



## Chapter 9

# LEVERAGING CONFIDENTIAL COMPUTING TO ENABLE SECURE INFORMATION SHARING

Samuel Chadwick, Scott Graham, James Dean and Matthew Dallmeyer

**Abstract** The emergence of the RISC-V Instruction Set Architecture incentivizes the critical infrastructure protection community to consider the use of emerging open-source security mechanisms to facilitate secure information sharing. An exemplar is Keystone, a Confidential Computing Consortium project, that offers an accessible open-source framework for building trustworthy secure hardware enclaves based on the RISC-V Instruction Set Architecture.

This chapter describes an attempt at extending Keystone to the Hi-Five Unmatched development platform and proposes enclave application development to effectively and affordably supplement deployed supervisory control and data acquisition devices with secure information sharing capabilities. Since the implementation of confidential computing principles axiomatically degrades real-time performance, the performance of supervisory control and data acquisition devices must be characterized to ensure that the devices enhanced with trusted execution environments meet operational requirements while supporting critical infrastructure operations with secure information sharing capabilities.

**Keywords:** Secure information sharing, confidential computing, Keystone enclave

## 1. Introduction

The persistent desire to securely share information drives the continuing evolution of mechanisms for enforcing information security that keeps pace with and responds to technological advancements. This research proposes the application of confidential computing principles to implement secure information sharing across a wide range of supervisory control and data acquisition used in critical infrastructure assets. Expressly, the research establishes the plausibility of building trusted

execution environments using commodity RISC-V personal computer hardware to supplement deployed SCADA devices with improved communication and operational security at reasonable, or even low, cost.

Keystone, a project of the Confidential Computing Consortium [2], is the first open-source framework for building customized trustworthy secure hardware enclaves based on the RISC-V Instruction Set Architecture [3]. The ability to port Keystone security monitor to new RISC-V hardware enables Linux distribution support and enclave application development. The use cases would exploit trusted execution environment primitives by equipping deployed SCADA devices with secure, isolated enclaves that appropriately segregate control mechanisms from data collection and sharing protocols.

This chapter describes an attempt at extending Keystone to the HiFive Unmatched development platform, the only commercially-available RISC-V development platform that satisfies the criteria of form factor standardization, commodity personal computer hardware compatibility, Linux operating system support and enhanced system-on-chip monitoring capabilities. This would support enclave application development to provide deployed supervisory control and data acquisition devices with secure information sharing capabilities effectively and affordably.

Since the implementation of confidential computing principles axiomatically degrades real-time performance, it is imperative to determine the performance overhead imposed by trusted execution environment implementations. This chapter describes synthetic benchmarking conducted for the HiFive Unmatched system that executed 20 compatible benchmarks from the Stress-NG benchmarking suite. At a price point of \$655, the HiFive Unmatched underperformed competitively-priced x86-64 commodity workstations. Nevertheless, as a Linux-capable, native RISC-V development platform that supports open-source trusted execution environment implementations, HiFive Unmatched sets the bar in the emerging RISC-V market.

## 2. Background and Related Work

This section discusses confidential computing and provides details about the RISC-V Instruction Set Architecture and Keystone enclave used in this work. Also, it discusses related work in the area.

### 2.1 Confidential Computing

The Confidential Computing Consortium is a Linux Foundation initiative that seeks to secure data in use through open collaboration. Commonly-deployed encryption techniques enforce confidentiality, in-

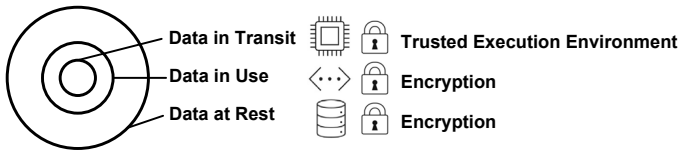


Figure 1. Security mechanisms applied to classical computing data states.

egrity and availability for data at rest in storage media and for data in transit across public and private networks. However, these techniques are limited by the conventional computing infrastructure. To adequately secure data in use, specifically during execution, computations must be performed in a hardware-based trusted execution environment (TEE) [2] or properly manipulate encrypted data without decrypting it first, as in the case of homomorphic computing, which is outside the scope of this work. Figure 1 illustrates the security mechanisms that apply to classical computing data states.

While a formal definition of a trusted execution environment has not been arbitrated, the Confidential Computing Consortium defines it as an environment that provides a level of assurance of data confidentiality, data integrity and code integrity [1, 5]. This work uses the definition of trusted execution environment interchangeably with variations commonly used by industry.

Many trusted execution environment implementations are proprietary, including the Intel Software Guard Extensions, ARM TrustZone and AMD Secure Encrypted Virtualization. Vendor-specific trusted execution environment implementations have two distinct disadvantages. The first is that intellectual property ties new features and bug fixes directly to vendors. The second is that different threat models have been ascribed to specific instruction set architectures. For example, Intel Software Guard Extensions focus on server and desktop application isolation, ARM TrustZone addresses vendor-provisioned mobile application isolation and AMD Secure Encrypted Virtualization focuses on virtual machine isolation.

