3-2022

Evaluating Secure Enclave Firmware Development for Contemporary RISC-V Workstations

Samuel D. Chadwick

EVALUATING SECURE ENCLAVE FIRMWARE DEVELOPMENT FOR CONTEMPORARY RISC-V WORKSTATIONS

THESIS

Samuel D. Chadwick, First Lieutenant, USSF

AFIT-ENG-MS-22-M-017

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-M-017

EVALUATING SECURE ENCLAVE FIRMWARE DEVELOPMENT FOR CONTEMPORARY RISC-V WORKSTATIONS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Samuel D. Chadwick, B.S.

First Lieutenant, USSF

March 2022

AFIT-ENG-MS-22-M-017

EVALUATING SECURE ENCLAVE FIRMWARE DEVELOPMENT FOR CONTEMPORARY RISC-V WORKSTATIONS

THESIS

Samuel D. Chadwick, B.S.
First Lieutenant, USSF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Lt Col James W. Dean, Ph.D.
Member

Matthew J. Dallmeyer, M.S.
Member

AFIT-ENG-MS-22-M-017

Abstract

The emergence of the open-source Reduced Instruction Set Computer Architecture - Five (RISC-V) Instruction Set Architecture (ISA) empowers developers and engineers, device manufacturers, and individual users with the unique opportunity to re-evaluate existing *Trusted Computing* paradigms. Emerging open-source security mechanisms facilitate the proliferation of *Confidential Computing* principles. These technology standards aim to provide *secure enclave* computing as a fundamental computing attribute, inherent within the RISC-V ISA specification. Security enforcement within these enclaves are handled by performing computation in memory-isolated, hardware-based, software-defined Trusted Execution Environments (TEEs). Keystone Enclave, an open-source project for building TEEs upon the RISC-V Privileged ISA, is one promising exemplar.

This research evaluates the firmware development procedures required to implement Keystone Enclave on new unsupported hardware. Expressly, this effort extends Keystone Security Monitor (SM) firmware components for use on the HiFive Unmatched development platform as a demonstration of Keystone Enclave's device portability claims. Furthermore, it proposes Keystone Software Development Kit (SDK) and Enclave Application (Eapp) development recommendations to supplement contemporary Application Specific Integrated Circuit (ASIC) RISC-V workstations with TEEs. Moreover, this research asserts that for the wide-spread adoption of *Confidential Computing* principles to occur, significant hardware, firmware, and software development advancements are required by all constituent parties.

AFIT-ENG-MS-22-M-017

Dedicated to

My Wife, for your unwavering support

My Daughter, for your inspiration

My Parents, for instilling a healthy work ethic.

"For I know the plans I have for you," declares the Lord, "plans to prosper you and not to harm you, plans to give you hope and a future."

-Jeremiah 29:11

Acknowledgements

To my academic advisor: Thank you for your mentorship, encouragement, and engagement throughout this research journey. Your expertise and confidence in my abilities have contributed immeasurably to my pursuit of life-long learning.

To my committee members: Your technical prowess, broad perspective, and constructive feedback have opened my aperture to exploration in academia. Your support and advice has been well received.

To my wife and daughter: Thank you immensely for standing beside me each and every step of the way along this adventure. Your love, dedication, and support gives me ample reason to continually strive to do my best. I love you, so much.

Samuel D. Chadwick

Table of Contents

Page

List of Figures

Figure Page

List of Tables

Table Page

List of Acronyms

AMD	Advanced Micro Devices
ANOVA	Analysis of Variance
APT	Advanced Packaging Tool
ARM	Advanced RISC Machines
ASIC	Application Specific Integrated Circuit
BBL	Berkeley Bootloader
Bogo OPs/s	Bogus Operations per Second
CCC	Confidential Computing Consortium
CIA	Confidentiality, Integrity, and Availability
CLT	Central Limit Theorem
CTO	Chief Technology Officer
CUT	Component Under Test
DDR4	Double Data Rate 4
DIP	Dual In-Line Package
DPKG	Debian Package Management System
DTB	Device Tree Blob
Eapp	Enclave Application
EOL	End of Life
ESM	Extended Security Maintenance
FPGA	Field Programmable Gate Array
FSBL	First Stage Bootloader
GB	Gigabyte
GCC	GNU Compiler Collection
GNU	GNU's Not Unix

H-Mode	Hypervisor Mode
hart	hardware thread
IBM	International Business Machines
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IP	Intellectual Property
IPC	Instructions per Clock Cycle
ISA	Instruction Set Architecture
ISO	International Organization for Standardization
ITB	Image Tree Blob
JTAG	Joint Test Action Group
L2	Level 2
LTS	Long Term Support
M-Mode	Machine Mode
MB	Megabyte
microSD	micro Secure Digital
Mini-ITX	Miniature-Information Technology eXtended
NVMe	Non-Volatile Memory Express
OpenSBI	Open Source Supervisor Binary Interface
OS	Operating System
pacman	Pacman Package Manager
PC	Personal Computer
PCIe	Peripheral Component Interconnect Express
PMP	Physical Memory Protection
PoR	Power on Reset
PTS	Phoronix Test Suite

QEMU	Quick EMUlator
RISC-V	Reduced Instruction Set Computer Architecture - Five
ROM	Read Only Memory
RPM	Red Hat Package Manager
RT	runtime
S-Mode	Supervisor Mode
SBI	Serial Binary Interface
SBL	Second Bootloader
SDK	Software Development Kit
SDRAM	synchronous dynamic random-access memory
SEV	Secure Encrypted Virtualization
SGX	Software Guard eXtensions
SM	Security Monitor
SoC	System on a Chip
SPI	Serial Peripheral Interface
SPL	Secondary Program Loader
SSD	Solid-State Drive
SSH	Secure Shell
SUT	System Under Test
TCB	Trusted Computing Base
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
U-Mode	User Mode
USB	Universal Serial Bus
VM	Virtual Machine

XML	eXtensible Markup Language
YUM	Yellowdog Updater Modified
ZSBL	Zeroth Stage Bootloader

EVALUATING SECURE ENCLAVE FIRMWARE DEVELOPMENT FOR CONTEMPORARY RISC-V WORKSTATIONS

I. Introduction

1.1 Motivation

The persistent desire to securely compute, store, and transport sensitive information drives a continuing evolution in the mechanisms for enforcing information security; keeping pace with, and responding to technological advancements. Today, sensitive data is encrypted while at rest in storage mediums and while in transit across networks –nevertheless, data in use remains vulnerable to threats that target application data within system memory [2]. This research advocates for the adoption of *Confidential Computing* principles to protect data in use by performing computation inside memory-isolated hardware-based Trusted Execution Environments (TEEs) [1]. To justify promoting the wide-spread adoption of these computing principles, emerging open-source secure enclave implementations must be evaluated against existing proprietary *Trusted Computing* solutions to delineate potential advantages, deficiencies, and developmental needs.

1.2 Problem Statement

The existing *Trusted Computing* paradigm employs closed-source, proprietary security mechanisms that impute *trust* by identifying expected behavior operating within particular hardware and software components [3]. *Confidential Computing* principles maintain this accepted model for ascribing *trust*; although, these conventions differ in practice by instead relying upon open-source code with open standards and open specifications to enforce security [4]. Keystone Enclave is one such open-source project. Built upon the RISC-V Privileged ISA specification, Keystone is supported by the Confidential Computing Consortium (CCC) and The Linux Foundation with the goal of providing a SDK for building customizable TEEs [1, 2].

Formally, Keystone Enclave v1.0.0 has been verified to function in emulation using Quick EMUlator (QEMU), on the FireSim Field Programmable Gate Array (FPGA) as a softcore processor, and on the (now discontinued) HiFive Unleashed native RISC-V development board. The Keystone documentation claims that migration to arbitrary RISC-V processors is possible with only minor modifications required to plant a silicon Root-of-Trust [1]. To evaluate Keystone's implementation portability claims, this research explores the necessary procedures and development contributions required to implement Keystone on the HiFive Unmatched development platform.

Irrespective of the requisite measures needed to port Keystone Enclave to new platforms, previous research has demonstrated that the performance overhead introduced by Keystone Enclave axiomatically degrades average system response times [5]. Accordingly, the operational performance of candidate hardware platforms must be sufficiently characterized to determine the feasibility and practicality of any particular hardware implementation. Subsequently, comparative benchmarking is conducted to characterize system performance across a variety of firmware and software configurations. For Keystone Enclave to competently champion the TEE paradigm promoted by the CCC, it must satisfy operational performance and security requirements while simultaneously offering its users clear implementation pathways.

1.3 Research Objectives

This research aims to assess feasibility and practicality assertions made by Keystone Enclave concerning device portability. It also seeks to characterize the performance of a native RISC-V Linux-capable workstation. Concertedly, these goals form pertinent criteria upon which TEE hardware candidates may be evaluated. The objectives of this research are encompassed by the following actions:

- Extend Keystone Enclave firmware components for use on the HiFive Unmatched (an ASIC, Linux capable, RISC-V development platform).
- Characterize the performance of the HiFive Unmatched board quantitatively using benchmarking.
- Assess system performance impacts pertaining to specific versions of these firmware and software components:
  - OpenSBI platform-specific reference implementation
  - U-Boot, the universal bootloader
  - The Linux Kernel & Distribution Linux Kernels
  - Ubuntu Distribution Releases
- Evaluate the Keystone SDK by configuring and modifying the development tools used to construct enclaves and Enclave Applications (Eapps).
- Verify Keystone Enclave compatibility with modern RISC-V workstations.

1.4 Hypothesis

The performance of contemporary RISC-V computers can be quantitatively characterized to substantiate their suitability for incorporating TEE functionality. Moreover, this research further hypothesizes that for *Confidential Computing* practices to burgeon, significant firmware and software contributions are required by all principal stakeholders.

1.5 Approach

To sufficiently determine if the HiFive Unmatched is a suitable candidate for Keystone Enclave, its performance must first be characterized. Performance characterizations are conducted by configuring the HiFive Unmatched board to run 20 compatible benchmarks, selected from the Stress-NG benchmarking suite, across three distinct Ubuntu distribution versions –each with unique distribution Linux kernels, U-Boot bootloaders, and OpenSBI versions. Once baseline performance thresholds have been conducted, appropriate firmware modifications are attempted to supplement the HiFive Unmatched with TEE capabilities. Expressly, this work catalogues the development processes and procedures required to extend the HiFive Unmatched with Keystone SM firmware. Upon integration of Keystone SM into the OpenSBI layer of the boot-flow, successive performance characterizations are conducted. Subsequent benchmarking runs repeat prior Stress-NG benchmarks; however, these configurations differ from initial baseline characterizations by substituting each of the three unmodified OpenSBI versions with the modified OpenSBI + Keystone SM implementation. With these benchmarking scores analyzed, Keystone Enclave portability claims are addressed with appropriate development and policy recommendations proposed.

1.6 Contributions

The contributions of this thesis to the fields of Cyber Security, Computer Networking, and Confidential Computing are outlined below:

- Demonstrated device portability claims of an open-source TEE project.

- Verified performance ramifications for recent Ubuntu Distribution Kernel releases on ASIC RISC-V systems.
- Implemented a platform evaluation framework to assess future RISC-V workstations and configurations.
- Assessed benchmarking characterizations to determine platform suitability for TEEs.
- Identified the need for Linux distribution publishers to directly support *Confidential Computing* projects.
- Proposed TEE policy and implementation recommendations for relevant communities, industries, and agencies.

1.7 Organization

This thesis document is grouped into six chapters. The remaining chapters of this work are as follows. Chapter II equips the reader with pertinent context by describing *Trusted Computing*, summarizing *Confidential Computing*, examining TEEs as a mechanism for security enforcement, listing prevalent examples of TEEs in industry, and introducing Keystone Enclave, an open-source TEE implementation based on the RISC-V ISA. Chapter III details the experimental bootloader and firmware modifications conducted that configure and extend Keystone SM for use on the HiFive Unmatched. Chapter IV reports the design of experiments and testing methodology applied to assess system performance across multiple bootloader, kernel, and distribution configurations. Chapter V presents benchmarking results, analysis, and observations that characterize system performance. Chapter VI offers concluding remarks, recommends best practices for accelerating the adoption of *Confidential Computing* principles, and outlines anticipated future efforts –including proposed Keystone Enclave performance characterizations, capable of ensuring that performance obligations are met while simultaneously enforcing strict security adherence.

II. Background & Related Work

2.1 Overview

This chapter offers relevant background information pertaining to *Trusted Computing*, *Confidential Computing*, the RISC-V ISA, and Keystone Enclave. Discussion begins by comparing and contrasting *Trusted Computing* paradigms against emerging *Confidential Computing* principles. Next, the open-source RISC-V ISA is described with particular attention given to the privileged specifications upon which Keystone Enclave is built. The chapter concludes by explicating the constituent components and supporting projects employed by Keystone Enclave.

2.2 Trusted Computing

The Trusted Computing Group (TCG) aims to provide secure computing technologies for "business-critical data and systems, secure authentication and strong protection of user identities, and the establishment of strong machine identity and network integrity" through open standards and specifications [6]. Notably, the TCG omits strong requirements for open-source code implementations, instead prioritizing the requisite standards and specifications necessary for Trusted Platform Module (TPM) specification compliance. The TCG is governed by a board comprised of 14 *Promoter* member companies and hundreds of elected *Contributor* advisors [6]. Currently, there are 14 TCG *Promoter* members. Listed alphabetically, they include the following companies: Advanced Micro Devices (AMD), Cisco, Dell, Google, Hewlett Packard Enterprise, HP, Huawei, International Business Machines (IBM), Infineon, Intel, Juniper Networks, Lenovo, Microsoft, and Toyota [6]. Collectively, these industry vendors, developers, manufactures, and infrastructure companies establish, verify, and validate *Trusted Computing* technologies by publishing their standards via the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

2.2.1 Trusted Platform Module

TPMs are hardware components that enforce *Trusted Computing* conventions by adhering to the TPM Library Specification. The current TPM Library Specification is named "Trusted Platform Module Library Specification, Family '2.0' Level 00 Revision 1.59" and is openly available under the ISO/IEC 11889:2015 standard publication [7]. Overwhelmingly, TPM hardware components utilize proprietary technologies to meet the openly published specification and are manufactured and sold by TCG *Promoter* and *Contributor* members to provide secure computing.

2.3 Confidential Computing

The Confidential Computing Consortium (CCC) is a Linux Foundation project community whose members focus on securing *data in use* through open collaboration. Commonly deployed encryption techniques enforce the full Confidentiality, Integrity, and Availability (CIA) triad for *data at rest* (i.e. within storage mediums) and for *data in transit* (i.e. across public or private networks). Nonetheless, these techniques are limited by the conventional computing infrastructure. To sufficiently protect *data in use* (i.e. during execution), computation must either be performed within a hardware-based TEE [8], or operate on a system with the ability to manipulate encrypted data without decrypting it first –as in homomorphic computing, which is not addressed by this research. Figure 1 illustrates the appropriate security mechanisms as they apply to classical computing data states.

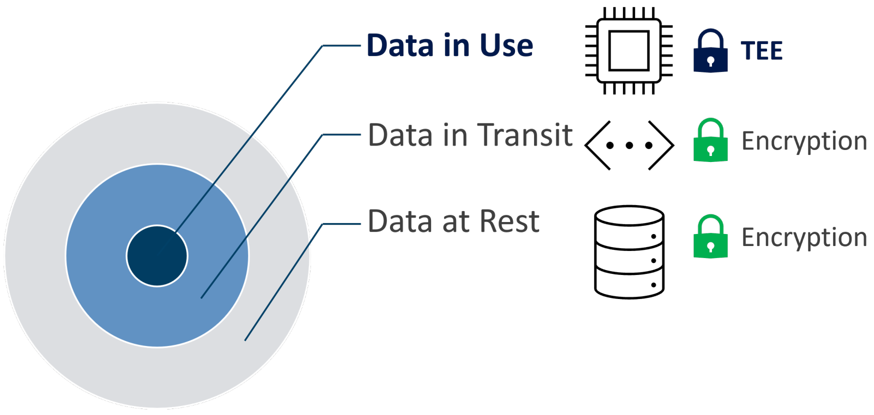


Figure 1. Security Mechanisms Applied to Classical Computing Data States

2.3.1 Trusted Execution Environments

While no formal definitions for TEEs have been arbitrated, the CCC defines TEEs "as an environment that provides a level of assurance of the following three properties: data confidentiality, data integrity, and code integrity" [8, 9]. Our work uses this definition interchangeably with variations commonly found within industry. Many prevalent TEE implementations are proprietary, including Intel Software Guard eXtensions (SGX), Advanced RISC Machines (ARM) TrustZone, and AMD Secure Encrypted Virtualization (SEV). Vendor specific TEE implementations pose two distinct disadvantages: (1) Intellectual Property (IP) ties both new features and bug fixes directly to vendors; and (2) different threat models have been ascribed to specific ISAs. By observation, Intel SGX targets server and desktop application isolation, ARM TrustZone addresses vendor-provisioned mobile application isolation, while AMD SEV focuses on virtual machine isolation. Markedly, Intel SGX extensions do not recognize side-channel analysis as an active threat vector [10]. Consequently, speculative execution attacks, such as Spectre and Meltdown, can exploit this threat model omission to compromise all *Trusted Computing* enclaves running on Intel SGX platforms [10]. Intel has since depreciated the SGX extensions and removed them from 11th generation and newer "Core™ " series processors.

To combat known *Trusted Computing* deficiencies, proprietary disadvantages must be weighed against emerging open-source alternatives. One promising project, Keystone Enclave, provides an extensible open-source TEE implementation for the RISC-V ISA. This research asserts that Keystone Enclave –along with supporting hardware, firmware, and software– may provide a viable avenue for extending native RISC-V computers with *Confidential Computing* capabilities while avoiding potential inadequacies found within existing proprietary TEE offerings.

2.4 The RISC-V ISA

ISAs provide technical specifications for interfacing processor hardware to low-level firmware. The RISC-V ISA is a free and open architecture aiming to enable a new era of processor innovation through open standard collaboration [11]. The RISC-V ISA describes a hardware thread as a "hart"; comparatively, these "harts" are equivalent in terminology to the colloquial term "core" widely accepted for existing x86_64 systems [11]. The RISC-V Instruction Set Manual is organized into two volumes: Volume I, Unprivileged ISA and Volume II, Privileged Architecture.

2.4.1 Unprivileged ISA

The Unprivileged RISC-V Instruction Set Manual accounts for the base integer architecture (designated by I) and additional optional instruction set extensions. The set standard of extensions currently defines multiply/divide operations "M", atomic operations "A", single and double-precision floating-point arithmetic "F" and "D", as well as compressed 16-bit instructions "C" [11]. The M, A, F, and D identifiers are standard extensions that are collectively referred to by "G". Our research employs 64-bit integer registers with all standard and compressed extensions –thus the particular

ISA descriptor used for this research is designated as RV64GC (where *RV* signifies RISC-V, *64* identifies the register width, *G* indicates the base ISA with standard extensions, and *C* shows support for compressed operations).

2.4.2 Privileged ISA

The Privileged RISC-V Instruction Set Manual was formally ratified on 4 December 2021 and describes all aspects of RISC-V systems beyond the unprivileged ISA [12]. The features pertinent to this work are Physical Memory Protection (PMP) and three of the four specified privilege levels: User Mode (U-Mode), Supervisor Mode (S-Mode), and Machine Mode (M-Mode). The fourth mode, Hypervisor Mode (H-Mode), has not been utilized for this work. Keystone Enclave makes appropriate use of these security primitives to enforce memory-isolated execution within TEEs [1]. Colloquially, these memory isolated environments are often referred to as *secure enclaves*. Because we utilize Keystone Enclave throughout the creation, execution, and destruction lifecycle of these enclaves, they are aptly named Keystone Enclaves.

2.4.3 Physical Memory Protection and Machine Mode Operations

The Privileged RISC-V Instruction Set Manual defines the PMP specification. PMP units are optional hardware components responsible for limiting physical memory address accesses by software to support secure processing and contain faults. [12] If included within the System on a Chip (SoC) design, PMP units provide "per-hart M-Mode control registers to allow physical memory access privileges to be specified for each physical memory region" [12].

2.5 Keystone Enclave

As a current project of the CCC, Keystone offers an accessible open-source framework supporting academia and industry with resources to build trustworthy secure hardware enclaves. Keystone is the first open-source framework for building customized TEEs [1]. It is designed for and built upon the RISC-V Privileged ISA. By leveraging trusted hardware, Keystone enables software-defined, hardware-enforced, memory-isolated execution beneath an untrusted Operating System (OS) [1]. Currently, Keystone supports the following three standard TEE primitives: (a) Secure Boot, (b) Secure Source of Randomness, and (c) Remote Attestation [1]. Figure 2 depicts the distinct Keystone Enclave components as they operate within the RISC-V ISA privilege levels, alongside the untrusted OS.

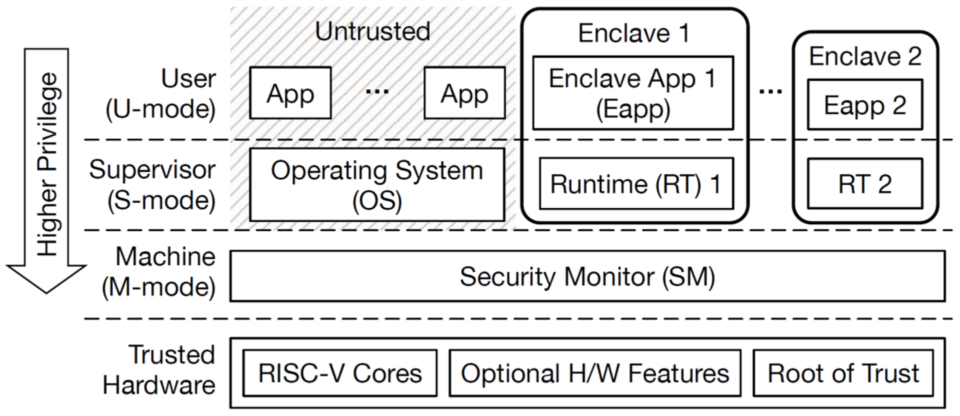


Figure 2. Compute System Operations with Keystone Enclave [1]

2.5.1 Keystone Security Monitor

As the core component for Keystone Enclave, the SM relies solely upon RISC-V standards for operation. This intentional design constraint promotes portability between various RISC-V hardware platforms. This research applies these design principles to port Keystone SM for use on the HiFive Unmatched development platform.

