PROVIDING SECURE BOOT OF OBFUSCATED OPCODES FOR CYBERSHIELD USING A TRUSTED EXECUTION ENVIRONMENT

by

Garrett Ray Perkins

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Cybersecurity

MONTANA STATE UNIVERSITY
Bozeman, Montana

May 2025

ACKNOWLEDGEMENTS

I would like to acknowledge my committee co-chairs, Dr. Brock LaMeres and Dr. Clemente Izurieta, for their mentorship and support throughout the entirety of this thesis. I would also like to acknowledge my committee members, Dr. Ann Marie Reinhold and Dr. Bradley Whitaker, for their support and guidance throughout this project. Montana State University Software Engineering and Cybersecurity Lab (MSU SECL) members provided valuable insights and discussions that helped advance this work.

NASA and Resilient Computing, LLC supported this research under award number 80NSSC23CA147, and subcontract number 4W9082, respectively. This research was conducted with the U.S. Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under contract 70RSAT24KPM000022. Any opinions contained herein are those of the authors and do not necessarily reflect those of NASA, Resilient Computing, LLC, or DHS S&T.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

NOMENCLATURE

TEE	Trusted Execution Environment
FPGA	Field Programmable Gate Array
SoC	System-on-Chip
RISC-V	Reduced Instruction Set Computer five
IoC	Indicator of Compromise
MSUSEL	Montana State University Software Engineering Lab
NASA	National Aeronautics and Space Administration
DHS	U.S. Department of Homeland Security

ABSTRACT

As Field Programmable Gate Arrays (FPGAs) become increasingly integrated into security-critical applications—particularly in space, IoT, and edge computing environments—the need for robust security mechanisms has grown paramount. Despite their versatility and performance, FPGAs were not initially designed with strong security in mind, leaving them vulnerable to attacks such as code injection and buffer overflows. This thesis introduces CyberShield, a Trusted Execution Environment (TEE) architecture integrated into the radiation-tolerant RadPC softcore processor to defend against such threats.

CyberShield enables secure boot with obfuscated opcodes, leveraging RadPC's Quad Modular Redundancy (QMR) and opcode diversification to prevent unauthorized code execution. By assigning unique opcode offsets to each core and validating integrity through an anti-voter module, CyberShield can detect code and command injection attacks. This thesis details the architectural modifications necessary to achieve secure boot from non-volatile memory, the encryption and bootloading process, and how the system mitigates injection-based attacks. Experimental validation confirms CyberShield's ability to resist buffer overflow exploits while preserving system reliability and performance. This work lays the foundation for further development of other TEE related security features for RadPC and embedded systems deployed in adversarial environments.

1

INTRODUCTION

The proliferation of edge computing and embedded systems has revolutionized modern computing paradigms. With data processing increasingly occurring at the network edge—close to sensors and user devices—systems are gaining new capabilities in speed, autonomy, and context awareness [2]. At the heart of many of these edge systems are Field Programmable Gate Arrays (FPGAs), valued for their reconfigurability, performance, and support for custom architectures. Particularly, System-on-Chip (SoC) FPGAs, which integrate programmable logic with embedded processors, have become critical in applications ranging from space avionics and industrial control systems to AI acceleration and IoT[3].

However, with this shift toward high-performance, real-time edge computing comes a significant expansion of the attack surface [4]. Originally designed with performance rather than security in mind, FPGAs are now deployed in hostile or high-stakes environments—such as space, defense, and multi-tenant cloud systems—where threats such as code injection, buffer overflow, and unauthorized access pose serious risks. These devices often operate without the security assurances available in conventional CPUs and OS-managed systems [5].

To address these challenges, Trusted Execution Environments (TEEs) have emerged as a foundational component for securing edge devices. TEEs establish isolated, tamper-resistant regions within a system where critical code and data can be executed securely [6]. While mainstream CPU vendors like ARM, Intel, and AMD have introduced TEE platforms like TrustZone, SGX, and SEV, these solutions are chip-specific and not accessible cross platform. Furthermore, most TEEs are developed for general-purpose processors, with few extensible or application-agnostic solutions targeting SoC FPGA platforms.

Based on the findings of the manuscript in Chapter 2 extensible TEEs are few and far between. While they do provide modularity and flexibility, they are still not suitable for all applications. The FPGA-based single board computer RadPC was originally designed as a radiation-tolerant space computer and is distinguished by its Quad Modular Redundancy (QMR) design, which allows it to recover from multiple single event effects (SEEs) caused by radiation. The FPGA allows real-time configuration of hardware to support field upgrades. This enables recovery procedures that has been found to be able to flush out radiation-induced faults. Because of its small form factor, e.g. an Artix 200T, there are limited resources to implement a TEE. Thus, much like many researchers in Chapter 2, this thesis focuses on a specific feature–secure boot. This thesis addresses the addition of a TEE that provides secure boot to CyberShield, a novel TEE architecture integrated with RadPC, a RISC-V-based radiation-tolerant microcontroller developed for small satellite applications. CyberShield builds on this resilience by incorporating secure boot mechanisms and opcode obfuscation to further harden the system against software-level cyber threats, providing both hardware and software redundancy.

CyberShield employs a bootloader-enabled TEE that receives encrypted executables via UART and stores them in non-volatile memory (NVM). It then decrypts and distributes obfuscated instruction sets—each with unique opcode offsets—to RadPC's four cores over separate UARTs. An anti-voter module monitors for anomalous behavior, raising an error flag if all four cores execute identical instructions, which would be an Indicator of Compromise (IoC). The obfuscated instructions allows CyberShield to resist and detect malicious code being excited as the result of an attack.

The implementation is validated through simulation and physical deployment on a Nexys A7-100T development board using Xilinx Artix-7 FPGAs. Testing included executing normal workloads, verifying secure boot functionality, and simulating buffer overflow attacks. Results demonstrate CyberShield's ability to boot safely, detect tampered execution, and

prevent unauthorized instruction execution, even under active attack conditions. Through obfuscation, hardware-enforced isolation, and recovery, CyberShield represents a robust approach to TEE development for small form factor FPGAs.

4

SOK: TRUSTED EXECUTION IN SOC-FPGAS

Contribution of Authors and Co-Authors

Manuscript in following chapter

Author: Garrett Perkins

Contributions: Developed study concept and design, data collection, analysis and interpretation of results, and wrote the manuscript.

Benjamin Macht

Contributions: Reviewed and presented the manuscript.

Lucas Ritzdorf

Contributions: Reviewed the manuscript.

Tristan Running Crane

Contributions: Reviewed the manuscript.

Brock LaMeres

Contributions: Obtained funding, provided feedback on the analyses and edited the manuscript.

Co-Author: Clemente Izurieta

Contributions: Obtained funding, provided feedback on the analyses and edited the manuscript.

Co-Author: Ann Marie Reinhold

Contributions: Obtained funding, provided feedback on the analyses and edited the manuscript.

Manuscript Information

Garrett Perkins, Benjamin Macht, Lucas Ritzdorf, Tristan Running Crane,

Brock LaMeres, Clemente Izurieta and Ann Marie Reinhold

IEEE i-ETC 2025

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
____ Accepted by a peer-reviewed journal
_X_ Published in a peer-reviewed journal

Publisher: IEEE
In-press

Abstract

Trusted Execution Environments (TEEs) have emerged at the forefront of edge computing to combat the lack of trust between system components. Field Programmable Gate Arrays (FPGAs) are commonly used as edge computers but were not created with security as a primary consideration. Thus, FPGA-based edge computers are increasingly the target of cyberattacks. We analyze the existing literature to systematize the applications and features of FPGA-based TEEs. We identified 27 primary studies related to different types of System-on-Chip FPGA-based TEEs. Across a wide range of applications and features, the availability of extensible solutions is limited. Most solutions focus on specific features and applications, whereas few solutions focus on feature-rich, comprehensive TEEs that can be utilized across computer systems. Whether TEEs are specific or extensible, the paucity of published studies provides evidence of research gaps. This SoK delineates these gaps revealing opportunities for researchers and developers.

Introduction

In the rapidly evolving landscape of the Internet of Things (IoT) and edge computing, the demand for secure environments (SEs) has grown markedly. With the increasing interconnectivity of devices, traditional computer systems are no longer able to rely on mutual trust among components, as a compromise in one area can lead to vulnerabilities in others [1]. This heightened risk has underscored the need for SEs that can adapt to the challenges posed by the evolving domain of secure computing [2–4].

Most major CPU vendors have introduced their own chip-specific Trusted Execution Environments (TEE) solutions. For example, *ARM TrustZone*, *Intel SGX*, and *AMD SEV*, each provide secure computing for their respective hardware. However, these chip-specific TEEs constrain developers to a singular platform creating a unique security challenge [5–7].

New solutions are emerging to address this challenge by providing more modular and flexible secure environments. Consequently, significant R&D efforts are being applied to TEEs and hardware-based solutions. Among these hardware solutions are FPGAs and ASICs. This paper focuses on FPGAs, which are application-agnostic, as opposed to ASICs, which are "application-specific" by definition. FPGAs also provide expanded I/O over ASICs while allowing real-time hardware configuration to support field upgrades. FPGAs are inherently modular and used across several applications, such as radar, Unmanned Aerial Vehicles (UAVs), Industrial Control Systems (ICS), data centers, neural networks, and space avionics [8–10]. These applications require that FPGAs be secure.

We explore how FPGA-based TEEs are currently being used to provide secure computing environments and the specific features that make them suitable for applications in IoT and other computing domains. By highlighting gaps in existing research and solutions that improve FPGA security, our study addresses the following research questions: **RQ1:** What are the applications of FPGA-based TEEs and which features do FPGA-based TEEs employ according to the literature? **RQ2:** What gaps exist in the field of FPGA-based TEEs according to the literature?

Methods

We searched two databases, ACM Digital Library and IEEE Xplore, identifying 109 peer-reviewed papers using the search strings and filters shown in Table 1. After applying inclusion criteria (Table 2), 27 papers remained for full evaluation.

We applied inclusion criteria focused on the convergence of TEEs, FPGAs, and cybersecurity. First, we included only papers that primarily addressed security concerns, excluding those not focused on security. Second, we considered only studies that demonstrated practical implementations or empirical evaluations, thus excluding theoretical papers and literature reviews. Furthermore, our review was limited to papers discussing System-on-Chip

(SoC)-based FPGA environments, excluding those involving non-SoC processors to maintain technological specificity. Last, we prioritized open-source systems, excluding studies reliant on proprietary platforms. This prioritization ensured the studies were universally accessible and modifiable. This meticulous selection process was critical to accurately mapping the landscape of FPGA-based TEEs, identifying their applications, and detailing the specific features they employ, directly addressing our research question.

Of the 109 papers, 75 were from ACM Digital Library, and 34 were from IEEE Xplore. After applying the inclusion criteria listed in Table 2, 31 papers remained: 17 from IEEE Xplore and 14 from ACM Digital Library. Despite meeting the inclusion criteria, four papers were removed from the pool of 31 due to lack of relevance, leaving 27 papers in the study. A stacked bar plot was made based on the number of papers published each year (Figure 1).

After selecting 27 papers, each was read to categorize the features and applications of these custom TEEs. Notable features and applications were separately categorized by paper in Table 3. This table does not include the papers [6, 11], and [12], as [6] and [11] are categorized as extensible TEEs and [12] is an implementation of [6].

We built a heatmap to identify which features are most commonly associated with each application and highlight areas of researcher attention (Figure 2). This aids in visualizing the distribution of features across various applications of FPGA-based TEEs and facilitates clear and immediate understanding of the landscape of FPGA-based TEEs.

Results & Discussion

The increasing rate of publications around TEEs and FPGAs indicates that these are both growing areas of research in the cybersecurity community (Figure 1)[13]. Despite recent growth in publications, research on FPGA-based TEEs remains limited, with only 27 relevant studies identified.

The majority of the 27 papers focus on applications and features. More specifically,

Table 1: Database search details, including search strings, filters, results.

Database	**Search String**	**Filters**	**Results**
ACM Digital Library	["trusted execution environment"] AND [fpga]	Past 5 years, Research Articles Only	75
IEEE Xplore	("All Metadata":"trusted execution environment") AND ("All Metadata":fpga)	2019-2024, Journals/Conferences	34

Table 2: Inclusion criteria and number of papers excluded for each criterion. Total count of excluded papers exceeds the 109 papers obtained from the initial search strings because some papers were excluded for not meeting multiple criteria.

Criteria	**Count of papers excluded**
Security Focused	4
Applied Research	16
Open-source Platform	28
System on Chip Based	63

24 papers address application-specific (15 papers) (Subsection 2) and feature-specific (9 papers)(Subsection 2) TEEs. Only two papers present a holistic approach to TEEs (Subsection 2). One paper presents a use case of a holistic approach. The application-based papers focus on the topics of accelerators, cloud computing, and attack mitigation (Table 3); the rest of the 24 papers are feature-driven. Root of Trust (RoT) and various memory security features are common, while features such as password recovery and upgraded page table walks are less common. Each paper presents a unique combination of applications and features (Figure 2).

Application Specific

Across the 27 selected papers, 15 constructed TEEs that served niche purposes. However, note that some of these "niche papers" developed TEEs that are multi-applicational (i.e., acceleration in cloud computing) but not fully extensible.

