Hardware-Rooted Device Identity for IoT: Integrating Silicon PUFs and DICE in RISC-V Embedded Systems

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 February 2026, accepted 5 March 2026, date of publication 17 March 2026, date of current version 13 April 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3674839

RESEARCH ARTICLE

Hardware-Rooted Device Identity for IoT: Integrating Silicon PUFs and DICE in RISC-V Embedded Systems

SILVIA SISINNI[1], LORENZO FERRO[1], ENRICO BRAVI[1], PABLO NAVARRO-TORRERO[2], EROS CAMACHO-RUIZ[2], MACARENA C. MARTÍNEZ-RODRÍGUEZ[2], PIEDAD BROX[2], AND ANTONIO LIOY[1]

[1]Dipartimento di Automatica e Informatica (DAUIN), Politecnico di Torino, 10129 Turin, Italy
[2]Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC/US, 41092 Seville, Spain

Corresponding author: Silvia Sisinni (silvia.sisinni@polito.it)

This research was supported in part by the Quantum-oriented Update to Browsers and Infrastructures for the PQ transition (QUBIP) Project with Grant Agreement No. 101119746 under the European Union (EU) Horizon Europe research and innovation programme and in part by the Secure Platform for ICT systems Rooted at the Silicon manufacturing process (SPIRS) Project with Grant Agreement No. 952622 under the EU H2020 research. This work was partially supported by the project SEcurity and RIghts in the CyberSpace (SERICS) (PE00000014) under the Piano Nazionale di Ripresa e Resilienza (PNRR) Ministero dell'Universitá e della Ricerca (MUR) program, funded by the European Union - NextGenerationEU. This publication is also part of the project PNRR-NextGenerationEU (NGEU) which has received funding from the MUR – Decreto Ministeriale (DM) 352/2022.

ABSTRACT The security of Internet of Things (IoT) and embedded systems critically depends on establishing trustworthy, unclonable device identities from the earliest boot stages. Conventional approaches often rely on externally stored secrets, which remain vulnerable to extraction and cloning, or heavyweight hardware modules, which are unsuitable for constrained platforms. In this work, we present a lightweight Hardware Root of Trust (HRoT) architecture that integrates silicon-based Physically Unclonable Functions (PUFs) with the Device Identifier Composition Engine (DICE) directly into the boot ROM of RISC-V embedded systems. Unlike existing RoTs, our design eliminates the need for persistent key storage by generating manufacturer-unknown identities on demand, and it extends trust across the system through DICE-based layered attestation. We further introduce lifecycle-aware mechanisms for identity provisioning, renewal, and secure zeroization, ensuring long-term device trustworthiness. The proposed architecture is implemented and evaluated on the open-source SPIRS RISC-V platform, embedding hardware accelerators for cryptography and PUF-bound secrets. Experimental results demonstrate strong resistance against physical and software attacks, compliance with IEC 62443 SL2+ requirements, robust lifecycle support, and negligible performance overhead (<1% increase in boot time). By bridging unclonable hardware identities with standardized attestation, this work establishes a scalable and interoperable foundation for secure authentication and continuous trust in IoT and embedded devices.

INDEX TERMS Hardware root of trust, IoT security, physical unclonable functions, DICE, RISC-V, secure boot, device identity provisioning.

I. INTRODUCTION

The rapid expansion of the Internet of Things (IoT) and the proliferation of embedded devices have significantly transformed various sectors, from industrial automation and healthcare to smart cities and intelligent home systems. As these devices increasingly collect, process and transmit sensitive data, ensuring robust security measures has become paramount [1], [2]. Secure device identities are essential for establishing trust within IoT ecosystems, enabling reliable authentication, access control, and secure communications [3]. This process, often referred to as *device personalization*, involves assigning unique identifiers or credentials to each device, enabling secure authentication, access control, and trusted communication. Traditional methods for identity provisioning typically rely on externally stored

The associate editor coordinating the review of this manuscript and approving it for publication was Parul Garg.

cryptographic keys or credentials in non-secure memory, exposing devices to unauthorized duplication, cloning, and identity theft [4]. Software-only authentication mechanisms, while flexible and inexpensive, lack resilience against software device compromise, sophisticated hardware attacks and physical tampering, exacerbating the need for more secure alternatives [5]. In contrast, hardware-based security approaches leverage isolated secure elements and dedicated cryptographic primitives to achieve stronger resistance against tampering and credential extraction [6], [7].

Motivated by these challenges, Hardware Root-of-Trusts (HRoTs) have emerged as a foundation for trustworthy devices. They support secure boot, ensuring only authorized software is executed, and enable attestation, providing verifiable evidence of system integrity [8], [9], [10]. As a result, Roots of Trust (RoTs) form the foundation of reliable device authentication and identity management in modern IoT deployments [11], [12]. Several open-source RoT frameworks have been introduced, such as OpenTitan [13], Caliptra [14], Keystone [15], and RECORD [16], each addressing different parts of the identity and trust chain. Recent designs also explore the integration of Physical Unclonable Functions (PUFs) as an unclonable hardware anchor for cryptographic key generation [17], [18].

Despite these advances, important limitations remain. Most existing solutions still rely on persistent storage of cryptographic identities, leaving them vulnerable to extraction or cloning [4], [18], [19], [20], [21]. Furthermore, comprehensive architectures that combine PUF-bound secrets with Device Identifier Composition Engine (DICE)-style dynamic identity derivation [22], in which each software layer derives its cryptographic identity from the measured sate of the preceding one, are not widely explored. Such integration is particularly critical for resource-constrained IoT devices, where adding external secure modules is impractical.

To address these gaps, this paper presents a HRoT architecture that embeds silicon-based PUFs and DICE logic directly within the boot Read-Only Memory (ROM) of a RISC-V system. Our approach, implemented on the SPIRS platform [23], enables on-device generation and renewal of unclonable cryptographic identities that remain unknown even to manufacturers. By provisioning identities at boot time and extending them across software layers through DICE, the design achieves robust resistance against device key extraction, cloning, and tampering, while supporting lifecycle-aware identity management. In doing so, the proposed system provides a scalable security foundation for constrained IoT and embedded environments.

Contributions and Novelty: Compared to existing hardware and software RoTs such as OpenTitan [13], Caliptra [14], Rolling DICE [24], Sancus [25], and Sanctum [26], this work makes the following specific contributions:

- *Boot ROM-anchored DICE with no persistent root secrets*: unlike OpenTitan and Caliptra, which rely on secrets stored inside a dedicated RoT block, and unlike Rolling DICE, which derives identities from software-managed secrets, our design derives the Unique Device Secret (UDS) directly from a silicon Ring Oscillator PUF (RO-PUF) inside the immutable boot ROM. No device-unique secret is stored in non-volatile memory at any stage.
- *Manufacturer-unknown device identity with deterministic regeneration*: in contrast to systems where the manufacturer provisions or can reconstruct the Device Root Key (DRK), the proposed architecture ensures that the root identity is generated on-device and remains unknown even to the manufacturer, while still being deterministically regenerated at each boot using helper data.
- *Full DICE layering extended to the host system and enclaves*: Unlike OpenTitan and Caliptra, where DICE layering is confined to the RoT subsystem, and unlike Sanctum and Sancus, which do not implement standardized DICE chains, our approach propagates DICE-derived identities from the boot ROM to the host firmware and to Keystone enclaves, enabling platform and enclave attestation under a unified model.
- *Lifecycle-aware identity management integrated in the RoT*: Existing lightweight RoTs typically stop at initial provisioning. This work explicitly supports identity provisioning, remote renewal, revocation, and secure zeroization without hardware replacement, while preserving the same PUF-anchored RoT.
- *Concrete implementation on a RISC-V System-on-Chip (SoC) with quantitative evaluation*: The proposed architecture is fully implemented on the SPIRS RISC-V platform, integrating hardware PUF, cryptographic accelerators, boot ROM logic, and Keystone Trusted Execution Environment (TEE) support. Experimental results show resistance to software and physical attacks and a boot-time overhead below 1%, demonstrating practicality for constrained IoT devices.