In the context of critical infrastructure protection, the disadvantages of vendor-specific trusted execution environment implementations must be weighed against emerging open-source alternatives. A Keystone enclave provides an extensible open-source trusted execution environment implementation for the RISC-V Instruction Set Architecture. This research asserts that Keystone and supporting hardware offer an avenue for extending deployed SCADA devices with secure information sharing

functionality while avoiding the deficiencies found in proprietary trusted execution environments.

## 2.2 RISC-V Instruction Set Architecture

The free and open RISC-V Instruction Set Architecture is intended to enable a new era of processor innovation through open standard collaboration [8]. The RISC-V Instruction Set Architecture is organized into an unprivileged instruction set architecture and a privileged instruction set architecture.

**Unprivileged Instruction Set Architecture.** The unprivileged instruction set architecture comprises the base integer architecture I and additional optional instruction set extensions. The set of standard extensions currently includes multiply/divide operations M, atomic operations A, single- and double-precision floating-point arithmetic operations F and D, respectively, and compressed 16-bit instructions C [8]. The M, A, F and D identifiers are standard extensions that are collectively referred to as G. This research employs 64-bit integer registers with all the standard and compressed extensions. Thus, the instruction set architecture descriptor used in the research is designated as RV64GC, where RV denotes RISC-V, 64 denotes the register width, G denotes the base instruction set architecture with standard extensions and C denotes support for compressed operations.

**Privileged Instruction Set Architecture.** The privileged instruction set architecture covers all aspects of RISC-V systems beyond the unprivileged instruction set architecture [9]. The features pertinent to this research are physical memory protection 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), is not employed in this research. Keystone makes appropriate use of these features to enforce isolated execution in trusted execution environments. These memory-isolated environments are often referred to as “secure enclaves.” Because Keystone is employed throughout the creation, execution and destruction lifecycles of the enclaves, they are named Keystone enclaves.

## 2.3 Keystone Enclave

As a current project of the Confidential Computing Consortium, Keystone offers an accessible open-source framework that provides academia and industry with resources for building trustworthy secure hardware

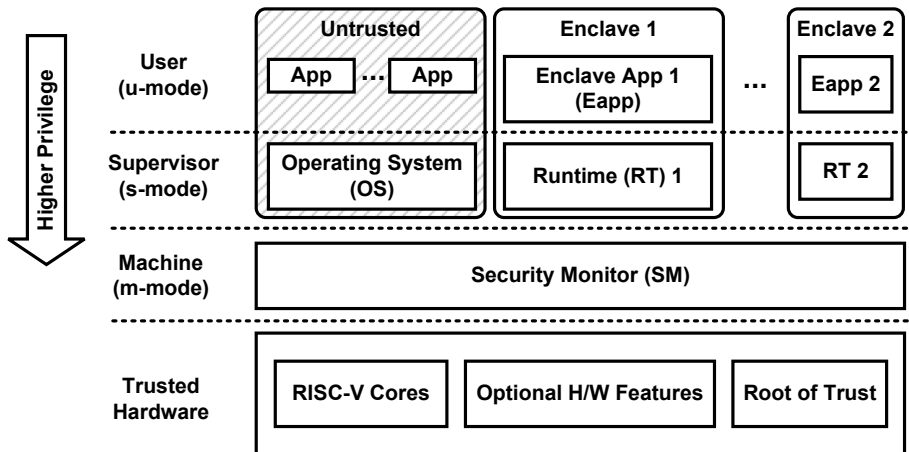


Figure 2. Compute system operations with Keystone [3].

enclaves. Keystone is the first open-source framework for building customized trusted execution environments [3]. It is designed for and built on the RISC-V privileged instruction set architecture. By leveraging trusted hardware, Keystone enables software-defined, hardware-enforced, isolated, memory-mapped execution beneath an untrusted operating system [3]. Currently, Keystone supports three standard trusted execution environment primitives, secure boot, secure randomness source and remote attestation [3]. Figure 2 shows the distinct Keystone components as they operate within the RISC-V Instruction Set Architecture privilege levels alongside an untrusted operating system.

**Keystone Security Monitor.** The Keystone security monitor, a core component of a Keystone enclave, relies entirely on the RISC-V standards for operation. This intentional design constraint promotes portability across RISC-V hardware platforms. This design principle is leveraged to port Keystone security monitor to SiFive’s HiFive Unmatched development platform, a multi-core, native RISC-V, application-specific integrated circuit (ASIC) computer. The Keystone security monitor achieves memory isolation for enclave runtimes and enclave applications by utilizing physical memory protection hardware built directly into each hardware application core [6]. The development platform features and specifications are provided later in this chapter. This research focuses on extending the Keystone security monitor to the HiFive Unmatched platform.

