Security-Aware Benchmarks for Performance Exploration of CHERI-Enabled Architectures

Spandan Das[1], Sayak Deb[2], Khushboo Qayyum[3], Sallar Ahmadi-Pour[1], Christoph Lüth[1,3], Rolf Drechsler[1,3]
[1]Institute of Computer Science, University of Bremen, Bremen, Germany
[2]Institute of Physics and Electrical Engineering, University of Bremen, Bremen, Germany
[3]Cyber-Physical Systems, DFKI, Bremen, Germany
{spandan, sayak, sallar, drechsler}@uni-bremen.de, {khushboo.qayyum, christoph.lueth}@dfki.de

***Abstract*—The Capability Hardware Enhanced RISC Instructions (CHERI) architecture provides fine-grained memory protection for systems, but introduces additional hardware overheads that may negatively impact performance. Evaluating and optimizing such secure architectures require benchmarks that explicitly exercise their security mechanisms. Despite the abundance of benchmarks for unhardened systems, security-aware benchmarks for CHERI-based architectures remain scarce. We address this gap by proposing a framework for generating security-aware benchmarks for CHERI-based RISC-V systems, leveraging the TestRIG tool and applying CHERI-specific post-processing to ensure valid capability usage. As a demonstration use case, we apply the generated benchmarks to evaluate an In-Memory Computing (IMC)-based acceleration of the CHERI tagged memory. Our results show best-case speedups between $6\,\%$ and $11\,\%$, while also identifying scenarios in which the acceleration proves no performance benefit.**

***Index Terms*—CHERI, Tag-based Architecture, Benchmark, In-Memory-Computation, Virtual Prototype**

I. Introduction

The widespread adoption of the Internet of Things (IoT) in security- and safety-critical domains makes memory safety a fundamental design requirement. A large fraction of exploitable vulnerabilities in such systems originates from memory-safety violations, motivating the integration of hardware-supported protection mechanisms directly into processor architectures [1]–[3]. The Capability Hardware Enhanced RISC Instructions (CHERI) architecture addresses this challenge by enforcing fine-grained spatial and temporal memory safety through capability-based protection [4]. Despite its security benefits, CHERI introduces additional hardware structures, which may significantly impact the performance and energy efficiency of IoT devices. Improving the performance of CHERI-based designs therefore requires systematic exploration and evaluation of architectural optimizations. Such optimization efforts critically depend on benchmarks that explicitly exercise CHERI's security mechanisms. However, existing benchmarks either target unhardened systems or require mature CHERI-capable environments running operating systems that are often unavailable during early development phases. As a result, performance-driven design space exploration for secure embedded architectures remains poorly supported.

This research has been supported by the German Research Foundation (DFG) with project EMBOSOM (grant nos. DR 287/42-1, LU 707/10-1), and the German Ministry for Research, Technology and Space (BMFTR) with projects ECXLplus (grant no 01IW24001) and ExaVerse (grant no 01IW25003).

This paper addresses this limitation by introducing a framework for generating security-aware benchmarks tailored to CHERI-enabled RISC-V systems. Our approach builds on Testing with Random Instruction Generation (TestRIG) and augments it with CHERI-specific instruction stream processing to produce executable benchmarks that correctly and systematically utilize capability-based memory operations. Unlike application-level benchmarks, the generated benchmarks are available early in development and can be deployed with minimal integration effort.

We evaluate the proposed framework using a Virtual Prototype (VP) and an illustrative case study modeling an In-Memory Computing (IMC)-based acceleration of a CHERI tagged memory. Our results show best-case speed-ups of $6\,\%$ and $11\,\%$ over an unaccelerated baseline, while also identifying parameter regimes in which acceleration yields no benefit. In summary, this work makes the following contributions:

- A security-aware benchmark generation framework for CHERI-enabled embedded systems.
- An experimental demonstration showing how the benchmarks quantify optimization potential and limits in a CHERI-based VP.
- A set of open-source security-aware benchmarks for CHERI-based RISC-V systems for early performance assessment.[1]

II. Preliminaries

A. CHERI

CHERI is a capability-based architectural extension that provides hardware-enforced spatial and temporal memory safety for systems programmed in memory-unsafe languages such as C and C++ [4], [5]. In CHERI-enabled RISC-V systems, conventional pointers are replaced with capabilities, which encode bounds, permissions, and validity metadata for memory regions. Capability tags are propagated through registers, caches, and memory, and are stored in a separate, non-addressable tagged memory that is accessible only via CHERI-specific instructions. Any illegal manipulation invalidates the tag and prevents further memory access through the corrupted capability. While these mechanisms improve security, they introduce additional metadata handling and memory accesses, which can affect performance in embedded systems. This work focuses on accelerating the CHERI tagged memory by using

[1]https://github.com/agra-uni-bremen/cheri-early-benchmarks

IMC to improve memory access, while preserving CHERI's security guarantees.