Together, these contributions deliver a HRoT that unites PUF-bound unclonable identities with DICE-based layered attestation in the boot ROM and adds lifecycle-aware provisioning, renewal, and zeroization, capabilities not simultaneously addressed by prior IoT-focused RoTs to the best of our knowledge. The hardware description and prototype implementation used in this work are openly available at https://github.com/torsec/manufacturing_tests.

The remainder of this paper is organized as follows. Section II reviews foundational concepts on hardware RoTs, device identity, PUFs, and DICE. Section III surveys related work and highlights the research gaps this work addresses. Section IV defines the threat model and security requirements. Section V presents the design and implementation of the proposed HRoT, including PUF integration, cryptographic modules, and the secure boot and DICE-based mechanisms embedded in the boot ROM. Section VI details the lifecycle of secure device identities, covering generation and provisioning, operational monitoring, and mechanisms for renewal and revocation. Section VII describes the

implementation on the SPIRS platform. Section VIII reports experimental evaluation and security analysis. Section X discusses implications, limitations. Finally, Section XI summarizes the contributions of this work and its significance for secure and scalable device identity management in IoT and embedded systems.

II. BACKGROUND

In modern computing and communication ecosystems, reliably establishing and verifying the unique identity of each device is fundamental to security, trustworthiness, and accountability [1], [2], [11], [27], [28]. Devices must actively prove their identity to peers, maintenance services, and cloud infrastructures. Even devices with identical hardware exhibit subtle physical variations, which can be leveraged to derive intrinsic unique identifiers (it is the fundamental idea on which PUFs are based). This section reviews the key building blocks we use to realize hardware-rooted identities and trustworthy boot on constrained platforms: secure device identity and trusted computing, the DICE architecture, PUFs, secure vs. measured boot, enclave-based TEEs with Keystone, and the most relevant requirements from International Electrotechnical Commission (IEC) 62443 [29].

A. SECURE DEVICE IDENTITY AND TRUSTED COMPUTING

Robust machine identity in networked systems requires more than practical uniqueness. Software-only identifiers such as Universally Unique Identifiers (UUIDs) and Media Access Control (MAC) addresses can be created, copied, or spoofed at will and therefore provide no authenticity or resistance to impersonation [30], [31], [32], [33]. In security-critical environments, devices must *authenticate* themselves using cryptographic *authenticators* that are uniquely bound to the asserting device and protected against extraction and cloning. Historically, authenticators (symmetric keys, asymmetric private keys, passwords/tokens) have been provisioned during manufacturing or first deployment to support verification across the device lifecycle [34]. However, when these authenticators are generated, stored, or used purely in software, numerous well-known attacks enable credential disclosure and migration, defeating identity guarantees [35], [36], [37], [38].

Institute of Electrical and Electronics Engineers (IEEE) 802.1AR addresses these risks by defining *Device Identities (DevIDs)*: cryptographic credentials anchored in protected hardware that enable strong, verifiable device authentication within Zero-Trust frameworks [27]. A DevID comprises: (i) a device-held private key, (ii) an X.509 certificate that binds the public key to the device identity and attributes, and (iii) a certificate chain to a trusted Certificate Authority (CA). A hardware DevID module confines use of the private key to authorized on-device operations, preventing key extraction and impersonation. The standard distinguishes immutable manufacturer-issued Initial DevIDs (IDevIDs) and locally scoped Local DevIDs (LDevIDs) for operational flexibility. While Trusted Platform Modules (TPMs) and secure elements implement DevIDs effectively, cost/area/power and integration complexity limit their adoption across low-cost, resource-constrained IoT platforms [39], [40], [41], [42]. As a result, there remains a gap between the security guarantees required by IEEE 802.1AR / Trusted Computing Group (TCG) and typical IoT implementations (Table 1).

TABLE 1. Comparison of IEEE 802.1AR and TCG security requirements with typical implementations in IoT and embedded devices.

IEEE 802.1AR and TCG Security Requirements	Implementation in IoT and Embedded Devices
Hardware-protected DevID storage	Generally absent in low-cost devices
Non-migratable private key security	Unsupported in software-only solutions
TPM-like signed attestation (quotes)	Rarely implemented due to lack of RoTs

However, identity alone is insufficient: verifiers also need assurance about the *software state* running on the device. Trusted Computing, promoted by TCG, complements hardware-anchored identity with integrity evidence collected and protected by three RoTs [43], [44]: the *Root of Trust for Measurement (RTM)* that measures firmware/software before execution, the *Root of Trust for Storage (RTS)* that securely stores those measurements and associated keys, and the *Root of Trust for Reporting (RTR)* that signs this evidence to enable *remote attestation* by external verifiers. The most widely adopted realization is the TPM, which maintains measurements in Platform Configuration Registers (PCRs) and uses Attestation Keys (AKs) to produce signed integrity quotes. This allows a verifier to confirm not only the device identity, but also that it booted into an expected configuration. While robust, TPM-based approaches remain too heavyweight for most IoT deployments.

Practical enforcement of these principles is achieved through *Secure Boot* and *Measured Boot*. Secure Boot validates each software stage using cryptographic signatures or hashes, halting execution if integrity checks fail [45]. Measured Boot, by contrast, records cryptographic measurements of loaded software into protected registers (e.g., TPM PCRs) without blocking execution [43]. Together, they ensure that devices prevent unauthorized firmware from running and provide attestable evidence of the executed configuration.

To bridge the IEEE 802.1AR/TCG vs IoT gap, we propose a lightweight on-device RoT that (i) avoids persistent secret storage, (ii) produces non-migratable identities bound to the executing firmware, and (iii) interoperates with standard attestation. This motivates our integration of *silicon PUFs* (for unclonable, on-demand secrets) with *DICE* (for identity derivation tied to measured boot) directly in the boot ROM, as detailed in the next sections.

B. DEVICE IDENTIFIER COMPOSITION ENGINE

Traditional RoTs such as the TPM provide strong identity and attestation but are impractical for many IoT and embedded systems due to area, cost, and power overheads. To address this gap, the TCG defined the DICE architecture, a lightweight RoT explicitly tailored for constrained devices while remaining compatible with TPM-based infrastructures [22], [46].

At its core, DICE anchors trust in a *UDS* provisioned at manufacturing. During boot, the UDS is combined with a cryptographic measurement of the *First Mutable Code (FMC)* to derive a *Compound Device Identifier (CDI)*:

$$CDI = \mathbf{Hash}(\text{UDS} \,||\, \mathbf{Hash}(\text{FMC})), \tag{1}$$

or

$$CDI = \mathbf{HMAC}(\text{UDS}, \mathbf{Hash}(\text{FMC})). \tag{2}$$

Any change in FMC configuration produces a different CDI, thereby binding the device identity directly to its integrity state.

DICE generalizes this mechanism into a *layered trust architecture*, in which each software layer measures the next, derives a fresh CDI, and uses it to generate cryptographic keys (Fig. 1, 2). Asymmetric keys derived from CDIs implicitly attest to the trustworthiness of the corresponding layer, while symmetric keys enable sealed storage or key-wrapping across reboots. If firmware is tampered with, subsequent CDI derivations diverge, preventing reuse of keys bound to trusted states.

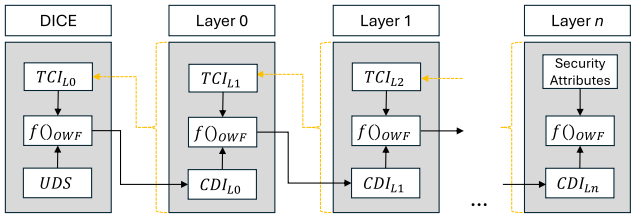

FIGURE 1. DICE layering architecture.