**Keystone Root-of-Trust.** Although the root-of-trust is typically depicted as a hardware component, Keystone also supports tamperproof software implementations. The research described in this chapter leverages this feature by employing modified first- and second-stage bootloaders to simulate the secure boot primitive. The research does not attempt to verify, validate or otherwise assess the cryptographic techniques employed by Keystone to realize trusted execution environment primitives. Instead, it supports Keystone’s portability claims by extending its use to previously-unsupported hardware.

**Keystone Enclave Applications.** With successful modifications to hardware-specific (m-mode) software (firmware), a HiFive Unmatched development platform equipped with Keystone could be configured to execute Keystone enclave applications. Application development would support any statically-compiled RISC-V binary as long as all the supporting libraries are included in Eyrie, the Keystone runtime environment. Specifically, secure enclave applications are envisioned that enable secure information sharing between SCADA devices across internal and external networks. Thus, sensitive SCADA operations could be appropriately decoupled from data collection tasks to shield critical infrastructure assets from untrusted actors and devices.

## 2.4 Related Work

Porting the Keystone security monitor to new hardware platforms is just an initial step on the path towards critical infrastructure device integration. To fully implement Keystone on contemporary RISC-V hardware, additional Linux kernel modifications will have to be baselined to support Linux distributions. Moreover, to encourage confidential computing practices, Linux distributions will likely need to provide flexible tools to facilitate the porting of Keystone enclaves to more devices. To justify the incorporation of trusted execution environments in deployed SCADA devices, strict performance requirements must also be maintained. The addition of secure enclave computing unavoidably impacts system performance. Therefore, characterization studies must be conducted to effectively evaluate trusted execution environment performance.

Tullos [7] has conducted performance characterizations of embedded RISC-V devices configured with Keystone implemented on field-programmable gate array (FPGA) hardware. As the RISC-V landscape matures, future performance characterizations must include ASIC hardware implementations with representative system evaluations for workstation-focused systems such as the HiFive Unmatched platform.

### 3. Experimental Configuration

The HiFive Unmatched development platform was selected due to its form factor standardization, commodity personal computer hardware compatibility, Linux operating system support and enhanced system-on-chip monitoring capabilities. HiFive Unmatched is currently the only commercially-available RISC-V development platform that satisfies the four desired criteria. In particular, it has the Mini-ITX form factor used by many AMD/Intel x86\_64 systems. This standard personal computer form factor enables straightforward hardware extensions via PCIe and NVMe interconnects. Moreover, HiFive 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, RISC-V workstation as opposed to an embedded system.

Importantly, the HiFive Unmatched is manufactured by SiFive, an industry leader in the RISC-V technology space. The SiFive leadership includes three co-founders of the RISC-V Instruction Set Architecture. Their involvement inspires confidence that SiFive will continue to support its products as the RISC-V specifications evolve.

**Development Board Specifications and Features.** The HiFive Unmatched platform is powered by a SiFive Freedom U740 system-on-chip, a multi-core, 64-bit dual-issue, superscalar RISC-V processor whose advertised performance is comparable to the ARM Cortex-A55. The Freedom U740 system-on-chip contains four Linux-capable U74 application cores that support RV64GC operations and includes a fifth S7 monitor core that supports RV64IMAC operations. All the cores have dual-issue in-order execution pipelines that support peak sustained execution rates of two instructions per cycle and maintain a fully-coherent 2 MB shared L2 cache. Additional board specifications include 16 GB DDR4 SDRAM, 32 MB Quad-SPI flash memory, MicroSD card expansion, Gigabit Ethernet, four USB 3.2 Gen 1 Type A ports, a microUSB JTAG console port, x16 PCIe Gen 3 expansion slot, M.2 M-Key slot for NVMe 2280 SSD modules and M.2 E-Key slot for Wi-Fi and Bluetooth modules.

Figure 3 shows the test platform configuration. The test configuration utilized the M.2 M-Key NVMe slot to leverage a 500 GB Samsung 980 PRO PCIe 4.0 SSD. Additional components were not strictly required in the research, but they enhanced performance by providing faster memory technology for testing in environments without wired Internet access. The PCIe expansion slot was not used in the research; the graphical capabilities are left for future investigations.