B. TestRIG

TestRIG is a verification-oriented framework for RISC-V processor implementations using constrained random instruction generation [6]. It interfaces with a Design under Verification (DUV) via the RISC-V Formal Interface (RVFI) and Direct Instruction Injection (DII), executing generated instructions and comparing architectural state against the Sail [7] formal model of the RISC-V Instruction Set Architecture (ISA). TestRIG supports multiple abstraction levels, ranging from high-level emulation to Register-Transfer Level (RTL) simulation. A key feature of the DII interface is that instruction execution does not require consistent program control flow, enabling high-throughput instruction generation. RVFI provides fine-grained visibility into architectural state after each instruction, allowing precise semantic validation. Communication between TestRIG and the DUV is implemented via standard network sockets, enabling flexible integration with simulators. In this work, we repurpose TestRIG's instruction generation infrastructure to form the basis of security-aware benchmark generation. By post-processing generated instruction streams to ensure valid capability usage, we enable meaningful performance evaluation of CHERI architectures beyond verification.

III. Related Works

Hardware-assisted spatial and temporal memory safety has been explored through various architectural techniques, including bounds-checking mechanisms such as HardBound [8] and SoftBound [9], tagged memory architectures such as ARM Memory Tagging Extension [10], and capability-based systems. CHERI represents the most mature capability-based memory safety architecture, providing fine-grained enforcement of pointer bounds and permissions in hardware [4]. Since its introduction, CHERI has been implemented and evaluated across multiple platforms, including software-based emulation [11], FPGA prototypes [12], and ASICs [5].

Benchmarking plays a central role in evaluating architectural optimizations, yet widely used benchmark suites such as SPEC [13], PARSEC [14], and Embench IoT [15] target general-purpose processors and do not exercise memory safety mechanisms. Studies evaluating Intel Memory Protection Extensions [16], ARM Memory Tagging Extension [17], and other tagged memory designs have therefore relied on specialized benchmarks focusing on bounds checks, tag manipulation, and fault handling latency. Such benchmarks are typically tailored to specific mechanisms and are difficult to reuse across architectures. For CHERI, the most prominent benchmarking effort targets ARM's Morello platform and is based on porting existing C/C++ applications to CHERI [18]. While effective at the application level, this approach depends on CheriBSD and assumes a mature execution environment, limiting its applicability for resource-constrained systems and early-stage design space exploration.

In contrast, this work enables early-stage, security-aware performance evaluation of CHERI-enabled RISC-V embedded systems. By generating instruction-level benchmarks that explicitly exercise CHERI's capability and tag-management and integrating them with TestRIG-based verification, our approach supports concurrent evaluation of performance and security correctness during system development.

IV. Benchmarking for Secure Embedded Systems

Meaningful performance evaluation of secure embedded systems requires benchmarking early in the system development cycle, when architectural design decisions remain flexible. While integrating security architectures such as CHERI into VPs enables early consideration of security aspects, existing benchmarks do not exercise the CHERI ISA extension for RISC-V. At the same time, application-level CHERI benchmarks require on a mature execution environment with an operating system and software stack, which are typically unavailable during early design space exploration.

To address this gap, we leverage the TestRIG framework to establish an early-stage benchmarking environment for CHERI-enabled systems. TestRIG provides configurable random instruction generation, which we repurpose to produce instruction streams that are subsequently refined to ensure correct and systematic use of CHERI capabilities. The resulting benchmarks enable quantitative performance evaluation of architectural optimizations while preserving CHERI's security guarantees.