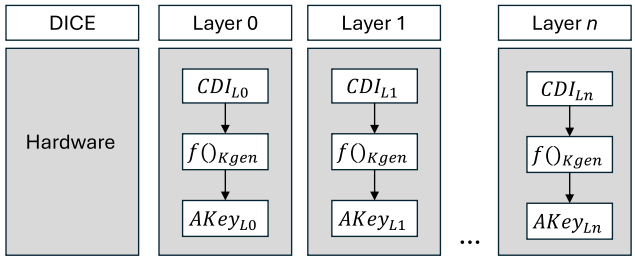

FIGURE 2. Asymmetric key generation example.

Finally, DICE supports a *certificate hierarchy* rooted in manufacturer-issued or embedded credentials, optionally extended by Embedded Certificate Authorities (ECAs) responsible for issuing certificates to higher layers (Fig. 3). Through this hierarchy, provenance and attestation evidence are propagated across the entire software stack.

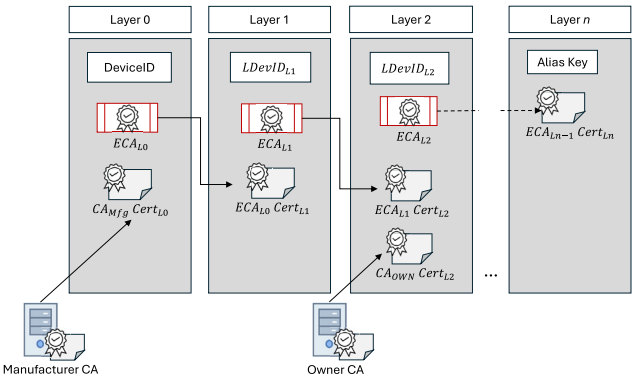

FIGURE 3. DICE Certificate Hierarchy.

Combining lightweight identity derivation with layered attestation, DICE provides a practical basis for establishing device trust within IoT deployments. It supports constrained hardware platforms but is still interoperable with current attestation standards, addressing some of the limitations of conventional RoTs.

C. PHYSICAL UNCLONABLE FUNCTIONS (PUFs)

Strong device identities rely on a unique and securely protected root secret (e.g., the IDevID secret in IEEE 802.1AR or the UDS in DICE). Storing this secret in non-volatile memory exposes it to extraction or cloning, particularly under physical attacks [19], [20]. PUFs address this challenge by exploiting intrinsic manufacturing variations in silicon to generate device-unique responses on demand. Because no fixed secret is stored, PUF-based identities are resistant to duplication and tampering, making them an attractive alternative for constrained and exposed devices [19], [21].

Strong PUFs, widely adopted in device authentication due to their ability to produce a large set of Challenge Response Pairs (CRPs), are known to be susceptible to modeling attacks that leverage machine-learning techniques to approximate their input–output behavior [47]. This vulnerability has prompted the development of multiple countermeasures aimed at reinforcing the robustness of different PUF architectures [48], [49]. By contrast, prediction-based attacks are generally less effective against weak PUFs, as these designs offer a limited CRP space, are mainly used for key generation and secure key storage, and typically do not expose their outputs externally, thereby reducing their attack surface [50], [51], [52].

Among PUF families, *RO-PUFs* are weak PUFs particularly suitable for SoC platforms. They rely on frequency differences across otherwise identical oscillator rings to derive unique bit patterns. RO-PUFs offer strong uniqueness and repeatability with low hardware overhead, and their simplicity makes them practical for integration in lightweight RoTs [53], [54]. An additional advantage is that the same

oscillator-based circuitry can provide entropy for True Random Number Generators (TRNGs), leveraging intrinsic jitter in oscillation frequencies [55], [56]. This dual role optimizes resource usage and provides both unclonable identities and internal randomness, reinforcing the overall trust anchor for IoT and embedded devices.

D. TRUSTED EXECUTION ENVIRONMENTS IN IoT

Beyond identity derivation and boot-time integrity, modern IoT platforms must also protect sensitive applications during runtime. *TEEs* provide this capability by establishing hardware-enforced isolation domains that preserve the confidentiality and integrity of code and data even in the presence of a compromised operating system or privileged software [57]. Unlike dedicated Hardware Security Modules (HSMs), TEEs operate within the main processor, offering a practical balance between flexibility and security for resource-constrained embedded devices.

Traditional TEEs, such as ARM TrustZone [58], adopt a coarse dual-world model (secure vs. normal world). More recent *enclave-based TEEs* refine this concept by enabling multiple mutually distrusting applications, called *enclaves*, to run in parallel on the same system. Each enclave's code and data are strongly isolated from the normal world, or Rich Execution Environment (REE), as well as from other enclaves, ensuring that even privileged system software cannot access or tamper with them. This application-oriented model provides stronger isolation and flexibility, which is especially relevant for IoT devices expected to run diverse workloads.

Among enclave-based frameworks, *Keystone* has emerged as a lightweight, open-source option tailored to RISC-V. It leverages the privileged Instruction Set Architecture (ISA) and Physical Memory Protection (PMP) extensions to enforce isolation through a trusted Security Monitor, which manages enclave lifecycle and mediates access to shared resources (Fig. 4) [15], [59]. Despite these advantages, Keystone's security ultimately depends on correct PMP configuration and on the trustworthiness of the Security Monitor itself, which must be anchored in Secure Boot and remote attestation. This dependency highlights the need for an integrated RoT capable of binding enclave execution to hardware-rooted identities and measured system integrity. In this work, Keystone serves as the TEE framework experimentally integrated and validated on our RISC-V platform (Section VII).

E. IEC 62443 (PART 4-2) FOR EMBEDDED DEVICES

The IEC 62443-4-2 standard [29] defines technical security requirements for embedded components in industrial automation, covering identification and authentication, use control, system integrity, confidentiality, and auditability. It introduces four Security Levels (SL1–SL4), reflecting increased resilience against adversaries ranging from casual attackers to highly resourced and sophisticated actors [60].

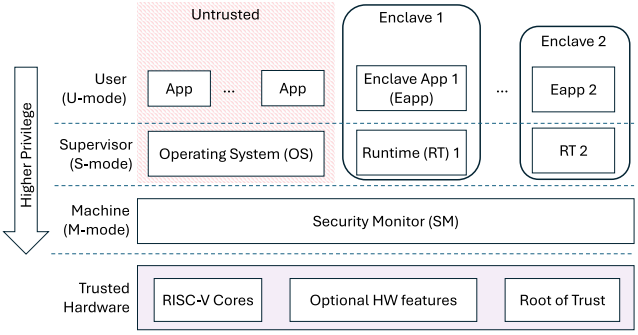

FIGURE 4. Keystone system architecture.

For IoT and embedded devices, compliance with SL2 and above is especially relevant, as it entails resistance to intentional attacks performed with publicly available tools and more sophisticated techniques. Meeting these requirements involves enforcing a signed boot process, binding device identities to non-migratable hardware roots, and supporting essential lifecycle operations such as credential renewal and revocation. The proposed architecture addresses these SL2+ requirements by providing a standards-aligned framework that remains practical for deployment on resource-constrained platforms.

III. RELATED WORK

A. HARDWARE ROOTS OF TRUST AND DICE-BASED ARCHITECTURES

Secure device identity provisioning and attestation are fundamental security requirements for modern embedded and IoT systems. A wide range of hardware-based designs and frameworks have been proposed to establish robust, verifiable device identities anchored in HRoTs. These efforts include both industry-driven open-source implementations and academic prototypes that aim to minimize the Trusted Computing Base (TCB) while maintaining strong security guarantees.

OpenTitan [13] is an important open-source silicon RoT project led by lowRISC and supported by industry partners (e.g., Google, Western Digital) It implements secure boot and device identity functions within a dedicated hardware block, including cryptographic engines and support for DICE-compliant certificates. However, OpenTitan is designed as a standalone RoT component. Extending its chain-of-trust to protect the main processor firmware and Operating System (OS) requires additional integration, and is not inherently provided in the base implementation. Moreover, its silicon complexity and resource requirements may be unsuitable for highly resource-constrained devices, which motivates exploration of more lightweight solutions.