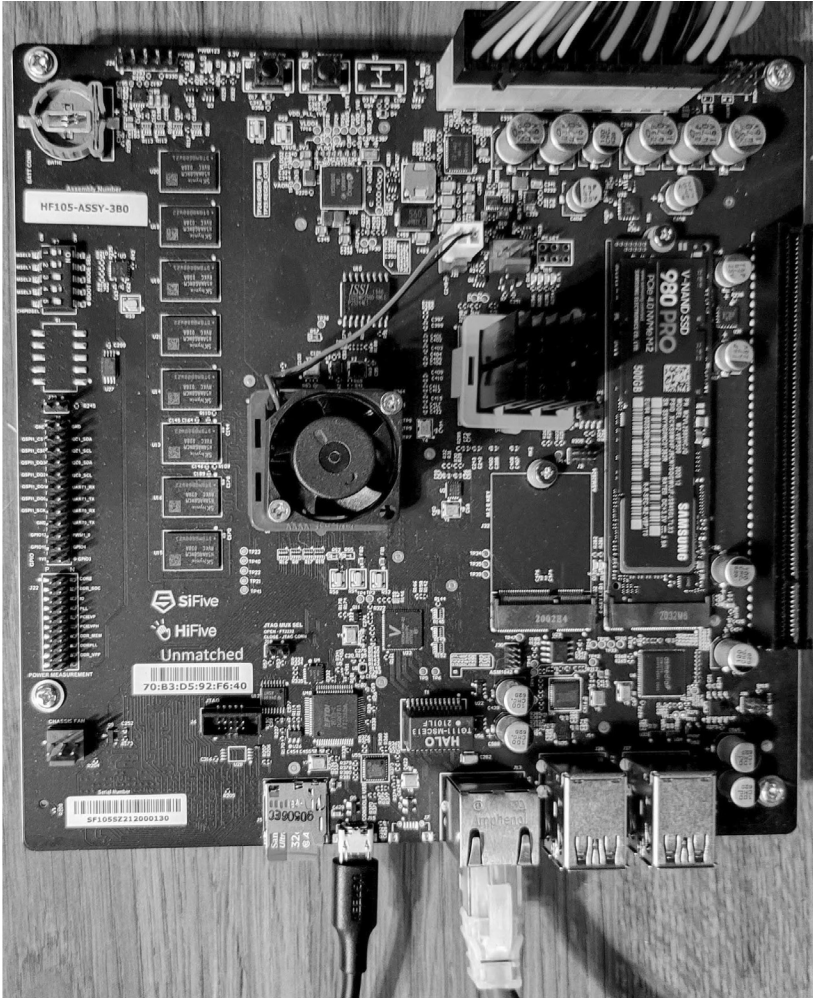


Figure 3. HiFive Unmatched Mini-ITX development platform configuration.

**Boot Flow Modifications.** The SiFive FU740-C000 manual details the boot process of the HiFive Unmatched platform [6]. Figure 4 shows the unmodified (standard) boot flow for the test HiFive Unmatched platform configuration. The boot operations proceed in the following order of precedence: power on reset (PoR) (0), zeroth stage bootloader (ZSBL) stored in on-chip mask read-only memory (1), first stage bootloader (U-Boot secondary program loader (SPL)) (2), secondary bootloader (SBL) containing the U-Boot image tree blob (ITB), device tree blob (DTB) and OpenSBI (3), EXTLINUX (4) and Linux kernel (5). In order to per-



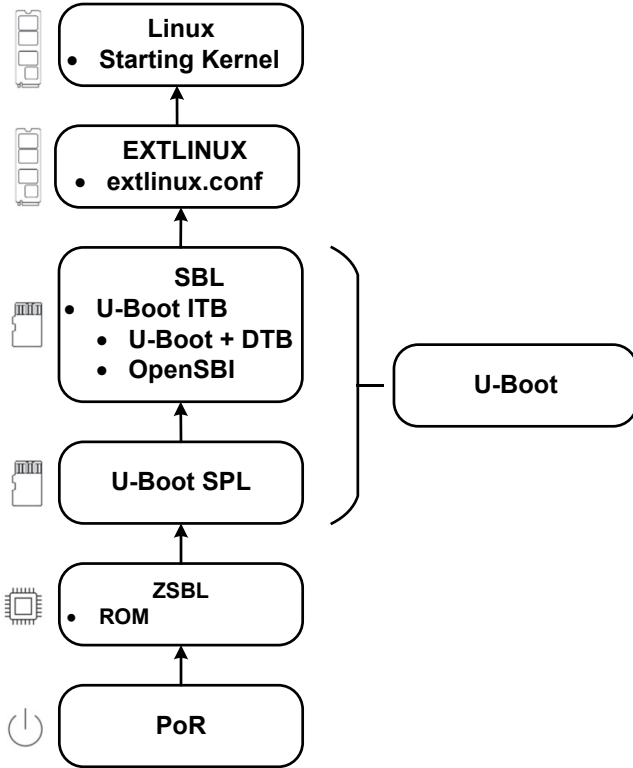


Figure 4. Standard boot flow.

form baseline performance characterizations of the HiFive Unmatched system without the Keystone security monitor, a preinstalled Ubuntu server image was employed. The bootable image was flashed to a microSD card, which was used to boot the HiFive Unmatched platform successfully with Ubuntu.