Hardware Acceleration Seven papers discuss custom TEEs and their features as they are applied to accelerators and acceleration. TEEs emphasizing hardware acceleration primarily feature memory security, enclaves, RoT, and attestation (Figure 2). *ShEF* implements a unique Shield module for secure data access [14]. Meanwhile, *TACC* separates memory management for in-package (internal) and off-package (external) memory [15]. *AccGuard* separates and isolates memory regions for use in multi-tenant cloud environments, whereas *AccShield* supports unified virtual memory across multiple accelerators, allowing them to securely share memory resources [16]. Paper [17] used a Software-Defined Interconnect block, a hardware block that dynamically controls and sets specific boundaries for memory regions. While secure memory is the most widespread hardware acceleration feature, other features are discussed in the literature.

Papers [14], [15], and [18] differ on cloud-specific use cases, but all take an enclave-based approach to TEEs. Papers [14], [18], and [16] required attestation with a root of trust for verification purposes. Other features were less prevalent across papers focused on hardware acceleration (e.g., Secure Boot, Security Monitor [SM], Key Monitoring, Physical Unclonable Functions) but are still important for securing hardware accelerators. Developers and researchers pursue these different features to secure TEEs focused on hardware acceleration.

Cloud and Remote Computing Papers on hardware acceleration almost always also focus on accelerators in a cloud computing environment (see references in Acceleration and Cloud Computing rows in Table 3). Papers already discussed in Section 2 are re-mentioned

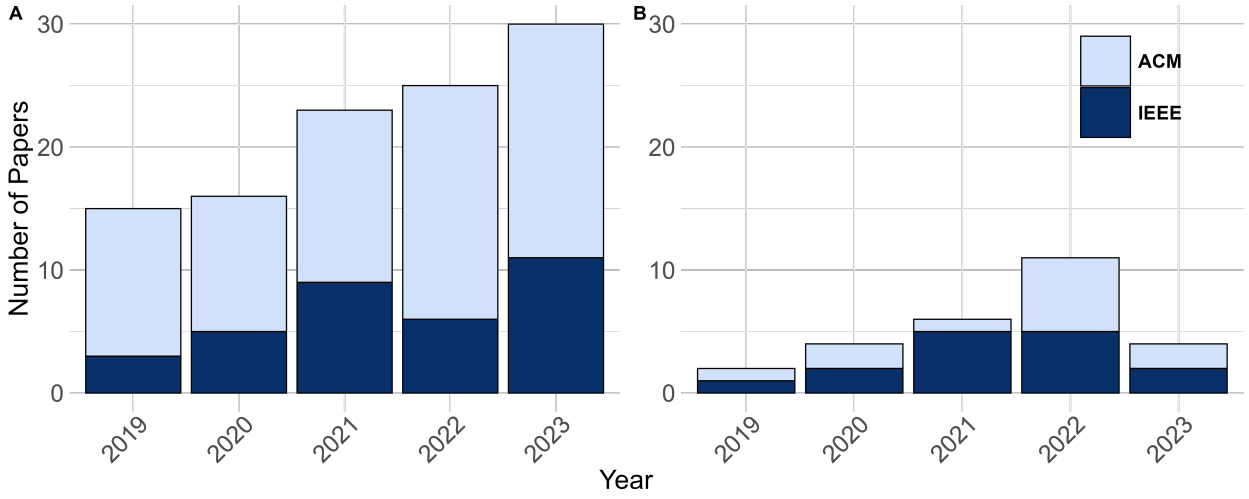

Figure 1: Stacked bar plot representing the number of papers published over the study period. Panel A is the 109 papers found using the search strings in Table 1. Panel B is the 27 papers after application of the inclusion criteria in Table 2. © 2025 IEEE.

but specific features are only discussed again here where relevant. Seven papers discuss custom-designed TEEs that implement security for cloud or remote-based FPGAs. Though applications are numerous for FPGA-based cloud computing, most papers found a need for security in a multi-tenant cloud environment. Key features such as attestation, memory security, enclaves, and RoTs are used to secure cloud environments that house accelerators [19].

Two papers discuss cloud and remote computing independent of hardware acceleration. Papers on *MeetGo* [20] and *Operon* [21] both provide TEEs for cloud and remote computing environments. *MeetGo* is a hardware-centric solution to insider threats in cloud computing. *MeetGo* implements a TEE that operates independently of the host systems architecture, restricting the administrator's access to users' data in the cloud. *MeetGo's* modularity was demonstrated when it was implemented as a cryptocurrency wallet and General-Purpose Graphics Processing Unit [20]. *Operon* [21] aims to provide secure, encrypted database operations while maintaining compatibility with existing SQL applications. Papers [14, 16, 18, 22, 23] also are applied to cloud and remote computing, but have already been discussed

in Section 2.

Attack Mitigation Trusted Execution Environments play a critical role in attack mitigation. Almost one-fifth of the literature focuses on attack-specific mitigation through custom TEE implementation. Side channel attacks (SCAs) are a significant threat to TEEs. *ChaosINTC* [24] and *REHAD* [25] both focus on SCA mitigation, interrupt-based and cache-based, respectively. *ChaosINTC* implements a dynamic interrupt delay mechanism alongside an interrupt handler to protect their TEE [24]. *REHAD* uses reconfigurable hardware to mitigate cached SCAs [25]. While SCAs are a threat to TEEs specifically, TEEs are also used to defend against other threats.

The remaining TEEs discussed in the literature focused on preventing diverse attack vectors. *TrustToken* features isolated execution and trusted user interaction to combat software-based assaults seeking information and unauthorized access [22]. Yet another TEE seeks to combat unauthorized access, specifically through Trojans, by implementing a Hardware Trojan detection, identification, and recovery mechanism [26]. Another attack vector, fault attacks, is mitigated by *SecWalk*, which protects virtual and physical memory against fault attacks[27]. From fault attacks to information leakage, TEEs often provide a first line of defense against bad actors.

IP Licensing Of the papers that do not discuss hardware accelerators, cloud computing, and attack mitigation, there are a few niche applications. Intellectual Property (IP) protection and licensing is a concern for [28] and [22] because of multi-tenant environments. These multi-tenant FPGA environments present new security risks; current solutions necessitate third-party involvement for key-programming and encryption. The aforementioned *TrustToken* [22] only permits trustworthy connections between third-party IP and the rest of the SoC, while Khan et el. [28] propose a Security framework for handling key storage and security monitoring.

Smart Grid Security Smart Grid Security [29] is a niche application that implements a TEE with dual-core isolation and secure boot based on a RoT. The niche applications of IP and grid security advance the field of SoC-FPGA-based TEEs, opening the door to apply TEEs to other computing areas. Applications of TEEs are slowly expanding as demonstrated by the papers centered around hardware accelerators, cloud computing, and attack mitigation.

The application of TEEs across various domains, from hardware acceleration to cloud computing and attack mitigation, showcases their versatility and growing importance in securing modern computing environments. The innovative use of enclaves, attestation, and memory isolation in these environments highlights the challenges associated with maintaining security in dynamic, resource-shared settings. Meanwhile, the application of TEEs in attack mitigation, particularly against SCAs and hardware Trojans, underscores the necessity of security mechanisms that can preempt and neutralize threats.

Although the focus on niche applications like IP licensing and smart grid security may seem specialized, these examples illustrate the broadening scope of TEE deployment. This trend reflects a growing recognition of the need for secure environments across all facets of computing, driving innovation and expansion in TEE capabilities.

Feature Specific

Nine papers focused on feature-specific TEEs. While all application-specific TEEs require a cadre of features, some researchers designed their TEEs with specific features in mind. These researchers put forth new contributions to the features TEEs can provide, however, not all features are implemented in tandem. These nine feature-specific papers focus on RoTs, attestation, memory security, secure boot, key management, and password recovery (Table 3). Some papers focus on a singular feature, while others focus on multiple (Figure 2).

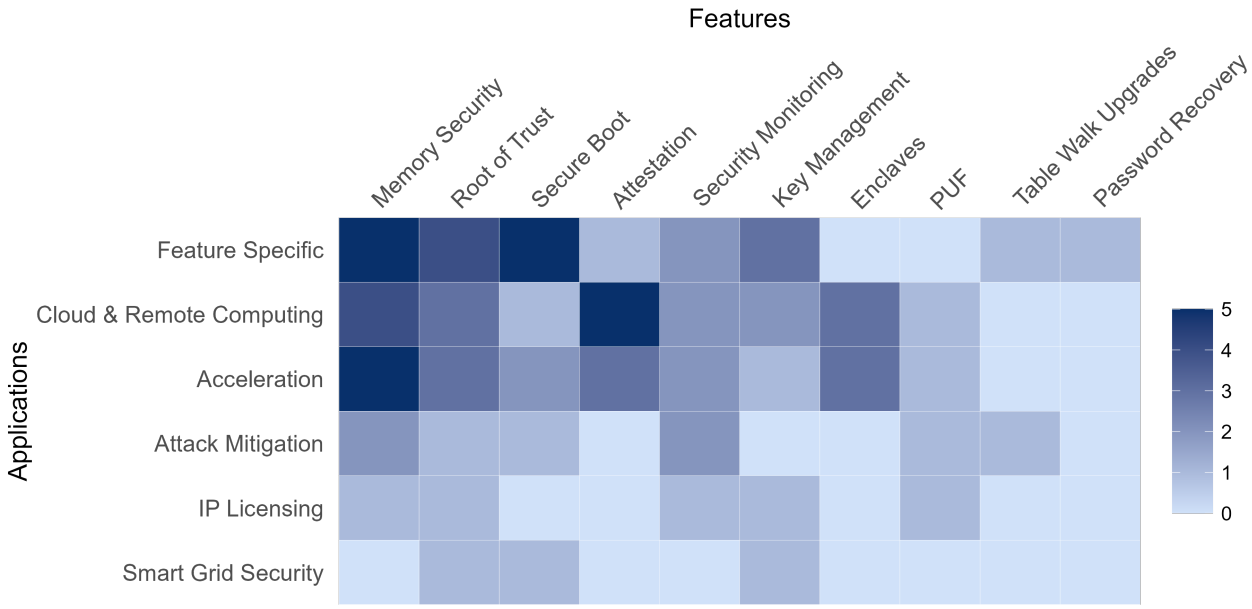

Figure 2: Heatmap of applications and their respective features in the pool of papers. Blue hue denotes number of papers discussing the features and applications indicated on axes.

A RoT is a foundational element in most TEE designs, providing an anchor point for other security features, such as attestation and secure boot. Attestation ensures that the software and hardware components of a system are trustworthy. Mutual attestation allows devices of the same rank and type to verify their mutual interaction; Turan and Verbauwhede [30] use a RoT to facilitate cryptographic verification, network communication, and decision-making, thus providing mutual attestation. Paper [31] employs a RoT, implemented using a Trusted Platform Module (TPM), as the basis for a protection-dedicated core in a multi-core RISC-V system. These feature-specific implementations show how RoTs provide the foundation for critical security features.

Memory security is crucial to the isolation of a TEE. Unrestricted or compromised memory access threatens entire system security. A notable memory-focused paper, *ARES*, implements a security mechanism designed for non-volatile memory (NVM) in embedded systems [32]. [32] aims to combat common memory attacks and issues with Non-Volatile

Table 3: Features and applications of Open-source, SoC-Based Trusted Execution Environments. © 2025 IEEE.

Topic	Paper
Applications	
Hardware Acceleration	[14–18, 22, 23]
Attack Mitigation	[22, 24–27]
Cloud and Remote Computing	[14, 16, 18, 20–23]
Feature Specific	[30–38]
IP Licensing	[22, 28]
Smart Grid Security	[29]
Features	
Attestation	[14, 16, 18, 20, 21, 30]
Enclaves	[14, 15, 18, 21]
Key Management	[18, 21, 28, 29, 33, 34, 37]
Memory Security	[14–17, 20, 22, 27, 32–36]
Page Table Walk Upgrades	[27, 35]
Password Recovery	[38]
Physical Unclonable Function	[22]
Root of Trust	[14, 18, 22, 29–31, 34, 37]
Secure Boot	[14, 15, 24, 29–31, 33, 34, 37]
Security Monitoring	[18, 22, 26, 28, 31, 38]

Memory (NVM) by implementing a novel Bonsai Merkle Tree (BMT) scheme and leveraging parallel recovery in FPGAs. Another NVM-focused TEE, [33], proposes a methodology for securely booting from NVM in insecure environments, leveraging the reconfigurable logic of the FPGA as a secure anchor point. The Trusted Memory-Interface Unit sits in the reconfigurable logic region of the FPGA and performs integrity and authenticity verifications of NVM data prior to executing any user application, ensuring a secure boot process. The focus on NVM-based solutions highlights the importance of secure memory access in ensuring the integrity of data.

Apart from NVM, memory encryption was the focus of a single paper [36]. [36] uses a special memory encryption unit that integrates directly with RISC-V architecture to encrypt memory using the lightweight ChaCha stream cipher which encrypts and decrypts quickly using the add-rotate-XOR (ARX) structure. Paper [36] also utilizes the RISC-

V Physical Memory Protection (PMP) unit to check load/store physical addresses against access restrictions. A spin-off of PMP presented by [35], Hybrid Physical Memory Protection (HPMP), blends segment-based memory protection with a permission table, combining the strengths of both approaches. This hardware-software co-design dynamically manages memory protection and allocates segments and permission tables. These memory security approaches highlight the essential role of protecting memory in ensuring TEE security and integrity.