Caliptra [14] is another open-source RoT developed within the Open Compute Project (OCP), targeting datacenter and edge infrastructures. It defines a RTM block that provides secure boot, measured boot, and attestation services for server-class SoCs. It integrates into a SoC

to measure and verify firmware components, deriving a cryptographic device identity (aligned with DICE principles) and reporting system state to external verifiers. Caliptra's design emphasizes scalability, interoperability, and even post-quantum cryptography support for large deployments. However, while Caliptra demonstrates a robust identity and attestation model for high-performance systems, its silicon complexity and hardware requirements make it unsuitable for deeply embedded or ultra-low-power IoT devices, as it explicitly targets datacenter-class SoCs such as Central Processing Units (CPUs), Graphics Processing Unit (GPUs) and Tensor Processing Units (TPUs) [61], [62].

In parallel, the academic community has explored lightweight RoT, focusing on minimal hardware additions for constrained devices. Sancus [25] (KU Leuven) and Sanctum [26] (MIT) are two representative examples that illustrate this approach. Sancus [25], [63] provides isolated execution and remote attestation for MSP430-class microcontrollers using device-unique secrets stored in protected memory, but relies exclusively on symmetric cryptography and does not address identity lifecycle management. Sanctum [26] targets RISC-V systems and uses a PUF to derive a device root key at boot, and enables enclave isolation. However, it requires custom ISA extensions and does not implement standardized layered attestation such as DICE.

Beyond hardware prototypes, several software frameworks explored DICE-based identities. "Rolling DICE" [24] implements lightweight attestation protocols for constrained devices, while Bravi et al. [64]) combine DICE with Manufacturer Usage Description (MUD) [65] to secure dynamic IoT scenarios. Other works integrate PUF technology with cryptography, e.g., Román et al. [66] combine Static Random Access Memory (RAM) (SRAM) PUFs with quantum-safe signatures on ESP32 devices. These approaches demonstrate feasibility, but do not integrate identity lifecycle support or hardware-anchored DICE in the boot ROM.

Compared to the works summarized in Table 2, the proposed design occupies a different position in the RoT design space. Existing open HRoTs such as OpenTitan [13] and Caliptra [14] concentrate trust functions inside a dedicated RoT subsystem, with device secrets stored internally and DICE layering confined to that block. Software-oriented approaches such as Rolling DICE [24] reduce hardware complexity, but rely on software-managed secrets and do not provide a hardware anchor against physical extraction or lifecycle attacks. Academic systems such as Sancus [25], [63] and Sanctum [26] introduce isolation and, in the latter case, PUF-derived keys, but do not combine a PUF-anchored root secret with standardized DICE layering and identity lifecycle management. In contrast, our architecture embeds the RoT directly in the immutable boot ROM of the host RISC-V system, derives the device root secret on demand from a silicon PUF without persistent storage, and propagates DICE-derived identities beyond the boot stage to the host firmware and Keystone enclaves. This combination enables secure boot, attestation, and identity renewal under a single hardware-rooted trust model, while remaining compatible with resource-constrained IoT platforms.

B. PUF-BASED AUTHENTICATION AND KEY GENERATION

A large amount of prior work uses PUFs as primitives for device authentication and key generation. Many of these approaches rely on strong PUFs, such as arbiter-based designs, due to their ability to generate a large number of CRPs. This enables remote challenge–response authentication, but also exposes strong PUFs to modeling attacks, where an adversary learns an approximate model of the PUF behaviour from observed CRPs. To mitigate these attacks, several works introduce protocol-level countermeasures rather than modifying the PUF circuit itself.

Within this class of approaches, some works focus on privacy-preserving authentication for IoT devices. In these schemes (e.g., [67], [68], [69]), the PUF is modeled as a Silicon-based Random Function (SiRF) and used within challenge–response protocols to reduce information leakage and prevent device tracking. Techniques such as response obfuscation, secret sharing, or session-specific derivations are applied to limit the exposure of individual CRPs. For example, Chen et al. [47] leverage Shamir's secret sharing to distribute trust across multiple responses and reduce information leakage from individual CRPs. While these schemes improve privacy during authentication, they still rely on repeated external access to PUF responses during normal operation. As a result, they operate primarily at the protocol level and remain decoupled from the device boot process and from system-wide trust establishment.

Other works aim to strengthen the PUF primitive through architectural modifications. Multi-line arbiter PUFs [48] and improved MA-APUF designs [49] modify the internal structure of arbiter PUFs to improve uniqueness, reliability, and resistance to machine-learning-based modeling attacks. These designs are mainly evaluated in terms of prediction accuracy, entropy, and hardware overhead, and are typically intended for authentication scenarios.

In contrast, weak PUFs, such as SRAM- and RO-based designs, expose only a limited CRP space and are commonly used for key generation rather than challenge–response authentication. Because weak PUFs outputs are typically accessed only during boot or provisioning and are not exposed through repeated queries, they are less susceptible to modeling attacks. Several surveys and experimental studies [50], [51], [52] discuss how prediction-based attacks mainly target strong PUFs, while weak PUFs are more often affected by reliability and noise-related issues.

Unlike strong-PUF or SiRF-based authentication schemes, this work uses a weak RO-PUF as a stable entropy source to derive a device-unique secret during the immutable boot phase. The PUF output is not used for repeated authentication challenges, but to anchor a HRoT that supports secure

TABLE 2. Comparison of selected hardware/software RoT proposals.

Project	Identity Root	Attestation	Identity Lifecycle	Notes
OpenTitan [13]	Stored keys	DICE (internal)	Limited	Standalone RoT, high complexity
Caliptra [14]	Stored keys	DICE (internal)	Limited	Datacenter/edge focus
Sancus [25], [63]	Stored secret	Symmetric	None	MSP430, symmetric-only
Sanctum [26]	PUF-derived key	Enclave attestation	None	Custom RISC-V ISA, no DICE
Rolling DICE [24]	Software keys	Lightweight DICE	None	IoT-ready, no hardware root
This work	PUF-protected UDS	DICE-based platform and enclave attestation	**Full**	Boot ROM integration, support to identity renewal/zeroization

boot, DICE-based identity derivation, attestation, and identity lifecycle management.

C. PUF DESIGN VARIANTS AND ALTERNATIVE USES

Beyond authentication and key generation, PUFs have been explored in a variety of alternative roles. One line of work studies intrinsic PUFs as sensors, where the sensitivity of PUF responses to environmental parameters such as temperature or voltage is exploited for sensing purposes. For example, intrinsic DRAM- and RO-based PUF sensors have been proposed to detect environmental changes while simultaneously providing device identification [70]. These designs intentionally use PUF instability as a signal and are therefore fundamentally different from identity-oriented uses.

Other works integrate PUFs into higher-level systems outside the boot and trust-establishment context. For instance, some recent studies explore PUFs in distributed or collaborative learning frameworks, where PUF-derived secrets are used to protect model updates or enforce device-level participation policies [71], [72]. In these cases, PUFs serve as supporting primitives within broader security or privacy-preserving protocols, rather than as roots of trust for the platform boot process.

While existing PUF-based works primarily focus on improving authentication protocols, enhancing PUF robustness against modeling attacks, or enabling specialized applications such as sensing or distributed security mechanisms, this work does not introduce a new PUF construction or authentication protocol. Instead, we use a weak RO-PUF as a stable entropy source to derive a device-unique secret that anchors a HRoT. Our focus is on integrating PUF-derived identity into the immutable boot ROM, combining it with DICE-based layering, and managing identity across the device lifecycle, rather than on optimizing PUF-level metrics.