Keystone SM achieves memory isolation for enclave runtimes and Eapps by utilizing PMP hardware built directly into each hart [13]. Precise development platform features, specifications, and configurations are provided in Chapter III. This work concentrates development efforts on extending Keystone SM for use on the Unmatched test platform.

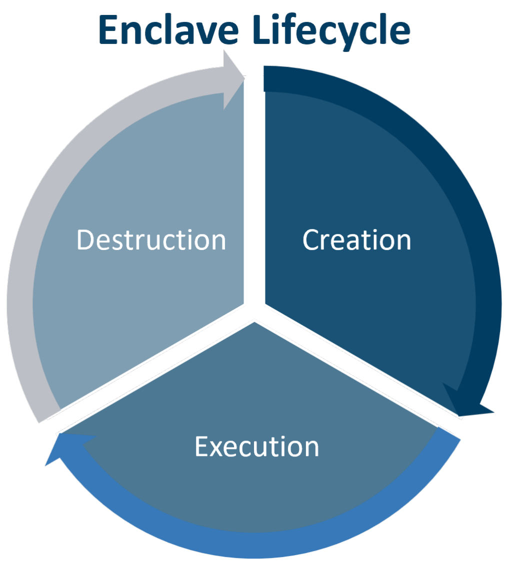


Figure 3. Keystone Enclave Lifecycle

During operation, Keystone SM facilitates secure enclave requests to the OS by calling the appropriate Keystone OpenSBI functions described in Table 1. Keystone enclaves experience three distinct phases during their lifecycle: creation, execution, and destruction. (These phases are illustrated in Figure 3.) Upon a creation request by the OS, the SM measures enclave memory, ensuring that the OS correctly loaded the enclave binaries into physical memory. Then, the SM hashes the page contents and the virtual addresses along with configuration data. At execution, the SM sets PMP entries and delegates control to the enclave entry point. After program completion, the runtime (RT) entity calls the exit function prompting the OS to initiate a destruction request. Upon a destruction call, the SM clears the enclave memory region prior to returning the memory address space to the OS [1].

Table 1. Keystone SM OpenSBI Function Calls [1]

Caller	OpenSBI Function Call	Description
OS	create	validate & measure the enclave
	run	start enclave & boot Eryie RT
	resume	resume enclave execution
	destroy	clean & release enclave memory
RT	stop	pause enclave execution
	exit	terminate the enclave
	attest	get signed attestation report
	random	get secure random values

2.5.2 Keystone Root-of-Trust

Although the Root-of-Trust is typically depicted as a hardware component (as shown in Figure 2), Keystone also supports tamper-proof firmware implementations. Our research leverages this feature, employing modified first and second-stage boot-loaders to simulate the Secure Boot primitive. Furthermore, this work does not attempt to verify, validate, or otherwise assess cryptographic techniques employed by Keystone to achieve TEE primitives –rather, this work examines Keystone Enclave portability claims by attempting to extend its use on unsupported hardware.

2.5.3 Keystone Modular Runtime: Eryie

Once the SM has isolated an enclave's physical memory, the enclave then initializes the enclave RT to run in S-Mode. Eryie has been developed by [1] as an exemplar RT enabling modular system-level abstraction for Eapps. With functionality analogous to an isolated kernel operating inside the enclave, Eryie satiates developers RT needs by including only the necessary capabilities required for specific Eapps in an effort to reduce the Trusted Computing Base (TCB) [1]. Because enclaves are created with

S-Mode privileges, they are capable of running the S-Mode RT, which in turn hosts Eapps in U-Mode. This defensive design permits only the enclave RT access to the shared memory buffer. Furthermore, these abstraction layers allow and encourage developers to substitute the Eryie RT with alternative microkernels, such as seL4 [1].

Currently, the Eryie RT does not support parallel multi-core enclave execution –instead, thread management is delegated to the enclave RT which then runs the multi-threaded Eapps on a single thread. Keystone SM could be extended to support parallel multi-threaded enclave execution by issuing multiple Keystone OpenSBI function calls across different cores. Future performance characterizations will be required to exhaustively evaluate the consequences of these design constraints. Moreover, extending the Keystone Enclave project to fully support multi-threaded applications exceeds the scope of this work.

2.5.4 The seL4 Microkernel

The seL4 microkernel "is a high-assurance, high-performance operating system microkernel" [14]. It has been formally verified for compiler correctness and implementation safety with provable trustworthiness [15]. These traits are desirable for use as a Keystone Enclave RT. Using the now discontinued HiFive Unleashed board, Keystone researchers have demonstrated seL4 functioning as the enclave RT [1]. As the Keystone Enclave project matures and evolves, alternative RTs, such as seL4 must be explored with new hardware platforms. The The seL4 microkernel is not used in this research; however, it has been identified as an additional recommended RT for future Keystone Enclave development. Keystone Enclave RT development opportunities are further discussed in Section 6.3.

2.5.5 Keystone Enclave Applications

With successful modifications to hardware specific M-Mode software (i.e. firmware), the HiFive Unmatched equipped with Keystone SM could conceivably be configured to support the Eryie runtime or the seL4 microkernel. With capable execution environments configured, future Keystone Eapps may be developed, verified, and validated. The Keystone SDK Eapp development for any statically compiled RISC-V binary, so long as all supporting libraries are also included within the Keystone runtime. Respectively, proposed *secure enclave* applications are envisioned that characterize system performance from within enclaves. Such insight would arm decision makers with system performance expectations for TEEs and would encourage the adoption of *Confidential Computing* principles. Proposed Eapps are presented in Section 6.3; however, Eapp development is beyond the scope of this research venture.

2.6 The RISC-V SBI Specification & The OpenSBI Library

The OpenSBI project provides "an open-source reference implementation of the RISC-V SBI specification for platform-specific firmwares executing in M-Mode" [16]. The primary component of OpenSBI is a generic, platform-independent static library (named `libsbi.a`) that implements the SBI interface. RISC-V platform and SoC vendors use this library to link their respective firmware and bootloader implementations to ensure SBI conformity. The OpenSBI project also provides platform-specific static libraries that integrate platform-dependent hardware manipulation functions with the `libplatsbi.a` and `libsbi.a` reference implementations. Collectively, these example firmwares replace the legacy `riscv-pk` bootloader (known as the "Berkeley Bootloader (BBL)") with U-Boot, the universal bootloader [16].

Keystone Enclave v1.0.0, released on 2 March 2021, transitioned their bootloader development from the BBL to the OpenSBI bootloader implementation. Subsequently, this work leverages the OpenSBI project, alongside the Keystone Enclave project, to build and modify novel OpenSBI implementations that incorporate Keystone SM function calls for enclave operations beneath the OS and host kernel. Platform-specific libraries were provided by SiFive to the OpenSBI project for the HiFive Unmatched on 27 July 2021 [16].

2.7 Related Work

Porting Keystone SM to new hardware platforms is only an initial step in the furtherance of *Confidential Computing* principles. To fully implement all Keystone Enclave components on contemporary RISC-V hardware, additional Linux Kernel modifications will need to be baselined into supporting Linux Distributions. Moreover, to encourage the adoption of *Confidential Computing* paradigms, Linux distributions will likely need to provide flexible tools to facilitate porting Keystone Enclave to more devices. To justify extending RISC-V compute systems with TEE technologies, strict performance requirements must also be maintained. The addition of *secure enclave* computing unavoidably impacts system performance [5]; to effectively evaluate TEE performance impacts, future characterization studies must be explored.

In his Master's Thesis, titled *Characterizing Security Monitor and Embedded System Performance Across Distinct RISC-V IP-Cores*, Tullos (2021) conducted relevant performance characterizations for embedded RISC-V devices configured with Keystone Enclave implemented on FPGA hardware [5]. As the RISC-V landscape matures, performance characterizations ought to include ASIC hardware implementations with representative system evaluations for workstation focused systems such as the HiFive Unmatched.

2.8 Summary

Chapter II presents sufficient background knowledge to articulate the distinctions between the*Trusted* and *Confidential Computing* paradigms. It succinctly describes foundational operations of the RISC-V ISA and encapsulates the security mechanisms employed to enforce *Confidential Computing* principles through the use of TEEs. Predominately, this chapter affords perspective by conveying the precise research domain targeted for exploration. Lastly, it recognizes the achievements and contributions by antecedent researchers that lay the groundwork for future Keystone Enclave development.

III. Experimental Platform Configurations

3.1 Overview

Chapter III begins by revealing the selection criteria used to pursue Keystone SM platform expansion. Discussion continues by specifying the relevant hardware features and capabilities of the HiFive Unmatched development platform. The exact hardware configuration used during experimentation is also documented. Subsequent sections contain expositions of the standard unmodified boot-flow along with precise experimental bootloader and firmware modification procedures. These firmware modifications configure and extend Keystone SM for use on the HiFive Unmatched. Relevant console output is presented in Figures 8 and 9 that showcase boot operations with both an unmodified and modified boot-flow. Additional console output is also shown in Figure 10 to substantiate successful Keystone SM integration into the OpenSBI SPL boot-flow layers.

3.2 The HiFive Unmatched by SiFive

The HiFive Unmatched development platform was selected by evaluating its merits against the following four criteria: (a) form factor standardization, (b) commodity Personal Computer (PC) hardware compatibility, (c) Linux OS support, and (d) enhanced SoC monitoring capabilities. Currently, the Unmatched is the only commercially available RISC-V development platform that satisfies each of the desired criterion. The Unmatched is uniquely positioned to facilitate desired research objectives by adopting the Mini-ITX form factor commonly used by many AMD/Intel x86_64 systems. This standard PC form factor enables straightforward hardware extensions via Peripheral Component Interconnect Express (PCIe) and Non-Volatile Memory Express (NVMe) interconnects. Out-of-the-box, the Unmatched is pre-configured with the OpenEmbedded Linux distribution; however, the Ubuntu distribution is leveraged for this work. Enhanced SoC monitoring capabilities are not pursued by this research, with envisioned use-cases proposed in Section 6.3.

Importantly, the Unmatched is a product by SiFive, an industry leader for RISC-V technologies. SiFive company leadership includes three of the RISC-V ISA co-founders: Yunsup Lee, Chief Technology Officer (CTO); Krste Asanović, Chief Architect; and Andrew Waterman, Chief Engineer. Strikingly, Krste Asanović, is a contributing author to the Keystone project [1]. SiFive involvement inspires confidence by incentivizing support for HiFive products as RISC-V specifications and Keystone Enclave implementations evolve.

Furthermore, the Unmatched succeeds the previous generation HiFive Unleashed (the first Linux-capable native RISC-V development platform). Although discontinued, the Unleashed platform was used to demonstrate the first Keystone Enclave implementation on an ASIC RISC-V computer. Moreover, the Unmatched is advertised as "the world's fastest native RISC-V development platform" which sets it apart from other platforms by positioning it as an independent, Linux-capable, native RISC-V workstation –rather than as an embedded system.

3.2.1 Hardware Features & Specifications

The Unmatched is powered by the closed-source SiFive Freedom U740 SoC: a multi-core, 64-bit dual-issue, superscalar RISC-V processor, with advertised performance comparable to the ARM Cortex-A55 [13]. The Freedom U740 contains four Linux-capable U74 application cores supporting RV64GC operations and includes a fifth S7 monitor core supporting RV64IMAC operations. All cores have dual-issue in-order execution pipelines that support a peak sustained execution rate of two Instructions per Clock Cycle (IPC) and maintain a fully-coherent 2 Megabyte (MB) shared Level 2 (L2) cache [13]. Additional board specifications include 16 Gigabytes (GBs) of Double Data Rate 4 (DDR4) synchronous dynamic random-access memory (SDRAM), a 32 MB Quad Serial Peripheral Interface (SPI) Flash, microSD card expansion, Gigabit Ethernet, four Universal Serial Bus (USB) 3.2 Gen 1 Type A ports, one microUSB Joint Test Action Group (JTAG) console port, one x16 PCIe Gen 3 expansion slot, one M.2 M-Key slot for NVMe 2280 SSD modules, and one M.2 E-Key slot for Wi-Fi/Bluetooth modules [13].

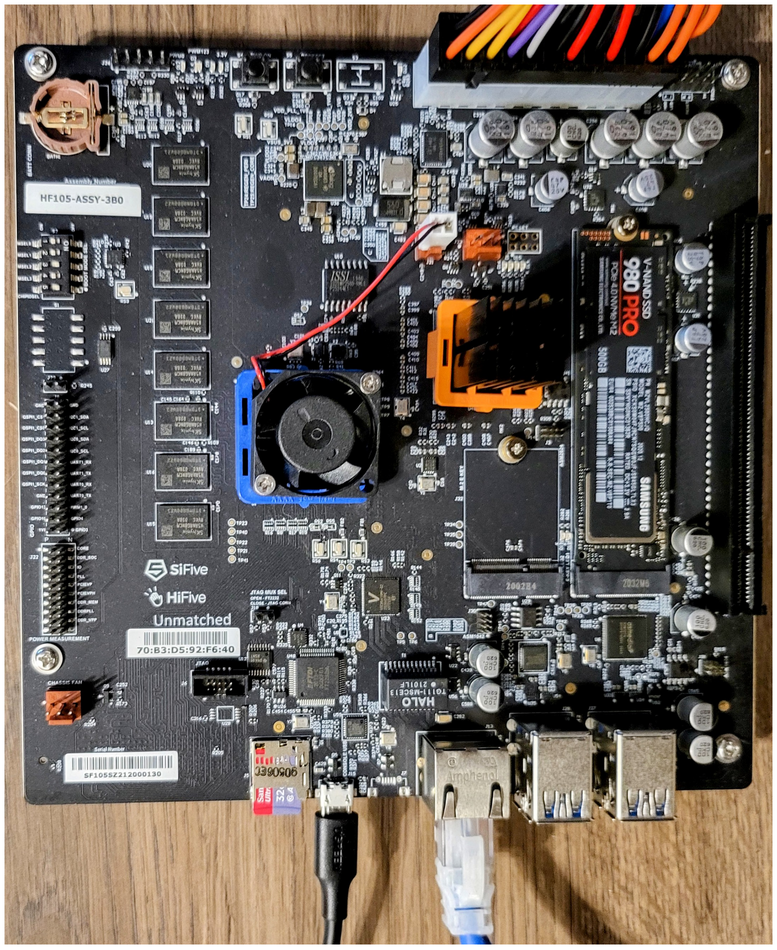


Figure 4. HiFive Unmatched Mini-ITX Development Platform, Hardware Configuration as Tested

The selected hardware configuration utilizes the M.2 M-Key NVMe slot to take advantage of a 500 GB Samsung 980 PRO PCIe 4.0 SSD. Additional components are not strictly required for system testing, but aid performance by providing faster memory technology and could allow for testing in environments without wired internet access. The PCIe expansion slot is not used for this research, with graphical capabilities left for future investigation. Figure 4 captures the test platform, as configured during experimentation.

3.3 Standard Test Configurations

There are few Linux distributions that officially support the RISC-V ISA; fewer still are those which support the Unmatched platform directly. At the time of development, only the OpenEmbedded and Ubuntu distributions officially offer compatible bootable images for the Unmatched. Although the included OpenEmbedded distribution could have been used to support Keystone SM portability, the OS is fundamentally designed for embedded systems. It lacks basic features typically found in other workstation-focused distributions, such as Ubuntu. Notably, OpenEmbedded does not include a package manager, such as Advanced Packaging Tool (APT), Debian Package Management System (DPKG), Pacman Package Manager (pacman), Red Hat Package Manager (RPM), or Yellowdog Updater Modified (YUM); thus, installing, updating, and removing testing packages and programs requires rebuilding independent static OS configurations. (The time required to build these static bootable images often exceeds eight hours and consumes 200 GB or more of disk space per configuration.) Ultimately, Ubuntu better suits the Unmatched platform by supporting existing workstation programs and work-flows without OS modifications.

3.3.1 Standardized Boot-Flow

The "SiFive FU740-C000 Manual" details the boot process stages for the HiFive Unmatched [13]. Typical boot operations proceed in the following order of precedence:

0. Power on Reset (PoR)
1. Zeroth Stage Bootloader (ZSBL), stored in an on-chip mask Read Only Memory (ROM)
2. First Stage Bootloader (FSBL), the U-Boot Secondary Program Loader (SPL)
3. Second Bootloader (SBL), containing the U-Boot Image Tree Blob (ITB), Device Tree Blob (DTB), and platform-specific OpenSBI reference implementation
4. EXTLINUX, for Linux Kernel version control and management
5. Distribution Linux Kernel, modified from the upstream mainline Linux Kernel maintained by Linus Torvalds

For baseline performance characterizations conducted on the system without Keystone SM, this boot-flow is preconfigured for Ubuntu and provided by Canonical within a preinstalled server image. By flashing this bootable image onto a microSD card, the Unmatched successfully boots into Ubuntu. Figure 5 delineates the unmodified boot-flow for standard test platform configurations without Keystone SM modifications. Where appropriate, PoR, SoC ROM, microSD, and NVMe M.2 SSD glyphs designate they physical location of each firmware stage as they resides in hardware on the Unmatched system.

3.3.2 Unmodified Ubuntu Preinstalled Server Images

To best ensure testing uniformity across numerous Ubuntu releases, only one build for each of the available preinstalled server images are downloaded from Canonical,

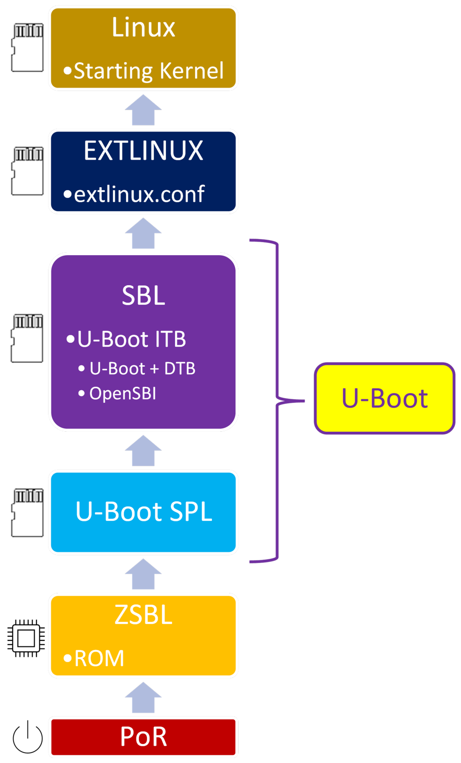


Figure 5. Standardized Boot Flow for the HiFive Unmatched, microSD

the publisher of Ubuntu [17, 18, 19]. The publishing date for these bootable images is 4 January 2022. Furthermore, once these images have been configured for use on the system, they are never updated and remain unpatched throughout performance testing. This practice is inadvisable for operational systems; however, updating is avoided to enforce strict software version control during testing. Canonical hosts daily builds of preinstalled server images for the Unmatched. Available major Ubuntu OS releases include the following distribution versions:

- 21.04 (Hirsute Hippo) [17]
- 21.10 (Impish Indri) Beta [18]

- 22.04 Long Term Support (LTS) (Jammy Jellyfish) Development Branch [19]

All preinstalled server images are downloaded and flashed to the microSD bootable medium for benchmarking, which is detailed in Chapter IV.

3.3.3 Configuring the M.2 NVMe Drive

By default, the Unmatched Boot Mode Selector Dual In-Line Package (DIP) switches are configured to boot from the microSD card slot. The provided microSD card has an advertised maximum read speed of 98 MB/s and maximum write speed of 58 MB/s, which is serviceable for initializing the system and preparing the NVMe drive. To substantially improve disk performance and reduce average system response time, EXTLINUX, the distribution Linux Kernel, and the Ubuntu OS are configured to run on the 500 GB Samsung 980 PRO PCIe 4.0 NVMe M.2 SSD. For comparison, this M.2 drive boasts read speeds of up to 7,000 MB/s and write speeds of up to 5,100 MB/s.

For version control and efficiency purposes, an NVMe-to-USB-C adapter is leveraged to preemptively flash corresponding preinstalled Ubuntu server images to the SSD prior to system configuration. Once the Unmatched board initializes from the microSD card (with the SSD drive prepared and physically mounted to the board), the following commands are executed to configure U-Boot for use with the NVMe device:

1. `sudo mount /dev/nvme0n1p1 /mnt` to mount the NVMe boot partition
2. `sudo chroot /mnt` to change the apparent root directory to the NVMe mount point
3. `nano /etc/default/u-boot` to edit the U-Boot configuration file

4. `U_BOOT_ROOT="root=/dev/nvme0n1p1"` to set the relevant environment variable

5. `u-boot-update` to save and update the U-Boot configuration file

6. `exit` to exit the `chroot` session

7. `sudo shutdown -h now` to shutdown the system

Upon reboot, the system is configured start Ubuntu from the NVMe drive. Figure 6 highlights the unmodified boot-flow for standard test platform configurations, after the M.2 NVMe SSD has been configured. Notably, the EXTLINUX and distribution Linux Kernel are the only boot-flow stages accessed by the M.2 NVMe SSD. At the time of development, U-Boot maintainers had not finalized support for flashing the FSBL and SBL to the on-board Quad SPI interface. Consequently, the microSD card is required to bring up the Unmatched, even when the NVMe drive is configured to host the entire bootable image. As of 4 December 2021, U-Boot maintainers have added support for flashing and booting directly from the Quad SPI, eliminating the requirement to boot from the microSD. This research does not rely upon the Quad SPI boot method; however, additional testing configurations utilizing this method are proposed in Section 6.3.