Secure boot is a critical feature in TEEs, ensuring that the system starts in a trusted state by verifying the authenticity and integrity of the bootloader and other essential components. The aforementioned [33] securely boots from NVM where the boot image is decrypted using the dynamically generated encryption key, and its integrity is verified by comparing the calculated hash against the stored token. Uniquely, [37] focuses on mitigating the threat of quantum computers on TEEs by implementing Secure Boot. The authors implement post-quantum secure boot using the eXtended Merkle Signature Scheme (XMSS) to protect the system's boot process from quantum computing attacks that could compromise traditional asymmetric cryptographic algorithms. This establishes a secure boot chain-of-trust from the RoT up to the operating system kernel ensuring the integrity of each boot stage [37].

In conjunction with secure boot, proper key management is essential to the security of TEE environments. Paper [34] proposes a novel approach to key management within the TEE by utilizing a flexible and secure boot procedure, complete isolation from the TEE domain, and exclusive secure storage for root keys. This ensures enhanced security and flexibility in key generation and maintenance.

A few papers focus on less mainstream features such as password recovery and page table walk upgrades. [38] implement a RISC-V processor, a secure coprocessor, and a password recovery engine connected through an AXI bus. The secure coprocessor includes

an instruction set architecture (ISA) monitor and secure cache for secure computing tasks, especially those involving sensitive data like passwords [38].

The diverse range of features explored across the literature highlights the components necessary for the deployment of TEEs in various computing contexts. The emphasis on foundational elements like RoTs and secure boot mechanisms underscores their role as the bedrock of secure system initialization and operation. These features establish and maintain trust, especially in environments where the integrity of both hardware and software must be assured.

Memory security, with its various implementations, is particularly crucial given the pervasive risk of unauthorized access or data breaches that could compromise the entire TEE. However, the focus on specific features like password recovery and page table walk upgrades, though less common, reflects the growing complexity and specialization of TEE functionalities as they are adapted to meet the needs of increasingly diverse and demanding applications. This progression suggests that future research will push what TEEs can achieve.

Extensible TEEs

The *HECTOR-V* and *Keystone* approaches provide modular and well-rounded TEEs, enabling users to plug and play rather than mix and match features and applications [6, 11].

HECTOR-V, concerns itself with side-channel attacks, arguing that, "TEEs, such as *Intel SGX* or *ARM TrustZone*, implemented on the main application processor, are insecure" [11]. Focusing on combating SCAs, these authors implement a heterogeneous multicore architecture that embeds a dedicated processor into the system to separate the secure and non-secure domains. Their RISC-V Secure Co-Processor (RVSCP) restricts I/O access and provides control-flow integrity (CFI) for secure applications. This TEE provides secure I/O using identifier-based secure communication channels between different devices in the system, which ensures that only authorized entities can access sensitive peripherals. The RVSCP

processor employs hardware-enforced CFI to safeguard applications running in *HECTOR-V* using a specialized hardware unit to monitor the control flow of applications. Overall, *HECTOR-V* aims to provide a secure architecture for trusted execution by combining a heterogeneous CPU architecture with secure coprocessor features, hardware control-flow integrity, and secure communication channels.

Lee et al. made a significant contribution to the TEE landscape when they created *Keystone*, "the first open-source framework for building customized TEEs" [6]. *Keystone* provides a comprehensive framework for implementing a modular TEE on an FPGA using RISC-V architecture. *Keystone* TEEs use enclaves and PMP to isolate different computing modes from accessing data. While memory security is critical, it is not the only feature *Keystone* TEEs provide. *Keystone* TEEs also provide a configurable security monitor (SM) that adds a trusted layer below the OS that can be configured to enforce TEE guarantees (e.g., policies and security primitives). In addition to the SM, the secure boot and attestation capabilities measure and verify the integrity of the SM and enclaves. The myriad features are accompanied by SCA mitigation as *Keystone* TEEs incorporate cache partitioning and other techniques to defend against side-channel attacks. In sum, Lee et al.'s comprehensive, open-source approach allows developers to have modularity and freedom when implementing and modifying a TEE created using the *Keystone* framework.

While both [11] and [6] present similar frameworks for TEEs, only one has been validated. The *Keystone* framework, implemented by [12], served as the architecture for a trusted IoT sensing system. The sensing system features *Keystone* and employs two types of Physically Unclonable Functions (PUFs)–one for the main device and one for the subordinate sensor. In this application, *Keystone* provides isolation from potentially untrusted operating systems and applications using its enclave system. The *Keystone* TEE integrates with a PUF, which serves as a hardware RoT that generates a unique, device-specific key for secure key management. This implementation of *Keystone* illustrates how its modularity and feature-

rich build allow multi-application realization.

Keystone and *HECTOR-V* are easily adaptable to any chip using the RISC-V instruction set, though not without foibles. *Keystone*, while highly modular and customizable, heavily relies on specific RISC-V hardware features, i.e. PMP. Physical Memory Protection also limits the number of memory regions that can be protected based on PMP entries. *HECTOR-V's* multicore architecture is complex to design and implement, particularly regarding the two communication between the cores. Along with the complex design, the hardware architecture and required resources of *HECTOR-V* could limit its adaptability. Though these two TEEs use non-chip-specific features that can be implemented across FPGA vendors, there are still some constraints when it comes to these frameworks.

Additionally, despite these advancements, several critical limitations persist that must be addressed. Performance overhead, particularly in memory encryption and secure boot processes, can slow down system operations, making TEEs less viable for resource-constrained environments like IoT devices and embedded systems, where efficiency is critical. Integration complexity, especially in heterogeneous architectures, complicates the seamless coordination between secure and non-secure domains, risking potential security gaps or performance bottlenecks. Additionally, while TEEs are designed to protect against many known threats, they remain vulnerable to emerging challenges such as quantum computing and advanced side-channel attacks. These limitations are crucial because they not only constrain the current utility of TEEs but also underscore the urgent need for ongoing research to develop more efficient, adaptable, and resilient security solutions.

Threats to Validity

We examine three potential threats to validity based on the classification scheme of [39] and [40].

Construct validity refers to how well the study identifies and categorizes TEEs. The

search strings may have failed to capture relevant papers. This threat was mitigated by checking references of the included papers for potential oversights. Another threat is the manual categorization of papers (e.g., application-specific or feature-specific TEEs). This relies on subjective judgment. To mitigate this, possible features and applications were reviewed and rechecked.

Content validity may be affected in two ways. First, if the inclusion criteria used to select the final 27 papers were too restrictive, this would result in excluding papers that offer theoretical frameworks or nascent areas of research. This threat was minimized by reading the abstracts of all 109 papers to ensure no relevant studies were excluded. Second, only IEEE or ACM were searched, possibly excluding relevant papers published elsewhere. This is not a significant threat because IEEE and ACM conference proceedings and journals are the primary outlets for publications on edge-computing security.

External validity relates to the ability to generalize the findings of this study. We do not perceive significant threats to the external validity of this study. The scope of our study is on SoC-FPGA TEEs. Within this scope, our research captures the state of the published research. However, extrapolating or generalizing findings beyond this scope to the broader landscape of edge computing is not advised.

Conclusion

This study systemizes SoC-FPGA TEEs, highlighting research gaps. Through the analysis of 109 papers sourced from IEEE Xplore and ACM Digital Library, a pool of 27 papers represented the current state of SoC-FPGA-based TEEs. These papers demonstrated the research challenges of implementing a robust, multi-featured, multi-application TEE, illustrated by the emphasis on application and feature-based TEEs. A robust, modular approach emerged in two papers combining critical features for a non-application-specific approach. The lack of publications related to SoC-FPGA-based TEEs that do not rely on

third-party technology reveals a gap in the literature and an opportunity for researchers and developers (Figure 1). Many papers emphasize specific applications or features, but few combine features to create extensible TEEs. These insights hold significance for future development of TEEs and emphasize the importance of secure computing across applications and platforms.

Acknowledgments

NASA and Resilient Computing, LLC supported this research under award number 80NSSC23CA147, and subcontract number 4W9082, respectively. This research was conducted with the U.S. Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under contract 70RSAT24KPM000022. Any opinions contained herein are those of the authors and do not necessarily reflect those of NASA, Resilient Computing, LLC or DHS S&T. Thank you to Yvette Hastings at MSU for assisting with Figs. 1 and 2.

Bibliography

[1] Pearson *et al.*, "Privacy, security and trust issues arising from cloud computing," in *2010 IEEE Second International Conf on Cloud Computing Technology and Science*, 2010, pp. 693–702.

[2] Xiao *et al.*, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[3] Zeyu *et al.*, "Survey on edge computing security," in *2020 International Conf on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2020, pp. 96–105.

[4] Zhang *et al.*, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[5] D. Kaplan, "Protecting vm register state with sev-es," Advanced Micro Devices, Inc., February 2017, white Paper. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf

[6] Lee *et al.*, "Keystone: an open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conf on Computer Systems*, ser. EuroSys '20. New York, NY, USA: ACM, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387532

[7] 2024. [Online]. Available: https://developer.arm.com/documentation/PRD29-GENC-009492/latest/

[8] Arm, "What is fpga?" [Online]. Available: https://www.arm.com/glossary/fpga

[9] J. Schneider and I. Smalley, "What is a field programmable gate array (fpga)?" May 2024. [Online]. Available: https://www.ibm.com/think/topics/field-programmable-gate-arrays

[10] [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/fpga-vs-structured-asic.html

[11] Nasahl *et al.*, "Hector-v: A heterogeneous cpu architecture for a secure risc-v execution environment," in *Proceedings of the 2021 ACM Asia Conf on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: ACM, 2021, p. 187–199. [Online]. Available: https://doi.org/10.1145/3433210.3453112

[12] Yoshida *et al.*, "Towards trusted iot sensing systems: Implementing puf as secure key generator for root of trust and message authentication code," in *Proceedings of the 10th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '21. New York, NY, USA: ACM, 2022. [Online]. Available: https://doi.org/10.1145/3505253.3505258

[13] B. A. Kitchenham, *et al.*, "Using mapping studies as the basis for further research–a participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638–651, 2011.

[14] Zhao *et al.*, "Shef: Shielded enclaves for cloud fpgas," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1070–1085.

[15] Zhu *et al.*, "Tacc: a secure accelerator enclave for ai workloads," in *Proceedings of the 15th ACM International Conf on Systems and Storage*, ser. SYSTOR '22. New York, NY, USA: ACM, 2022, p. 58–71. [Online]. Available: https://doi.org/10.1145/3534056.3534943

[16] Ren *et al.*, "Accshield: a new trusted execution environment with machine-learning accelerators," in *2023 60th ACM/IEEE DAC*, 2023, pp. 1–6.

[17] Kolimbianakis *et al.*, "Software-defined hardware-assisted isolation for trusted next-generation iot systems," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '22. New York, NY, USA: ACM, 2022, p. 139–146. [Online]. Available: https://doi.org/10.1145/3477314.3508378

[18] Ren *et al.*, "Accguard: Secure and trusted computation on remote fpga accelerators," in *2021 IEEE iSES*, 2021, pp. 378–383.

[19] Jasti *et al.*, "Security in multi-tenancy cloud," in *44th Annual 2010 IEEE International Carnahan Conf on Security Technology*, 2010, pp. 35–41.

[20] Oh *et al.*, "Meetgo: A trusted execution environment for remote applications on fpga," *IEEE Access*, vol. 9, pp. 51 313–51 324, 2021.

[21] Wang *et al.*, "Operon: an encrypted database for ownership-preserving data management," *Proc. VLDB Endow.*, vol. 15, no. 12, p. 3332–3345, Aug. 2022. [Online]. Available: https://doi.org/10.14778/3554821.3554826

[22] Ahmed *et al.*, "Trusted ip solution in multi-tenant cloud fpga platform," in *2022 IEEE 8th WF-IoT*, 2022, pp. 1–6.

[23] Ince *et al.*, "Token-based authentication and access delegation for hw-accelerated telco cloud solution," in *2022 IEEE 11th International Conf on CloudNet*, 2022, pp. 109–117.

[24] Zhu *et al.*, "Chaosintc: A secure interrupt management mechanism against interrupt-based attacks on tee," in *2023 60th ACM/IEEE DAC*, 2023, pp. 1–6.

[25] Mao *et al.*, "Rehad: Using low-frequency reconfigurable hardware for cache side-channel attacks detection," in *2020 IEEE EuroS&PW*, pp. 704–709.

[26] Malekpour *et al.*, "Hardware trojan detection and recovery in mpsocs via on-line application specific testing," in *2019 IEEE 22nd International Symposium on DDECS*, 2019, pp. 1–6.

[27] Schilling *et al.*, "Secwalk: Protecting page table walks against fault attacks," in *2021 IEEE International Symposium HOST*, pp. 56–67.

[28] Khan *et al.*, "Utilizing and extending trusted execution environment in heterogeneous socs for a pay-per-device ip licensing scheme," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2548–2563, 2021.

[29] Chen *et al.*, "A risc-v system-on-chip based on dual-core isolation for smart grid security," in *2022 IEEE 6th Conf EI2*, 2022, pp. 1278–1283.

[30] Turan *et al.*, "Propagating trusted execution through mutual attestation," in *Proceedings of the 4th Workshop on System Software for Trusted Execution*, ser. SysTEX '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3342559.3365334

[31] Stoyanov *et al.*, "Secure heterogeneous architecture based on risc-v and root-of-trust," in *Proceedings of the 24th International Conf on Computer Systems and Technologies*, ser. CompSysTech '23. New York, NY, USA: ACM, 2023, p. 19–23. [Online]. Available: https://doi.org/10.1145/3606305.3606312

[32] Zou *et al.*, "Ares: Persistently secure non-volatile memory with processor-transparent and hardware-friendly integrity verification and metadata recovery," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 1, feb 2022. [Online]. Available: https://doi.org/10.1145/3492735

[33] Streit *et al.*, "Secure boot from non-volatile memory for programmable soc architectures," in *2020 IEEE International Symposium HOST*, 2020, pp. 102–110.