IV. THREAT MODEL AND SECURITY REQUIREMENTS

This section defines the threat model and the associated security requirements for the proposed robust device identity architecture integrating DICE with a silicon PUF tailored for RISC-V-based embedded and IoT systems. Our threat model considers the complexity and diverse deployment environments of modern IoT devices utilizing the Keystone TEE framework, with particular focus on hardware and firmware security.

A. THREAT MODEL

Our threat model considers adversaries with varying capabilities, targeting hardware, firmware, and software layers of IoT and embedded systems.

- *Physical Attacker*: Physical attackers have the capability to intercept, modify, inject, or replay signals external to the device package. This adversary is physically close to the device but does not possess the capability to directly tamper with internal silicon components such as CPU logic, the embedded PUF circuit, internal cryptographic accelerators, or boot ROM. However, external interfaces, such as network connections, debugging ports (e.g., JTAG), memory buses, and external storage, are susceptible to physical manipulation.
- *REE Software Attacker*: Adversaries controlling the REE possess extensive capabilities, including modification of the operating system kernel, drivers, system calls interception, and manipulation of firmware. Such attackers may attempt to illicitly access sensitive data, particularly the UDS, undermining the secure device identity. Additionally, attackers can launch unauthorized enclave instances or modify existing ones, posing risks to the integrity of enclave applications and the confidentiality of their data.
- *TEE Software Attacker*: Software components within an enclave-based TEE (including Security Monitor (SM), Runtime (RT), and Enclave Application (Eapp)) can contain residual vulnerabilities that may be exploited by sophisticated adversaries. A TEE software attacker seeks to leverage these vulnerabilities to compromise enclave isolation, manipulate or corrupt the firmware, and potentially extract critical device identity secrets such as the UDS.
- *Side-Channel Attacker*: These attackers exploit indirect information leakage via measurable side-effects (e.g., timing, power consumption, electromagnetic emissions, or cache utilization) to compromise sensitive data,

including cryptographic keys and confidential device identifiers.

- *Identity Lifecycle Attacker*: Attackers may target identity lifecycle stages (e.g., generation, provisioning, operational usage, renewal, and deletion) to compromise device authentication secrets. For instance, an adversary might attempt unauthorized extraction or modification of identity secrets during provisioning or exploit weaknesses during identity renewal procedures. Such compromises could lead to persistent impersonation, denial-of-service due to identity conflicts, or loss of trust in devices' cryptographic identities over their operational lifespan.

With respect to PUF-based identity reconstruction, the threat model assumes that an adversary may obtain access to externally stored metadata, including Helper Data (HD) and Challenge Masks (CMs), but does not have access to raw PUF responses, repeated noisy measurements, or intermediate reconstruction values. In particular, PUF evaluation and identity reconstruction are confined to the immutable boot ROM and protected hardware logic, and their outputs are not exposed to software executing after the boot transition. This assumption is consistent with prior work on Error Correcting Code (ECC)-based Helper Data Algorithms (HDAs) and reflects the intended deployment of the proposed HRoT.

B. MITIGATION AND SECURITY CONTROLS

To effectively address the threats previously outlined, we integrated in our architecture a comprehensive set of security controls spanning both hardware and firmware layers, enhanced by robust identity lifecycle management strategies.

Our architecture implements an immutable hardware-based RoT, integrating DICE capabilities with silicon-based PUF. This combined approach securely generates cryptographic identities directly within the device, thereby eliminating the need to store sensitive authentication data persistently. Critical authentication secrets, such as DICE-compliant private keys, remain confined within the hardware-based TEE, thus significantly reducing vulnerabilities associated with physical extraction and identity theft. Furthermore, dedicated hardware cryptographic accelerators are utilized to enhance security and efficiency in cryptographic operations. These accelerators minimize potential side-channel vulnerabilities, such as timing analysis attacks, by providing isolated, hardware-managed execution environments. This approach not only strengthens cryptographic processes but also reduces software complexity, thereby decreasing the attack surface.

An immutable, ROM-based Root of Trust for Detection (RTD) establishes the foundation for the secure boot process, ensuring the integrity of firmware from the earliest stages of device initialization. By validating cryptographic signatures of firmware components, this mechanism robustly prevents unauthorized firmware modifications originating from either REE or TEE attackers, thus safeguarding the integrity of critical security functions, such as those implemented in the Keystone SM, throughout the device lifecycle. Moreover, the integration of a DICE Layering Architecture, at the boot ROM and firmware level, provides robust hardware-rooted cryptographic identities for all enclaves within the TEE. Such DICE-compliant identities facilitate standardized attestation procedures and strong EApps authentication, enabling rapid detection and response to any software compromises affecting enclave trustworthiness.

In addition to these hardware and firmware measures, our design incorporates secure identity lifecycle management procedures. This includes secure on-device identity generation, protected identity provisioning processes incorporating integrity and confidentiality controls, as well as mechanisms for identity renewal and revocation. Collectively, these practices enable effective detection of untrusted device identities and misuse of legitimate identities during device operation, ensuring the prompt revocation and renewal of identities that have been compromised or become outdated. This significantly reduces long-term security risks associated with persistent cryptographic identities.

Table 3 summarizes the mapping between the threat classes defined in Sec. IV-A, the corresponding mitigation mechanisms, and the evaluation evidence presented in Sec. VIII.

C. SECURITY REQUIREMENTS

The security controls presented above explicitly fulfill several embedded device security requirements. Specifically, our architecture addresses the following Embedded Device Requirements (EDRs):

- (EDR 1) Integrity of Executed Code: All code executed within enclaves is subject to strict integrity verification, mitigating risks from malicious or unauthorized dynamically loaded code.
- (EDR 2) Protection Against Malicious Software: Secure boot, enclave-based TEEs, hardware accelerators, and continuous attestation jointly prevent execution of untrusted firmware/software and detect compromises in enclave applications.
- (EDR 3) Physical Tamper Resistance and Detection: The integration of silicon-based PUF and dedicated hardware accelerators provides resilience against physical tampering aimed at extracting secrets or altering device behavior. PUF-derived identity secret is never exposed externally, not even during provisioning.
- (EDR 4) Secure Provisioning of Manufacturer Trust Anchors: Device identities are generated on-device using the PUF, ensuring that cryptographic RoTs are securely established within hardware and remain confidential.
- (EDR 5) Secure Provisioning for Asset Owners: Our architecture supports controlled provisioning of additional credentials by asset owners (e.g., end users or

TABLE 3. Mapping between threat classes, mitigation mechanisms, and evaluation results in Sec. VIII.

Threat class (Sec. IV-A)	Mitigation mechanisms	Evaluation evidence (Sec. VIII)
Physical attacker	PUF-derived root secret; no persistent keys; boot-time DRK check; PMP isolation	PUF replacement (Table 10); metadata tampering (Table 9); PUF robustness evaluated in [56]
REE software attacker	Secure boot; DICE derivation before REE transition; root identity isolation	Secure boot tests (Table 8); REE key access denied (Table 10)
TEE software attacker	SM-managed keys; PMP-based enclave isolation; enclave-specific CDI derivation	Enclave key access denied (Table 10)
Side-channel attacker	Constant-time hardware crypto; early-boot confinement of DRK reconstruction	Stated mitigations and limits; no power/EM tests (Sec. VIII-B)
Identity lifecycle attacker	Authenticated provisioning; CSR and metadata checks; fail-secure boot on DRK/cert mismatch	Replay and metadata attacks rejected (Table 9)

enclave developers), based on the device's intrinsic trustworthiness. Thanks to TEE isolation, these credentials remain distinct yet interoperable with manufacturer identities.

- (EDR 6) Integrity of the Boot Process: An immutable boot ROM enforces secure boot verification, detecting and blocking compromised firmware or unauthorized boot sequences from the very first instruction set.
- (EDR 7) Lifecycle Management of Device Identities: Our design explicitly supports identity generation, revocation, and renewal, ensuring that compromised or outdated identities can be promptly and securely replaced. This capability is critical for maintaining continuous trust in long-lived IoT deployments.