3.4 Modified Test Configurations

After each of the three targeted Ubuntu releases are prepared for use on the HiFive Unmatched, modifications ensue to integrate Keystone SM into their respective boot-flows. Due to the physical separation of boot-flow components across microSD and NVMe devices, development modifications to the U-Boot firmware become more practical. An independent Linux environment –complete with constituent SDKs, code repositories, and toolchains– is required to pursue firmware development. This effort

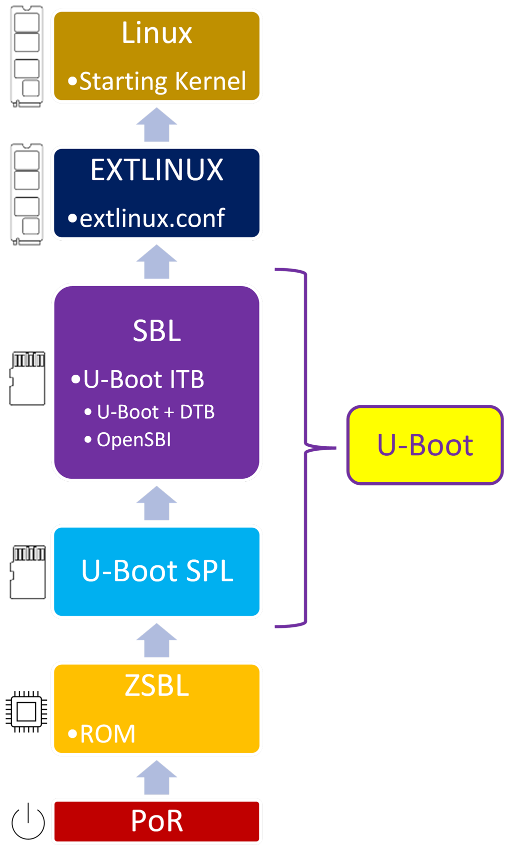


Figure 6. Standardized Boot Flow for the HiFive Unmatched, NVMe M.2 SSD

conducts development work on a Kubuntu 20.04 Virtual Machine (VM) –although the specific Linux distribution used for development is mostly arbitrary. Notably, complete Keystone Enclave integration with the Unmatched platform requires distribution kernel modifications. Because the Unmatched represents the bleeding-edge of RISC-V hardware, at the time of writing, Canonical has not yet incorporated RISC-V ISA distribution kernel support into their existing open-source kernel repository. Consequently, without source code for the precise Ubuntu Distribution Kernels used during testing, Keystone Enclave on the Unmatched lacks support for kernel-specific drivers that interface OS enclave requests with the underlying Keystone OpenSBI function calls described in Section 2.6. Accordingly, this effort addresses the integration of Keystone SM into the HiFive Unmatched boot-flow; however, the continued development, integration, and testing of additional Keystone Enclave components is left for future research.

3.4.1 Boot-Flow Modifications

To configure Keystone SM for use with the Unmatched, modifications to the platform-specific OpenSBI reference implementation (detailed in Section 2.6) are conducted to integrate SM function calls into the U-Boot bootloader firmware executing in M-Mode. The OpenSBI project enables modifications to the HiFive Unmatched boot-flow by publishing the Freedom U740-specific library (named `sifive_u740.c`); the first release of this library occurred on 27 July 2021 [16]. With the source code to this platform-specific OpenSBI reference implementation, along with the supporting RISC-V GNU's Not Unix (GNU) Compiler Toolchain, integration of Keystone SM becomes possible.

OpenSBI is extended with Keystone SM functionality by using an *out-of-tree* platform build configuration, supported by the OpenSBI build process. This configuration build is contained within the Keystone SDK. Because Keystone does not officially support the Unmatched development platform, several Keystone Enclave components require manual intervention and patching to successfully build. Markedly, older versions of the OpenSBI and Linux Kernel repositories are archived as submodules within the Keystone Enclave repository to ensure continued compatibility with legacy hardware. To support the Unmatched, these submodules are replaced with current versions of their respective repositories and necessarily break compatibility with older platforms, such as the HiFive Unleashed. Figure 7 highlights the boot-flow

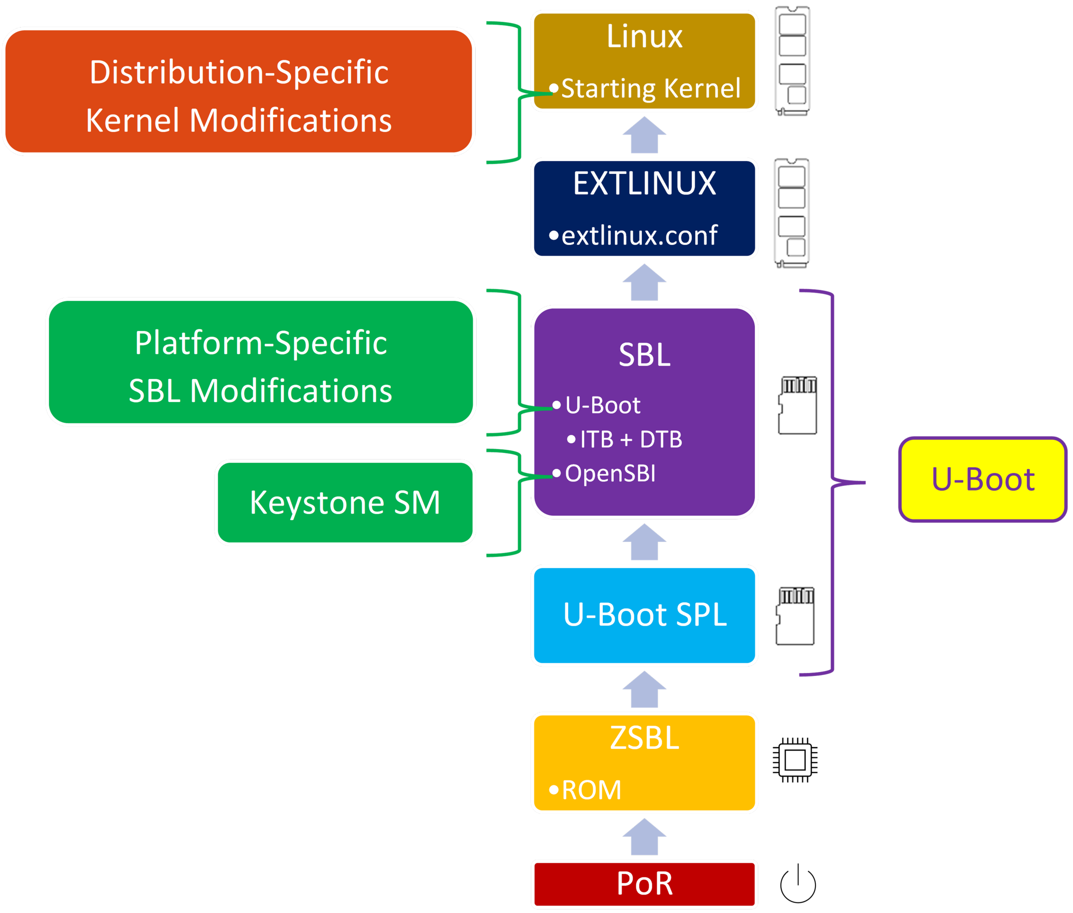


Figure 7. Modified Boot Flow for the HiFive Unmatched with Keystone SM

modifications required to implement Keystone SM on the Unmatched test platform.

After OpenSBI is configured on the Kubuntu host to include the SM, Keystone build scripts leverage the RISC-V GNU Toolchain to generate the modified `fw_dynamic.bin` OpenSBI platform configuration binary used to build U-Boot proper. Development proceeds by using the U-Boot repository build processes to generate the `u-boot.itb` Image Tree Blob (ITB) and `u-boot-spl.bin` Secondary Program Loader (SPL) binary files from the `fw_dynamic.bin` file created by the Keystone SDK. These two files are generated from the U-Boot working directory and comprise the U-Boot bootloader that is later flashed onto the microSD card for testing, which is described in Chapter IV. U-Boot `make` commands are executed in the U-Boot build directory and are included below:

```
CROSS_COMPILE=riscv64-unknown-linux-gnu- make sifive_unmatched_defconfig
CROSS_COMPILE=riscv64-unknown-linux-gnu- make -j N
```

(`N` represents the number of jobs created to complete the `make` request and should be implemented according to unique development hardware support.)

3.4.2 Kernel Modifications

As alluded to in the opening paragraph of Section 3.4, source code is not available for RISC-V builds of the Ubuntu Distribution Linux Kernel. Indeed, distribution Linux kernels (such as the RISC-V specific Ubuntu kernels required by this effort) are maintained independently by their respective publishers. Distribution Linux kernels serve as downstream derivatives of The Linux Kernel mainline (maintained by Linus Torvalds) that incorporate distribution-specific features. (One pertinent example is support for Snap packages, a software packaging and distribution platform developed by Canonical). Rather than abandoning the kernel modifications implemented by the Keystone Enclave build process, development continues by applying and cross-compiling Keystone SM and SiFive patches to available RISC-V supported builds of The Linux Kernel (Version `5.13.0-19.19`) for future RV64GC hardware targets and alternative open-source distributions. By default, the Keystone kernel build process produces the `Image.gz` Linux Kernel for the OpenEmbedded distribution; this is an artifact of existing hardware support for the discontinued HiFive Unleashed development platform. Completing this build stage leverages existing `makefile` scripts to generate the `hifive-unmatched-a00.dtb` Device Tree Blob (DTB). This firmware component contains the specific board hardware descriptors needed to boot the experimental OpenSBI + Keystone SM build. For this research, all performance characterizations utilize unmodified Ubuntu Distribution Linux Kernels. Appropriately, conducted Linux Kernel modifications support Keystone Enclave; however, they remain unused by this work and are provided for future development.

3.4.3 Bootable Image Preparation

Lastly, development concludes by creating a bootable image file and flashing it to the microSD card. To build modified Ubuntu disk images, begin by flashing the desired prebuilt server image to the microSD card. Once this process completes, utilize the `dd` tool to overwrite the ITB and DTB boot partitions with the modified U-Boot bootloader. These commands overwrite the two U-Boot partitions with the OpenSBI + Keystone SM enabled implementation:

```
sudo dd if=u-boot-spl.bin of=/dev/sdX* seek=34
sudo dd if=u-boot.itb of=/dev/sdX seek=2082
```

*`X` represents a place holder for the appropriate mount point.

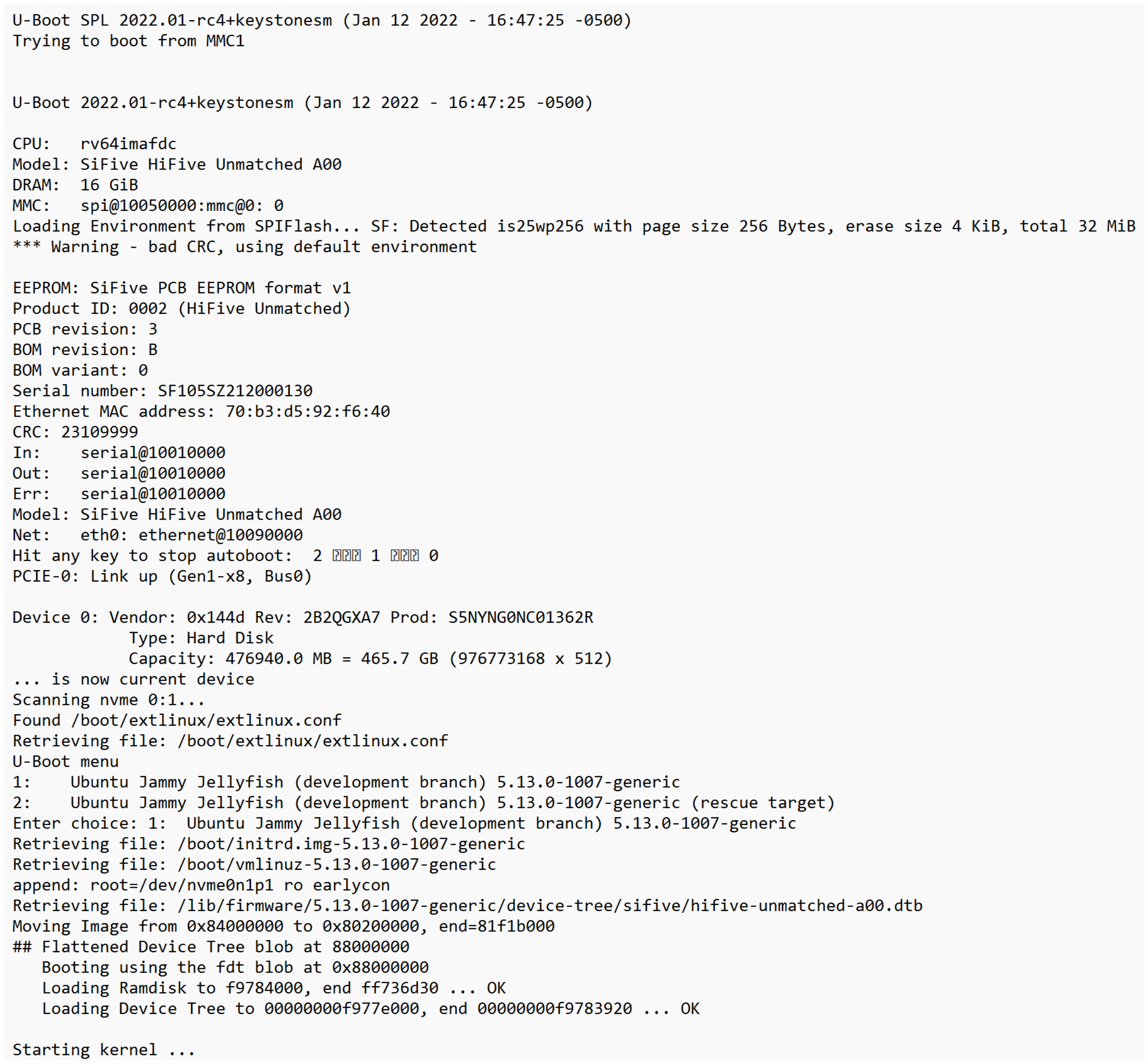
Finally, repeat the `dd` commands to overwrite the U-Boot partitions on the microSD card for each Ubuntu release selected for testing. Bring up the system by inserting the modified bootable microSD card into the HiFive Unmatched and pressing the PoR button. Upon booting the device, observe the serial console output to verify successful U-Boot modifications. Once the device boots, set the `U_BOOT_ROOT` environment variable as described in Section 3.3.3 to use the preconfigured NVMe drive with the Ubuntu OS. Figure 8 shows the standard serial console output of the unmodified Ubuntu 22.04 LTS system, while Figure 9 substantiates the modified U-Boot SPL build. Figure 10 captures the Ubuntu terminal upon system login.

3.5 Summary

Chapter III provides a synopsis describing the hardware selection criteria used to guide Keystone platform portability efforts. This chapter also conveys experimental development procedures by outlining a broad, repeatable framework for modifying and configuring bootloader firmware for the Unmatched platform. By physically isolating development to the microSD card, relevant boot-flow elements are extended and integrated into the Unmatched platform without modifications to the Ubuntu OS. Critically, chartered development exposes the need for Linux distribution publishers to integrate novel RISC-V ISA Linux kernels into existing open-source distribution kernel repositories. If Linux distribution publishers were to support TEEs mechanisms directly, *Confidential Computing* principles are more likely to propagate throughout the RISC-V hardware, firmware, and software industries.

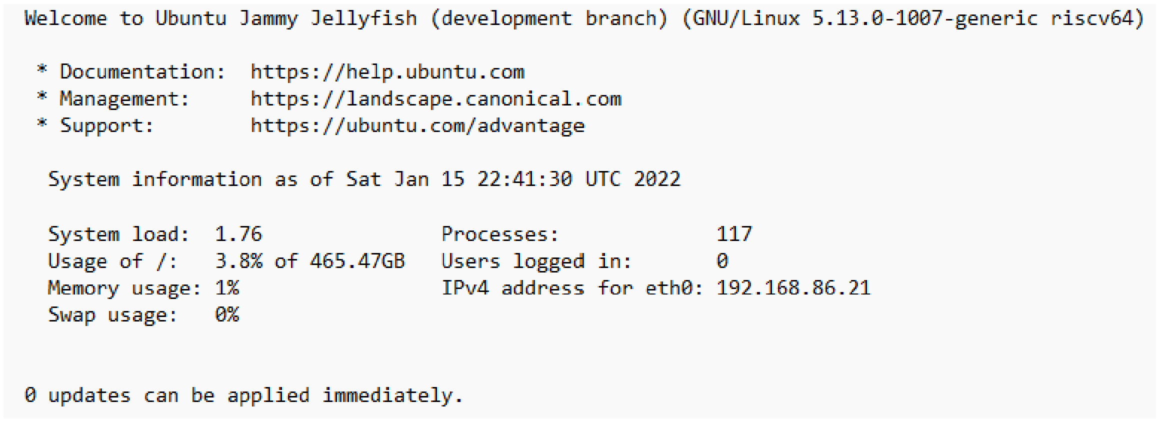
```
U-Boot SPL 2021.07+dfsg-0ubuntu10 (Dec 03 2021 - 14:16:56 +0000)
Trying to boot from MMC1


U-Boot 2021.07+dfsg-0ubuntu10 (Dec 03 2021 - 14:16:56 +0000)

CPU:   rv64imafdc
Model: SiFive HiFive Unmatched A00
DRAM:  16 GiB
MMC:   spi@10050000:mmc@0: 0
Loading Environment from nowhere... OK
EEPROM: SiFive PCB EEPROM format v1
Product ID: 0002 (HiFive Unmatched)
PCB revision: 3
BOM revision: B
BOM variant: 0
Serial number: SF105SZ212000130
Ethernet MAC address: 70:b3:d5:92:f6:40
CRC: 23109999
In:    serial@10010000
Out:   serial@10010000
Err:   serial@10010000
Model: SiFive HiFive Unmatched A00
Net:   eth0: ethernet@10090000
Hit any key to stop autoboot:  2 □□□ 1 □□□ 0
PCIE-0: Link up (Gen1-x8, Bus0)

Device 0: Vendor: 0x144d Rev: 2B2QGXA7 Prod: S5NYNG0NC01362R
            Type: Hard Disk
            Capacity: 476940.0 MB = 465.7 GB (976773168 x 512)
... is now current device
Scanning nvme 0:1...
Found /boot/extlinux/extlinux.conf
Retrieving file: /boot/extlinux/extlinux.conf
791 bytes read in 3 ms (256.8 KiB/s)
U-Boot menu
1:    Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic
2:    Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic (rescue target)
Enter choice: 1:  Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic
Retrieving file: /boot/initrd.img-5.13.0-1007-generic
100347184 bytes read in 109 ms (878 MiB/s)
Retrieving file: /boot/vmlinuz-5.13.0-1007-generic
29497344 bytes read in 33 ms (852.4 MiB/s)
append: root=/dev/nvme0n1p1 ro earlycon
Retrieving file: /lib/firmware/5.13.0-1007-generic/device-tree/sifive/hifive-unmatched-a00.dtb
10529 bytes read in 8 ms (1.3 MiB/s)
Moving Image from 0x84000000 to 0x80200000, end=81f1b000
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Using Device Tree in place at 0000000088000000, end 0000000088005920

Starting kernel ...
```

Figure 8. U-Boot SPL Version: 2021.07+dfsg-0ubuntu10



```
U-Boot SPL 2022.01-rc4+keystonesm (Jan 12 2022 - 16:47:25 -0500)
Trying to boot from MMC1


U-Boot 2022.01-rc4+keystonesm (Jan 12 2022 - 16:47:25 -0500)

CPU:   rv64imafdc
Model: SiFive HiFive Unmatched A00
DRAM:  16 GiB
MMC:   spi@10050000:mmc@0: 0
Loading Environment from SPIFlash... SF: Detected is25wp256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB
*** Warning - bad CRC, using default environment

EEPROM: SiFive PCB EEPROM format v1
Product ID: 0002 (HiFive Unmatched)
PCB revision: 3
BOM revision: B
BOM variant: 0
Serial number: SF105SZ212000130
Ethernet MAC address: 70:b3:d5:92:f6:40
CRC: 23109999
In:    serial@10010000
Out:   serial@10010000
Err:   serial@10010000
Model: SiFive HiFive Unmatched A00
Net:   eth0: ethernet@10090000
Hit any key to stop autoboot:  2 ▯▯▯ 1 ▯▯▯ 0
PCIE-0: Link up (Gen1-x8, Bus0)

Device 0: Vendor: 0x144d Rev: 2B2QGXA7 Prod: S5NYNG0NC01362R
            Type: Hard Disk
            Capacity: 476940.0 MB = 465.7 GB (976773168 x 512)
... is now current device
Scanning nvme 0:1...
Found /boot/extlinux/extlinux.conf
Retrieving file: /boot/extlinux/extlinux.conf
U-Boot menu
1:    Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic
2:    Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic (rescue target)
Enter choice: 1:  Ubuntu Jammy Jellyfish (development branch) 5.13.0-1007-generic
Retrieving file: /boot/initrd.img-5.13.0-1007-generic
Retrieving file: /boot/vmlinuz-5.13.0-1007-generic
append: root=/dev/nvme0n1p1 ro earlycon
Retrieving file: /lib/firmware/5.13.0-1007-generic/device-tree/sifive/hifive-unmatched-a00.dtb
Moving Image from 0x84000000 to 0x80200000, end=81f1b000
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Loading Ramdisk to f9784000, end ff736d30 ... OK
   Loading Device Tree to 00000000f977e000, end 00000000f9783920 ... OK

Starting kernel ...
```

Figure 9. U-Boot SPL Version: 2022.01-rc4+keystonesm



```
Welcome to Ubuntu Jammy Jellyfish (development branch) (GNU/Linux 5.13.0-1007-generic riscv64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Jan 15 22:41:30 UTC 2022

  System load:  1.76               Processes:             117
  Usage of /:   3.8% of 465.47GB   Users logged in:       0
  Memory usage: 1%                 IPv4 address for eth0: 192.168.86.21
  Swap usage:   0%


0 updates can be applied immediately.
```

Figure 10. Ubuntu 22.04 (Jammy Jellyfish) Terminal Output

IV. Design of Experiments & Testing Methodology

4.1 Overview

Chapter IV exhibits the design of experiments and testing methodology used to construct performance characterizations for the HiFive Unmatched development platform. To evaluate the Unmatched as a hardware candidate for TEE supplementation, a thorough understanding of system performance in relation to application requirements must be determined. Unavoidably, the addition of TEEs will impact compute system performance [5]. Therefore, it remains imperative to quantify the performance overheads imposed by Keystone Enclave and its constituent components. Pointedly, this research seeks to investigate the passive impacts Keystone SM has on synthetic benchmarking performance and kernel boot times for the HiFive Unmatched. Performance characterization assessments consist of three independent case studies, each containing two experiments. Experiment I measures synthetic benchmarking performance and Experiment II measures distribution kernel boot times. Performance characterizations are derived from these experiments and are analyzed in Chapter V. Resulting benchmark scores and boot times are then interpreted to characterize the overall system performance of the HiFive Unmatched as both a native RISC-V workstation and as a candidate hardware platform for TEE augmentation.