[34] Hoang *et al.*, "Trusted execution environment hardware by isolated heterogeneous architecture for key scheduling," *IEEE Access*, vol. 10, pp. 46 014–46 027, 2022.

[35] Du *et al.*, "Accelerating extra dimensional page walks for confidential computing," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: ACM, 2023, p. 654–669. [Online]. Available: https://doi.org/10.1145/3613424.3614293

[36] Cilardo *et al.*, "Memory encryption support for an fpga-based risc-v implementation," in *2021 16th International Conf on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1–5.

[37] Kumar *et al.*, "Post-quantum secure boot," in *Proceedings of the 23rd Conf on Design, Automation and Test in Europe*, ser. DATE '20. San Jose, CA, USA: EDA Consortium, 2020, p. 1582–1585.

[38] Xi *et al.*, "A heterogeneous risc-v soc for confidential computing and password recovery," in *2022 7th International Conf on Integrated Circuits and Microsystems (ICICM)*, 2022, pp. 500–504.

[39] Cook *et al.*, *Quasi-experimentation: Design & Analysis Issues for Field Settings*. Houghton Mifflin, 1979. [Online]. Available: https://books.google.com/books?id=BFNqAAAAMAAJ

[40] Campbell *et al.*, *Experimental and Quasi-experimental Designs for Research*. R. McNally, 1966. [Online]. Available: https://books.google.com/books?id=kFtqAAAAMAAJ

CYBERSHIELD: SECURE BOOT FOR OBFUSCATED INSTRUCTION CODES

Contribution of Authors and Co-Authors

Manuscript in following chapter

Author: Garrett Perkins

Contributions: Developed the TEE and Cybershild, data collection, and wrote the manuscript.

Co-Author: Hezikaih Austin

Contributions: Assisted in the research and development of CyberShield.

Co-Author: Tristan Running Crane

Contributions: Assisted in the research and development of CyberShield.

Co-Author: Chris Major

Contributions: Assisted in the research and development of CyberShield.

Co-Author: Clemente Izurieta

Contributions: Obtained funding, provided feedback during the design process and reviewed the manuscript.

Co-Author: Brock LaMeres

Contributions: Obtained funding, provided feedback during the design process and reviewed the manuscript.

Manuscript Information

Garrett R. Perkins, Hezekiah A. Austin, Tristan Running Crane, Chris Major, Clemente Izurieta, and Brock LaMeres

IEEE International Conference on Space Mission Challenges for Information Technology / Space Computing Conference (SMC-IT/SCC 2025)

Status of Manuscript:
____ Prepared for submission to a peer-reviewed journal
____ Officially submitted to a peer-reviewed journal
_X_ Accepted by a peer-reviewed journal
____ Published in a peer-reviewed journal

Publisher: IEEE
Submission Date: April 4nd 2024

Abstract

The increasing reliance on Field Programmable Gate Arrays (FPGAs) in security-critical applications underscores the need for robust protection mechanisms against cyber threats such as buffer overflow and injection attacks. This paper presents Cybershield, a novel integration of a Trusted Execution Environment (TEE) within RadPC, a radiation-tolerant softcore processor featuring Quad Modular Redundancy (QMR). Leveraging RadPC's four-core architecture, Cybershield's TEE employs secure boot from non-volatile memory to initialize RadPC's cores with obfuscated instruction codes. The opcode obfuscation and a hardware-based anti-voter mechanism prevent the execution of unauthorized code and detect Indicators of Compromise (IoCs). By implementing secure boot and opcode obfuscation, Cybershield mitigates common attack vectors while maintaining software redundancy and recoverability. Experimental validation demonstrates the system's ability to detect buffer overflow attacks and prevent unauthorized code execution. While the integration of a TEE introduces computational overhead and development constraints due to RadPC's bare-metal environment, this work lays the foundation for combining hardware and software redundancy to enhance the security of embedded systems.

Introduction

The rapid growth of the Internet of Things (IoT) and edge computing has fundamentally transformed the way data is processed and analyzed. Unlike traditional cloud computing, edge computing shifts data processing closer to the source, enabling real-time insights, reduced latency, and minimized bandwidth usage. This paradigm shift has fueled advancements in a wide range of industries, from healthcare and automotive systems to industrial automation and telecommunications. However, as these edge devices become increasingly interconnected and intelligent, their attack surfaces expand, exposing them to

many cybersecurity threats.

At the heart of many edge devices are Field Programmable Gate Arrays (FPGAs), which offer unparalleled flexibility, reconfigurability, and performance[**?**]. FPGAs combine programmable logic with embedded processors and have become particularly attractive for edge computing due to their ability to handle complex workloads such as artificial intelligence (AI), signal processing, and real-time control. Despite their advantages, FPGAs were not originally designed with robust security in mind. As they transition from specialized applications to widespread use in critical industries including but not limited to radar, Unmanned Aerial Vehicles (UAVs), Industrial Control Systems (ICS), data centers, neural networks, and space avionics, the potential for malicious attacks targeting both their hardware and software components has grown significantly[1, 2].

This increasing reliance on FPGAs in security-critical applications highlights the urgent need to address their vulnerabilities. From intellectual property theft to fault injection and side-channel attacks, FPGAs face a diverse array of threats[**?** **?**]. To ensure the reliability and security of edge computing systems, new mechanisms must be developed to protect the hardware, software, and communication protocols of FPGAs.

Trusted Execution Environments (TEEs) offer a promising solution to enhance the security of FPGAs. TEEs come in many forms with many features. Most major CPU vendors have introduced their own chip-specific TEEs. These TEEs, i.e., *ARM TrustZone*[1][2], *Intel SGX*[3], and *AMD SEV*[4] leverage hardware-based security features to protect against vulnerabilities that traditional process isolation methods cannot address [3]. By isolating sensitive operations and providing secure boot mechanisms, TEEs can protect edge devices

[1]https://www.arm.com/technologies/trustzone-for-cortex-a

[2]https://www.arm.com/technologies/trustzone-for-cortex-m

[3]https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

[4]https://www.amd.com/en/developer/sev.html

from tampering and unauthorized access. This paper presents the novel design and integration of a TEE with a radiation-tolerant softcore processor, RadPC, implemented on a commercial FPGA platform. This integration is known as CyberShield.

Related Works

Trust Execution Environments specific to FPGAs come in many different forms. Some TEEs host myriad features and have more general applications, whereas others are use-specific and intended for a single purpose [**?**].

Feature-rich TEEs aim to provide comprehensive security solutions by integrating elements such as memory encryption, secure boot, attestation, and root of trust, making them suitable for multi-tenant environments and dynamic workloads [**?**]. Conversely, application-specific TEEs are tailored to address distinct security challenges, such as mitigating side-channel attacks or securing intellectual property within FPGA deployments [**?**].

Secure boot is a fundamental security feature in TEEs, ensuring that the system initializes in a trusted state by verifying the authenticity and integrity of critical components. Several works have explored different aspects of secure boot in TEEs within FPGAs.

Recent advancements in secure boot architectures have explored the integration of hardware-based security measures to improve the efficiency and robustness of firmware verification. Loo et al. [4] propose a secure boot implementation for RISC-V using an FPGA-based approach. Their method introduces a hardware security block at the Register Transfer Level (RTL) to generate a SHA-512 digest, significantly reducing the computational burden on software-based secure boot mechanisms. The study highlights that while secure firmware incurs a 35% increase in boot time and a 3.3 MB increase in binary size, the FPGA implementation offsets performance costs by accelerating cryptographic computations, achieving an 1132% improvement in execution time over software-based

hashing.

Another advancement in secure boot architectures is CARE, a lightweight attack-resilient secure boot framework designed for RISC-V-based SoCs [5]. Unlike traditional secure boot mechanisms that focus solely on detecting malware presence, CARE introduces an onboard recovery mechanism that ensures compromised devices can autonomously restore their firmware to a trusted state. The framework integrates a Code Authentication and Resilience Engine (CARE) that verifies firmware integrity using a hardware-accelerated HMAC-SHA256 cryptographic core. In the event of a detected compromise, CARE employs a dedicated recovery engine to re-flash only the corrupted memory regions from a secure backup ROM, preventing unauthorized modifications and minimizing system downtime. By leveraging Physical Memory Protection (PMP) and secureIbex features of the RISC-V processor, CARE mitigates fault injection and side-channel attacks while maintaining a minimal 8% performance overhead.

One of the most notable RISC-v-based TEEs, Keystone, incorporates secure boot as a critical security primitive, ensuring that only authenticated and unmodified software is executed within its TEE [6]. The secure boot process in Keystone begins with a hardware-based root of trust, where a tamper-proof bootloader or cryptographic engine measures the integrity of the Security Monitor (SM) at system reset. This process generates a fresh attestation key, which is securely stored in the SM's isolated memory and signed using a hardware-visible secret. During boot, Keystone verifies the integrity of enclave code and runtime components before execution, preventing unauthorized modifications or tampering. By leveraging RISC-V's Physical Memory Protection (PMP) and a minimal trusted computing base (TCB), Keystone ensures strong memory isolation and attestation capabilities, making it adaptable for various deployment scenarios, from cloud environments to embedded systems.

Another notable approach by Streit et al. focuses on securely booting from non-volatile

memory (NVM) in insecure environments by leveraging the reconfigurable logic of an FPGA as a secure anchor point. The proposed methodology integrates a Trusted Memory-Interface Unit within the FPGA's reconfigurable logic region, enabling integrity and authenticity verification of NVM data before executing any user applications. The boot image is decrypted using the dynamically generated encryption key, and its integrity is verified by comparing the calculated hash against the stored token [7].

Background

RadPC

The softcore processor RadPC was chosen for integration with a TEE. RadPC is a custom radiation-tolerant space computer designed for small satellites with a reduced instruction set (RISC-V 32I). The radiation tolerance of RadPC specifically relates to Single Event Effects (SEEs), which are strikes that cause inadvertent switching in logic circuits. To mitigate single-event effects, RadPC uses an architectural approach in which the detection and recovery of SEEs is abstracted from software developers. RadPC is developed in VHSIC Hardware Description Language (VHDL) and implemented as a logic design. RadPC expands upon the traditionally used Triple Modular Redundancy (TMR) to Quad Modular Redundancy (QMR) to withstand two SEEs simultaneously (Figure 3). RadPC is a QMR microcontroller implemented on an FPGA with a RISC-V instruction set, and implements majority voting as a fault mitigation strategy, which requires a minimum of three cores to achieve a majority [8, 9]. However, RadPC researchers discovered that by including a fourth core, they gain additional capabilities: specifically, they have enough time to perform a partial reconfiguration (PR) of a faulted core, enabling a more comprehensive repair procedure beyond a simple reset. Because PR takes longer than a register reload, the fourth core "buys time" during recovery. It also has the added benefit of allowing a second fault to occur while still maintaining a majority vote. When a fault is detected in

one of RadPC's CPUs, its respective registers are reloaded with the correct values using the majority voter. In the event of an unrecoverable fault using the voting approach, a full CPU can be partially reconfigured and registers reloaded to fully recover the system [10, 11]. These two recovery methods, complemented by memory scrubbers, provide resilience from SEEs [8, 10–12].

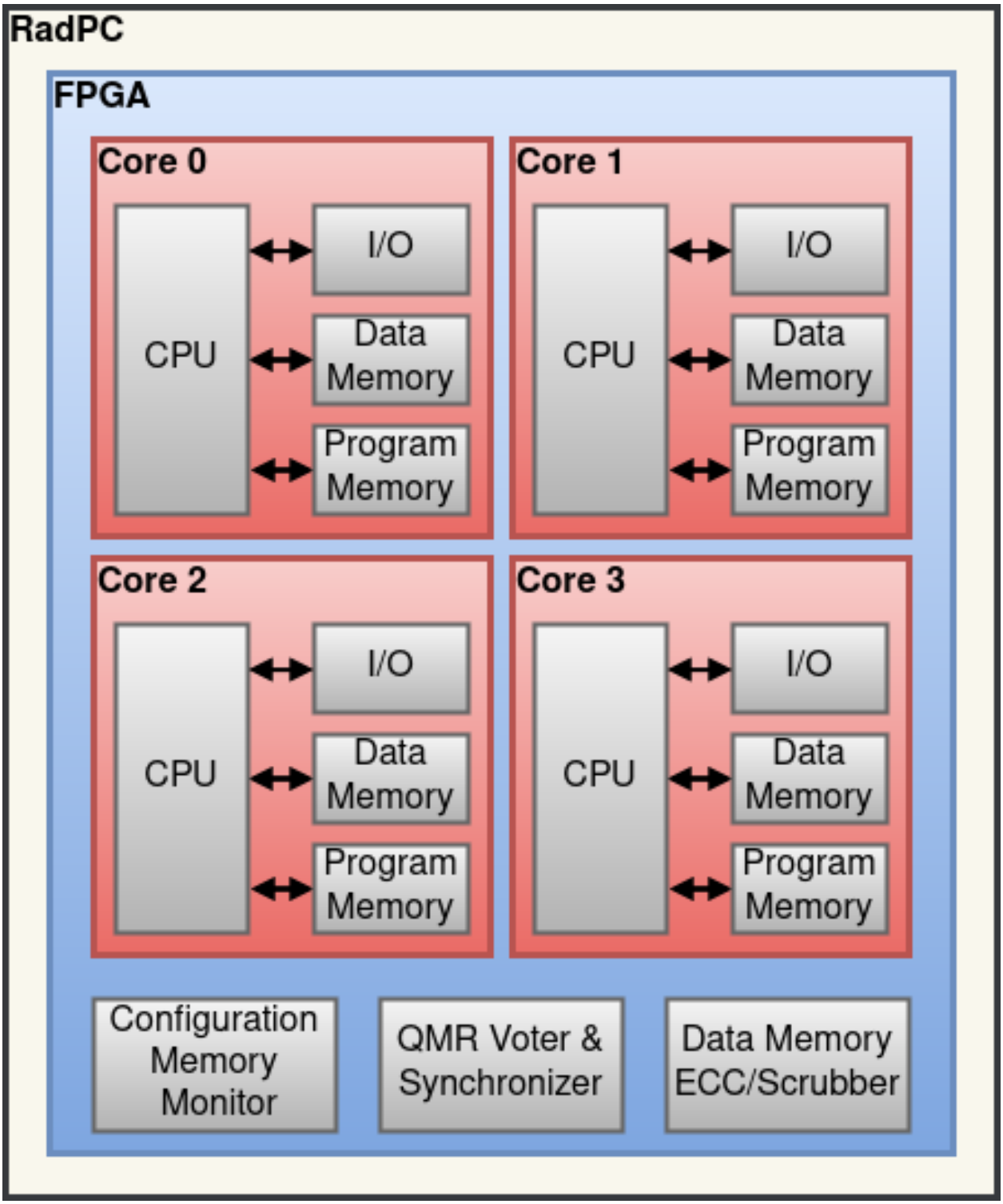

Figure 3: Block diagram of the QMR architecture of the RadPC FPGA system [12].