Compliance Note: While presented here as general security requirements, these measures align with IEC 62443-4-2 recommendations for embedded devices, particularly at Security Level 2 and above, demonstrating that the proposed architecture is consistent with both academic and industrial security baselines.

V. HARDWARE ROOT OF TRUST ARCHITECTURE

This section presents the design and implementation of the proposed HRoT, which forms the foundation of our security architecture. Our approach integrates multiple hardware modules directly within the immutable boot ROM, anchoring trust at the earliest instruction set and eliminating dependence on mutable storage for critical secrets.

Figure 5 illustrates the high-level architecture. Concretely, the HRoT unifies: (i) a Ring Oscillator (RO)-based PUF/TRNG for unclonable identity derivation and true randomness; (ii) a Secure Hash Algorithm 2 (SHA-2) engine supporting multiple variants for integrity and key-derivation tasks; (iii) an EdDSA25519 digital signature accelerator ensuring efficient, constant-time signature verification and generation; and (iv) a boot ROM that orchestrates secure boot, DICE layering, and certificate issuance. By embedding these capabilities into the boot ROM, our HRoT simultaneously implements the four classical RoT roles (RTD / RTM / RTR / RTS), protecting critical RoTs functionalities and secrets by leveraging RISC-V PMP hardware primitive. All the modules are connected by using an Advanced eXtensible Interface (AXI)-Lite interface that allows being invoked during the operations.

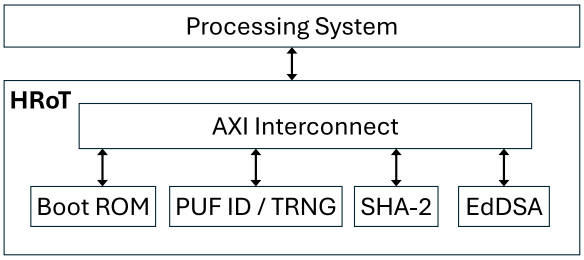

FIGURE 5. Proposed Hardware Root of Trust (HRoT) integrating PUF, SHA-2, EdDSA, and Boot ROM logic. Each module contributes to the four classical RoT functions (RTD, RTM, RTR, RTS).

A. PHYSICAL UNCLONABLE FUNCTION IMPLEMENTATION

The PUF anchors device identity at the hardware level. Our implementation employs RO pairs to generate challenge–response values. The PUF module generates a response as a bitstream generated by concatenating bits derived from values selected after a series of comparisons. The challenge-response is obtained by comparing two ROs, each incrementing its own counter. When one of the counters reaches saturation, the values of the most suitable selected bits from the slower counter are used to generate the ID. After a sequence of challenges, the PUF response is produced, serving as an identifier generator. Furthermore, a hardware challenge-selection mechanism is incorporated to discard the least reliable challenges, thereby enhancing the overall stability of the PUF response.

The basis for this PUF is a configurable RO built from three inverter stages and an enable stage. Each inverter stage uses four parallel inverters driven by the same input, with the output chosen through a multiplexer controlled by two bits. The final stage provides two inputs to support row and

TABLE 4. Comparison of FPGA RO-PUF implementations.

Characteristic	Ideal value	This work	PRO-PUF [54]	XCRO [73]
Functions	–	**RO-PUF + TRNG + EM-aware layout**	Programmable-delay RO-PUF	XOR-CRO (anti-EM layout)
$\mathbf{HD_{intra}}$	0% – perfect reproducibility (same response under identical conditions)	**0.2%** (after selection)	1.99%	2.63%
$\mathbf{HD_{inter}}$	50% – perfect uniqueness across devices	**49.2–49.5%**	44.79%	48.85%
Uniformity	50% – equal probability of 0s and 1s in each response	**48.8–51.8%**	49.74%	–
Bit aliasing	50% – balanced probability of each bit across multiple devices	**49.1–51.5%**	48.63%	–
Resources	Lower = better	4 CLBs	Not reported; claims $> 100\times$ efficiency vs. classic RO/CRO	4 CLBs (7 delay units)

column enablement within an RO bank and completes the feedback loop through four selectable paths, also controlled by two bits. The design can be duplicated using the eight 6-input Look-Up Tables (LUTs) available in each Configurable Logic Block (CLB) of Xilinx Series 7 devices, where the multiplexer functions are implemented directly within the LUTs, avoiding the use of extra logic or routing resources. The structure, therefore, provides a total of eight configuration bits that allow 256 configurations to be selected, which is equivalent to implementing 512 different ROs in each CLB of the Xilinx Series 7 devices. Therefore, an array of only 4×4 CLBs is sufficient to host 32 configurable ROs (equivalent to $32 \times 256 = 8192$ ROs) capable of providing up to $32K$ bits at the output of the PUF/TRNG. More detailed info about this module is described in [74].

Table 4 summarizes the performance of the proposed RO-PUF compared with other Field Programmable Gate Array (FPGA) implementations reported in the literature. Since the metrics used to characterize PUFs and TRNGs are statistical, a large sample set is needed to obtain meaningful results. For PUFs on programmable devices, this is typically addressed by placing multiple instances of the design in different locations on the chip or by using several FPGA boards. Yet, studies rarely report data from more than a hundred distinct samples. However, here, we used a test system that leverages the configurable RO architecture to overcome this limitation. A first system test is built by integrating ten PUF instances that replicate the structure into 8×15 CLBs composed of 128 different PUFs that extracts up to 1920 bits per PUF, enabling the evaluation of 1280 distinct PUFs or TRNGs on a single development board.

The average HD_{inter} values were obtained from the 10 instances across the 256 RO configurations. For each configuration, an enrollment phase calls the PUF 10 times to generate a reference output. The Hamming distance between this reference and 10 responses from each configuration of the other nine instances is then computed, totaling 1,024,000 evaluations. The average HD_{intra} values for the 256 configurations and 10 instances are obtained after an enrollment step similar to the previous case. Here, the Hamming distance is computed between the reference output and 10 additional responses of the same PUF under the same configuration (204,800 calls in total). The enrollment process is able to identify comparisons (challenges) that negatively affect PUF reliability. These challenges are recorded in a selection mask used to filter them out in subsequent evaluations. The resulting HD_{intra} shows the values after discarding 10% of the comparisons. The results reported in Table 4 are obtained from a single Kintex-7 FPGA board using ten spatially distinct PUF instances. Environmental robustness was evaluated separately across three boards under controlled voltage and temperature variations, as detailed in [56].

Experimental evaluation confirms high-quality identifiers: intra-device Hamming distance (HD_{intra}) within the range 0%-2% and as low as 0-0.2% after applying the challenge selection strategy, inter-device Hamming distance (HD_{inter}) within 49.2%–49.5%, and balanced uniformity and bit-aliasing metrics within 48–52%. This steady closeness to the ideal values of those metrics demonstrates the strong uniqueness and reliability of the presented PUF design. Moreover, as indicated in [74], the PUF used here confirms the trends reported in the characterization under non-nominal temperature and voltage conditions presented in [56], where a similar test system with ten instances of the PUF was run in three different boards. In that study, the PUF behavior was evaluated across a core voltage range of 0.95–1.05 V and an operating temperature range of 10–50 ° C, using the nominal condition of 1.0 V and 30 ° C as reference.

Furthermore, the embedded TRNG passes NIST SP 800-22 [75] and NIST SP 800-90b [76] statistical tests, and it is fully compliant with ISO standards [77], [78]. To perform the randomness analysis, the minimum number of bits required by the procedures provided by NIST to evaluate the entropy

associated with the source (one million bits) was captured for each of the 20 IPs in the test systems. Although NIST tests allow successive TRNG outputs to be concatenated to form longer sequences, the original structure of the data was maintained as 31 sequences of 32,768 bits. These results validate both the uniqueness and entropy of the proposed PUF/TRNG module, ensuring resilience against cloning and replay attacks.