In order to implement the Keystone security monitor on the HiFive Unmatched platform, OpenSBI was used as the interface between the bootloader and platform-specific firmware executing in m-mode. OpenSBI is an independent RISC-V Foundation project that provides an open-source reference implementation of a platform-independent static library to implement a serial binary interface [4]. OpenSBI also provides platform-specific support, including Freedom-U740-specific libraries required to modify the HiFive Unmatched development kit.

To construct the modified bootable microSD card image, the Keystone security monitor was built in OpenSBI using an out-of-tree plat-

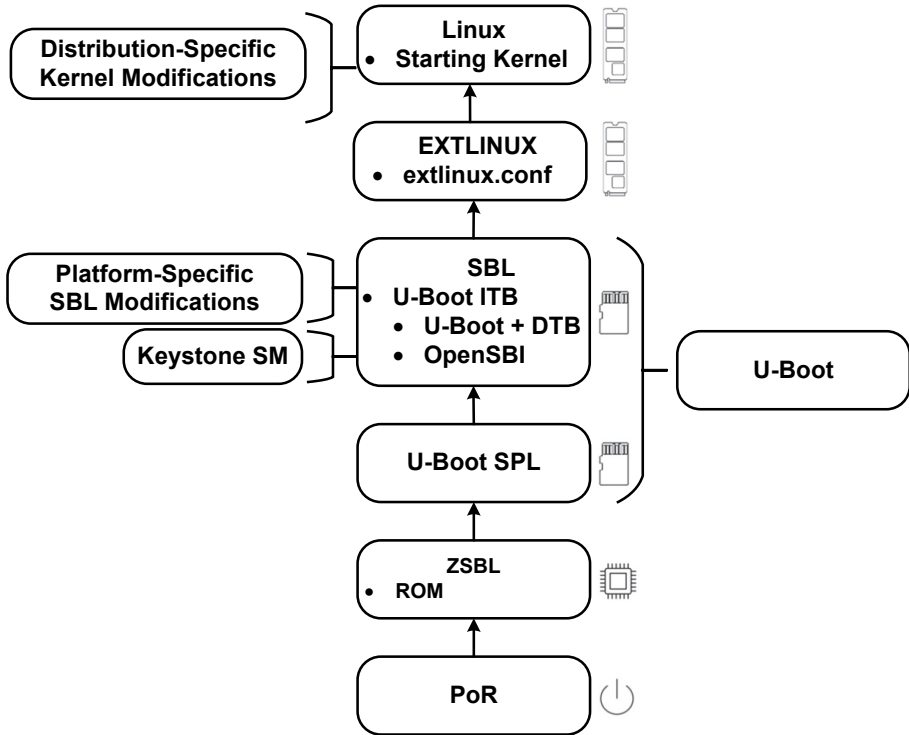


Figure 5. Modified boot flow with the Keystone security monitor.

form build configuration supported by the OpenSBI toolchain. All the development work was conducted on an external x86\_64 Kubuntu Linux workstation, although the Linux distribution used for development was arbitrary. Because Keystone does not officially support the HiFive Unmatched development platform, several Keystone components required the manual application of patch files provided by SiFive; the modifications included platform-specific changes to OpenSBI, U-Boot and the Linux kernel. Figure 5 highlights the boot flow modifications required to implement the Keystone security monitor on the test platform.

After the Keystone security monitor was configured in the OpenSBI platform build, the `fw_dynamic.bin` platform configuration binary used by U-Boot was created. Next, the `u-boot.itb` image tree blob and `u-boot-spl.bin` binary files required to build the U-Boot bootloader were created. These two files, which comprise the U-Boot bootloader, were flashed to the microSD card image.

**Kernel Modifications.** After successfully building the modified bootloader with the Keystone security monitor, the next step was to build the Linux kernel in order to apply the Keystone security monitor and SiFive patches and cross-compile the build for the RV64GC target. The Keystone kernel build process produced the `Image.gz` Linux kernel for the OpenEmbedded distribution; this is an artifact of hardware support for the discontinued HiFive Unleashed development platform. The build used a `makefile` script to generate the `hifive-unmatched-a00.dtb` device tree blob containing the specific board hardware descriptor. The Linux kernel for the Ubuntu distribution was employed for unmodified device performance characterization. These Linux kernels, created by Canonical, are provided in pre-built server images for HiFive Unmatched. Performance characterizations of the modified Linux kernel for the OpenEmbedded distribution are available for future testing efforts.

**Root Filesystem and Distribution.** Since the research did not require building a root filesystem from scratch, pre-built server images provided by Canonical were employed. The research used various daily builds of Ubuntu distributions, including 21.04 (Hirsute Hippo), 21.10 (Impish Indri) and 22.04 LTS (Jammy Jellyfish).