Cybershield

RadPC's architectural QMR approach that resists and recovers from radiation-induced faults was extended into Cybershield, enhancing its resilience against malware attacks, particularly buffer overflow exploits (Figure 4. Cybershield employs instruction code obfuscation by assigning unique opcode offsets to each core and updating the software binaries accordingly, preventing uniform opcode execution and adding resistance to injection attacks [13]. This approach leverages the flexibility of FPGA-based implementations and RISC-V architecture to ensure that, even if a vulnerability is exploited, opcode discrepancies among cores trigger immediate detection and recovery mechanisms. The Cybershield microcontroller runs multiple obfuscated cores in parallel, with an anti-voter module monitoring instruction registers to detect any anomaly caused by malware injection[14]. Malware that is injected into CyberShield will be replicated across each core and upon execution each core will see the same Opcode. This by design is impossible and will be flagged as an anomaly. While CyberShield is implemented on RadPCs' four cores, the obfuscation defense mechanism in CyberShield itself only requires two cores to work effectively; detecting matching opcodes across two different cores is sufficient to trigger an anomaly flag. This architectural innovation not only increases security through hardware-level diversity but also provides an additional defense layer by making unauthorized code execution significantly more difficult. By integrating RadPC's proven reconfiguration capabilities with advanced obfuscation techniques, Cybershield offers a robust solution for securing embedded systems in critical applications.

In the original proof of concept of CyberShield, the obfuscated instruction decoders and associated software binaries were embedded in the VHDL description of the system before implementation. This posed a problem because the system could not be bootloaded like a traditional microcontroller. Instead, changing the software required a new bitstream to be generated. This paper proposes a TEE that provides secure boot with obfuscated opcodes for

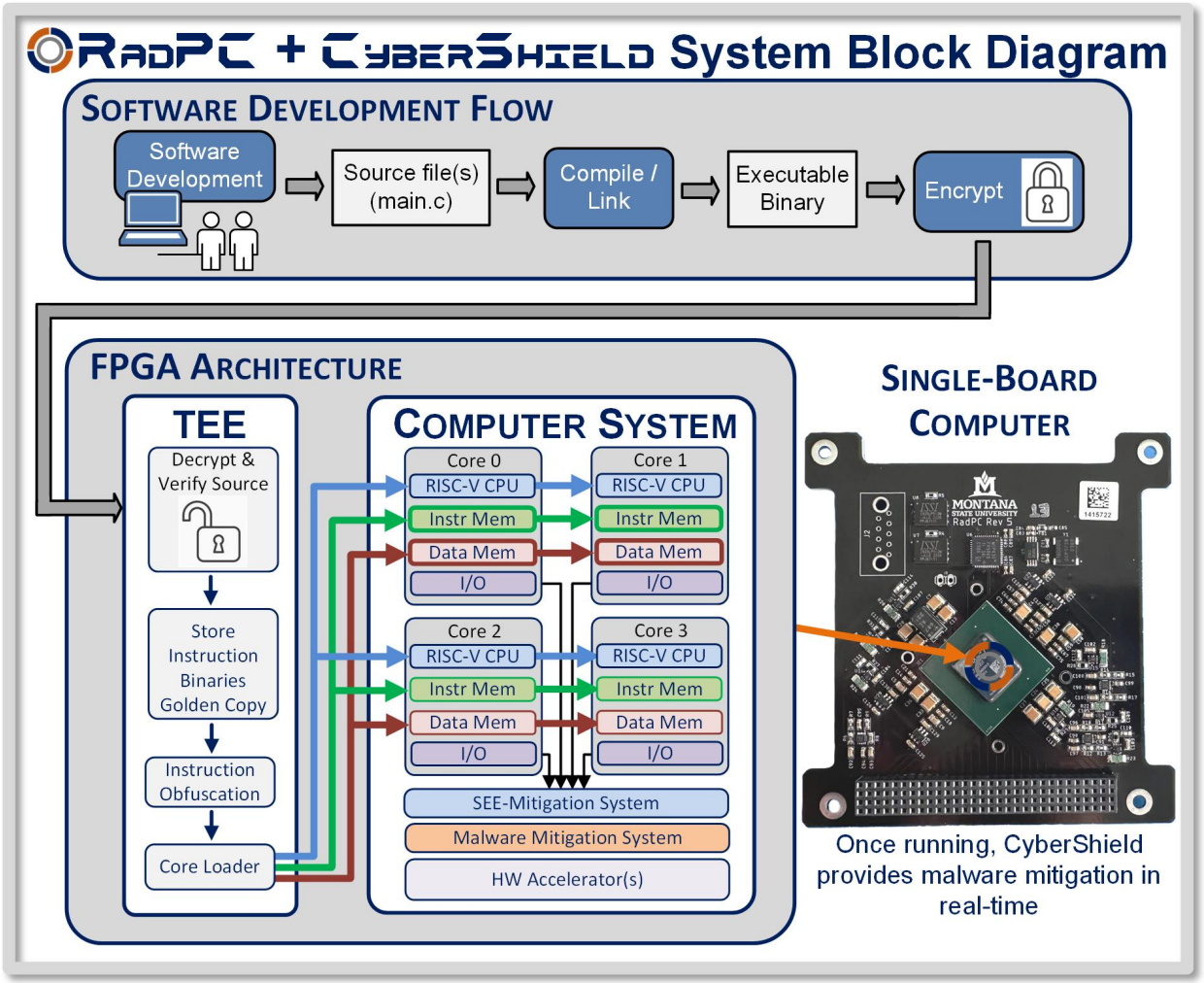

Figure 4: System block diagram of RadPC + TEE i.e. Cybershield. Illustrates the software development flow, FPGA architecture, and the single-board computer.

CyberShield, bootloading it like a traditional microcontroller. These architectural changes to the original version of CyberShield enhance both security and reliability by addressing the bootloading challenges of the original version.

Design

The first modification to the CyberShield proof-of-concept was to modify the system so that it could have software binaries loaded into its instruction memory (IMEM) (e.g.,

bootloading) over a UART serial port. This allows the system to accept executables in real time. Based on this work, a TEE that bootloaded CyberShield was developed that could be implemented on the same FPGA. The TEE needed four key functionalities. First, to be able to accept an encrypted executable via a UART between CyberShield and an external computer. Second, to read and write the encrypted executable to the onboard NVM serial flash chip. Third, to be able to bootload CyberShield over an internal UART on the FPGA between the TEE and CyberShield's four cores. Last, the TEE needed to be able to obfuscate the software binaries before bootloading CyberShield.

The TEE was integrated into the CyberShield architecture as a separate softcore processor and deployed on a Nexys A7-100T development board. This board utilizes an Artix-7 Xilinx FPGA. The TEE and CyberShield are "wrapped" at the top level but are separate entities within the VHDL hierarchy. The architecture of the TEE is similar to the cores of CyberShield but with limited functionality and only necessary peripherals, i.e., SPI, UART, and GPIO. The other signals are tied to `0`, `(others => '0')`, or `open` for security. Both the TEE and CyberShield have been modified to include five UART peripherals. These UARTs are used by the TEE to bootload each individual CyberShield core instead of each core receiving the same executable.

The executable for the TEE is compiled using GCC on the developer's computer then the TEE is bootloaded over a UART. Once the TEE has been bootloaded it is ready to receive the encrypted executable for CyberShield. The executable for CyberShield is first compiled using GCC and then encrypted using a one-time pad. The executable is then ready to be sent in one-byte blocks over UART to the TEE.

Writing the executable to Non-volatile Memory

In order for the TEE to accept the executable for CyberShield a circular buffer was implemented. The circular buffer was implemented due to memory constraints. The TEE

and CyberShield share IMEM and DMEM which are both 12.288 kB. The test executable for CyberShield was 24.308 kB, making it too large for the TEE to accept in one block. Using a terminal, the developer can send blocks of up to 11520 bytes to the TEE to then be written over SPI to the NVM. The user specifies the size of the incoming block to the TEE, and then using a Python script sends the specified number of bytes over UART to the TEE at a Baudrate of 115200 with a one millisecond delay between bytes. The TEE stores the block of the executable in a byte array. Once the block of the executable has been accepted into the TEE it is ready to write to NVM.

The TEE communicates with the NVM via a Serial Peripheral Interface (SPI). The SPI peripheral on the TEE operates in 3-pin mode at 50 kHz, with the Chip Select (CS) line being manually toggled. Once a block of the executable is in the buffer, the developer can use the terminal to tell the TEE to write that block to the NVM. The TEE software keeps track of the number of bytes written along with the addresses to which that data was written. The data is written in 256-byte pages, and after each page, the TEE must toggle CS and reestablish communication with the NVM. Once the program is done writing to NVM, the circular buffer is reset and ready to receive another block of the executable. This process is repeated until the entire encrypted executable has been stored. Once this process is complete, the executable will remain in memory until it is cleared. The TEE is now ready to read from NVM and bootload CyberShield.

Booting CyberShield with an Encrypted Executable

Booting CyberShield Over One UART Without Offset (Figure 5) The first step in testing the system was to verify that the TEE could bootload the CyberShield QMR system without any instruction code obfuscation. To bootload CyberShield from NVM, the TEE requires two key components: the one-time pad decryption key and the instruction code offsets specific to CyberShield. Both of these have been manually programmed into the TEE

software. The CyberShield cores' bitstreams are generated with their respective offsets when the executable is compiled.

To successfully bootload CyberShield the TEE must communicate using two different communication protocols: SPI and UART. When the executable was generated each 32-bit (four bytes) instruction was broken into four, single-byte pieces. The TEE reads the encrypted executable one byte at a time from the NVM and decrypts it in real-time. Initially, this test was performed using a single UART interface to transmit instructions to the CyberShield system without any instruction offsets.

Booting CyberShield Over Four UARTs Without Offset (Figure 6) After verifying that CyberShield could be bootloaded using a single UART, the next step was to test bootloading over four separate UART channels, one for each core. The process remained similar, but instead of using a single UART for all cores, the TEE distributed the executable to each core over their respective UART channel. At this stage, no offsets were applied to the instructions, meaning all four cores received identical instructions. This setup more closely resembled the intended operational configuration of the system and allowed for independent verification of each core's ability to execute the received instructions. The successful execution of this setup demonstrated that the TEE could correctly handle multi-channel UART communication while maintaining synchronization.