4.2 Experiment Objectives

To characterize the performance of HiFive Unmatched development platform, three unique case studies are constructed to run two distinct experiments. In direct support of the characterization and benchmarking objectives outlined in Section 1.3, the following ancillary objectives are defined:

- Assign performance scores to the HiFive Unmatched system, using the Stress-NG benchmarking suite for each of the six experimental configurations.
- Assess distribution kernel boot times, using timestamps reported by internal system logs.

Collectively, the results from these experiments form a substantive base to evaluate the HiFive Unmatched as a hardware platform target for future Keystone Enclave integration.

4.3 System Under Test

Figure 11 captures the functional block diagram for the SUT and Component Under Test (CUT). Control Parameters are not depicted; however, precise configuration specifications are detailed in Section 4.8. All conducted experiments for each of the three case studies utilize the same SUT. Case study configurations are presented in Tables 5, 6, and 7.

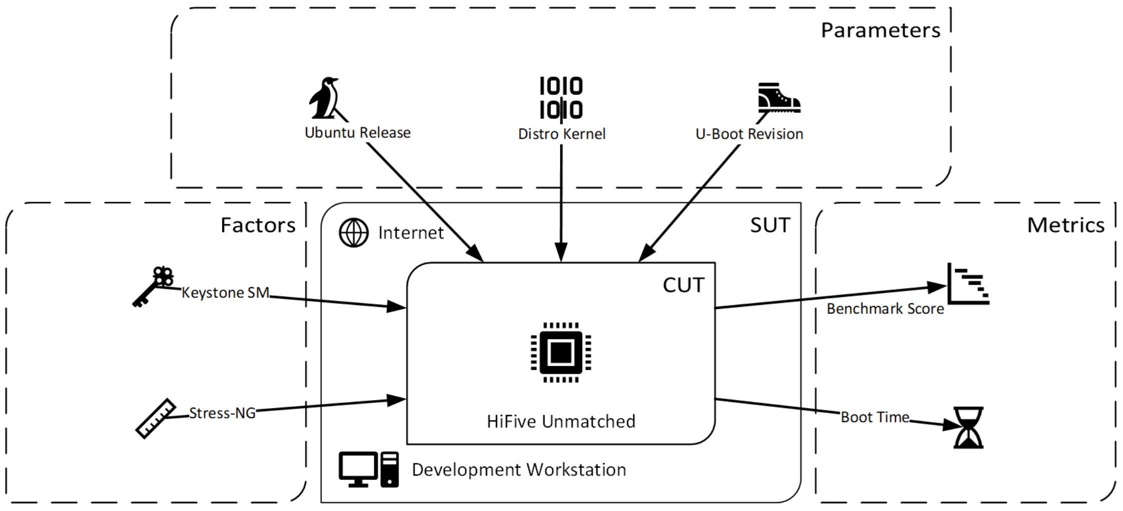


Figure 11. Functional Block Diagram - SUT: HiFive Unmatched

4.4 Assumptions

Previous work by [5] evaluates Keystone Enclave performance on RISC-V Intellectual Property (IP) softcore processors, instantiated on FPGA hardware. An ASIC Keystone Enclave implementation is assessed by [1], leveraging the HiFive Unleashed development platform during testing. Benchmarking results and performance characterizations from [1] and [5] indicate that the primary contributors to performance overhead are enclave creation, Eapp raw binary size, and Eryie RT boot operations –assuming single-threaded binary execution without context switching and that the particular hardware implementation used maintains similar memory Read/Write speeds between enclave disk storage and OS disk storage.

As discussed in Section 3.3.3, the Read/Write speeds for the microSD card (where the U-Boot bootloader and modified OpenSBI + Keystone SM reside) are dramatically slower than the Read/Write speeds for the M.2 NVMe SSD (where the distribution Linux Kernel and OS are stored). Although complete enclave testing has not yet been implemented nor evaluated, prospective evaluators must consider the underlying memory technology that supports TEE configurations. Anticipated enclave performance for the Unmatched is expected to result in substantially greater average system response times when compared to operations performed without enclave support. Because development work for this effort is limited to porting Keystone SM (and thus, does not evaluate benchmarking performance within enclaves), the expected performance differences between unmodified and modified boot-flows should be negligible for system bring-up procedures. These experimental assumptions are asserted prior to conducting performance characterizations:

- Keystone SM function calls are inaccessible without kernel support; their passive impact to performance is intertwined with distinct OpenSBI revisions. Performance variations, if any, are presumed to result from differences in OpenSBI implementations, more so than from specific Keystone SM modifications.

- Successive benchmarking runs are interpreted as independent events.
- A sample size of 30 is suitable for applying the Central Limit Theorem (CLT).
- Self-reported distribution kernel timestamps are trustworthy.
- Security mechanisms, such as encryption, built into Keystone SM are not assessed.

4.5 Control Parameters

The hardware configuration identified in Section 3.3.3 and Section 3.4 is held constant for both characterization studies. Table 2 lists the control parameters and summarizes the tested hardware configuration.

Table 2. Control Parameters

Parameter	Description
Processor	SiFive RISC-V (4 Cores)
Motherboard	SiFive HiFive Unmatched A00
Memory	16GB
Disk	Samsung SSD 980 PRO 500GB + 32GB SD32G
Chipset	SiFive FU740-C000 RISC-V SoC
File-System	ext4

4.6 Independent Variables

Preinstalled server images for each Ubuntu release contain constituent distribution kernels, bootloaders, and compilers. Experiments within each case study are purposefully restricted to testing across equivalent Ubuntu releases (e.g. the same OS release number, such as 22.04). The U-Boot SPL boot-flow layer contains the OpenSBI firmware where Keystone SM modifications are integrated into the system.

Therefore, to isolate otherwise confounding variables, only the U-Boot bootloader is altered between configurations within individual case studies. Importantly, not all distribution kernels, U-Boot revisions, and compiler versions are compatible with every major OS release. Consequently, firmware variations between different Ubuntu releases necessarily contain differing independent variable values; however, these distinctions are separated through the use of case studies and should not be compared directly. To glean insight about system performance across case studies, further testing is required. Tables 5, 6, and 7 in Section 4.8 describe precise case study configurations. With the notable exception of benchmarking, independent variables for Experiment I and Experiment II and are identical and are captured in Table 3.

Table 3. Independent Variables

Variable	Value	Description
OS	21.04, 21.10, 22.04 LTS	Ubuntu Distribution Release
Kernel	5.11.0.1007-generic, 5.13.0-1004-generic, 5.13.0.1007-generic	Ubuntu Distribution Linux Kernel
Bootloader	2021.01+dfsg-3ubuntu9, 2021.07+dfsg-0ubuntu8, 2021.07+dfsg-0ubuntu10, 2022.01-rc4+keystonesm	U-Boot SPL Revision
Compiler	GCC 10.3.0, GCC 11.2.0	Compiler Name & Version
Benchmark #	1, 2, 3, ... 20	Compatible Benchmarks from Stress-NG Benchmarking Suite

4.7 Response Variables

Response variables for Experiment I include synthetic benchmarking scores for each of the 20 compatible benchmarks selected from the Stress-NG benchmarking suite and are described in Table 9 [20]. Experiment II response variables capture distribution kernel boot times for each of the six testing configurations. Table 4 highlights the response variables for each experiment.

Table 4. Response Variables

Variable	Units	Description
Benchmark Score	Numerical Value	Measured in Bogus Operations per Second (Bogo OPs/s)
Boot Time	Time	Distribution Kernel Boot Time, Measured in Seconds

4.8 Experiment Design

Three case studies are explored to evaluate the passive impact Keystone SM may have on HiFive Unmatched system performance. For each case study, two experiments are conducted: Experiment I - Synthetic Benchmarking; and Experiment II - Distribution Kernel Boot Time. Tables 5, 6, and 7 reflect the case study arrangements and capture the precise configurations used for each experiment. To ease the developmental burden, the same OpenSBI + Keystone SM bootloader implementation is integrated into the U-Boot SPL for each of the three modified configurations.

4.8.1 Experiment I - Synthetic Benchmarking

The first experiment leverages the Phoronix Test Suite (PTS) (described in Section 4.9.2) to download, install, manage, and operate the Stress-NG benchmarking suite. Once configured on the SUT, 20 compatible tests are performed 30 times each, for both configurations within each case study. Iterative benchmarking runs are first performed on the Unmatched platform (as configured in Section 3.3) running standard Ubuntu preinstalled server images. Once the unmodified benchmarking scores are tabulated as a baseline, testing progresses by repeating benchmarking runs across the modified Ubuntu releases –with each configuration prepared as described in Section 3.4.

4.8.1.1 Synthetic Benchmarking Procedure

The procedure for Experiment I is performed by executing the following steps using Secure Shell (SSH) to interact with the SUT:

1. After initial Ubuntu logon, download the PTS `*.deb` package using the `wget` command [21].

2. Install PTS using `sudo dpkg -i *.deb`.

3. Install PTS dependencies using `sudo apt --fix-broken install`.

4. Launch PTS using the `phoronix-test-suite` command.

5. Connect PTS to OpenBenchmarking.org by logging in with
   `phoronix-test-suite openbenchmarking-login`.
   (Requires a free OpenBenchmarking.org account; not required for testing, but useful for automated logging.)

6. Install the Stress-NG benchmarking suite with
   `phoronix-test-suite install stress-ng`.

7. Install Stress-NG dependencies with
   `phoronix-test-suite install-dependencies stress-ng`.

8. Execute `export FORCE_TIMES_TO_RUN=30` to set the environment variable that controls the number of runs per benchmark to 30. (The default value is dynamic, starting with three runs and increases based upon standard deviation.)

9. Begin benchmarking with `phoronix-test-suite benchmark stress-ng`.

10. Enter the requested reporting data for result uploads and select all compatible tests for testing.

11. Wait for each benchmark within the Stress-NG suite to perform 30 runs, sequentially for benchmarks #1, 2, 3 ... through 20.

12. Upon test completion, indicate `Y` to upload relevant configuration information, system logs, and benchmark results to OpenBenchmarking.org [22].

13. Repeat this procedure for each of the six configurations.

4.8.2 Experiment II - Distribution Kernel Boot Time

The second experiment measures distribution kernel boot times at system start-up by logging `stdout` from the serial console. Kernel boot time measurements are self-reported by the SUT and begin when the `Starting Kernel ...` and `[0.000000]` timestamp messages print to the screen. Distribution kernel boot time measurements conclude by recording the timestamp reported by `stdout` that immediately precedes the `Welcome to Ubuntu` console message. Because initial timestamps are always zero, the difference of these two measurements reflects the distribution kernel boot time. To restrict start-up measurements to only the kernel boot-flow layer, the time taken by the system to initialize the OS and present the login prompt is excluded from evaluation. For each case study, testing begins with the standard configuration described in Section 3.3 and ends with the modified configuration detailed in Section 3.4. Distribution kernel boot time measurements are only taken from the CUT following graceful shutdowns; boot performance after faults have not been examined. Case studies are explored sequentially with 30 independent boot time measurements performed before switching test configurations.

4.8.2.1 Distribution Kernel Boot Time Procedure

Experiment II requires a serial console connection between the CUT and the host development workstation via the JTAG protocol over USB. The procedure for

Experiment II consists of the following steps:

1. For each configuration, after initial Ubuntu logon, power down the system with `sudo shutdown -h now`.

2. Establish a new serial connection from the CUT to the SUT development host. (PuTTY is used to connect and log `stdout` with the Windows OS; however this procedure can also be accomplished using other native Linux tools.)

3. Configure PuTTY to capture `stdout` using session logging features and open the serial connection.

4. From off, press the `PWR ON` button to bring-up the CUT.

5. Capture `stdout` messages to store the `Starting Kernel ...` and `Welcome to Ubuntu` timestamps.

6. Once the system completes booting, logon and power down the CUT to prepare it for the next testing run using `sudo shutdown -h now`.

7. Save and close the PuTTY session to record logs for further analysis.

8. Repeat this procedure 30 times for each of the two configurations within the three case studies.

4.8.3 Case Study A: Ubuntu 21.04 - Hirsute Hippo

Case Study A assesses performance for the CUT running Ubuntu 21.04, Hirsute Hippo. Released on 22 April 2021, Ubuntu 21.04 represents the oldest Ubuntu release officially available for the Unmatched platform. As of January 2022, Ubuntu 21.04 is the only tested release that has reached End of Life (EOL) and no longer receives standard support by Canonical [23]. Accordingly, the Ubuntu Distribution Kernel used by Case Study A differs by a major version number (`5.11.0-` vs. `5.13.0-`). Uniquely, configurations in Case Study A rely upon an older major version of the GNU Compiler Collection (GCC) compiler (`10.3.-` vs `11.2.-`). Notably, the U-Boot bootloader used by the Standard A configuration (major revision `2021.01-`) lacks full support for graceful shutdowns. Due to the rapid pace of OS version releases for the Unmatched, this bug will not be retroactively fixed for older firmware; instead users of Ubuntu Release 21.04 will need to physically hold down the `PWR ON` button after the `sudo shutdown -h now` command is issued and completed. Table 5 contrasts the modified Keystone SM configuration with the standard Ubuntu 21.04 preinstalled server image configuration provided by Canonical [17]. Firmware implementation differences are emphasized with red text.

Table 5. Case Study A Configurations: Ubuntu 21.04 (Hirsute Hippo)

Configuration:	Standard A	Modified A
Processor	SiFive RISC-V (4 Cores)	SiFive RISC-V (4 Cores)
Motherboard	SiFive HiFive Unmatched A00	SiFive HiFive Unmatched A00
Memory	16GB	16GB
Disk	Samsung SSD 980 PRO 500GB + 32GB SD32G	Samsung SSD 980 PRO 500GB + 32GB SD32G
Chipset	SiFive FU740-C000 RISC-V SoC	SiFive FU740-C000 RISC-V SoC
OS	Ubuntu 21.04	Ubuntu 21.04
Kernel	5.11.0-1007-generic (riscv64)	5.11.0-1007-generic (riscv64)
Bootloader	U-Boot SPL 2021.01+dfsg-3ubuntu9	U-Boot SPL 2022.01-rc4+keystonesm
Compiler	GCC 10.3.0	GCC 10.3.0
File-System	ext4	ext4

4.8.4 Case Study B: Ubuntu 21.10 - Impish Indri

Case Study B assesses performance for the CUT running Ubuntu 21.10, Impish Indri (Beta). Released on 14 October 2021, Ubuntu 21.10 represents another interim Ubuntu release. With only nine months of Standard Support, Ubuntu 21.10 is expected to reach EOL in July 2022 [23]. Unlike Hirsute Hippo, Impish Indri shares major version implementations of the Ubuntu Distribution Linux Kernel (`5.13.0-`), the GCC compiler (`11.2.-`), and U-Boot SPL (`2021.07-`) with Ubuntu 22.04, Jammy

Jellyfish. Testing release 21.10 narrows the U-Boot SPL development time gap between the Standard B and the Modified B configuration firmware implementation dates. Table 6 contrasts the modified Keystone SM configuration with the standard Ubuntu 21.10 preinstalled server image configuration provided by Canonical [18]. Firmware implementation differences are emphasized with green text.

Table 6. Case Study B Configurations: Ubuntu 21.10 (Impish Indri)

Configuration:	**Standard B**	**Modified B**
Processor	SiFive RISC-V (4 Cores)	SiFive RISC-V (4 Cores)
Motherboard	SiFive HiFive Unmatched A00	SiFive HiFive Unmatched A00
Memory	16GB	16GB
Disk	Samsung SSD 980 PRO 500GB + 32GB SD32G	Samsung SSD 980 PRO 500GB + 32GB SD32G
Chipset	SiFive FU740-C000 RISC-V SoC	SiFive FU740-C000 RISC-V SoC
OS	Ubuntu 21.10	Ubuntu 21.10
Kernel	5.13.0-1004-generic (riscv64)	5.13.0-1004-generic (riscv64)
Bootloader	U-Boot SPL 2021.07+dfsg-0ubuntu8	U-Boot SPL 2022.01-rc4+keystonesm
Compiler	GCC 11.2.0	GCC 11.2.0
File-System	ext4	ext4

4.8.5 Case Study C: Ubuntu 22.04 - Jammy Jellyfish

Case Study C assesses performance for the CUT running Ubuntu 22.04 LTS, Jammy Jellyfish (Development Branch). Ubuntu 22.04 is currently under active development and is categorized under the *Development Branch* moniker. Accordingly, Canonical has not yet elevated the release to *beta* status. Jammy Jellyfish represents the only distribution OS tested that has five years of promised hardware and maintenance upgrades as well as ten years of promised Extended Security Maintenance (ESM) [23]. From the developers' perspective, Ubuntu 22.04 LTS makes the most sense as a foundation for expanding TEE firmware support. Table 7 contrasts the modified Keystone SM configuration against the standard Ubuntu 22.04 preinstalled server image configuration provided by Canonical [19]. Firmware implementation differences are emphasized with blue text.

Table 7. Case Study C Configurations: Ubuntu 22.04 (Jammy Jellyfish)

Configuration:	Standard C	Modified C
Processor	SiFive RISC-V (4 Cores)	SiFive RISC-V (4 Cores)
Motherboard	SiFive HiFive Unmatched A00	SiFive HiFive Unmatched A00
Memory	16GB	16GB
Disk	Samsung SSD 980 PRO 500GB + 32GB SD32G	Samsung SSD 980 PRO 500GB + 32GB SD32G
Chipset	SiFive FU740-C000 RISC-V SoC	SiFive FU740-C000 RISC-V SoC
OS	Ubuntu 22.04	Ubuntu 22.04
Kernel	5.13.0-1007-generic (riscv64)	5.13.0-1007-generic (riscv64)
Bootloader	U-Boot SPL 2021.07+dfsg-0ubuntu10	U-Boot SPL 2022.01-rc4+keystonesm
Compiler	GCC 11.2.0	GCC 11.2.0
File-System	ext4	ext4

4.9 Data Collection & Analysis Tools

To perform experimentation, several free and open-source resources are used; Table 8 captures the software tools and services leveraged to conduct testing.

Table 8. Data Collection and Analysis Tools

Name	Version	Description
OpenBenchmarking.org	-	Cross-Platform, Open-Source, Automated, Centralized Testing Ecosystem
PTS	10.8.0	Open-Source, Automated Benchmarking Platform
PuTTY	0.76	Free SSH and telnet client for Windows
R	4.1.1	The R Project for Statistical Computing
RStudio	1.4.1717	Integrated Development Environment (IDE) for R
Stress-NG	0.13.02	Linux stress tool developed by Colin King of Canonical
Stress-NG for PTS	1.4.0	eXtensible Markup Language (XML) and install.sh setup scripts to configure Stress-NG for PTS

4.9.1 OpenBenchmarking.org

OpenBenchmarking.org is an open-source, cross-platform, centralized testing ecosystem that offers a collaborative, open test platform with standardized testing profiles and suite management system tools for distributing and standardizing benchmarks

[22]. Synthetic benchmark results from this research are made available online at OpenBenchmarking.org and are discussed in Chapter V [24, 25, 26, 27, 28, 29].

4.9.2 Phoronix Test Suite

The PTS is an open-source, automated benchmarking platform offering comprehensive testing capabilities to facilitate effective, reproducible, and automated benchmarking [30]. From the Linux terminal, PTS provides an extensible framework for adding benchmark tests with integrated remote management systems to schedule, install, and archive system test data, results, and installation logs. Benchmarking results are optionally configured to automatically record to OpenBenchmarking.org where they are stored for further analysis, explored in Chapter V.

4.9.3 Stress-NG

The Stress-NG benchmarking suite is an open-source Linux stress tool, developed by Colin King of Canonical. Stress-NG contains over 270 stress tests intended to exercise the various physical subsystems of a computer and various OS kernel interfaces [20]. Selected tests from the upstream Stress-NG Git repository (Version `0.13.02`) have been incorporated into the PTS implementation of Stress-NG (Version `1.4.0`) using XML and bash scripts for configuration. Stress-NG benchmark scores are recorded in Bogo OPs/s, with higher scores indicating better performance. Table 9 itemizes and describes the specific benchmarks used by this experiment.

4.10 Summary

Chapter IV expresses the experimental design and testing methodologies implemented that assess the HiFive Unmatched development platform as a potential hardware candidate for complete Keystone Enclave integration. Three case studies are

Table 9. **Compatible Benchmarks from Stress-NG 1.4.0**

#	Benchmark	Description
1	MMAP	Memory Map
2	NUMA	Non-Uniform Memory Access
3	MEMFD	Anonymous Kernel Memory Management
4	Atomic	Atomic Operations
5	Crypto	MD5, SHA-256, SHA-512, scrypt, NT, yescrypt
6	Malloc	Memory Allocation
7	Forking	CPU Forking
8	IO_uring	Asynchronous Input/Output
9	SENDFILE	Read/Write
10	CPU Cache	Cache Thrashing
11	CPU Stress	Integer, Multiply, Floating Point, and Double Precision
12	Semaphores	Shared Resources
13	Matrix Math	Two- and Three-Dimensional Matrix Operations
14	Vector Math	128-bit Vector Operations
15	Memory Copying	memcpy() Method Operation
16	Socket Activity	IPv4, TCP Congestion Control
17	Context Switching	Memory Clobbering
18	Glibc C String Functions	Copying, Concatenating, Comparing, & Searching Strings
19	Glibc Qsort Functions	Quick Sort Method
20	System V Message Passing	Kernel Level System Calls

explored, each with two distinct experiments, to evaluate synthetic benchmark performance and distribution kernel boot times for the Unmatched system running various Ubuntu OS releases both with and without Keystone SM firmware modifications. From these experiments, a substantive performance characterization is formed in Chapter V in support of future TEE development and the permeation of *Confidential Computing* principles.