**Bootable Image.** The development concluded by creating a bootable image file and flashing it to a microSD card. The modified Ubuntu image was built by flashing the desired pre-built server image to the microSD card to create a default bootable medium. The `dd` tool was then used to overwrite the image tree blob and device tree blob boot partitions with the modified U-Boot bootloader and the included Keystone security monitor.

The OpenEmbedded distribution was built by creating an empty image file, which was partitioned with the appropriate disk identifiers. The bootloader partitions were written, following which the root filesystem was created and mounted. Next, the root filesystem was unpacked and the Linux kernel packages were copied to the root filesystem. Following this, the Linux kernel was installed, the image tree blob and device tree blob were copied to the correct partitions and the `extlinux.conf` file for EXTLINUX was created. Finally, the newly-created image file was flashed to the microSD card and the root partition was resized.

Upon inserting the microSD card into the HiFive Unmatched platform and booting the device, the serial console shown in Figure 6 was displayed. The `U_BOOT_ROOT` environment variable was then set to use

COM4 - PuTTY

---

```
U-Boot SPL 2022.01-rc2-00024-g3144ba23bf (Nov 18 2021 - 00:26:34 -0500)
Trying to boot from MMC1
```

```
U-Boot 2022.01-rc2-00024-g3144ba23bf (Nov 18 2021 - 00:26:34 -0500)
```

```
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:  0
PCIE-0: Link up (Gen1-x8, Bus0)

Device 0: unknown device
starting USB...
Bus xhci_pci: Register 4000840 NbrPorts 4
Starting the controller
USB XHCI 1.00
scanning bus xhci_pci for devices... 3 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found

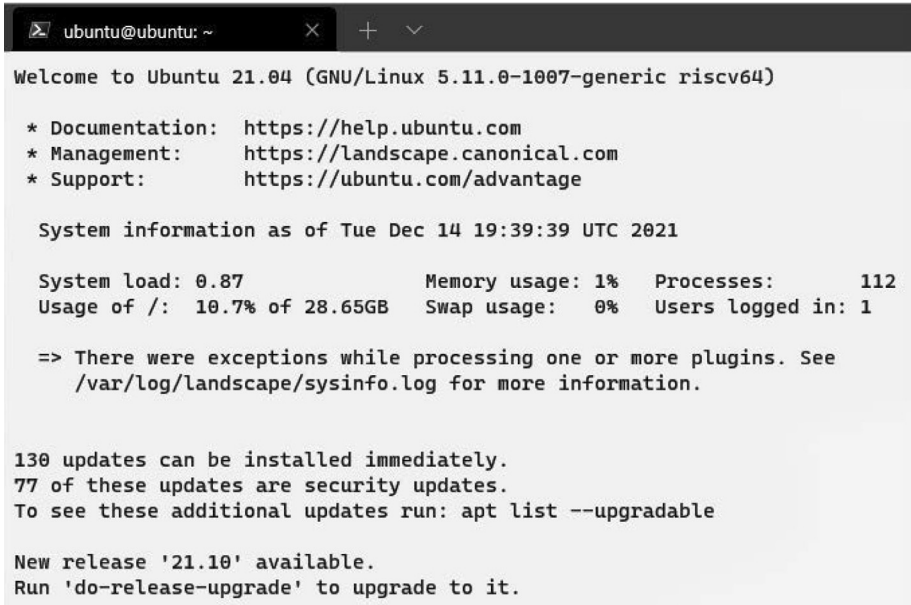
Device 0: unknown device
switch to partitions #0, OK
mmc0 is current device
scanning bus for devices...
```

Figure 6. U-Boot serial console output.

the preconfigured NVMe drive with the Ubuntu operating system. Figure 7 shows the Ubuntu terminal after system login.

## 4. Proposed Development

The ability to port Keystone security monitor to new RISC-V hardware enables Linux distribution support and enclave application develop-

A terminal window titled 'ubuntu@ubuntu: ~' with window control buttons. The terminal output includes: 'Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-1007-generic riscv64)', links for documentation, management, and support, system information as of Tue Dec 14 19:39:39 UTC 2021, system load (0.87), memory usage (1%), processes (112), usage of / (10.7% of 28.65GB), swap usage (0%), and users logged in (1). It also displays a warning about exceptions while processing plugins, 130 updates available (77 security updates), and a new release '21.10' available.

```
ubuntu@ubuntu: ~
Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-1007-generic riscv64)

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

System information as of Tue Dec 14 19:39:39 UTC 2021

System load: 0.87           Memory usage: 1%    Processes:      112
Usage of /:  10.7% of 28.65GB  Swap usage:  0%    Users logged in: 1

=> There were exceptions while processing one or more plugins. See
   /var/log/landscape/sysinfo.log for more information.

130 updates can be installed immediately.
77 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '21.10' available.
Run 'do-release-upgrade' to upgrade to it.
```

Figure 7. Ubuntu 21.04 (Hirsute Hippo) terminal.

ment, especially for critical infrastructure protection applications. These use cases would exploit trusted execution environment primitives by equipping SCADA devices with secure, isolated enclaves that segregate control mechanisms from data collection and sharing protocols.