Booting CyberShield Over Four UARTs with Offset (Figure 7) The final test introduced instruction code obfuscation to assess its impact on the bootloading process. Figure 7 presents the results of bootloading CyberShield with obfuscated instructions. In this stage, the TEE applied predefined offsets to the instruction sequences before transmission. Three of the four cores received obfuscated instructions, while one core continued to run standard RISC-V 32I instructions. The obfuscation process altered only the last byte of each 32-bit instruction, ensuring that execution remained functionally equivalent but obscured from

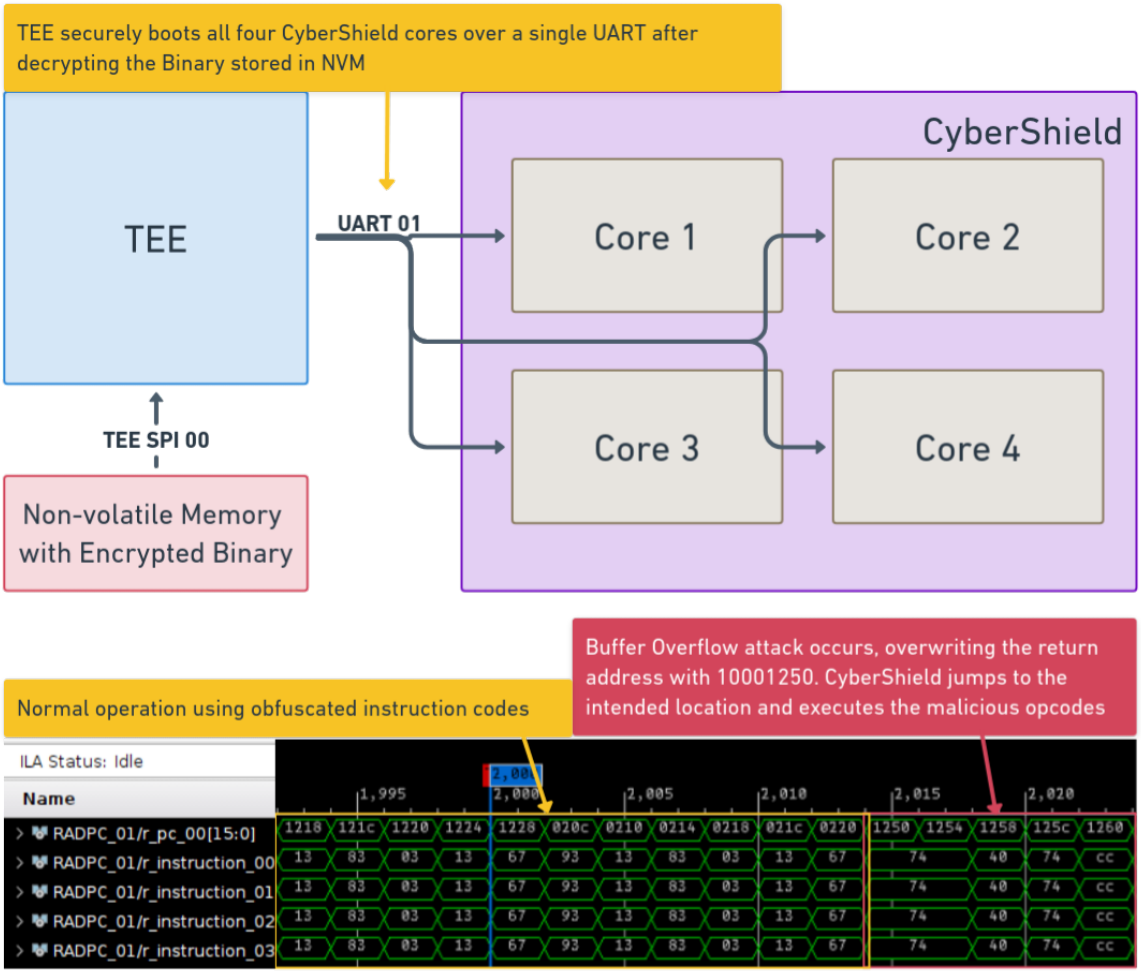

Figure 5: Diagram of CyberShield + TEE and NVM coupled with its ILA outputs. The diagram shows CyberShield being bootloaded via one UART with no obfuscation. The ILA shows the four cores with the program counter running through a dummy function and the vulnerable overflow function and then falling victim to a buffer overflow attack.

direct analysis.

Additionally, the TEE had to determine when to stop applying offsets to avoid corrupting non-executable sections of memory. The `ret` instruction sequence `(67 80 00 00)` served as a marker for the end of the IMEM executable section. Once the final occurrence of this sequence was detected, the TEE ceased applying offsets, leaving the following DMEM data unaffected.

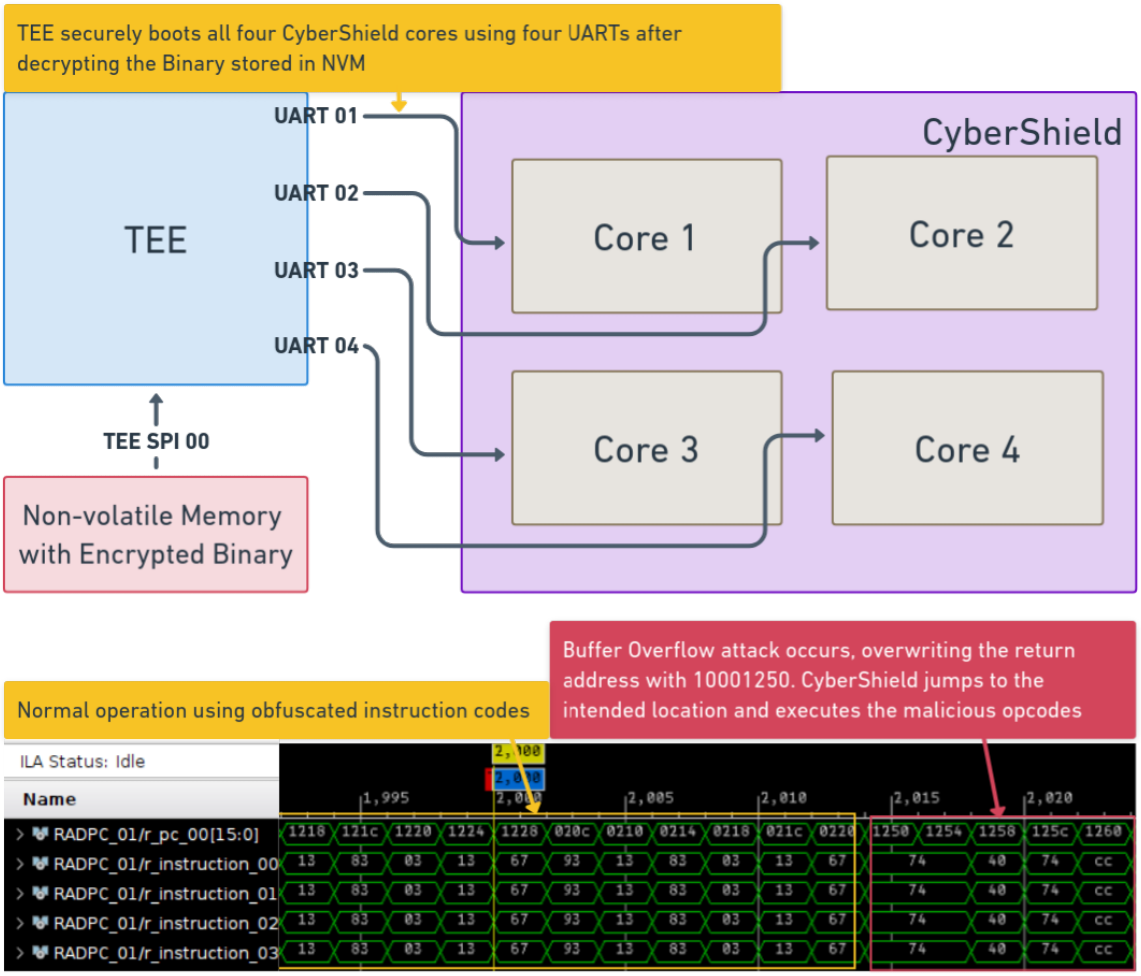

Figure 6: Diagram of CyberShield + TEE and NVM coupled with its ILA outputs. The diagram shows each individual CyberShield core being bootloaded via four UARTs with no obfuscation. The ILA shows the four cores with the program counter running through a dummy function and the vulnerable overflow function and then falling victim to a buffer overflow attack.

Once all four cores were bootloaded with their respective obfuscated instructions, CyberShield executed the program successfully, confirming that the obfuscation mechanism did not interfere with normal execution while increasing security.

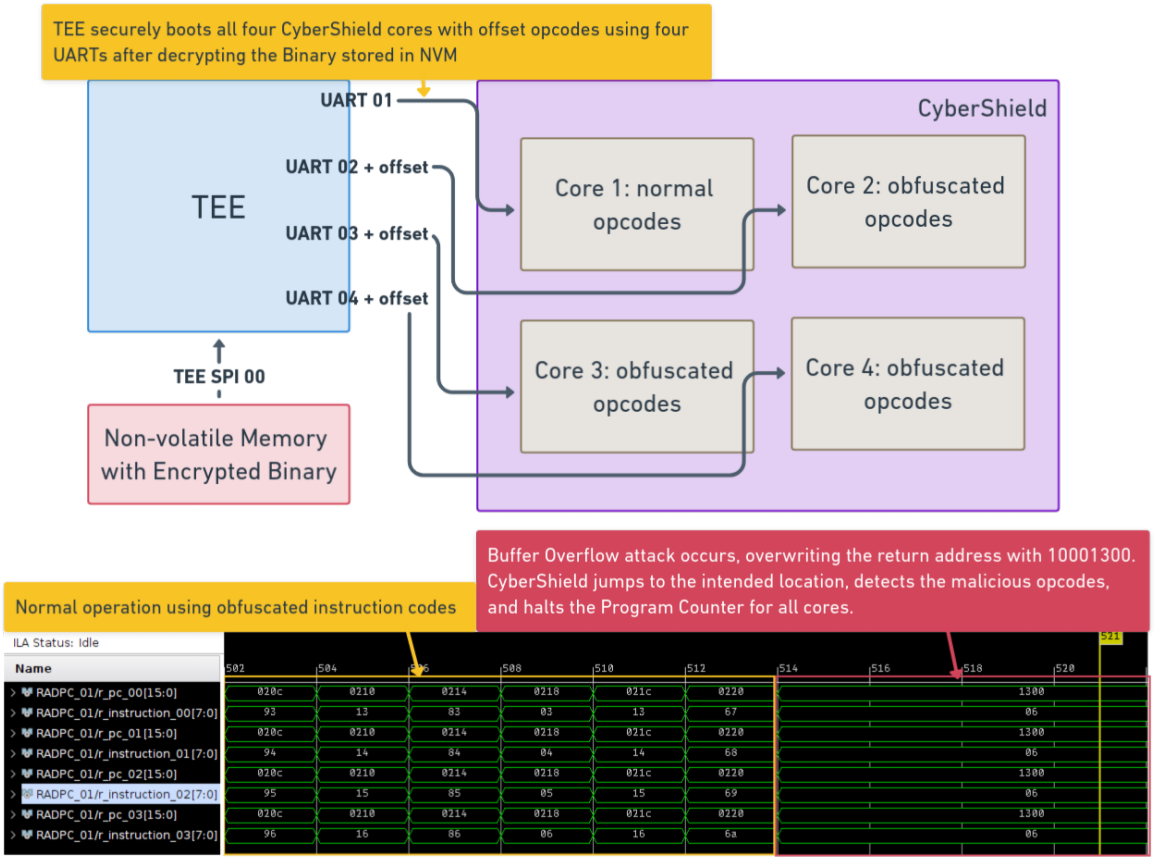

Figure 7: Diagram of the CyberShield architecture with CyberShield and TEE integration, accompanied by ILA outputs. The diagram illustrates how the TEE securely bootloads each CyberShield core with its assigned opcode offset: Core 1 executes standard RISC-V opcodes, while Cores 2–4 operate with unique, offset opcodes. The ILA shows the cores running offset opcodes during normal operation. When a buffer overflow attack occurs, all four cores are forced to execute identical opcodes. The anti-voter in CyberShield detects that all four cores are running the same opcode and freezes the program counter to prevent the cores from executing the malicious code.

Hardware Error Handing, e.g Anti-Voter

While the software for the TEE plays a crucial role in providing security and redundancy to CyberShield there are also critical hardware components. RadPC contains a voting system for each of its four cores in order to provide hardware redundancy in the case of a SEE. This voter compares all the instructions to ensure agreement and then passes the confirmed

instructions. The anti-voter, inspired by RadPC's voting system, does the exact opposite. The anti-voter checks that no two instructions are the same as they are executed. If any of the cores share an instruction, this is an Indicator of Compromise (IoC), and an error flag is raised. When the error flag is raised, two actions occur: (1) the PC is halted to prevent CyberShield from executing any malicious instructions, and (2) both the PC active LED and the IoC LED turn red (Figures 7 and 8). In the future, this flag will be used by the TEE to reboot CyberShield after an attack.

Results

Cybershield was deployed on a Nexys A7 FPGA Development Board featuring the Artix-7 100T FPGA for testing. Cybershield was loaded onto the FPGA using Vivado Design Suite. Bootloading the TEE and loading the executable into the TEE for NVM were done over the serial terminal. A demo counting program was created to simulate CyberShield's software.

CyberShield's general operating mode is centered around securing the vulnerable interface between microcontrollers and the outside world. Any microcontroller that receives external inputs, whether through serial ports, network connections, or other interfaces, is susceptible to attacks. CyberShield replaces the traditional microcontroller in such systems with a hardened, fault-tolerant design capable of detecting when malware has been injected. It accomplishes this by obfuscating opcodes across multiple redundant cores. If malware enters the system, it is blindly replicated across all cores. Because each core expects different opcodes at the same program counter (due to the obfuscation), the malware causes all cores to present the same opcode—a scenario that is architecturally impossible under normal conditions and is immediately flagged as an anomaly.

CyberShield operates in tandem with a TEE, which plays two key roles. First, at system boot, the TEE receives an encrypted executable, decrypts it, applies the opcode obfuscations

across the redundant cores, and bootloads the CyberShield microcontroller. The second role of the TEE—currently targeted for future implementation—is to assist in recovery when malware is detected within the CyberShield cores, potentially orchestrating reconfiguration or system healing. Once bootloading is complete, the TEE is physically disconnected from any external inputs, eliminating it as a potential attack surface. Because the TEE is disconnected from the outside world, attackers cannot target the TEE directly; instead, an attack would occur through CyberShield's standard I/O interfaces, where CyberShield's architecture is specifically designed to detect and mitigate such threats.

Buffer Overflow Attack

To test the resiliency of the obfuscated opcodes for CyberShield a buffer overflow attack was designed to overwrite the stack and have the Program Counter (PC) return to a location outside of IMEM. The payload for the buffer overflow attack was delivered over UART 04 on CyberShield from a serial terminal. A switch was used to change the serial terminal windows connection between TEE UART 00 and CyberShield UART 04 so that CyberShield could be attacked.