Moreover, as discussed later, the response of the PUF as an identifier will be used to obfuscate a random number that will serve as the system key. To this end, we have evaluated the behaviour of this application, which is based on a HDA [79] using a ECC based on majority vote to compensate bit instability. Several scenarios were considered, using between 25% and 100% of the available comparisons to obfuscate and recover keys of different sizes (256 to 4098 bits) using different repetition codes from 5 to 13. In all the cases, the keys are recovered successfully using the correct PUF instance and the process is repeated one hundred times. However, different PUF instances never generate the key. To evaluate testability and security, obfuscation and key recovery processes have been performed one million times with successful results for a key size of 512 bits. As detailed in [79], HDA is not necessarily secure against physical attacks. The leakage-resistance of HDA is out-of the-scope of this work. However, our approach never reveals information about the PUF response as stated in the Threat Models Section, making more difficult a potential attack.

B. SHA-2 IMPLEMENTATION

The hardware SHA-2 module in the proposed HRoT follows a lightweight architecture derived from [80] and optimized for seamless integration within the RoT of a RISC-V platform. The design prioritizes low resource utilization and high operating frequency, achieving efficient hash computation suitable for embedded and IoT devices.

A central feature of the architecture is the *Message Schedule Unit*, implemented around a Non-Linear Feedback Shift Register (NLFSR). During the initial loading phase, the first sixteen message blocks are stored in the NLFSR, which subsequently generates the following coefficients on the fly. This streaming approach eliminates the need for precomputed message scheduling, reducing latency and memory footprint. The coefficients generated by the NLFSR feed directly into the concurrent SHA-2 round computations, improving throughput and parallelism. Furthermore, the on-the-fly generation ensures a constant number of cycles per block, enhancing the timing determinism of the hash operation, a key property for dependable and side-channel-resilient execution in security-critical environments.

The proposed design supports all standardized SHA-2 variants (i.e., SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256) through parameterized configuration. To enable this flexibility, the corresponding initial hash values and the k constants for each variant are stored in on-chip ROM. A dedicated control unit coordinates the operation of the message scheduler, round computation, and padding logic, ensuring protocol compliance with FIPS 180-4 [81]. An input memory and decoder manage data synchronization between the external interface and the hashing core, while a padding submodule autonomously handles the message padding required by the SHA-2 hashing algorithm.

Within the overall HRoT, the SHA-2 core contributes to firmware integrity validation during secure-boot and to the derivation of the CDI in the DICE layering chain. The AXI-Lite interface allows the boot ROM to invoke the module directly during boot operations, providing an efficient link between hardware hashing and the attestation mechanism. To strengthen system dependability, all SHA-2 instances operate with continuous monitoring of voltage, temperature, and clock-integrity anomalies, ensuring resilience against low-cost fault-injection attempts.

The SHA-2 module was synthesized on a Xilinx Kintex UltraScale device, achieving up to 1.86 Gb/s throughput and 11.3 Mb/s/slice efficiency for the SHA-256 variant, with a resource occupation of only 165 slices. The comparative results reported in Table 5 are based on synthesis on a Virtex-7 device, in line with the configurations adopted in [82], [83], [84], and [85], to enable a fair, device-class-consistent comparison. Compared with representative SHA-2 implementations in the literature, the proposed design achieves the lowest area occupation and the highest area efficiency, while maintaining a competitive throughput. This efficiency stems from the compact message-scheduling architecture and reduced memory footprint, which make the core particularly well suited for resource-constrained HRoT deployments. The resulting SHA-2 core thus combines high efficiency with timing determinism and side-channel resilience, key properties for secure and dependable operation within a HRoT.

C. EDDSA25519 IMPLEMENTATION

The EdDSA25519 scheme is a high-performance and side-channel-resilient digital signature primitive built on Twisted Edwards curves over the prime field $\mathbb{F}_p$, with $p = 2^{255} - 19$. Conforming to RFC 8032 [86] and FIPS 186-5 [87], it provides deterministic signatures, constant-time execution, and compact credentials (32-byte public keys and 64-byte signatures), making it ideal for embedded RoTs.

The curve equation involves specific parameters defined modulo a large prime, and its use of unified addition laws streamlines arithmetic operations. Public keys are generated through scalar multiplication of a base point $\mathcal{G}$, while signature creation combines hashing, modular arithmetic, and scalar multiplication following the RFC standard. The scalar multiplication scheme employs a constant-time "double-and-add" algorithm. Each 256-bit scalar is processed bit by bit from the most significant to the least significant, starting at the identity point. For each bit, the accumulator point is

TABLE 5. Comparison of recent SHA-2 hardware implementations on FPGA.

Algorithm	Work / Platform	Area (Slices)	Freq. (MHz)	Throughput (Gb/s)	Efficiency (Mb/s/Slice)
SHA-256	[82], Virtex-5	387	202.54	1.580	4.190
	[83], Virtex-4	610	170.75	1.345	2.200
	[84], Virtex-5	1895	411.30	3.290	1.740
	This work, Virtex-7	**282**	**191.13**	**1.630**	**5.784**
SHA-512	[82], Virtex-5	874	176.06	2.200	2.580
	[85], Virtex-5	1080	129.00	0.826	0.765
	This work, Virtex-7	**576**	**173.52**	**2.221**	**3.856**

doubled, and the base point is conditionally added only if the bit is one. This uniform control flow prevents timing and power variations, eliminating side-channel leakage from data-dependent stalls and guaranteeing execution determinism, essential properties for dependable HRoT subsystems. Moreover, it removes the need for precomputation, and minimizes memory requirements.

The hardware accelerator, described in [88], integrates two parallel modular adders/subtractors, a pipelined four-level Karatsuba multiplier, and a dedicated modulo-L reduction unit orchestrated by a finite-state controller. Arithmetic operations employ Extended Projective Coordinates ($X : Y : T : Z$) to avoid modular inversions during point operations, while interleaved reduction techniques optimize modular arithmetic with both the prime p and the curve order L. This pipeline enables overlap of multiplication and reduction phases, achieving constant latency per operation and uniform power signatures. A configurable control port allows the same engine to support scalar and modular multiplications for other cryptographic blocks within the RoT.

The EdDSA accelerator was synthesized on a Xilinx Zynq SoC and Kintex-7 device. Resource utilization and performance are summarized in Table 6, together with representative state-of-the-art implementations. Despite operating at a lower frequency, the proposed accelerator achieves the highest throughput among comparable FPGA realizations thanks to its parallel modular pipeline and full Digital Signal Processing (DSP) utilization. The use of Extended Projective Coordinates eliminates inversion bottlenecks, while interleaved modular reduction ensures stable latency and data-independent timing. From a dependability standpoint, the design enforces constant control flow and deterministic latency across all scalar values, guaranteeing reproducibility and side-channel resistance. Within the HRoT, the EdDSA25519 core enables fast and verifiable firmware authentication and secure key derivation for DICE layering, complementing the SHA-2 hashing engine and strengthening the overall trust chain.

As an overview, the HRoT presented in this work incorporates countermeasures against timing attacks by providing time-constant implementations across its modules. In addition, it addresses Side-channel Attacks (SCA) threats, including power analysis and fault injection attacks. In the latter case, dedicated monitoring logic is included to raise an alarm upon the detection of anomalies in control signals, as well as variations in temperature or power supply.

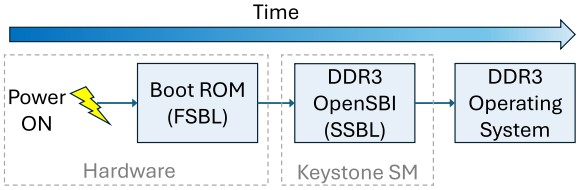

FIGURE 6. Keystone boot process.