V. Observations & Analysis

5.1 Overview

Chapter V evaluates the results produced by experimentation conducted in Chapter IV and provides informative performance characterizations of the HiFive Unmatched system running the Ubuntu OS, both with and without bootloader support for Keystone SM firmware. Observations, results, and analysis are organized according to the particular case study under investigation and present findings for each of the two conducted experiments. This chapter also addresses research difficulties associated with directly comparing results across disparate case studies. Research discoveries presented in this chapter aid future work, discussed in Chapter VI, by proposing recommendations for continued Keystone SM integration onto new hardware and for the broader promotion of *Confidential Computing* practices by all stakeholders.

5.2 Performance Characterizations

Performance characterizations for the Unmatched begin by exploring each of the three case studies proposed in Chapter IV. For each case study, synthetic benchmarking experiments are investigated first, followed by an analysis of distribution kernel boot times. To evaluate the statistical significance of these findings, the One-Way ANOVA Test is performed. The application of the ANOVA test necessitates that the underlying data comprises a normal distribution and that the variances of data are equal. For all statistically significant findings presented in Chapter V, quantile-quantile plots for each benchmark in Experiment I and for reported boot times in Experiment II are shown as paired-observation differences in relation to the theoretical normal distribution line. All normal distribution probability plots and comparative box plots are provided in Appendices A, B, and C for Case Studies A, B, and C, respectively. With the ANOVA assumptions satisfied, the reminder of this chapter discusses the testing hypotheses, significant observations, and comparative analyses for each case study.

5.3 Testing Hypotheses

5.3.1 Experiment I - ANOVA Hypotheses

The hypotheses for the One-Way ANOVA Tests remain consistent across case studies and for all benchmarks evaluated within Experiment I. Recall that Stress-NG benchmark scores are recorded in Bogo OPs/s; higher scores indicate better performance. The null and alternate hypotheses for all synthetic benchmarking tests are provided below:

- H_0^I: There is no significant synthetic benchmarking performance* difference between the Standard and Modified configurations.
- H_1^I: Synthetic benchmarking performance differs significantly between the Standard and Modified configurations.

5.3.2 Experiment II - ANOVA Hypotheses

The Hypotheses for the One-Way ANOVA Tests remain consistent across case studies and for all boot observations within Experiment II. Distribution kernel boot times are measured in seconds; lower scores indicate better performance. The null and alternate hypotheses for distribution kernel boot time tests are as follows:

- H_0^{II}: There is no significant difference in distribution kernel boot times** between the Standard and Modified configurations.
- H_1^{II}: Distribution kernel boot times differ significantly between the Standard and Modified configurations.

5.4 Case Study A: Ubuntu 21.04 - Hirsute Hippo

Described in Section 4.8.3, Case Study A investigates the Standard A configuration compared against the Modified A configuration as detailed in Sections 3.3 and 3.4.

5.4.1 Case Study A, Experiment I: Synthetic Benchmarking Performance

Upon initial observation, the mean Stress-NG Benchmark Scores for Ubuntu 21.04 (found in Appendix A, Table 16) appear to indicate a difference in performance. With the exceptions of NUMA, MEMFD, Crypto, and Malloc, the other 16 mean benchmark scores for the Modified A configuration outperform those of the Standard A configuration. To narrow the exploration space, the ratio of mean execution scores are taken for each of the 20 benchmarks. The six benchmark tests with the greatest normalized score difference are listed below, with red text indicating higher mean scores for the Modified A configuration, and black text representing the Standard A configuration:

1. Benchmark #4 - Atomic: Modified A +44.40%
2. Benchmark #17 - Context Switching: Modified A +29.67%
3. Benchmark #6 - Malloc: Standard A +21.03%
4. Benchmark #14 - Vector Math: Modified +19.63%
5. Benchmark #18 - Glibc C String Functions: Modified +19.51%
6. Benchmark #13 - Matrix Math: Modified +19.28%

Figure 12 contains box plots for the six previously identified tests with the greatest normalized score differences. To characterize performance for the Stress-NG Benchmark Suite as a whole, Figure 13 showcases the box plots of the geometric means for each test replication. The Modified A configuration is contrasted against the Standard A configuration through the use of red plots.

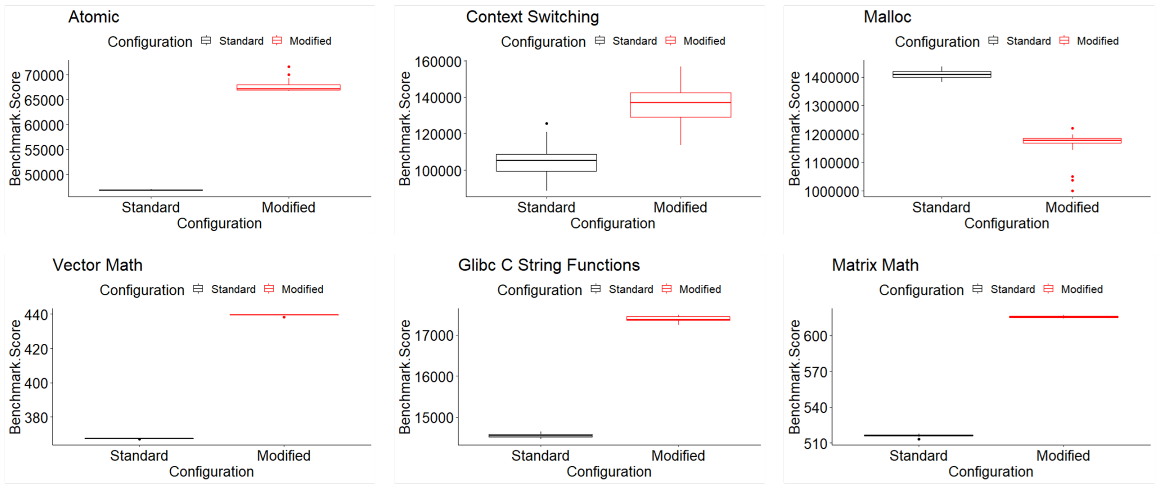


Figure 12. Case Study A, Experiment I: Stress-NG, Selected Box Plots

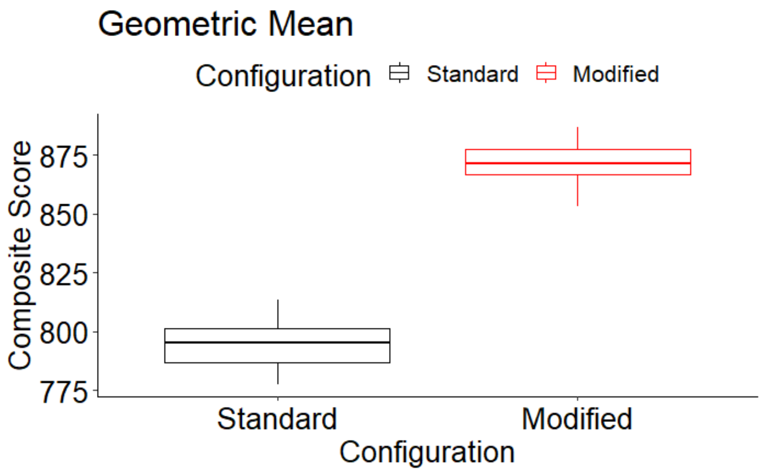


Figure 13. Case Study A, Experiment I: Stress-NG Geometric Mean Box Plot

To holistically characterize the overall synthetic benchmarking performance for Case Study A, the geometric mean of all benchmark scores are taken to determine if the difference between the Standard A and Modified A configurations are statistically significant. To evaluate the hypothesis for Experiment I, the ANOVA test is performed for the six previously identified benchmarks and for the geometric mean composite score. Table 10 summarizes the six test results and the geometric mean composite score for all benchmarks. If the null hypothesis H_0^I is rejected, red text signifies better performance for the Modified A configuration, while black text indicates better performance for the Standard A configuration. If the null hypothesis H_0^I fails to reject, there is no meaning ascribed to the text color. For Case Study A, Experiment I, the null hypothesis H_0^I is rejected, meaning that there is a statistically significant performance difference ($p < 0.01$) between the two configurations. The Modified A configuration –with its updated platform-specific, Keystone SM enabled, OpenSBI reference implementation– outperforms the Standard A configuration.

Table 10. Case Study A, Experiment I: Selected & Composite ANOVA Results

#	Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
4	Atomic	10069	$2.2e^{-16}$	Reject H_0^I
17	Context Switching	145.9	$2.2e^{-16}$	Reject H_0^I
6	Malloc	708.37	$2.2e^{-16}$	Reject H_0^I
14	Vector Math	2672950	$2.2e^{-16}$	Reject H_0^I
18	Glibc C String Functions	43454	$2.2e^{-16}$	Reject H_0^I
13	Matrix Math	205317	$2.2e^{-16}$	Reject H_0^I
-	Geometric Mean	962.43	$2.2e^{-16}$	Reject H_0^I

5.4.2 Case Study A, Experiment II: Distribution Kernel Boot Time

Again, the distribution kernel boot time box plots (shown in Figure 14) appear to indicate a difference in distribution kernel boot times for Case Study A. To confirm the visual suspicions, the ANOVA test is performed as shown in Table 11.

Table 11. Case Study A, Experiment II: Distribution Kernel Boot Time ANOVA Results

Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
Distribution Kernel Boot Time	4096.7	$2.2e^{-16}$	Reject H_0^{II}

For Case Study A, Experiment II, the null hypothesis H_0^{II} is rejected; the distri-

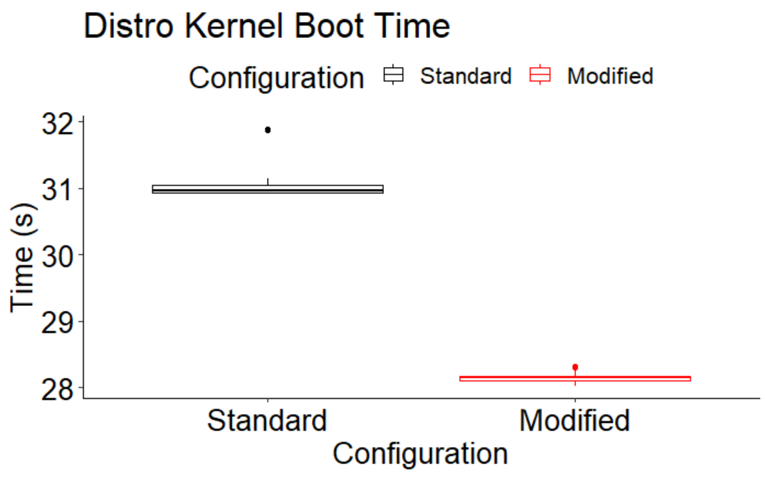


Figure 14. Case Study A, Experiment II: Distro Kernel Boot Time Box Plot (Lower is Better)

bution kernel boot time for the Modified A configuration is significantly faster than the boot time for the Standard A configuration ($p < 0.01$). By applying the geometric mean across the 30 test replicates, the Modified A configuration is shown to reduce distribution kernel boot times by 2.88 seconds per boot –a 9.28% improvement over the Standard A configuration.

5.5 Case Study B: Ubuntu 21.10 - Impish Indri

Defined in Section 4.8.4, Case Study B examines the Modified B configuration as it compares to the Standard B configuration detailed in Sections 3.3 and 3.4.

5.5.1 Case Study B, Experiment I: Synthetic Benchmarking Performance

In contrast to Case Study A, the mean Stress-NG Benchmark Scores for Ubuntu 21.10 (found in Appendix B, Table 18) do not appear to indicate a difference in performance. Again, the exploration space is reduced by taking the ratio of mean execution scores for each of the 20 benchmarks. Noticeably, these normalized score differences appear much smaller than those found in Case Study A. The six benchmark tests with the greatest normalized score difference are listed below, with green text indicating higher mean scores for the Modified B configuration, and black text representing the Standard B configuration:

1. Benchmark #8 - IO_uring: Standard B +3.28%
2. Benchmark #9 - SENDFILE: Modified B +1.90%
3. Benchmark #10 - CPU Cache: Modified B +1.39%
4. Benchmark #17 - Context Switching: Standard B +1.26%
5. Benchmark #20 - System V Message Passing: Standard B +1.09%
6. Benchmark #3 - MEMFD: Standard B +0.68%

Figure 15 contains box plots for the six previously identified tests with the greatest normalized score differences. To capture performance across the entire Stress-NG benchmarking suite, Figure 16 showcases the box plot of the geometric means for each test replication. The Modified B configuration is differentiated from the Standard B configuration through the use of green plots.

To sufficiently characterize the performance of Case Study B for the Stress-NG benchmarking suite, the geometric mean of all benchmark scores are taken to determine if the difference between the Standard B and Modified B configurations are statistically significant. To evaluate the hypothesis for Experiment I, the ANOVA test is performed for the six previously identified benchmarks and for the geometric mean composite score. Table 12 summarizes these seven test results. If the null hypothesis H_0^I is rejected, green text signifies better performance for the Modified B configuration, while black text indicates better performance for the Standard B configuration. If the null hypothesis H_0^I fails to reject, there is no meaning assigned

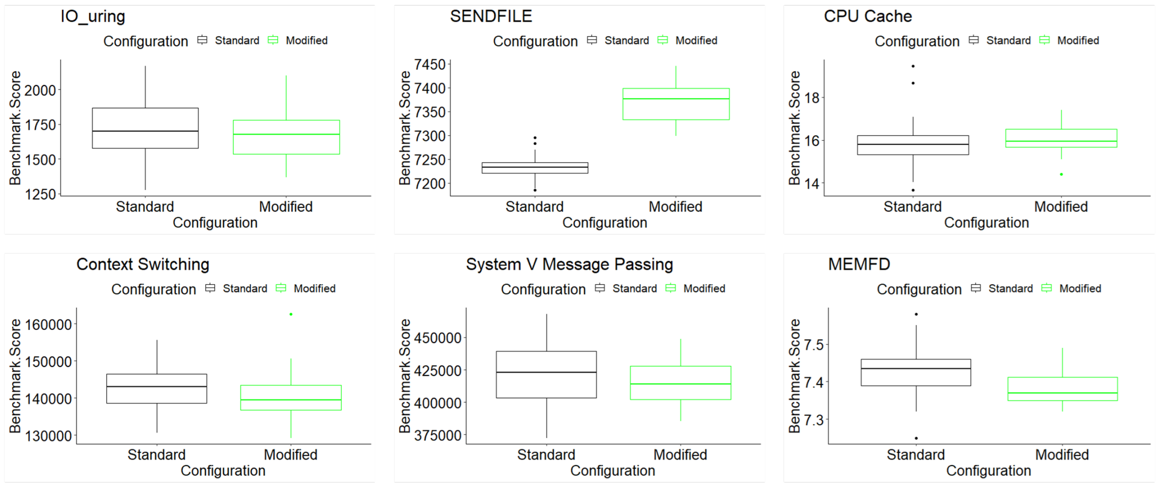


Figure 15. Case Study B, Experiment I: Stress-NG, Selected Box Plots

to the text color. For Case Study B, Experiment I, the null hypothesis H_0^I fails to reject; therefore, there is not a statistically significant performance difference ($p < 0.01$) between the Standard B and Modified B configurations. When comparing synthetic benchmark performance between U-Boot SPL revision `2021.07+dfsg-0ubuntu8` and revision `2022.01-rc4+keystonesm`, it is concluded that the addition of Keystone SM firmware modifications do not passively impact performance when compared to the Standard B configuration. In other words, adding the Keystone SM firmware has little to no impact on the Stress-NG Benchmark scores.

Table 12. Case Study B, Experiment I: Selected & Composite ANOVA Results

#	Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
8	IO_uring	1.1591	0.2861	Fail to Reject H_0^I
9	SENDFILE	238.63	$2.2e^{-16}$	Reject H_0^I
10	CPU Cache	0.7686	0.3843	Fail to Reject H_0^I
17	Context Switching	1.1076	0.297	Fail to Reject H_0^I
20	System V Message Passing	0.7684	0.3843	Fail to Reject H_0^I
3	MEMFD	9.7126	0.002845	Reject H_0^I
-	Geometric Mean	0.7257	0.3978	Fail to Reject H_0^I

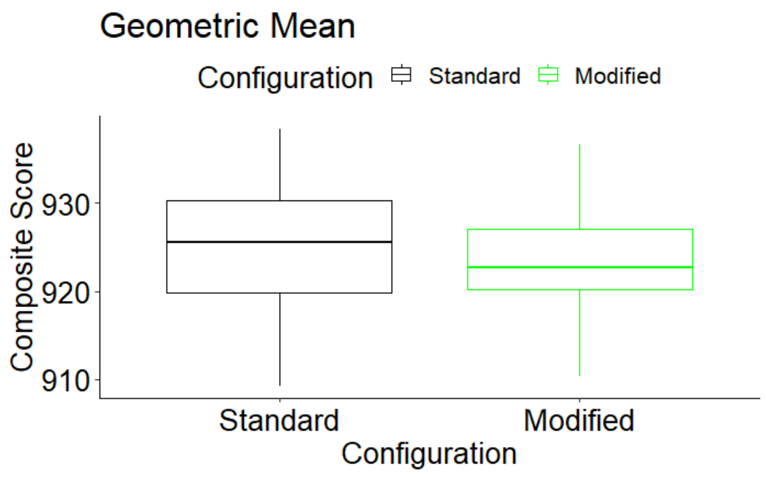


Figure 16. Case Study B, Experiment I: Stress-NG Geometric Mean Box Plot

5.5.2 Case Study B, Experiment II: Distribution Kernel Boot Time

Upon review of the distribution kernel boot time box plots (shown in Figure 17), the mean boot times suggest that there may be a significant difference in distribution kernel boot times for Case Study B. Further investigation is warranted; to prove significance, the ANOVA test is performed as shown in Table 13.

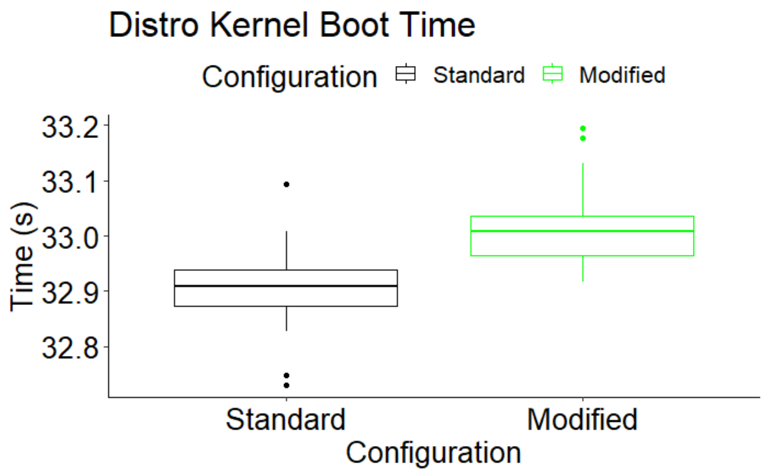


Figure 17. Case Study B, Experiment II: Distro Kernel Boot Time Box Plot (Lower is Better)

Table 13. Case Study B, Experiment II: Distribution Kernel Boot Time ANOVA Results

Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
Distribution Kernel Boot Time	38.224	$6.803e^{-8}$	Reject H_0^{II}

For Case Study B, Experiment II, the null hypothesis H_0^{II} is rejected; the distribution kernel boot time for the Modified B configuration significantly differs from the Standard B configuration ($p < 0.01$). In contrast to Experiment I, Experiment II concludes that the addition of Keystone SM firmware passively affect distribution kernel boot times significantly. The Modified B configuration is shown to increase distribution kernel boot times by 0.11 seconds per boot –a 1.00% reduction compared to the Standard B configuration.

5.6 Case Study C: Ubuntu 22.04 - Jammy Jellyfish

Specified in Section 4.8.5, Case Study C explores performance differences between the Standard C configuration and the Modified C configuration detailed in Sections 3.3 and 3.4.

5.6.1 Case Study C, Experiment I: Synthetic Benchmarking Performance

Similar to Case Study B, the mean Stress-NG Benchmark Scores for Ubuntu 22.04 (found in Appendix C, Table 20) do not appear to indicate a performance difference between testing configurations. Analysis begins with a reduction to the exploration space by taking the ratio of mean execution scores for all 20 benchmarks. The six benchmarks with the greatest normalized score difference are summarized below, with blue text representing higher mean scores for the Modified C configuration and black text for the Standard C configuration:

1. Benchmark #13 - Matrix Math: Standard C +2.02%

2. Benchmark #7 - Forking: Standard C +1.35%
3. Benchmark #10 - CPU Cache: Modified C +1.23%
4. Benchmark #3 - MEMFD: Standard C +0.82%
5. Benchmark #1 - MMAP: Modified C +0.65%
6. Benchmark #8 - IO_uring: Modified C +0.64%

Figure 18 contains box plots for the six selected benchmarks with the greatest normalized score differences. To express performance across the entire Stress-NG benchmarking suite, Figure 19 compares the box plots of the geometric means for each test replication. The Modified C configuration are represented with blue plots.