The attack to be delivered over the serial terminal was on a modified version of the test counting program. This program contained a "Dummy" function that had a `printf` statement with unregulated input length. This function would continue to write to memory beyond the length of the buffer as long as data was being supplied. The disassembly for the test counting C program was used to determine where outside of IMEM the program would return to when attacked. The stack and surrounding data were repeatedly overwritten with the address `10001250` or `10001300` which is a location in Data Memory (DMEM) (Figure 5, 6, & 7).

The attack was tested on three different subsets of CyberShield. First, the attack was tested on CyberShield in which all four cores had been bootloaded with a single

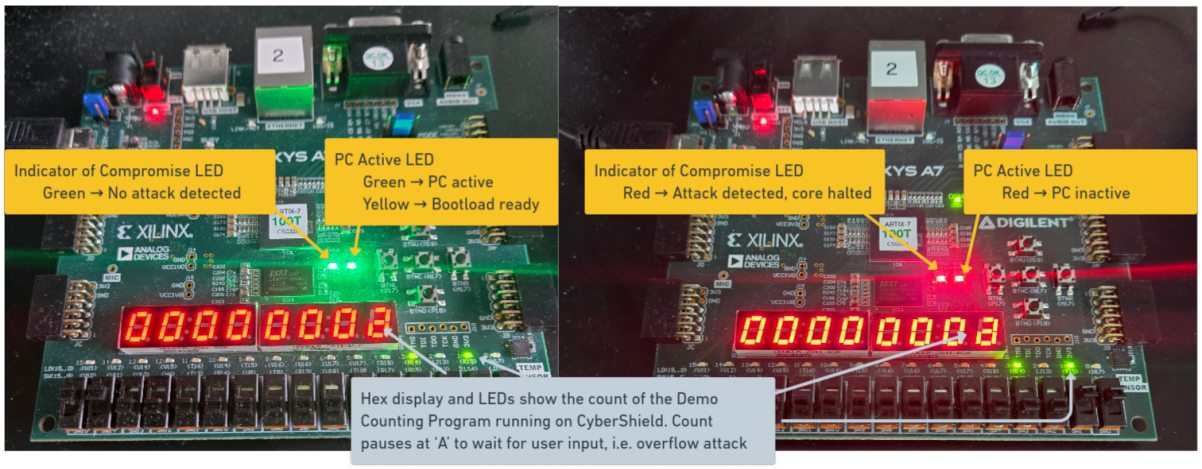

Figure 8: Two Nexys A7 boards demonstrating CyberShield's runtime behavior. Left board: The green LED on the left indicates that CyberShield has booted successfully and no indicator of compromise (IoC) has been detected. The green LED on the right functions as a PC active flag, signifying that the cores are currently executing instructions normally. Right board: In contrast, the left red LED indicates that an IoC has been detected, signaling that the system has identified an attack. The right red LED, which normally serves as the PC active flag, now indicates that all cores have been halted to prevent further execution of potentially malicious instructions.

UART (Figure 5) and didn't have instruction obfuscation. Second, the attack was tested on CyberShield in which all four cores were bootloaded using separate UARTs without obfuscation (Figure 6). Last, the attack was tested on the full Cybershield system (Figure 7). When Cybershield is attacked the PC jumps to the address in DMEM and attempts to execute the "instruction" at that location. As a result, the opcodes of all four CyberShield cores become identical, indicating compromise.

This attack demonstrated how the obfuscation of the cores can defend against buffer overflows and other types of injection attacks. In the absence of a stack guard or address space layout randomization (ASLR), the redundancy of a quad-core system with obfuscated opcodes can defend against overflow and injection attacks. The ability to securely boot and reboot in the event of an attack ensures the continued operation of Cybershield even in the event of compromise.

Conclusion

The principal results of this work demonstrate a proof-of-concept design for integrating a TEE within the RadPC architecture to create CyberShield. Once integrated, the TEE provided secure boot of obfuscated instructions to detect and defeat injected malware. This implementation also enhances the security of CyberShield by enabling secure boot using an encrypted executable. The TEE also provides a mechanism for software recovery in the event of an IoC. These enhancements collectively strengthen the resilience of CyberShield against a range of cyber threats, especially command injection and buffer overflow attacks.

Applications & Advantages

The addition of a TEE into CyberShield offers significant advantages over the baseline RadPC architecture. As an embedded system, RadPC lacks many security features that are standard in general-purpose computing platforms, such as ASLR and StackGuard, which serve as defenses against injection and buffer overflow attacks [15]. By incorporating a TEE, CyberShield gains secure boot, in conjunction with opcode obfuscation, which provides real-time protection against such attacks [16]. This, in conjunction with, the capability to securely boot and reboot from NVM with an encrypted executable ensures robust software redundancy. Secure boot and encryption are particularly valuable for embedded systems that rely on persistent storage and may lack frequent software updates.

Limitations

Despite the security enhancements introduced by the TEE, certain limitations remain. This proof-of-concept implementation has not been extensively tested under adversarial conditions, leaving open questions about its resilience to advanced side-channel attacks and hardware-level exploits. The buffer overflow attack was detected, but further testing is needed to ensure it can detect other IoCs. While opcode obfuscation enhances security,

it introduces compatibility challenges when integrating with existing software toolchains or debugging frameworks. Additionally, the integration of a TEE imposes additional computational overhead, which may affect performance, particularly in resource-constrained embedded environments. This system runs on a 100T FPGA, and smaller chips may not be conducive to Cybershield.

Future Work

Future research will focus on refining the security and performance of the CyberShield TEE implementation. Detecting other types of attacks outside of buffer overflows with ensure a more robust security framework. Moreover, adapting this TEE framework to other embedded architectures and security-critical applications, such as aerospace, medical devices, and industrial control systems, could broaden its impact. Finally, improving software recovery mechanisms through runtime integrity verification and anomaly detection techniques will further strengthen CyberShield's resilience against software compromises. Addressing these areas will ensure that CyberShield remains a secure and adaptable embedded computing platform capable of countering modern cyber threats.

Acknowledgments

This research was supported in part by NASA under award number 80NSSC23CA147 and by Resilient Computing, LLC under subcontract number 4W9082. Any opinions contained herein are those of the authors and do not necessarily reflect those of NASA or Resilient Computing, LLC.

Bibliography

[1] A. Ltd., "What is fpga?" [Online]. Available: https://www.arm.com/glossary/fpga

[2] J. Schneider and I. Smalley, "What is a field programmable gate array (fpga)?" May 2024. [Online]. Available: https://www.ibm.com/think/topics/field-programmable-gate-arrays

[3] P. Jauernig, A.-R. Sadeghi, and E. Stapf, "Trusted execution environments: Properties, applications, and challenges," *IEEE Security & Privacy*, vol. 18, no. 2, pp. 56–60, 2020.

[4] T. L. Loo, M. K. Ishak, and K. Ammar, "Design and implementation of secure boot architecture on risc-v using fpga," *Microprocessors and Microsystems*, vol. 101, p. 104889, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933123001321

[5] A. Dave, N. Banerjee, and C. Patel, "Care: Lightweight attack resilient secure boot architecture with onboard recovery for risc-v based soc," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 516–521.

[6] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, 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. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387532

[7] F.-J. Streit, F. Fritz, A. Becher, S. Wildermann, S. Werner, M. Schmidt-Korth, M. Pschyklenk, and J. Teich, "Secure boot from non-volatile memory for programmable soc architectures," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 102–110.

[8] C. M. Major, A. Bachman, C. Barney, S. Tamke, and B. J. LaMeres, "Radpc: A novel single-event upset mitigation strategy for field programmable gate array–based space computing," *Journal of Aerospace Information Systems*, vol. 18, no. 5, pp. 280–288, 2021. [Online]. Available: https://doi.org/10.2514/1.I010859

[9] J. Williams, C. Barney, Z. Becker, J. Davis, C. Major, B. J. LaMeres, and B. Whitaker, "Radpc@scale: A novel approach to radpc single event upset mitigation strategy," *2022 IEEE Aerospace Conference*, 2022.

[10] B. J. LaMeres and C. Gauer, "Dynamic reconfigurable computing architecture for aerospace applications," *2009 IEEE Aerospace Conference*, 2009.

[11] C. Gauer, B. J. LaMeres, and D. Racek, "Spatial avoidance of hardware faults using fpga partial reconfiguration of tile-based soft processors," *2010 IEEE Aerospace Conference*, 2010.

[12] H. A. Austin, "Fault injection system for fpga-based space computers," Ph.D. dissertation, Montana State University-Bozeman, College of Engineering, 2023.

[13] L. L. Ritzdorf, C. Barney, C. M. Major, T. R. Crane, H. Austin, B. Macht, C. Izurieta, and B. J. LaMeres, "Evaluating the effectiveness of obfuscated instruction codes for malware resistance," in *2023 Intermountain Engineering, Technology and Computing (IETC)*, 2023, pp. 67–72.

[14] T. T. Running Crane, "Using instruction code obfuscation to defeat malware attacks," Ph.D. dissertation, Montana State University-Bozeman, College of Engineering, 2023.

[15] M. A. Butt, Z. Ajmal, Z. I. Khan, M. Idrees, and Y. Javed, "An in-depth survey of bypassing buffer overflow mitigation techniques," *Applied Sciences*, vol. 12, no. 13, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/13/6702

[16] Z. Shao, Q. Zhuge, Y. He, and E.-M. Sha, "Defending embedded systems against buffer overflow via hardware/software," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, 2003, pp. 352–361.

CONCLUSION

This thesis presents CyberShield, a Trusted Execution Environment (TEE) designed to enhance the security of RadPC, an embedded FPGA system, through secure boot and opcode obfuscation. Developed as a hardware-software extension to the radiation-tolerant RadPC processor, CyberShield addresses software redundancy in the system.

CyberShield's design introduces several key innovations. First, it implements a TEE that operates independently of the main processor and is capable of securely receiving, decrypting, and bootloading encrypted executables. Second, it enhances protection against injection attacks through instruction opcode obfuscation, assigning each RadPC core a unique offset. This architectural asymmetry is monitored by an anti-voter module, which can detect and respond to anomalies that indicate malicious behavior.

CyberShield is most similar to Operon in its applications and features, but more closely aligned with HECTOR-V architecturally. All three systems represent trusted execution environments (TEEs) for embedded systems, but differ in focus and execution (Figure 9). CyberShield emphasizes malware detection and recovery through redundancy and opcode obfuscation: each redundant core is loaded with uniquely offset instructions, making injected malware detectable when identical opcodes appear across cores. Like HECTOR-V, CyberShield includes a dedicated TEE implemented as a separate RISC-V core, responsible for secure bootloading and future recovery support. Operon, by contrast, uses a dual-core model within a shared environment, isolating secure and insecure domains but lacking a fully independent TEE core. HECTOR-V hardens security at the processor level, embedding defenses against hardware Trojans and side-channel attacks. Together, these designs highlight the range of strategies for securing embedded systems—from dynamic anomaly detection to execution isolation to hardware-rooted protection.

Experimental validation confirmed CyberShield's ability to detect buffer overflow

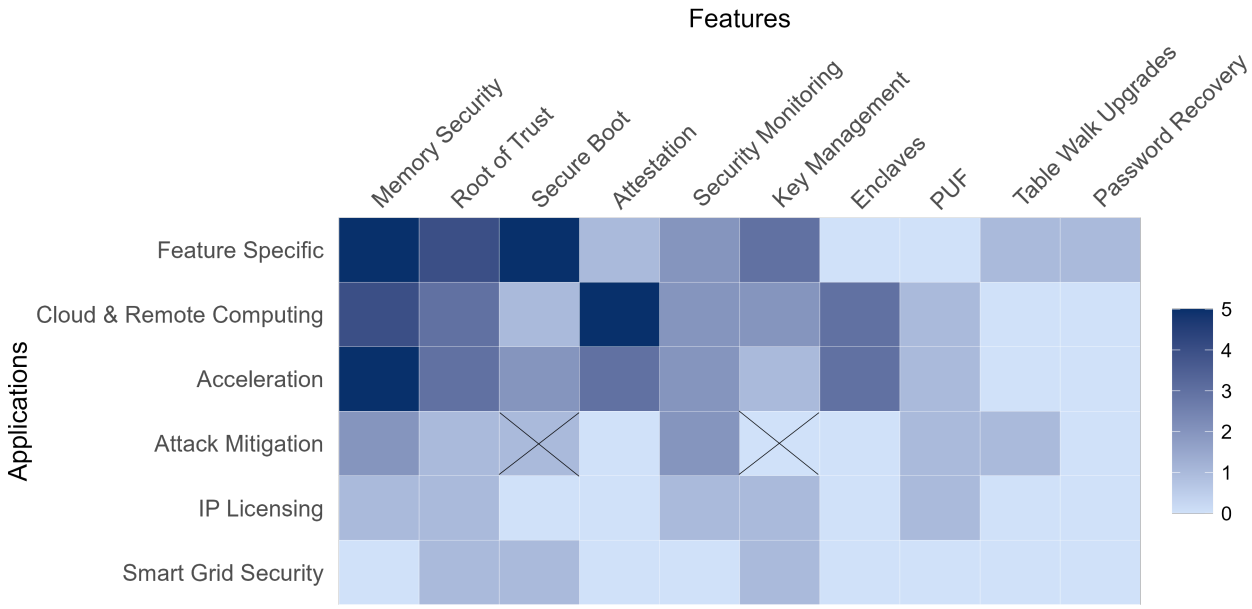

Figure 9: Heatmap of applications and their respective features in the pool of papers. Blue hue denotes the number of papers discussing the features and applications indicated on the axes. Boxes marked with an "X" indicate where CyberShield aligns with the identified features and applications.

attacks by triggering an IoC whenever uniform instructions are observed across cores. The system was tested in three stages: single UART, unobfuscated boot, quad-UART unobfuscated boot, and quad-UART obfuscated boot. At each stage, the system was attacked using a buffer overflow attack. Only the obfuscated system was able to detect the attack. These results demonstrate that secure boot, combined with obfuscated opcodes can serve as an effective defense mechanism.