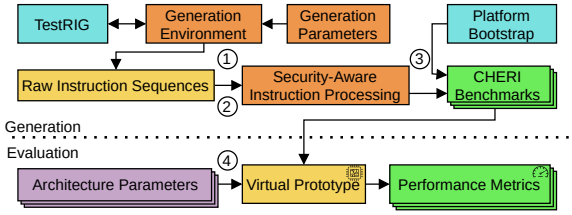

Figure 1: Overview of our generation framework for early and security-aware benchmarks.

Figure 1 provides an overview of the benchmark generation workflow. First, a customized generation environment connects to TestRIG and requests instruction streams with configurable instruction distributions (①). Since these sequences are intended for functional verification, they are post-processed to construct security-aware instruction streams that execute correctly under CHERI's capability and tagging rules (②). The processed streams are then combined with platform-specific bootstrap code to build executable benchmarks (③). To demonstrate the applicability of the framework, we apply the generated benchmarks to evaluate a CHERI-specific architectural optimization in a VP (④). Specifically, we model an IMC-based acceleration of CHERI tagged memory and compare performance against an unmodified VP to quantify potential speed-ups and identify performance limits.

A. Security-Aware Benchmark Generation

Instruction sequences generated by TestRIG are designed for functional verification and intentionally violate architec-

Algorithm 1 Security-aware processing of instruction streams

Input: Raw instruction stream I_{raw}
Output: Security-aware instruction stream I_{sec}
CapSize ← Size of a capability in number of bits
for each instruction I in raw instruction stream I_{raw} **do**
  ct ← any capability register
  rt ← any integer register
  addr ← any data space address divisible by *CapSize*
  if I is cjal, cjalr, B-type or regular J-type instruction **then**
    Set Immediate of I to 4
  else if I is jalr.cap or cinvoke **then**
    Set capability register of I to cs1
    Append sequence to I_{sec}:
    auipcc cs,0; **cincoffsetimm** cs,cs,12
  else if I is jalr.pcc **then**
    rs ← rs1 of I
    Append sequence to I_{sec}:
    auipcc ct,0; **cincoffsetimm** ct,ct,16;
    cgetoffset rs,ct
  else if I is sc.cap or csc **then**
    if I has valid immediate **then**
      Set immediate of I to 0
    end if
    cs ← cs1 of I
    Append sequence to I_{sec}:
    auipcc cs,0; **li** rt,addr; **csetaddr** cs,cs,rt
  else if I is sc.ddc **then**
    rs ← rs1 of I
    Append sequence to I_{sec}:
    auipcc ct,0; **li** rt,addr; **csetaddr** ct,ct,rt;
    cspecialw ddc,ct; **andi** rs,rs,CapSize
  else if I is ebreak, unknown or illegal instruction **then**
    Skip I and do not append
  end if
  Append I to I_{sec}
end for

tural invariants such as valid control flow, initialization, and memory access semantics. While useful for bug detection, such behavior is unsuitable for performance benchmarking, where meaningful execution depends on architecturally valid instruction streams. Consequently, TestRIG output must be filtered and transformed before using as a benchmark.

In a first step, we filter generated instruction streams to remove instructions and sequences that would trap or terminate execution on a correct RISC-V CHERI implementation. While this yields executable instruction streams, they still do not correctly exercise CHERI's capability mechanisms. In a second step, we post-process the filtered streams to enforce proper capability preparation and usage. CHERI enforces strict bounds, permission, and validity checks on memory accesses which, without proper capability setup, will fail for most generated memory operations. When these operations fail, they don't reach CHERI-specific hardware such as the tagged memory. To address this, we identify instruction patterns that require valid capabilities and insert or rewrite instruction sequences to construct capabilities with appropriate bounds, permissions, and tags. This ensures that capability instructions execute successfully and fully utilize the CHERI-enabled hardware.

Algorithm 1 summarizes the instruction stream processing and illustrates how specific instruction types are augmented with capability setup sequences. After post-processing, the resulting security-aware instruction streams execute to completion without violating CHERI's architectural constraints. Instruction distributions can be further controlled through TestRIG's generation parameters, allowing the benchmarks to emphasize specific CHERI instructions or approximate application-level behavior. This enables targeted and reproducible performance evaluation of CHERI microarchitectures.