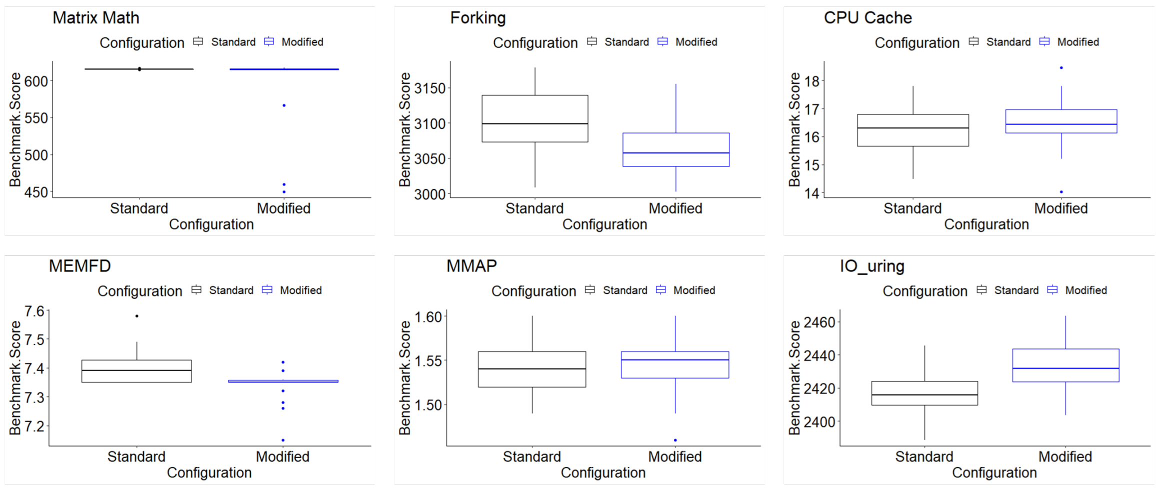


Figure 18. Case Study C, Experiment I: Stress-NG, Selected Box Plots

For Case Study C, performance characterizations are achieved by evaluating compatible tests within the Stress-NG benchmarking suite. To assess the passive impact Keystone SM may have on performance, the geometric mean of all benchmark scores are taken to determine if differences between the Standard C and Modified C configurations are statistically significant. To evaluate the hypothesis for Experiment I,

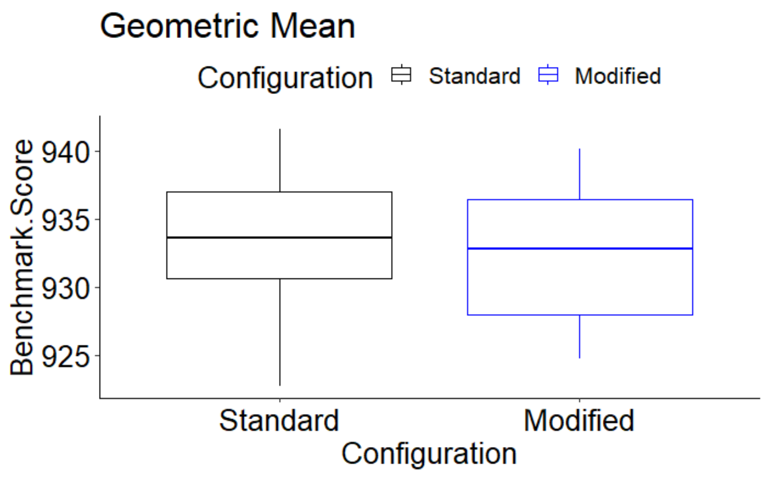


Figure 19. Case Study C, Experiment I: Stress-NG Geometric Mean Box Plot

the ANOVA test is performed for the six previously identified benchmarks and for the geometric mean composite score. Table 14 summarizes these seven test results. If the null hypothesis H_0^I is rejected, blue text signifies better performance for the Modified C configuration, while black text indicates better performance for the Standard C configuration. If the null hypothesis H_0^I fails to reject, there is no meaning attributed to the text color. For Case Study C, Experiment I, the null hypothesis H_0^I fails to reject. As found for Case Study B, there is not a statistically significant performance difference ($p < 0.01$) between the Standard C and Modified C configurations. When comparing synthetic benchmark performance between U-Boot SPL revision `2021.07+dfsg-0ubuntu10` and revision `2022.01-rc4+keystonesm`, it is concluded that the introduction of Keystone SM firmware modifications do not passively impact performance when compared against the Standard C configuration.

Table 14. **Case Study C, Experiment I: Selected & Composite ANOVA Results**

#	Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
13	Matrix Math	2.6027	0.1121	Fail to Reject H_0^I
7	Forking	14.996	0.0002761	Reject H_0^I
10	CPU Cache	0.8007	0.3746	Fail to Reject H_0^I
3	MEMFD	24.079	$7.859e^{-6}$	Reject H_0^I
1	MMAP	0.1154	0.7353	Fail to Reject H_0^I
8	IO_uring	20.171	$3.426e^{-5}$	Reject H_0^I
-	Geometric Mean	0.7707	0.3836	Fail to Reject H_0^I

5.6.2 Case Study C, Experiment II: Distribution Kernel Boot Time

After examining the distribution kernel boot time box plots (shown in Figure 20), there are no immediate indicators that suggest a significant difference in distribution kernel boot times. Notably, three outliers are discovered in the Standard C configuration; however, their scale is dwarfed by the sole outlier for the Modified C configuration. Nevertheless, without these outliers, the box plot boxes remain noticeably flat. Upon investigation into the serial console log files, the outlying boot time for the Modified C configuration occurs in test Replicate Ten, where the `random: crng init done` step is added to the kernel start-up procedure. This additional kernel task is automatically scheduled based upon myriad factors, and configures the kernel's random number generator [31]. To convey differences, Figure 21 displays Replicate Nine from the Modified C configuration on the top, contrasted against Replicate Ten shown on the bottom. Replicate Ten for the Modified C configuration is the only test that experiences this system-wide, random number generation anomaly. To prove significance, the ANOVA test is performed as shown in Table 15.

Table 15. **Case Study C, Experiment II: Distribution Kernel Boot Time ANOVA Results**

Benchmark	F-Value	P-Value	Significance ($P < 0.01$)
Distribution Kernel Boot Time	0.3307	0.5674	Fail to Reject H_0^{II}

For Case Study C, Experiment II, the null hypothesis H_0^{II} fails to reject; the

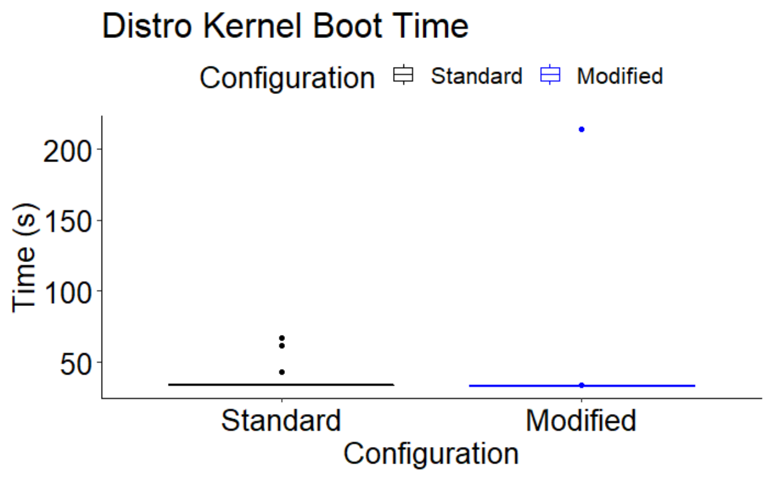
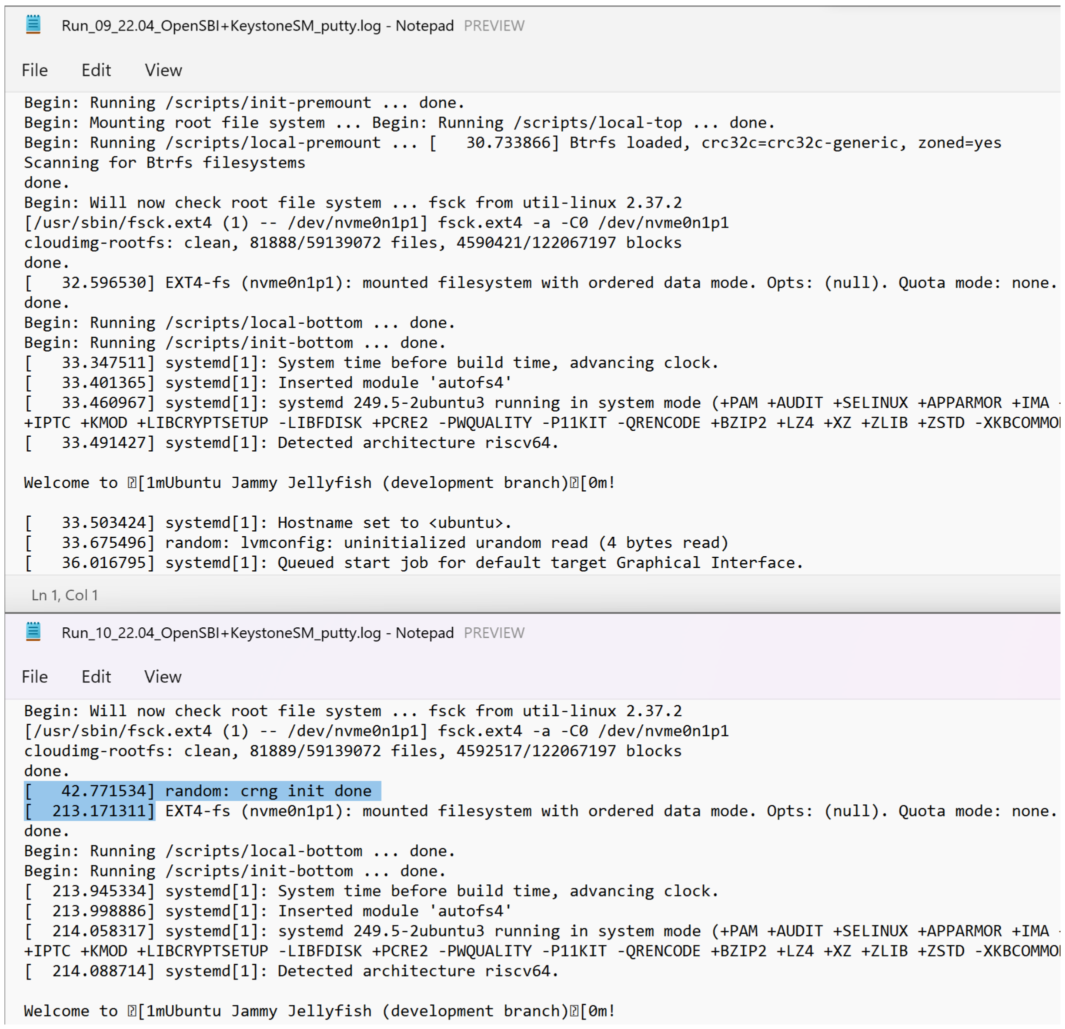


Figure 20. Case Study C, Experiment II: Distro Kernel Boot Time Box Plot (Lower is Better)

distribution kernel boot time for the Modified C configuration does not significantly differ than that of the Standard C configuration ($p < 0.01$). Finally, Experiment II, concludes that the addition of Keystone SM firmware does not significantly affect distribution kernel boot times.

5.7 Comparing Data Across Case Studies

The temptation to directly compare performance results across disparate case studies is not easily satiated. Without experimental design constraints imposed to limit the interactions between independent variables, –and perhaps most importantly, the inherent constraint imposed by firmware incompatibilities between each OS release– evaluating relative performance across case studies risks overlooking the delicate interactions buried within low-level firmware. OpenSBI, for example, is not listed as an independent variable in Table 3. Nevertheless, the particular OpenSBI implementations used by this work rely upon specific U-Boot SPL revisions, which may or may not be suitable for integrating Keystone SM into the OpenSBI boot-flow layer. Understandably, previous research by [5] premises analysis on the assumption



```
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Running /scripts/local-premount ... [   30.733866] Btrfs loaded, crc32c=crc32c-generic, zoned=yes
Scanning for Btrfs filesystems
done.
Begin: Will now check root file system ... fsck from util-linux 2.37.2
[/usr/sbin/fsck.ext4 (1) -- /dev/nvme0n1p1] fsck.ext4 -a -C0 /dev/nvme0n1p1
cloudimg-rootfs: clean, 81888/59139072 files, 4590421/122067197 blocks
done.
[   32.596530] EXT4-fs (nvme0n1p1): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
[   33.347511] systemd[1]: System time before build time, advancing clock.
[   33.401365] systemd[1]: Inserted module 'autofs4'
[   33.460967] systemd[1]: systemd 249.5-2ubuntu3 running in system mode (+PAM +AUDIT +SELINUX +APPARMOR +IMA -
+IPTC +KMOD +LIBCRYPTSETUP -LIBFDISK +PCRE2 -PWQUALITY -P11KIT -QRENCODE +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -XKBCOMMOI
[   33.491427] systemd[1]: Detected architecture riscv64.

Welcome to [1mUbuntu Jammy Jellyfish (development branch)[0m!

[   33.503424] systemd[1]: Hostname set to <ubuntu>.
[   33.675496] random: lvmconfig: uninitialized urandom read (4 bytes read)
[   36.016795] systemd[1]: Queued start job for default target Graphical Interface.
```

```
Begin: Will now check root file system ... fsck from util-linux 2.37.2
[/usr/sbin/fsck.ext4 (1) -- /dev/nvme0n1p1] fsck.ext4 -a -C0 /dev/nvme0n1p1
cloudimg-rootfs: clean, 81889/59139072 files, 4592517/122067197 blocks
done.
[   42.771534] random: crng init done
[  213.171311] EXT4-fs (nvme0n1p1): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
[  213.945334] systemd[1]: System time before build time, advancing clock.
[  213.998886] systemd[1]: Inserted module 'autofs4'
[  214.058317] systemd[1]: systemd 249.5-2ubuntu3 running in system mode (+PAM +AUDIT +SELINUX +APPARMOR +IMA -
+IPTC +KMOD +LIBCRYPTSETUP -LIBFDISK +PCRE2 -PWQUALITY -P11KIT -QRENCODE +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -XKBCOMMOI
[  214.088714] systemd[1]: Detected architecture riscv64.

Welcome to [1mUbuntu Jammy Jellyfish (development branch)[0m!
```

Figure 21. Case Study C, Experiment I, Modified C, Replicate Ten: Outlier Anomaly

that the passive impact to synthetic benchmarking performance caused by the introduction of Keystone SM is ineffectual. This work has corroborated that underlying assumption, but only so long as the particular firmware implementation used remains otherwise unchanged. In practice, supplementing unsupported hardware with TEEs necessitates low-level bootloader modifications, often requiring different or modified versions of firmware for each boot-flow layer –many of which are not readily compatible. This challenge is not easily overcome by the individual developer; yet could be overcome by Linux distribution publishers directly.

5.8 Summary

The observations, results, and analysis showcased in Chapter V substantiate assumptions made by prior research [1, 5]. Although Case Studies B and C did not find a statistically significant difference between their respective synthetic benchmarking performance characterizations, Case Study B, Experiment II exposes the potential for performance consequences with respect to distribution kernel boot times. It is speculated that the relatively minimal difference in development time between their U-boot SPL revisions is responsible for similar performance between configurations in Case Study C; however, as the development time between U-Boot SPL revisions increases, the possibility for degraded boot performance is revealed.

Empirically, it is reasonable to craft an experiment that isolates Keystone SM modifications from respective underlying OpenSBI reference implementations to definitively conclude where performance is affected within the boot-flow. Regrettably, such an approach yields little practical relevance. Ultimately, Keystone SM implementation complexities add significant development costs and are time intensive to construct. Subsequently, characterizing the performance of the system with boot-flow modifications and without added Keystone SM function calls does not realistically represent a typical use case –either the system is used without TEE modifications, or it is modified to add TEE security features. Consequently, as new bootloader firmware is developed, Keystone SM will need to be purpose-built for future firmware and hardware compatibility to ensure system performance is maintained. Case Study A demonstrates the significant impact to performance imparted by low-level firmware, beneath the Linux Kernel, and within only one year of development. If *Confidential Computing* principles and practices are to flourish, TEE implementation development must keep pace with the rapidly evolving RISC-V ecosystem.

VI. Conclusion

6.1 Overview

Chapter VI provides a distilled synopsis of this research effort, covering achieved goals, explored hypotheses, and TEE implementation recommendations. Grounded by developmental experience, this work calls upon industry stakeholders to intentionally promote *Confidential Computing* through open collaboration and by investing development resources into the adoption of TEE capabilities as inherent requirements for future Linux Distribution OS builds. This work closes by proposing related future research topics that further the actualization of *Confidential Computing* principles to secure *data in use*.

6.2 Research Contributions

Of the research contributions listed in Section 1.6, each of the following items are successfully demonstrated by this work to the benefit of the Cyber Security, Computer Networking, and Confidential Computing domains:

- Demonstrated device portability claims of an open-source TEE project, by supplementing the HiFive Unmatched development platform with Keystone SM.
- Verified performance ramifications for three distinct Ubuntu Distribution Kernel releases on an ASIC RISC-V system.
- Implemented a platform evaluation framework to assess RISC-V workstation suitability and configurations for TEE augmentation.
- Assessed performance characterizations constructed by examining synthetic benchmarking performance and by measuring distribution kernel boot times.

- Identified the need for Linux distribution publishers to provide RISC-V compatible, Distribution Linux Kernel source code to directly support *Confidential Computing* projects.

- Proposed *Confidential Computing* policy and implementation recommendations –such as having distribution publishers integrate TEE capabilities directly into mainline firmware– for relevant communities, industries, and agencies.

6.3 Future Work

In addition to the research explored by this effort, several related efforts are identified for future investigation. The topics listed below represent ancillary functionality supported by the HiFive Unmatched platform, and enhanced Keystone Enclave components with supporting development tools:

- This research did not attempt to explore the capabilities of the fifth, S7 monitor core, present within the Freedom U740 SoC. Because all five hardware threads (harts) share a common L2 cache, opportunities may exist to directly observe Keystone processor operations within *secure enclaves* to better understand future implementation best practices.

- The PCIe slot on the Unmatched supports various AMD graphics cards. Keystone Enclave impacts to graphical performance remain unexplored.

- Support for flashing Unmatched bootloader firmware directly to on-board ROM via the Quad-SPI interface would improve the underlying memory speeds for Keystone enclaves. Additional performance characterizations could model various implementations to evaluate speedup and security enhancements.

- Keystone Enclave version `1.0.0`, does not fully support context switching and multi-threaded operations. These limitations will inherently hinder system performance. As Keystone itself is developed, continued performance characterizations could examine the performance disparities across multi-threaded versus single-threaded applications.

- Comparative performance analysis from within enclaves remains contingent upon the development of compatible Keystone-Linux driver modules and open-source distribution kernels. Without the source code for RISC-V compatible Ubuntu Distribution Kernels, performance characterizations within enclaves are currently only achievable by switching to an alternate Debian-based Linux distribution. While not officially supported, development for The Linux Kernel anticipates adding platform-specific RISC-V support in upcoming releases.
- With complete Keystone Enclave integration, security mechanisms may be tested and enclave performance for the Unmatched may be more deeply characterized through the development of new Eapps.

6.4 Summary

In contrast to the *Trusted Computing* paradigm, the Confidential Computing Consortium promotes TEEs as a security mechanism for protecting *data in use* through open collaboration, open specifications, and open-source code implementations [4]. While there are significant benefits to this open approach, current offerings are limited in scope and have not yet achieved wide-spread adoption. Keystone Enclave, for example, only officially supports one RISC-V hardware platform (which has subsequently been discontinued). Based upon this work, the portability of Keystone Enclave to new hardware will require broader support from Linux distribution publishers directly. For Keystone Enclave to establish itself as the default TEE implementation for the RISC-V ISA, it must rival its *Trusted Computing* counterparts in popularity and capabilities, while also providing broad platform support. To realize this goal, Keystone Enclave would be best implemented as an inherent attribute within pre-configured bootable disk images –with Eapp development support baked directly into existing IDE tools. Only then could Keystone supplant the existing TPM approach for attributing trust in contemporary RISC-V compute systems. Ultimately, the successful integration of these new computing paradigms directly into the RISC-V ISA and onto new hardware platforms relies upon dedicated open-source contributors in collaboration with distribution publishers to convince stakeholders of the benefits that *Confidential Computing* principles offer.

From the onset of this research, official hardware support by Canonical for the Unmatched platform has been paramount –with OpenEmbedded providing the only other officially supported Linux distribution OS. Unsurprisingly, developmental investments are required to build and incorporate platform-specific firmware and TEE software features into existing Linux distribution releases. Therefore, for Ubuntu distribution publishers to maximize their return on investment, TEE development needs to prioritize Keystone Enclave integration for Ubuntu LTS releases, which support an estimated 95% of Ubuntu users [23].

In addition to concerns regarding Linux distribution support, there are also concerns regarding the long term support of specific hardware devices. The HiFive Unmatched is a hardware product sold by SiFive, which is a relative newcomer to the industry. While initial fervor surrounding their RISC-V products appear promising, SiFive has not publicly committed to any long-term hardware support plans. Launched on 1 February 2018, the previous generation HiFive Unleashed platform has been discontinued and supplanted by the Unmatched platform on 29 October 2020. As a direct consequence of the rapid growth and adoption seen by RISC-V ISA, hardware platform manufacturers have elected to move onto newer, more ambitious projects, rather than prioritize support for older hardware. Unfortunately, the Unmatched platform falls victim to these same business strategy decisions as is reported by Phil Dworsky, the Global Head of Strategic Alliances for SiFive, Inc: "With such great ecosystem adoption, demand has exceeded our already high expectations, and we're close to selling out our production inventory. Given the challenge of supply chain issues that we overcame for the first run of these boards (issues that we continue to face), we've decided to focus on the next generation SiFive HiFive development systems rather than trying to put together another build of the HiFive Unmatched platform in 2022" [32]. As a business, this decision by SiFive appears financially justifiable; however, to champion *Confidential Computing* paradigms, SiFive ought to provide product owners with some level of support assurance for existing platforms in the long-term.

The RISC-V ISA is new; it does not yet rival the market prevalence of the AMD/Intel x86_64 or ARM ISAs. In the midst of this research effort, on 4 December 2021, the privileged ISA specification was officially ratified, with few compatible hardware optimized applications or devices in existence. Nevertheless, experimentation with RISC-V hardware, firmware, and software demonstrates a renewed interest in ISA development. As RISC-V matures and the need for TEEs expand, firmware implementation compatibility will need to be resolved by device manufactures and distribution publishers to provide TEEs as an intrinsic system capability –enforced by device hardware, supported by platform firmware, and configured through readily available software applications. Necessarily, the status quo of proprietary computer architectures with undisclosed security mechanisms will no longer suffice for tomorrow's data security needs. Subsequently, this work aims to promote *Confidential Computing* as an open-source alternative to the *Trusted Computing* paradigm, contributing innovative solutions for securing *data in use*.

Appendix A. Case Study A - Performance Results

All test data for Case Study A, Experiment I is hosted by OpenBenchmarking.org and made publicly available at [24] and at [25].