Supplementing deployed SCADA devices with open-source trusted execution environments would be a practical secure information sharing solution that avoids the large-scale replacement of proprietary implementations. By leveraging Keystone, the RISC-V Instruction Set Architecture and a growing list of compatible commodity personal computer hardware components, legacy SCADA devices can be made extensible by augmenting information sharing responsibilities with emerging native RISC-V devices. Promising scenarios would employ capable RISC-V platforms such as HiFive Unmatched to empower decision makers by operating as intermediary confidential computing networks that securely obtain relevant data from SCADA devices, process the data in secure enclaves and transmit encrypted information for secure collection.

As the RISC-V landscape evolves, it is anticipated that confidential computing practices will be adopted widely. In particular, low-cost native RISC-V platforms with commodity personal computing hardware augmented with open-source trusted execution environments – such as

OpenBenchmarking.org	Phoronix Test Suite 10.6.1
<b>SiFive RISC-V (4 Cores)</b>	Processor
<b>SiFive HiFive Unmatched A00</b>	Motherboard
<b>SiFive FU740-C000 RISC-V SoC</b>	Chipset
<b>16GB</b>	Memory
<b>Samsung SSD 980 PRO 500GB + 32GB SD32G</b>	Disk
<b>Ubuntu 22.04</b>	OS
<b>5.13.0-1006-generic (riscv64)</b>	Kernel
<b>GCC 11.2.0</b>	Compiler
<b>ext4</b>	File-System

Figure 8. Baseline performance configuration.

Keystone – would be used to affordably facilitate confidentiality, integrity and availability across all the data states.

## 5. Experimental Results and Analysis

Before endowing deployed SCADA devices with trusted execution environments and secure information sharing features, it is important to obtain a thorough understanding of system performance requirements. Specifically, it is imperative to determine the performance overhead imposed by a trusted execution environment implementation. In the case of Keystone and the HiFive Unmatched system, synthetic benchmarking can yield insights for evaluating system performance.

Baseline benchmarking was conducted to evaluate SiFive’s performance claims about the HiFive Unmatched system. Figure 8 specifies the baseline performance configuration.

The Phoronix Test Suite, an open-source, comprehensive testing and benchmarking tool, was employed to run 20 compatible benchmarks from the Stress-NG benchmarking suite (version 1.4.0). Table 1 describes the benchmarks used in the baseline performance evaluation.

Table 2 shows the baseline performance for the benchmark stress tests in bogo-ops/s, where higher scores indicate better performance. Each benchmark test was trialed three times or trialed repeatedly until a standard deviation of less than one was obtained.

The results establish an upper threshold for typical HiFive Unmatched system performance. With the inclusion of trusted execution environments via Keystone, subsequent benchmark scores are expected to decline. At a price point of \$655, the HiFive Unmatched dramatically

Table 1. Selected benchmarks from Stress-NG 1.4.0.

Benchmark	Description
MMAP	Memory map
NUMA	Non-uniform memory access
MEMFD	Anonymous kernel memory management
Atomic	Atomic operations
Crypto	MD5, SHA-256, SHA-512, <code>scrypt</code> , NT, <code>yescrypt</code>
<code>malloc</code>	Memory allocation
Forking	CPU forking
<code>io_uring</code>	Asynchronous input/output
SENDFILE	Read/write
CPU Cache	Cache thrashing
CPU Stress	Integer, multiplication, floating point, double precision
Semaphores	Shared resources
Matrix Math	Two- and three-dimensional matrix operations
Vector Math	128-bit vector operations
Memory Copying	<code>memcpy</code> method operation
Socket Activity	IPv4, TCP congestion control
Context Switching	Memory clobbering
<code>glibc</code> C String Functions	<code>glibc</code> C string functions
<code>glibc</code> Qsort Functions	<code>glibc</code> Qsort functions
System V Message Passing	System V message passing

underperformed competitively-priced x86\_64 PC workstations. For the CPU stress test, the average time required to complete the benchmark for all publicly-listed systems at [openbenchmarking.org](https://openbenchmarking.org) was only 1.8 minutes whereas it exceeded 16 minutes for HiFive Unmatched. Compared against a quad-core ARM Cortex-A55 system-on-chip, the HiFive Unmatched system recorded an average of 208.20 bogo-ops/s that outperformed the average 156.28 bogo-ops/s recorded by the ARM Cortex-A55 system-on-chip; this meets the advertised HiFive Unmatched performance claims for CPU operations. Substantially cheaper ARM alternatives, such as the Raspberry Pi 400, which does not fully support ARM TrustZone and hardware-enforced trusted execution environments, handily doubled the CPU stress performance achieved by HiFive Unmatched. Nevertheless, as a Linux-capable, native RISC-V development platform that supports open-source trusted execution environment implementations, HiFive Unmatched sets the bar in the emerging RISC-V market.