CyberShield's TEE presents a replicable model for bootloading obfuscated instruction sets via UART, storing and decrypting executables from non-volatile memory, and integrating seamlessly with bare-metal embedded systems. Importantly, this framework maintains compatibility with the existing RadPC QMR design, preserving its resilience to radiation-induced faults while introducing new defenses against software-level cyber threats. These two pieces provide combinational hardware and software redundancy.

Despite its usefulness, CyberShield is not without limitations. The system introduces computational overhead and increased development complexity, particularly due to the need for custom encryption, bootloading coordination, and VHDL integration. Furthermore, opcode obfuscation may limit certain types of debugging or real-time introspection. These trade-offs must be considered when deploying CyberShield in resource-constrained or time-critical applications. Future work will explore dynamic opcode re-obfuscation, detection of other IoCs, and integration with future versions of RadPC.

In conclusion, CyberShield offers a novel, practical, and effective solution to software redundancy and really time detection of IoCs at boot time and runtime. By combining hardware diversity, software obfuscation, and TEE principles, this architecture strengthens embedded security for mission-critical systems operating under real-world constraints. As edge devices and FPGA-based systems continue to proliferate, solutions like CyberShield will play a vital role in protecting the integrity and reliability of tomorrow's computing infrastructure.

Bibliography

[1] H. A. Austin, "Fault injection system for fpga-based space computers," Ph.D. dissertation, Montana State University-Bozeman, College of Engineering, 2023.

[2] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2021.

[3] C. Bobda, J. M. Mbongue, P. Chow, M. Ewais, N. Tarafdar, J. C. Vega, K. Eguro, D. Koch, S. Handagala, M. Leeser, M. Herbordt, H. Shahzad, P. Hofste, B. Ringlein, J. Szefer, A. Sanaullah, and R. Tessier, "The future of fpga acceleration in datacenters and the cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 3, Feb. 2022. [Online]. Available: https://doi.org/10.1145/3506713

[4] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[5] NCC Group, "Fpgas: Security through obscurity?" https://www.nccgroup.com/us/research-blog/fpgas-security-through-obscurity/, 2021, accessed: 2025-03-21.

[6] O. Hosam and F. BinYuan, "A comprehensive analysis of trusted execution environments," in *2022 8th International Conference on Information Technology Trends (ITT)*, 2022, pp. 61–66.

[7] Pearson *et al.*, "Privacy, security and trust issues arising from cloud computing," in *2010 IEEE Second International Conf on Cloud Computing Technology and Science*, 2010, pp. 693–702.

[8] Xiao *et al.*, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[9] Zeyu *et al.*, "Survey on edge computing security," in *2020 International Conf on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2020, pp. 96–105.

[10] Zhang *et al.*, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[11] D. Kaplan, "Protecting vm register state with sev-es," Advanced Micro Devices, Inc., February 2017, white Paper. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf

[12] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, 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. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387532

[13] 2024. [Online]. Available: https://developer.arm.com/documentation/PRD29-GENC-009492/latest/

[14] A. Ltd., "What is fpga?" [Online]. Available: https://www.arm.com/glossary/fpga

[15] J. Schneider and I. Smalley, "What is a field programmable gate array (fpga)?" May 2024. [Online]. Available: https://www.ibm.com/think/topics/field-programmable-gate-arrays

[16] [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/fpga-vs-structured-asic.html

[17] Nasahl *et al.*, "Hector-v: A heterogeneous cpu architecture for a secure risc-v execution environment," in *Proceedings of the 2021 ACM Asia Conf on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: ACM, 2021, p. 187–199. [Online]. Available: https://doi.org/10.1145/3433210.3453112

[18] Yoshida *et al.*, "Towards trusted iot sensing systems: Implementing puf as secure key generator for root of trust and message authentication code," in *Proceedings of the 10th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '21. New York, NY, USA: ACM, 2022. [Online]. Available: https://doi.org/10.1145/3505253.3505258

[19] B. A. Kitchenham, *et al.*, "Using mapping studies as the basis for further research–a participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638–651, 2011.

[20] Zhao *et al.*, "Shef: Shielded enclaves for cloud fpgas," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1070–1085.

[21] Zhu *et al.*, "Tacc: a secure accelerator enclave for ai workloads," in *Proceedings of the 15th ACM International Conf on Systems and Storage*, ser. SYSTOR '22. New York, NY, USA: ACM, 2022, p. 58–71. [Online]. Available: https://doi.org/10.1145/3534056.3534943

[22] Ren *et al.*, "Accshield: a new trusted execution environment with machine-learning accelerators," in *2023 60th ACM/IEEE DAC*, 2023, pp. 1–6.

[23] Kolimbianakis *et al.*, "Software-defined hardware-assisted isolation for trusted next-generation iot systems," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '22. New York, NY, USA: ACM, 2022, p. 139–146. [Online]. Available: https://doi.org/10.1145/3477314.3508378

[24] Ren *et al.*, "Accguard: Secure and trusted computation on remote fpga accelerators," in *2021 IEEE iSES*, 2021, pp. 378–383.

[25] Jasti *et al.*, "Security in multi-tenancy cloud," in *44th Annual 2010 IEEE International Carnahan Conf on Security Technology*, 2010, pp. 35–41.

[26] Oh *et al.*, "Meetgo: A trusted execution environment for remote applications on fpga," *IEEE Access*, vol. 9, pp. 51 313–51 324, 2021.

[27] Wang *et al.*, "Operon: an encrypted database for ownership-preserving data management," *Proc. VLDB Endow.*, vol. 15, no. 12, p. 3332–3345, Aug. 2022. [Online]. Available: https://doi.org/10.14778/3554821.3554826

[28] Ahmed *et al.*, "Trusted ip solution in multi-tenant cloud fpga platform," in *2022 IEEE 8th WF-IoT*, 2022, pp. 1–6.

[29] Ince *et al.*, "Token-based authentication and access delegation for hw-accelerated telco cloud solution," in *2022 IEEE 11th International Conf on CloudNet*, 2022, pp. 109–117.

[30] Zhu *et al.*, "Chaosintc: A secure interrupt management mechanism against interrupt-based attacks on tee," in *2023 60th ACM/IEEE DAC*, 2023, pp. 1–6.

[31] Mao *et al.*, "Rehad: Using low-frequency reconfigurable hardware for cache side-channel attacks detection," in *2020 IEEE EuroS&PW*, pp. 704–709.

[32] Malekpour *et al.*, "Hardware trojan detection and recovery in mpsocs via on-line application specific testing," in *2019 IEEE 22nd International Symposium on DDECS*, 2019, pp. 1–6.

[33] Schilling *et al.*, "Secwalk: Protecting page table walks against fault attacks," in *2021 IEEE International Symposium HOST*, pp. 56–67.

[34] Khan *et al.*, "Utilizing and extending trusted execution environment in heterogeneous socs for a pay-per-device ip licensing scheme," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2548–2563, 2021.

[35] Chen *et al.*, "A risc-v system-on-chip based on dual-core isolation for smart grid security," in *2022 IEEE 6th Conf EI2*, 2022, pp. 1278–1283.

[36] Turan *et al.*, "Propagating trusted execution through mutual attestation," in *Proceedings of the 4th Workshop on System Software for Trusted Execution*, ser. SysTEX '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3342559.3365334

[37] Stoyanov *et al.*, "Secure heterogeneous architecture based on risc-v and root-of-trust," in *Proceedings of the 24th International Conf on Computer Systems and Technologies*, ser. CompSysTech '23. New York, NY, USA: ACM, 2023, p. 19–23. [Online]. Available: https://doi.org/10.1145/3606305.3606312

[38] Zou *et al.*, "Ares: Persistently secure non-volatile memory with processor-transparent and hardware-friendly integrity verification and metadata recovery," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 1, feb 2022. [Online]. Available: https://doi.org/10.1145/3492735

[39] Streit *et al.*, "Secure boot from non-volatile memory for programmable soc architectures," in *2020 IEEE International Symposium HOST*, 2020, pp. 102–110.

[40] Hoang *et al.*, "Trusted execution environment hardware by isolated heterogeneous architecture for key scheduling," *IEEE Access*, vol. 10, pp. 46 014–46 027, 2022.

[41] Du *et al.*, "Accelerating extra dimensional page walks for confidential computing," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: ACM, 2023, p. 654–669. [Online]. Available: https://doi.org/10.1145/3613424.3614293

[42] Cilardo *et al.*, "Memory encryption support for an fpga-based risc-v implementation," in *2021 16th International Conf on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1–5.

[43] Kumar *et al.*, "Post-quantum secure boot," in *Proceedings of the 23rd Conf on Design, Automation and Test in Europe*, ser. DATE '20. San Jose, CA, USA: EDA Consortium, 2020, p. 1582–1585.

[44] Xi *et al.*, "A heterogeneous risc-v soc for confidential computing and password recovery," in *2022 7th International Conf on Integrated Circuits and Microsystems (ICICM)*, 2022, pp. 500–504.

[45] Cook *et al.*, *Quasi-experimentation: Design & Analysis Issues for Field Settings*. Houghton Mifflin, 1979. [Online]. Available: https://books.google.com/books?id=BFNqAAAAMAAJ

[46] Campbell *et al.*, *Experimental and Quasi-experimental Designs for Research*. R. McNally, 1966. [Online]. Available: https://books.google.com/books?id=kFtqAAAAMAAJ

[47] P. Jauernig, A.-R. Sadeghi, and E. Stapf, "Trusted execution environments: Properties, applications, and challenges," *IEEE Security & Privacy*, vol. 18, no. 2, pp. 56–60, 2020.

[48] T. L. Loo, M. K. Ishak, and K. Ammar, "Design and implementation of secure boot architecture on risc-v using fpga," *Microprocessors and Microsystems*, vol. 101, p. 104889, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933123001321

[49] A. Dave, N. Banerjee, and C. Patel, "Care: Lightweight attack resilient secure boot architecture with onboard recovery for risc-v based soc," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 516–521.

[50] F.-J. Streit, F. Fritz, A. Becher, S. Wildermann, S. Werner, M. Schmidt-Korth, M. Pschyklenk, and J. Teich, "Secure boot from non-volatile memory for programmable soc architectures," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 102–110.

[51] C. M. Major, A. Bachman, C. Barney, S. Tamke, and B. J. LaMeres, "Radpc: A novel single-event upset mitigation strategy for field programmable gate array–based space computing," *Journal of Aerospace Information Systems*, vol. 18, no. 5, pp. 280–288, 2021. [Online]. Available: https://doi.org/10.2514/1.I010859

[52] J. Williams, C. Barney, Z. Becker, J. Davis, C. Major, B. J. LaMeres, and B. Whitaker, "Radpc@scale: A novel approach to radpc single event upset mitigation strategy," *2022 IEEE Aerospace Conference*, 2022.

[53] B. J. LaMeres and C. Gauer, "Dynamic reconfigurable computing architecture for aerospace applications," *2009 IEEE Aerospace Conference*, 2009.

[54] C. Gauer, B. J. LaMeres, and D. Racek, "Spatial avoidance of hardware faults using fpga partial reconfiguration of tile-based soft processors," *2010 IEEE Aerospace Conference*, 2010.

[55] L. L. Ritzdorf, C. Barney, C. M. Major, T. R. Crane, H. Austin, B. Macht, C. Izurieta, and B. J. LaMeres, "Evaluating the effectiveness of obfuscated instruction codes for malware resistance," in *2023 Intermountain Engineering, Technology and Computing (IETC)*, 2023, pp. 67–72.

[56] T. T. Running Crane, "Using instruction code obfuscation to defeat malware attacks," Ph.D. dissertation, Montana State University-Bozeman, College of Engineering, 2023.

[57] M. A. Butt, Z. Ajmal, Z. I. Khan, M. Idrees, and Y. Javed, "An in-depth survey of bypassing buffer overflow mitigation techniques," *Applied Sciences*, vol. 12, no. 13, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/13/6702

[58] Z. Shao, Q. Zhuge, Y. He, and E.-M. Sha, "Defending embedded systems against buffer overflow via hardware/software," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, 2003, pp. 352–361.

[59] M. Schneider, R. J. Masti, S. Shinde, S. Capkun, and R. Perez, "Sok: Hardware-supported trusted execution environments," 2022. [Online]. Available: https://arxiv.org/abs/2205.12742

[60] R. R.V. and K. A., "Secure boot of embedded applications - a review," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 291–298.

[61] V. B. Y. Kumar, N. Gupta, A. Chattopadhyay, M. Kasper, C. Krauß, and R. Niederhagen, "Post-quantum secure boot," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1582–1585.

[62] K. Cahit, "Internal validity: A must in research designs," *Educational Research and Reviews*, vol. 10, no. 2, pp. 111–118, 2015.