Table 16. Case Study A, Experiment I: Mean Stress-NG Benchmark Scores

#	Benchmark	Standard A	Modified A
1	MMAP	1.38	1.60
2	NUMA	14.36	13.58
3	MEMFD	6.77	5.87
4	Atomic	46857.45	67660.51
5	Crypto	75.12	66.61
6	Malloc	1409525.71	1164625.24
7	Forking	2323.14	2383.93
8	IO_uring	2362.79	2457.60
9	SENDFILE	5657.93	6155.78
10	CPU Cache	12.93	14.83
11	CPU Stress	169.48	201.75
12	Semaphores	108462.91	114693.97
13	Matrix Math	516.16	615.70
14	Vector Math	367.28	439.39
15	Memory Copying	35.65	39.15
16	Socket Activity	158.04	175.52
17	Context Switching	104485.05	135486.47
18	Glibc C String Functions	14548.96	17387.27
19	Glibc Qsort Functions	4.80	5.67
20	System V Message Passing	268451.33	319548.71

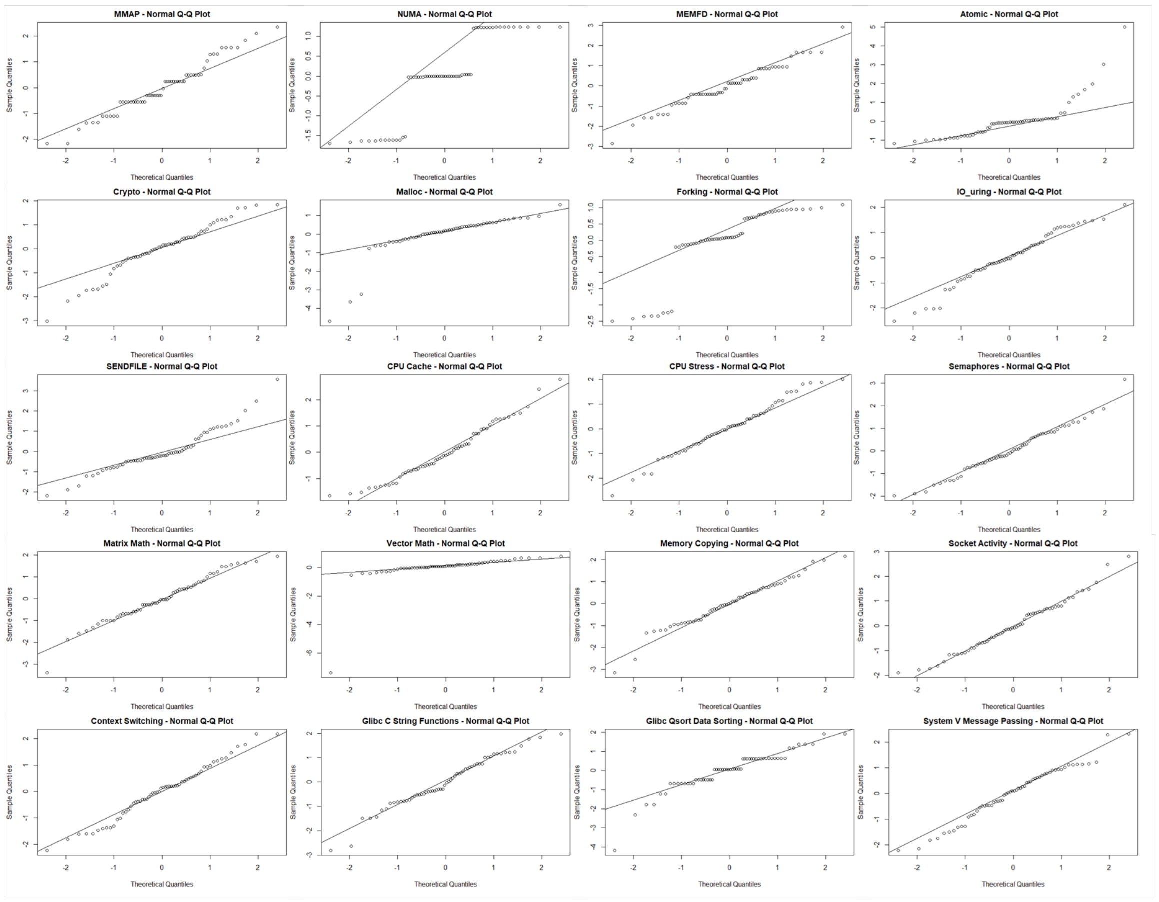


Figure 22. Case Study A, Experiment I: Stress-NG Normality Plots

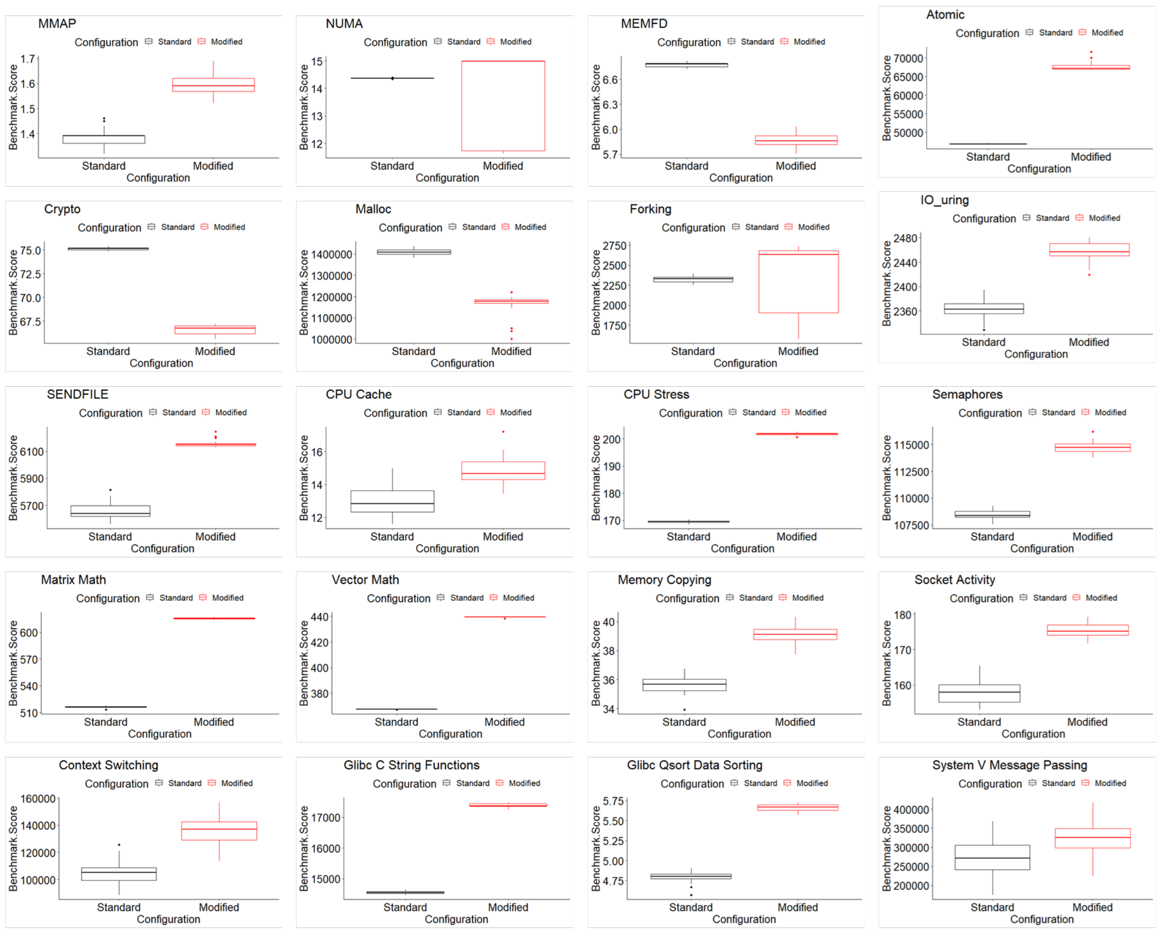


Figure 23. Case Study A, Experiment I: Stress-NG Box Plots

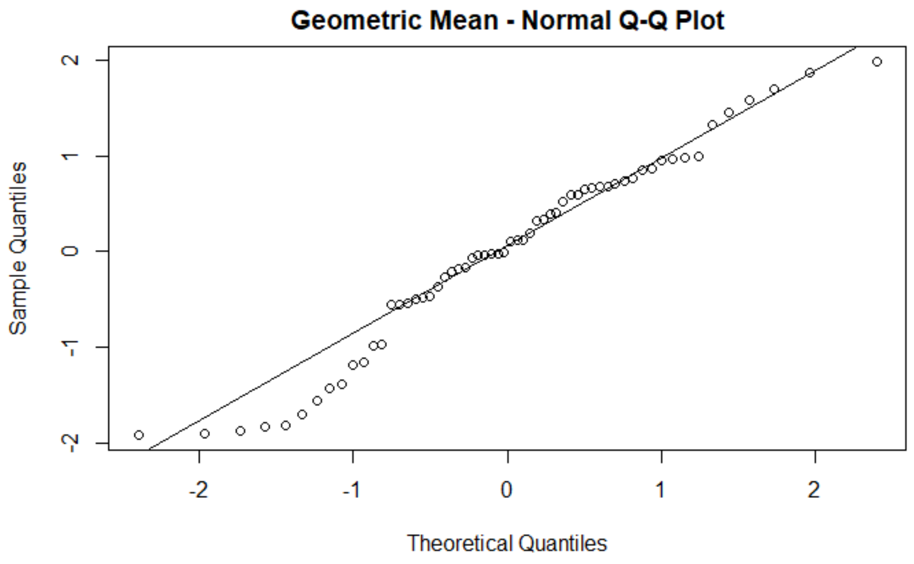


Figure 24. Case Study A, Experiment I: Geometric Mean Normality Plot

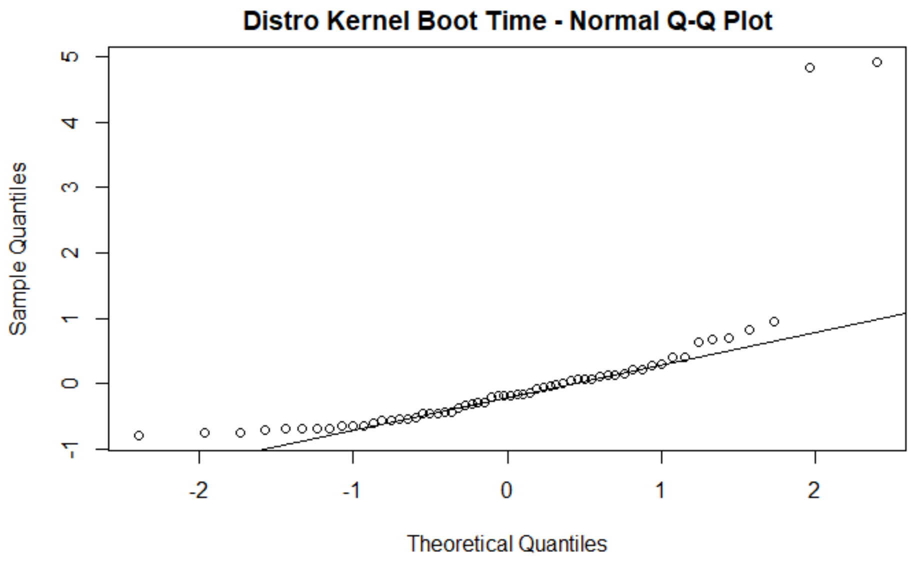


Figure 25. Case Study A, Experiment II: Distro Kernel Boot Time Normality Plot

Table 17. Case Study A, Experiment II: Distro Kernel Boot Times

Trial	Standard A	Modified A
1	30.963945	28.229685
2	30.965997	28.085595
3	30.922420	28.159412
4	31.149824	28.228413
5	30.922848	28.133499
6	31.078958	28.153457
7	30.919152	28.147347
8	30.912270	28.157276
9	31.009178	28.143392
10	30.931176	28.109970
11	31.008353	28.165040
12	30.924798	28.320418
13	30.947571	28.061136
14	30.988063	28.100808
15	31.867709	28.108550
16	31.077997	28.170307
17	31.086902	28.094023
18	30.906151	28.169525
19	31.009128	28.176449
20	30.930963	28.299135
21	30.961724	28.129820
22	31.064020	28.056050
23	30.929987	28.081011
24	30.924806	28.272892
25	31.884534	28.123794
26	30.951201	28.028366
27	30.944342	28.180584
28	31.011210	28.277500
29	31.065896	28.170315
30	30.946964	28.208300

Appendix B. Case Study B - Performance Results

All test data for Case Study B, Experiment I is hosted by OpenBenchmarking.org and made publicly available at [26] and at [27].

Table 18. Case Study B, Experiment I: Mean Stress-NG Benchmark Scores

#	Benchmark	Standard B	Modified B
1	MMAP	1.55	1.54
2	NUMA	12.55	12.57
3	MEMFD	7.43	7.38
4	Atomic	55509.75	55461.9
5	Crypto	90.62	90.6
6	Malloc	1564134.87	1564485.64
7	Forking	3116.55	3102.47
8	IO_uring	1715.76	1661.34
9	SENDFILE	7232.24	7369.45
10	CPU Cache	15.86	16.08
11	CPU Stress	207.69	207.72
12	Semaphores	118148.88	118670.48
13	Matrix Math	616.51	615.54
14	Vector Math	438.9	439.55
15	Memory Copying	40.61	40.56
16	Socket Activity	178.98	178.05
17	Context Switching	142523.18	140756.09
18	Glibc C String Functions	18678.23	18706.07
19	Glibc Qsort Functions	5.65	5.65
20	System V Message Passing	419487.61	414944.87

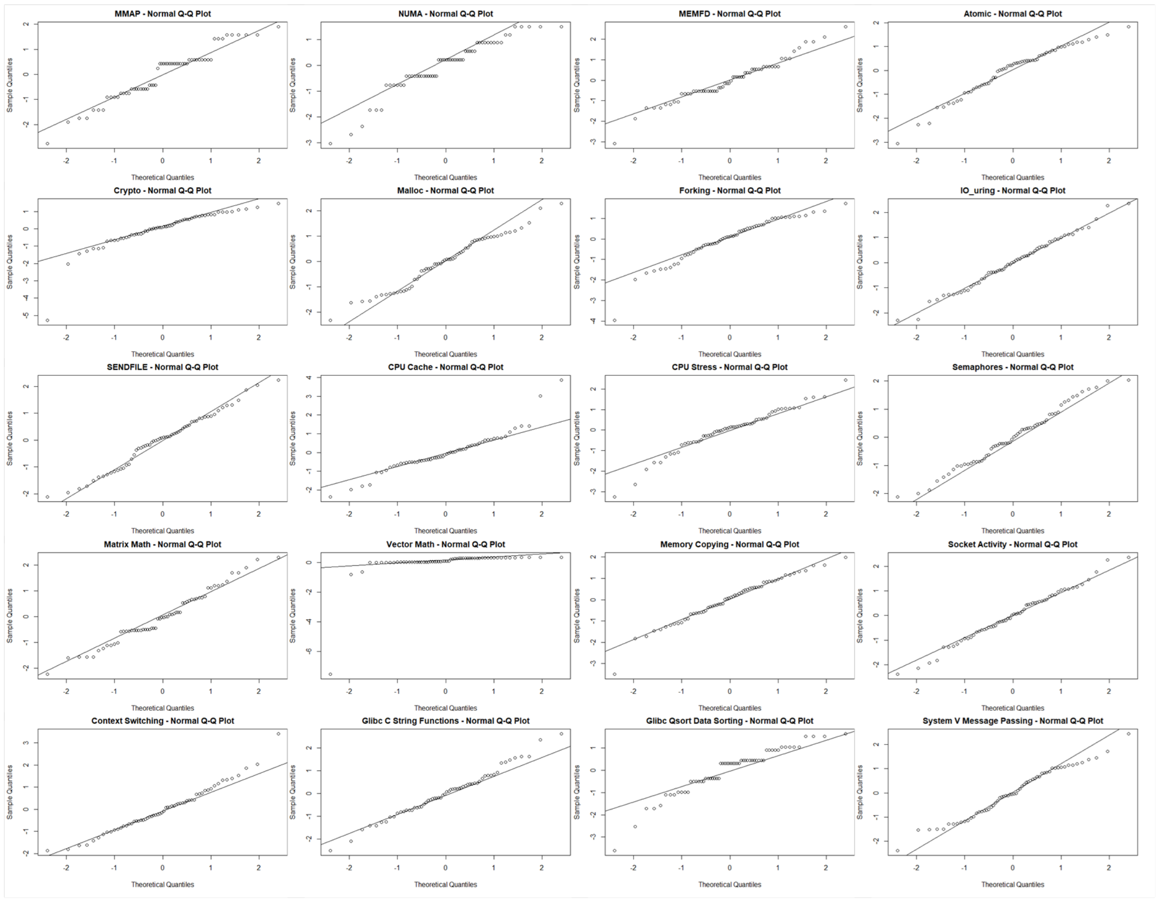


Figure 26. Case Study B, Experiment I: Stress-NG Normality Plots

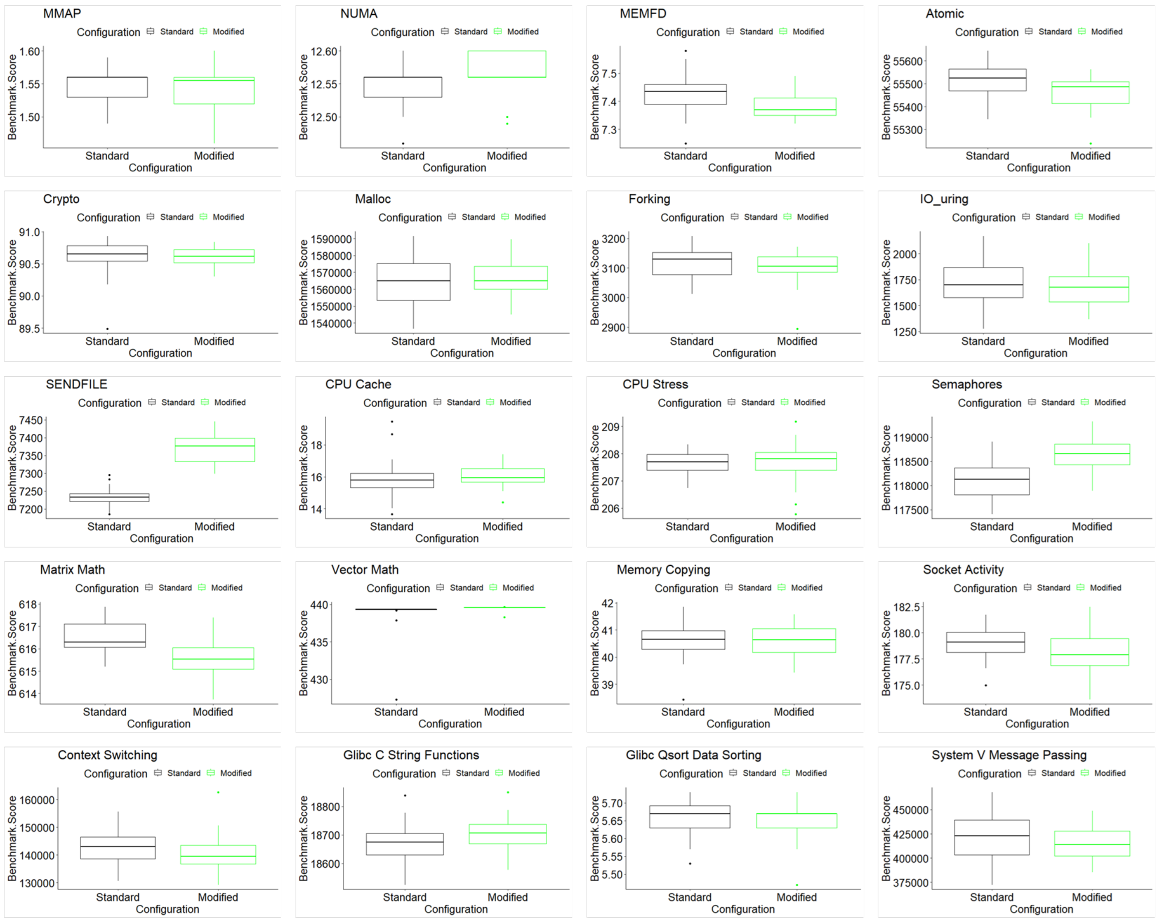


Figure 27. Case Study B, Experiment I: Stress-NG Box Plots

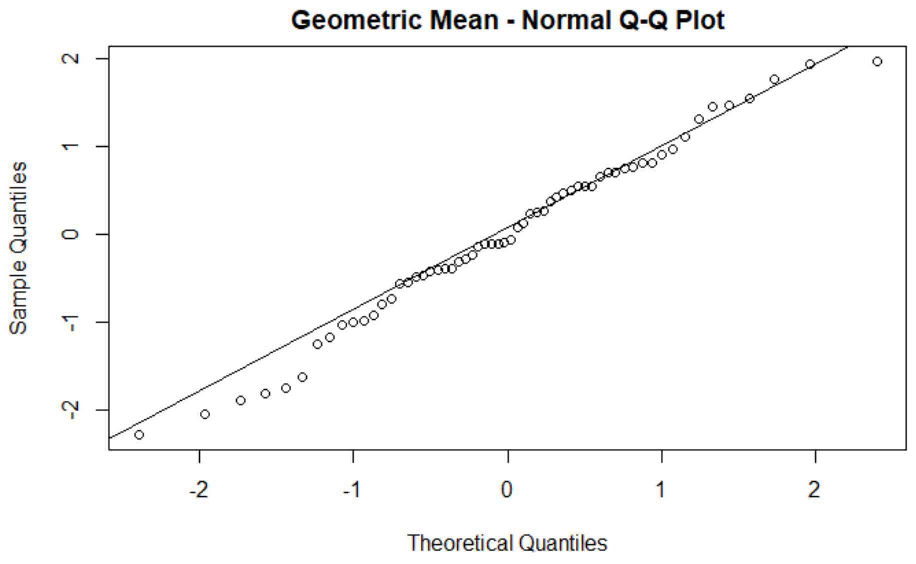


Figure 28. Case Study B, Experiment I: Geometric Mean Normality Plot

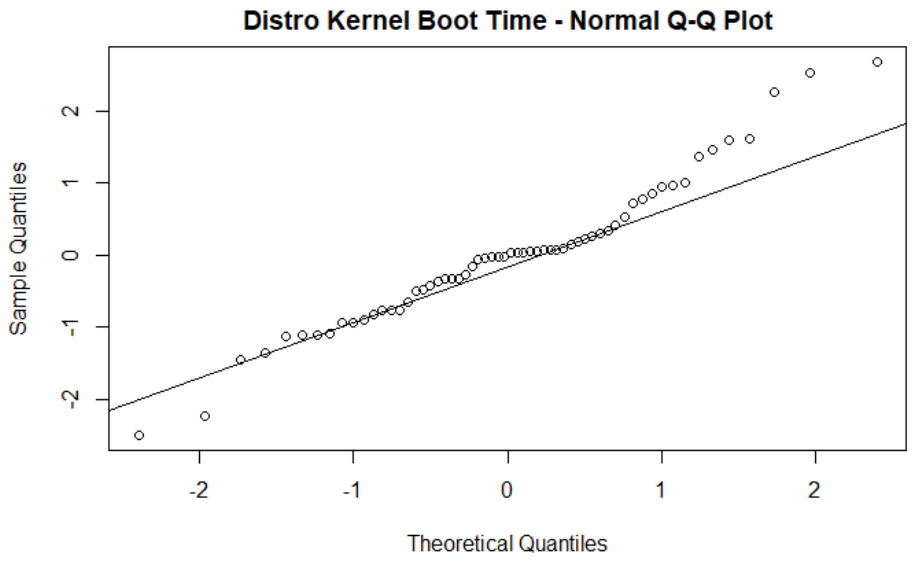


Figure 29. Case Study B, Experiment II: Distro Kernel Boot Time Normality Plot

Table 19. **Case Study B, Experiment II: Distro Kernel Boot Times**

Trial	Standard B	Modified B
1	32.972553	33.113763
2	32.893361	33.131111
3	32.928561	33.000038
4	32.902336	32.993266
5	32.828423	32.916931
6	32.900896	33.020891
7	32.914856	32.983640
8	32.903442	33.072295
9	32.922985	32.995531
10	32.748880	32.924180
11	32.954572	32.952664
12	32.827139	32.952531
13	32.827101	33.023415
14	33.092525	32.955567
15	32.871654	33.023595
16	32.975677	33.017077
17	32.859157	33.175979
18	32.963886	32.996045
19	33.007617	33.194733
20	32.971142	33.021102
21	32.909000	32.939591
22	32.920174	33.016104
23	32.875623	32.965158
24	32.910672	33.038942
25	32.907581	33.129157
26	32.941582	33.046832
27	32.730712	32.995161
28	32.917820	33.023413
29	32.851740	32.964779
30	32.908949	32.961315

Appendix C. Case Study C - Performance Results

All test data for Case Study C, Experiment I is hosted by OpenBenchmarking.org and made publicly available at [28] and at [29].

