robert Watson (University of Cambridge)

CloudCAP: Capability-based Isolation for Cloud-Native Applications
Prof Peter Pietzuch (Imperial College London)

HD-Sec: Holistic Design of Secure Systems on Capability Hardware
Professor Michael Butler (University of Southampton)

SCorCH: Secure Code for Capability Hardware
Dr Giles Reger (The University of Manchester)
Prof Daniel Kroening (University of Oxford)

Department for Digital, Culture Media & Sport

EPSRC
Engineering and Physical Sciences Research Council

CONCLUSION

Some potential software research areas

- **Clean-slate OSes and languages**

  Current research has focused on incremental CHERI adoption within current software and languages. How would we design new OSes, languages, etc., assuming CHERI as an ISA baseline?

- **Compilers, language runtimes, and JITs**

  How can we mitigate the performance overheads of more pointer-dense executions, such as with language runtimes? Are vulnerabilities in code generated by compilers and JIT susceptible to mitigation using CHERI? How does CHERI break or potentially improve current compiler analyses and optimization?

- **Further C/C++ protections with CHERI**

  We have focused on spatial, referential, and temporal memory safety for C/C++. But the CHERI primitives could assist with data-oriented protections, garbage collection, type checking, etc. Could these improve security, and at what performance cost?

- **Safe and managed languages**

  Languages such as Java, Rust, C#, OCaml, etc., offer strong safety properties, but frequently depend on C/C++ runtimes and FFI-linked native code. Can CHERI provide stronger foundations for higher-level language stacks?

- **Virtualization**

  Can memory protection usefully harden hypervisors? Can we compartmentalize hypervisors? Can CHERI offer a better mechanism for virtualizing code than an MMU?

- **Debuggers and tracing**

  Debugging/tracing tools rely on high levels of privilege to operate. How can we reduce their privilege to mitigate vulnerabilities in these tools? With stronger architectural semantics, is new dynamic analysis possible?

- **Software compartmentalization tools**

  Granular software compartmentalization offers vulnerability mitigation through privilege reduction and strong encapsulation. How should current applications be refactored, and new applications be designed, to accomplish maintainable and more secure software?

- **Security evaluation and adversarial research**

  What is the impact of CHERI on known vulnerabilities and attack techniques? How does a CHERI-aware attacker change their behavior? Could formal models and proofs support stronger security arguments for CHERI?

Conclusion

- CHERI primitives developed over 10 years of hardware-software-semantics co-design
- Enables fine-grained memory protection and scalable software compartmentalization
- Will appear in the Arm Morello board shipping 2021Q4, along with our complete open-source CHERI software stack
- Many opportunities for future research into enabled software models, architectural and microarchitectural enhancements

  http://www.cheri-cpu.org/

- Watson, Moore, Sewell, and Neumann. **An Introduction to CHERI**, UCAM-CL-TR-941, September 2019.