B. IMC-based CHERI Acceleration

To demonstrate the applicability of the generated benchmarks for early-stage design exploration, we evaluate a representative CHERI-specific optimization using a VP. Specifically, we model an IMC-based acceleration of the CHERI tag update mechanism and compare system performance with and without acceleration. CHERI maintains capability validity using tag bits stored in a dedicated, non-addressable tag memory. During capability store instructions (e.g., `sc.ddc`, `sc.cap`), the corresponding tag bit must be updated alongside the data. A conventional implementation requires transferring the tag-containing memory word to the processor, modifying the tag, and writing it back, resulting in additional memory traffic and latency. We model an IMC-based tag memory in which tag updates are performed directly within the memory array, allowing the processor to issue a tag-modification command without transferring full memory words. This abstraction reduces data movement over the system interconnect while preserving CHERI's security semantics.

To capture the diversity of potential IMC implementations, we model the tag update operation using a parametrized latency in the VP. By sweeping this parameter, we identify the latency range in which IMC-based tag updates provide a net performance benefit, as well as regimes where acceleration becomes ineffective. While the evaluation focuses on a single optimization, it serves as an illustrative case study demonstrating how the proposed benchmarks enable quantitative analysis of CHERI-specific architectural trade-offs early in the development.

V. Evaluation

For the evaluation, we integrate the CHERI extension for RISC-V into an open-source RISC-V VP [19], [20] and apply the generated benchmarks to both a baseline CHERI VP and an extended VP modeling an IMC-based tag update mechanism. The IMC functionality is abstracted and evaluated within the VP to enable early-stage performance exploration. In the IMC-enabled VP, the latency of the in-memory tag update operation (referred to as IMC-time) is parametrized from $0\,\mathrm{ns}$ to $30\,\mathrm{ns}$ to model a broad range of possible accelerator implementations. We generate three benchmarks with accelerated instruction occurrences ranging from $8\,\%$ to $20\,\%$ (referred to as `TRG_8`, `TRG_15`, `TRG_20`), demonstrating the configurability of the benchmark generation process. All experiments were performed on an Intel(R) Core(TM) Ultra 5 125U 12-Core Processor with $4.3\,\mathrm{GHz}$ and $64\,\mathrm{GB}$ of memory.

Figure 2 shows the normalized speed-up of the IMC-accelerated VP relative to the baseline (red line) while sweeping the IMC-time. For an idealized IMC-time of $0\,\mathrm{ns}$, speed-ups between $1.06\times$ and $1.11\times$ are observed. Performance

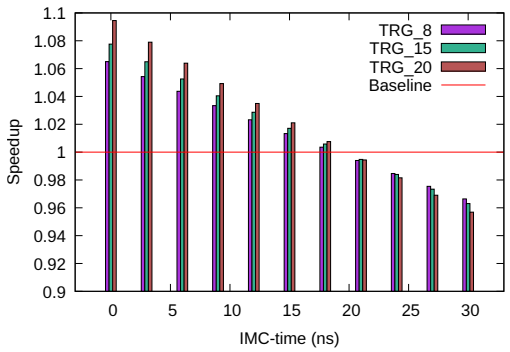

Figure 2: Processor speedup versus IMC-time for security-aware benchmarks.

benefits diminish as IMC-time increases, with acceleration becoming ineffective for latencies above approximately 20 ns.

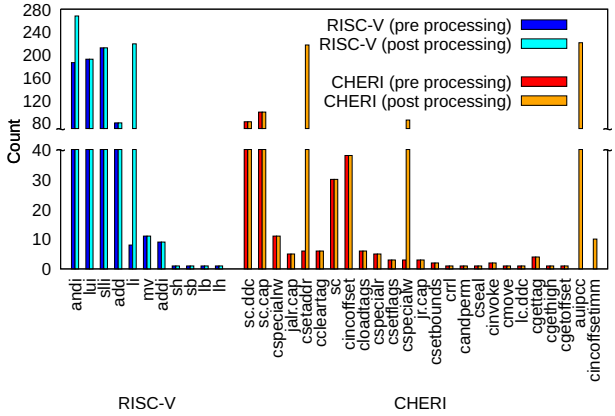

Figure 3: Instruction counts before and after security-aware instruction processing.

These results illustrate how the benchmarks support early identification of performance boundaries and feasibility regions for architectural optimizations.

Figure 3 shows the instruction distribution of benchmark (`TRG_20`), to illustrate the effects of security-aware instruction post-processing. Additional capability-management instructions such as `csetaddr`, `cspecialw`, `auipcc` and `li` are introduced to ensure valid capability setup and control flow. For example, capability-based jumps require both correct capability preparation and well-defined target addresses.