Table 20. Case Study C, Experiment I: Mean Stress-NG Benchmark Scores

#	Benchmark	Standard C	Modified C
1	MMAP	1.54	1.55
2	NUMA	12.55	12.57
3	MEMFD	7.4	7.34
4	Atomic	55481.43	55442.68
5	Crypto	91.11	91.23
6	Malloc	1560492.94	1561792.31
7	Forking	3102.26	3060.98
8	IO_uring	2417.09	2432.63
9	SENDFILE	7060.47	7052.38
10	CPU Cache	16.29	16.49
11	CPU Stress	207.65	207.52
12	Semaphores	118998.2	119280.52
13	Matrix Math	615.37	603.19
14	Vector Math	439.44	439.4
15	Memory Copying	40.44	40.41
16	Socket Activity	177.26	178.09
17	Context Switching	140637.39	140386.74
18	Glibc C String Functions	18696.99	18656.71
19	Glibc Qsort Functions	5.67	5.67
20	System V Message Passing	365319.82	364639.1

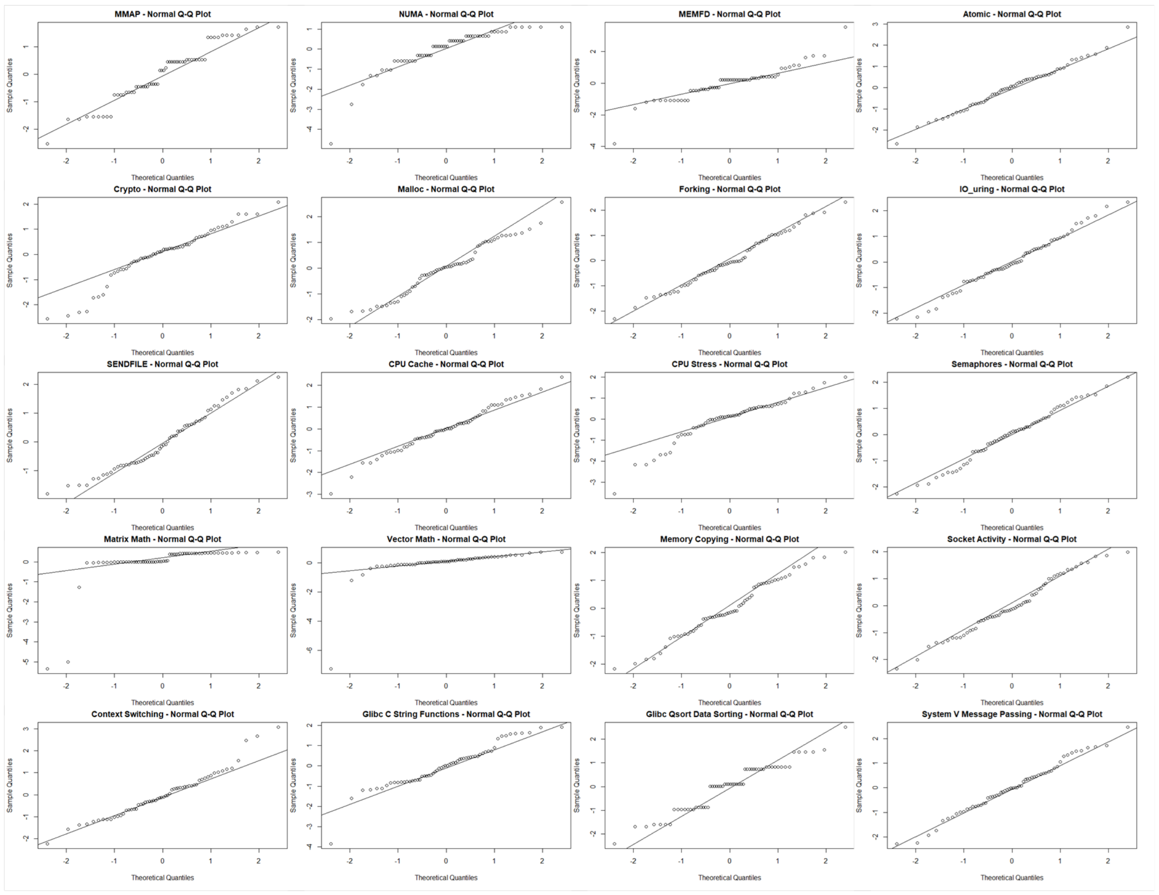


Figure 30. Case Study C, Experiment I: Stress-NG Normality Plots

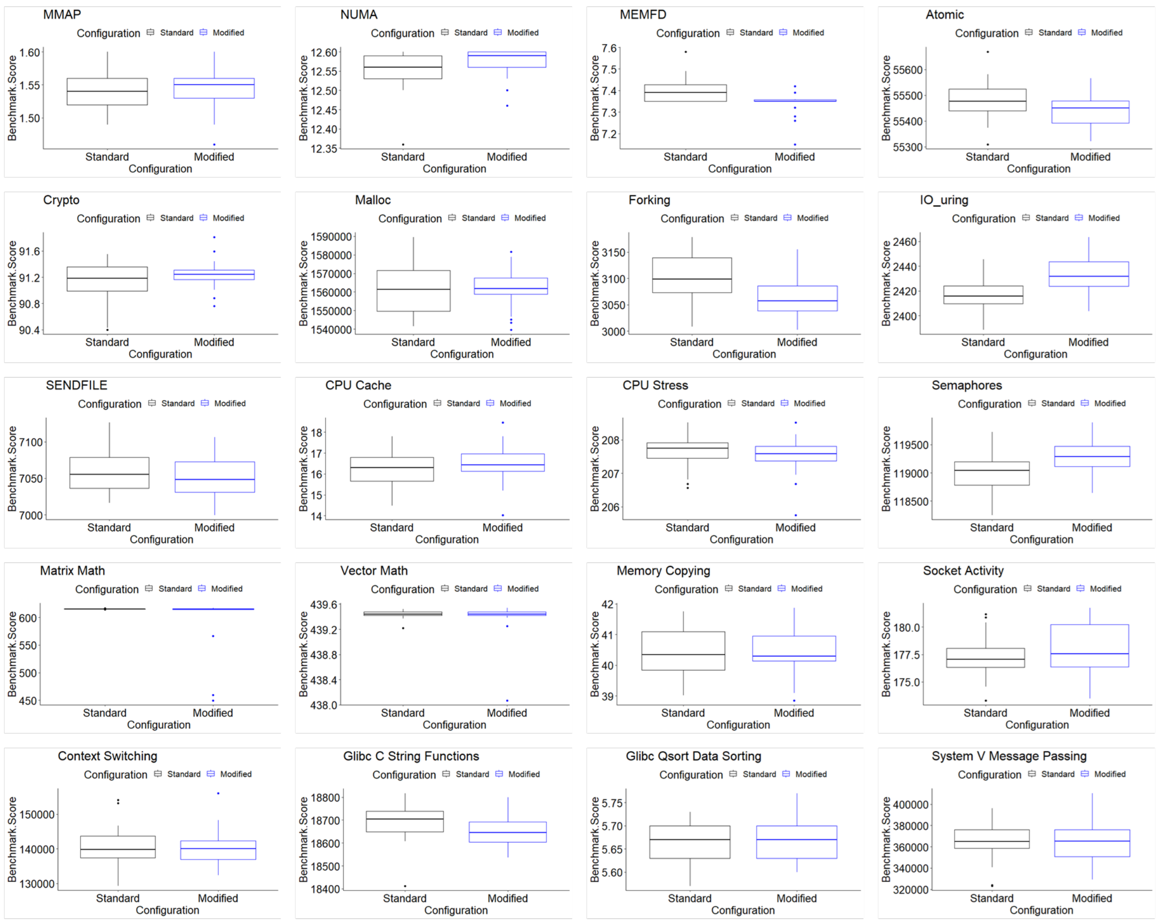


Figure 31. Case Study C, Experiment I: Stress-NG Box Plots

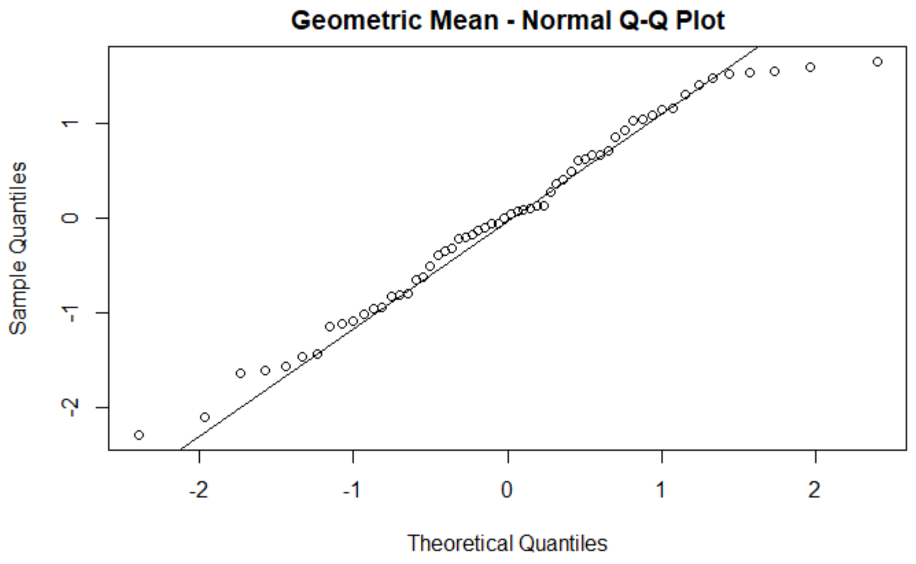


Figure 32. Case Study C, Experiment I: Geometric Mean Normality Plot

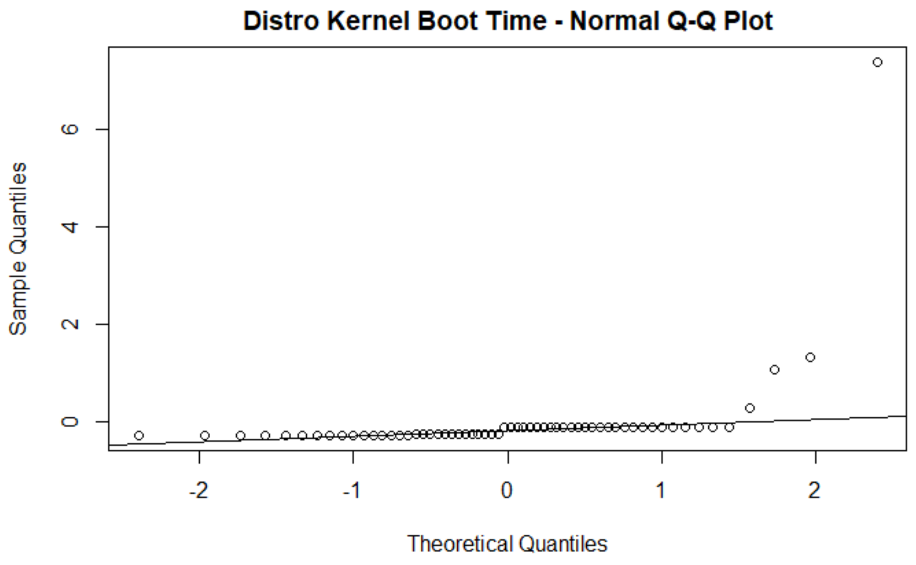


Figure 33. Case Study C, Experiment II: Distro Kernel Boot Time Normality Plot

Table 21. Case Study C, Experiment II: Distro Kernel Boot Times

Trial	Standard C	Modified C
1	33.558727	33.367219
2	33.552439	33.495271
3	33.563576	33.388498
4	33.494949	33.472340
5	33.600749	33.460749
6	33.515780	33.479305
7	33.531073	33.666528
8	33.525829	33.410226
9	33.425846	33.491427
10	33.492501	214.088714
11	33.506480	33.398014
12	33.550357	33.414981
13	33.511719	33.453451
14	42.954706	33.381126
15	33.448077	33.380998
16	33.434992	33.396855
17	33.499800	33.406144
18	33.490950	33.411460
19	61.453072	33.476956
20	33.538023	33.401841
21	33.580878	33.470354
22	33.507777	33.625561
23	33.563844	33.467020
24	67.503070	33.343182
25	33.607811	33.474277
26	33.536188	33.462070
27	33.491140	33.436515
28	33.590006	33.469924
29	33.458395	33.323981
30	33.488426	33.364029

Bibliography

1. D. Lee, D. Kohlbrenner, S. Shinde, K. Asanovic, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20, 2020.

2. T. L. Foundation. (2019) Confidential computing consortium. [Online]. Available: https://confidentialcomputing.io/ [Accessed: 2021-12-04]

3. Trusted Computing Group, "Trusted Platform Module Library, Part 1: Architecture, Family "2.0", Level 00 Revision 01.59," 2019.

4. Confidential Computing Consortium. (2021) Confidential Computing Consortium Scope. [Online]. Available: https://confidentialcomputing.io/scope/ [Accessed: 2021-12-04]

5. J. C. Tullos, "Characterizing security monitor and embedded system performance across distinct risc-v ip-cores," Master's thesis, Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB, OH 45433-7765, 2021.

6. T. C. Group. (2019) Trusted computing. [Online]. Available: https://trustedcomputinggroup.org/trusted-computing/ [Accessed: 2022-01-19]

7. F. s. L. . R. . Trusted Platform Module Library Specification, "Iso/iec 11889-1:2015 information technology - tpm library - part 1: Architecture," 2015. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c066510_ISO_IEC_11889-1_2015.zip [Accessed: 2022-01-19]

8. C. C. Consortium, "White paper: A technical analysis of confidential computing," 2021. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/03/CCC-Tech-Analysis-Confidential-Computing-V1.pdf [Accessed: 2021-12-04]

9. M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, vol. 1, pp. 57–64, 2015.

10. G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, 2019, pp. 142–157.

11. A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, "The RISC-V Instruction Set Manual Volume I: Unprivileged ISA," *Document Version 20191213-draft*, vol. I, pp. 1–238, 2019.

12. ——, "The RISC-V Instruction Set Manual Volume II: Privileged Architecture," *Document Version 20211203-draft*, vol. II, pp. 1–155, 2021.

13. S. Inc., *SiFive FU740-C000 Manual*, SiFive, Inc. [Online]. Available: https://sifive.cdn.prismic.io/sifive/de1491e5-077c-461d-9605-e8a0ce57337d_fu740-c000-manual-v1p3.pdf [Accessed: 2021-12-04]

14. G. Heiser, "The sel4 microkernel: An introduction," *Banking*, vol. 0.3, no. June, p. 33, 2020. [Online]. Available: https://sel4.systems/Foundation\%0Ahttps://sel4.systems/About/seL4-whitepaper.pdf [Accessed: 2022-01-19]

15. G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "Sel4: Formal verification of an operating-system kernel," *Communications of the ACM*, vol. 53, no. 6, pp. 107–115, 2010.

16. W. D. Corporation, "Risc-v open source supervisor binary interface (opensbi)," https://github.com/riscv-software-src/opensbi, 2021.

17. U. R. Canonical. (2022) Ubuntu 21.04 (hirsute hippo). [Online]. Available: https://cdimage.ubuntu.com/ubuntu/releases/21.04/release/ [Accessed: 2022-01-04]

18. ——. (2022) Ubuntu 21.10 (impish indri) beta. [Online]. Available: https://cdimage.ubuntu.com/releases/21.10/beta/ [Accessed: 2022-01-04]

19. ——. (2022) Ubuntu 22.04 lts (jammy jellyfish) daily build. [Online]. Available: http://cdimage.ubuntu.com/ubuntu-server/daily-preinstalled/current/ [Accessed: 2022-01-04]

20. C. King. (2022) Stress-ng: Linux stress tool. [Online]. Available: https://github.com/ColinIanKing/stress-ng [Accessed: 2022-01-04]

21. P. T. Suite. (2022) Phoronix testing suite, release 10.8.1. [Online]. Available: https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.8.1_all.deb [Accessed: 2022-01-04]

22. OpenBenchmarking.org. (2022) Cross-platform, open-source, automated, centralized testing ecosystem. [Online]. Available: https://openbenchmarking.org/ [Accessed: 2022-01-04]

23. U. A. Canonical. (2022) Ubuntu release cycle. [Online]. Available: https://ubuntu.com/about/release-cycle [Accessed: 2022-02-01]

24. OpenBenchmarking.org. (2022) 2201177-eald-afit22m31. [Online]. Available: https://openbenchmarking.org/result/2201177-EALD-AFIT22M31 [Accessed: 2022-01-04]

25. ——. (2022) 2201183-eald-afit22m64. [Online]. Available: https://openbenchmarking.org/result/2201183-EALD-AFIT22M64 [Accessed: 2022-01-04]

26. ——. (2022) 2201123-eald-afit22m16. [Online]. Available: https://openbenchmarking.org/result/2201123-EALD-AFIT22M16 [Accessed: 2022-01-04]

27. ——. (2022) 2201124-eald-afit22m82. [Online]. Available: https://openbenchmarking.org/result/2201124-EALD-AFIT22M82 [Accessed: 2022-01-04]

28. ——. (2022) 2201062-eald-afit22m08. [Online]. Available: https://openbenchmarking.org/result/2201062-EALD-AFIT22M08 [Accessed: 2022-01-04]

29. ——. (2022) 2201085-eald-afit22m8. [Online]. Available: https://openbenchmarking.org/result/2201085-EALD-AFIT22M87 [Accessed: 2022-01-04]

30. P. T. Suite. (2022) Open-source, automated benchmarking. [Online]. Available: https://phoronix-test-suite.com/ [Accessed: 2022-01-04]

31. T. L. K. Documentation. (2022) The kernel's command line parameters. [Online]. Available: https://docs.kernel.org/admin-guide/kernel-parameters.html?highlight=crng [Accessed: 2022-02-02]

32. Hackster.IO. (2022) Sifive discontinues its hifive unmatched risc-v pc boards following "Supply Chain Issues". [Online]. Available: https://www.hackster.io/news/ [Accessed: 2022-02-02]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 24–03–2022	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE Evaluating Secure Enclave Firmware Development for Contemporary RISC-V Workstations	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Chadwick, Samuel D., 1Lt, USSF	**5d. PROJECT NUMBER** 21G195
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering an Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-22-M-017
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory 2241 Avionics Circle WPAFB OH 45433-7765 Attn: Pranav Patel COMM 937-656-9045 Email: pranav.patel.2@us.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RYDA **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The emergence of the open-source RISC-V ISA empowers developers and engineers, device manufactures, industry leaders, nation-states, adversaries and allies alike with the unique opportunity to re-evaluate existing Trusted Computing paradigms. Emerging open-source security mechanisms facilitate the proliferation of Confidential Computing principles. These technology standards aim to provide secure enclave computing as a fundamental computing attribute, inherent within the RISC-V ISA specification. Security enforcement within these enclaves are handled by performing computation in memory-isolated, hardware-based, software-defined TEEs. This research evaluates the firmware development procedures required to implement Keystone Enclave on new unsupported hardware. Expressly, this effort extends Keystone SM firmware components for use on the HiFive Unmatched development platform as a demonstration of Keystone Enclave's device portability claims. Furthermore, it proposes Keystone SDK and Eapp development recommendations to supplement contemporary ASIC RISC-V workstations with TEEs. Moreover, this research asserts that for the wide-spread adoption of Confidential Computing principles to occur, significant hardware, firmware, and software development advancements are required by all constituent parties.

15. SUBJECT TERMS

Confidential Computing, Keystone Enclave, PMP, RISC-V ISA, TEE, Trusted Computing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Scott Graham, AFIT/ENG
U	U	U	UU	108	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565 x4581; scott.graham@afit.edu