Table 2. Baseline performance.

Benchmark	Score (bogo-ops/s)
MMAP	1.55
NUMA	12.66
MEMFD	7.49
Atomic	55,245.91
Crypto	91.75
malloc	1,572,568.61
Forking	3,163.54
io_uring	2,440.31
SENDFILE	7,338.63
CPU Cache	16.52
CPU Stress	208.20
Semaphores	119,929.34
Matrix Math	617.00
Vector Math	440.98
Memory Copying	34.48
Socket Activity	177.85
Context Switching	144,396.00
glibc C String Functions	18,746.18
glibc Qsort Functions	5.68
System V Message Passing	375,212.46

## 6. Conclusions

Supplementing deployed SCADA devices with open-source trusted execution environments is a practical secure information sharing solution that eliminates the large-scale replacement of proprietary implementations. By leveraging Keystone, the RISC-V Instruction Set Architecture and a growing list of compatible commodity personal computer hardware components, legacy SCADA devices can be made extensible by augmenting information sharing responsibilities with emerging native RISC-V devices. Promising scenarios would employ capable RISC-V platforms such as HiFive Unmatched to empower decision makers by operating as intermediary confidential computing networks that securely obtain relevant data from operational SCADA devices, process the data in secure enclaves and transmit encrypted information for secure collection.

The RISC-V Instruction Set Architecture is new and does not yet rival the market share of AMD/Intel x86\_64 and ARM instruction set architectures. It was only in December 2021 that the RISC-V privileged instruction set was officially ratified for a few compatible hardware-optimized applications and devices. Nevertheless, the experimentation



with RISC-V hardware and software demonstrates a renewed interest in instruction set architecture development. Clearly, proprietary computer architectures with undisclosed security mechanisms will not suffice for future data security applications. Therefore, it is important to advocate open technologies that offer innovative solutions for securing data in use. As the RISC-V landscape evolves, it is anticipated that confidential computing practices will be adopted widely. In particular, low-cost native RISC-V platforms with commodity personal computing hardware augmented with open-source trusted execution environments like Keystone would be attractive because they can affordably facilitate information confidentiality, integrity and availability across all the data states.

Future research will focus on Keystone enclave application development for comparative benchmarking as well as on Keystone enclave development.

The views expressed in this chapter are those of the authors, and do not reflect the official policy or position of the U.S. Air Force, U.S. Space Force, U.S. Department of Defense or U.S. Government. This document has been approved for public release; distribution unlimited (Case #88ABW-2021-1035).

## References

- [1] Confidential Computing Consortium, A Technical Analysis of Confidential Computing (v1.1), The Linux Foundation, San Francisco, California ([confidentialcomputing.io/wp-content/uploads/sites/85/2021/03/CCC-Tech-Analysis-Confidential-Computing-V1.pdf](https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/03/CCC-Tech-Analysis-Confidential-Computing-V1.pdf)), 2021.
- [2] Confidential Computing Consortium, What is the Confidential Computing Consortium? The Linux Foundation, San Francisco, California ([confidentialcomputing.io](https://confidentialcomputing.io)), 2022.
- [3] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanovic and D. Song, Keystone: An open framework for architecting trusted execution environments, *Proceedings of the Fifteenth European Conference on Computer Systems*, article no. 38, 2020.
- [4] opensbi Contributors, RISC-C Open Source Supervisor Binary Interface, GitHub ([github.com/riscv-software-src/opensbi](https://github.com/riscv-software-src/opensbi)), 2021.
- [5] M. Sabt, M. Achemlal and A. Bouabdallah, Trusted execution environment: What it is and what it is not, *Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 57–64, 2015.

- [6] SiFive, SiFive FU740-C000 Manual (v1p6), San Mateo, California ([sifive.cdn.prismic.io/sifive/1a82e600-1f93-4f41-b2d8-86ed8b16acba\\_fu740-c000-manual-v1p6.pdf](https://sifive.cdn.prismic.io/sifive/1a82e600-1f93-4f41-b2d8-86ed8b16acba_fu740-c000-manual-v1p6.pdf)), 2021.
- [7] J. Tullos, Characterizing Security Monitor and Embedded System Performance Across Distinct RISC-V IP-Cores, M.S. Thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2021.
- [8] A. Waterman and K. Asanovic (Eds.), The RISC-V Instruction Set Manual Volume I: Unprivileged ISA, Document Version 20191213, RISC-V Foundation, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, California, 2019.
- [9] A. Waterman, K. Asanovic and J. Hauser (Eds.), The RISC-V Instruction Set Manual Volume II: Privileged Architecture, Document Version 20211203, RISC-V Foundation, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, California, 2021.