Overall, this evaluation demonstrates that the proposed benchmark framework enables quantitative analysis of CHERI-specific optimizations in early development stages. Even in this focused case study, the benchmarks expose both achievable best-case improvements (6 % to 11 %) and fundamental limits of acceleration under realistic timing constraints.

VI. Conclusion

Evaluating and optimizing secure embedded systems requires benchmarks that explicitly exercise hardware-enforced security mechanisms. This work addresses the lack of such benchmarks for CHERI-enabled architectures by introducing a framework for generating security-aware instruction-level benchmarks that are available early in the system development process. By leveraging TestRIG and CHERI-specific instruction stream processing, the proposed framework enables quantitative performance evaluation and architectural exploration while preserving CHERI's security semantics. Using VPs, we showed how the benchmarks expose both performance opportunities and inherent limits of CHERI-specific optimizations. In a representative case study modeling an IMC-based tag memory, the benchmarks revealed best-case speed-ups of 6 % to 11 %, as well as latency regimes in which acceleration becomes ineffective. More broadly, the proposed approach provides a foundation for early-stage evaluation of security mechanisms before mature software stacks are available. Future work will extend the framework toward generating more complex program structures and richer control-flow patterns while retaining security awareness.

References

[1] A. Mosenia *et al.*, "A comprehensive study of security of Internet-of-Things," *IEEE Transactions on emerging topics in computing*, vol. 5, no. 4, pp. 586–602, 2016.

[2] Y. Lu *et al.*, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2018.

[3] V. E. Moghadam *et al.*, "Memory integrity techniques for memory-unsafe languages: A survey," *IEEE Access*, vol. 12, pp. 43 201–43 221, 2024.

[4] J. Woodruff *et al.*, "The CHERI capability model: Revisiting RISC in an age of risk," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 457–468, 2014.

[5] R. Grisenthwaite *et al.*, "The Arm Morello evaluation platform—Validating CHERI-based security in a high-performance system," *IEEE Micro*, vol. 43, no. 3, pp. 50–57, 2023.

[6] A. Joannou *et al.*, "Randomized testing of RISC-V CPUs using direct instruction injection," *IEEE Design & Test*, vol. 41, no. 1, pp. 40–49, 2024.

[7] A. Armstrong *et al.*, "ISA semantics for ARMv8-a, RISC-V, and CHERI-MIPS," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–31, 2019.

[8] J. Devietti *et al.*, "Hardbound: Architectural support for spatial safety of the C programming language," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 2, pp. 103–114, 2008.

[9] S. Nagarakatte *et al.*, "SoftBound: Highly compatible and complete spatial memory safety for C," in *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2009, pp. 245–258.

[10] S. Jero *et al.*, "TAG: Tagged architecture guide," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–34, 2022.

[11] F. Bellard, "QEMU: A generic and open source machine emulator and virtualizer," 2003.

[12] S. Amar *et al.*, "CHERIoT: Complete memory safety for embedded devices," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 641–653.

[13] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[14] C. Bienia *et al.*, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, 2008, pp. 72–81.

[15] D. Patterson *et al.*, "Embench IOT 2.0 and DSP 1.0: Modern embedded computing benchmarks," *Computer*, vol. 58, no. 5, pp. 37–47, 2025.

[16] O. Oleksenko *et al.*, "Intel MPX explained: A cross-layer analysis of the Intel MPX system stack," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, pp. 1–30, 2018.

[17] M. Li *et al.*, "NanoTag: Systems support for efficient byte-granular overflow detection on ARM MTE," *arXiv preprint arXiv:2509.22027*, 2025.

[18] X. Sun *et al.*, "Sweet or sour CHERI: Performance characterization of the Arm Morello platform," in *Proceedings of the 2025 IEEE International Symposium on Workload Characterization*. IEEE, 2025.

[19] V. Herdt *et al.*, "Extensible and configurable RISC-V based virtual prototype," in *2018 Forum on Specification & Design Languages (FDL)*. IEEE, 2018, pp. 5–16.

[20] V. Herdt *et al.*, "RISC-V VP." [Online]. Available: https://github.com/agra-uni-bremen/riscv-vp