The HRoT was synthesized on the Pynq-Z2 board for comparison purposes and evaluated against the resource utilization reported for OpenTitan [92]. It should be noted that this comparison is not strictly fair, as the implementations target different platforms. Nevertheless, as shown in Table 7, the proposed RoT exhibits a smaller area than OpenTitan. Furthermore, a direct area comparison with other RoT solutions is of limited relevance, since their implementations are typically based on embedded microcontrollers (e.g., Sactus) or RISC-V processors (e.g., Caliptra), where security algorithms are executed in software; consequently, the reported area mainly reflects the footprint of the soft-core processors. In contrast, the HRoT presented in this work relies on dedicated, fully hardware-based implementations of the security primitives, which fundamentally differentiates it from other RoT architectures.

D. HARDWARE-ANCHORED SECURE BOOT PROCESS

The original Keystone architecture, which serves as the starting point of our implementation, employs a three-stage boot process (illustrated in Fig. 6) which begins with the immutable boot ROM embedded within the HRoT. Upon power-on or system reset, the boot ROM initializes hardware components, including peripherals and Double Data Rate Type 3 (DDR3) memory, and then loads all partitions stored on the Secure Digital (SD) card into memory. Execution control is subsequently transferred to the OpenSBI firmware [93], which functions as a Second-Stage Boot Loader (SSBL). In the Keystone implementation, OpenSBI incorporates Keystone's SM as an integrated extension. The extended firmware initializes essential platform components

TABLE 6. Comparison of Ed25519 hardware implementations.

Work / Platform	Slices (CLBs)	DSPs	Freq. (MHz)	Signatures/s	Verifications/s
[89] STM32F401	–	–	84	154	63
[90] Zynq SoC	2192	0	137.5	–	–
[91] Zynq SoC	4128	16	82	303	272
This work, Zynq SoC	**4124**	**108**	**62.5**	**508**	**654**

TABLE 7. Occupation in CLBs of the OpenTitan and this work RoT .

HRoT	CLBs	Platform
OpenTitan [92]	6729	Zynq-7020
This work RoT	6327	Zynq-7000

and establishes a TEE managed by the SM. Finally, control passes to the Linux kernel, marking the completion of the boot sequence and initiating the device's runtime state.

A fundamental characteristic of the boot mechanism is that each stage implicitly trusts its predecessor, accepting configurations and parameters without independent verification. For example, the OS depends entirely on the integrity of the bootloader and lacks the capability to autonomously distinguish between legitimate and compromised bootloader states. Thus, achieving comprehensive boot security requires initiating protection from the very first stage of the boot sequence, specifically within the immutable boot ROM. Establishing a robust hardware-level RTD, capable of detecting software compromises, enables each trust relationship to be securely traced back to the platform's initial trusted state, thus protecting subsequent boot stages against unauthorized modifications.

In a classical secure boot paradigm, each stage verifies the integrity and authenticity of the next before execution. The RTD and the subsequent components forming the Chain of Trust for Detection (CTD) employ cryptographic verification techniques to distinguish legitimate software from tampered code. Commonly used cryptographic techniques include:

- *Digital Signatures*: Provide robust security without requiring RTD modifications when software components are updated. Only the signature verification logic and the public keys from authorized providers are needed.
- *Hash Functions*: Favored in resource-constrained systems for their computational simplicity, though they require RTD updates after each software change.
- *Encryption*: Provides additional confidentiality benefits and avoids RTD updates upon software changes, but does not support Execution In Place (XIP).

Our implementation adopts digital signatures due to their superior flexibility in accommodating updates without altering the RTD or CTD. Specifically, we employ the EDDSA25519 digital signature scheme for its strong cryptographic assurances and computational efficiency. To enhance both performance and robustness, verification is accelerated

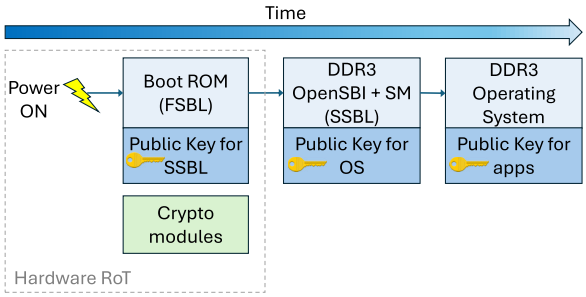

FIGURE 7. Steps in the secure boot chain.

through hardware cryptographic cores (see Sec. V-C), enabling efficient Ed25519 signature validation. In this approach, each boot component securely retains the public verification key necessary to authenticate the subsequent stage, as depicted in Fig. 7. The boot ROM stores the public key for verifying the SSBL (OpenSBI with integrated SM), while OpenSBI holds the key required to validate the Linux kernel.

Although confidentiality of verification keys is not required, their integrity is critical. Any unauthorized key replacement could allow execution of malicious software signed by untrusted entities. To prevent such tampering, the initial verification key is embedded directly within the immutable hardware boot ROM, physically preventing modification (see Fig. 8). The boot ROM employs this embedded key to authenticate the SSBL, while the SSBL contains the public key needed to verify the OS kernel, thus preserving the integrity of the entire boot sequence. Authenticated components stored on the SD card include their corresponding digital signatures, which are loaded into DDR3 memory together with the components. This integrity mechanism can be further extended to validate kernel modules and user-level applications by embedding additional public keys within the OS kernel, leveraging the Linux Integrity Measurement Architecture (IMA) module's appraise functionality.

Comprehensive validation of the secure boot mechanism was carried out across all critical components: the OpenSBI firmware (including SM), Linux kernel, boot parameters, and root filesystem. Given the performance constraints typical of IoT devices, our implementation enforces mandatory secure boot verification only for the SSBL through hardware logic in the boot ROM. Authentication of the OS and filesystem remains optional and can be selectively enabled by firmware

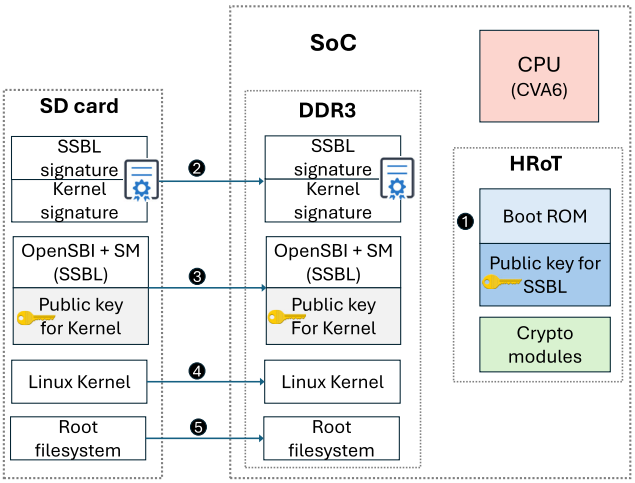

FIGURE 8. Components involved in Secure Boot process.

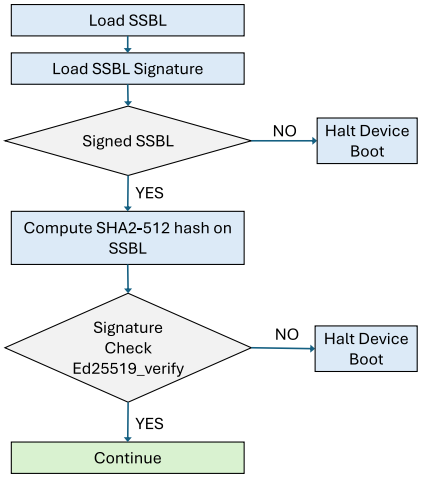

FIGURE 9. Firmware verification steps performed by the boot ROM RTD.

or software providers to extend the CTD up to the OS or application level. Fig. 9 summarizes the firmware verification steps executed by the boot ROM (RTD).

E. CHAIN OF TRUST ESTABLISHMENT WITH DICE

Building upon the hardware-enforced secure boot foundation described in Section V-D, our architecture embeds the DICE logic directly into the immutable boot ROM, forming a cohesive HRoT. This integration enables secure derivation of cryptographically verifiable device identities and supports critical security processes, including remote attestation of system components throughout the device's runtime. The foundational element of the HRoT is the integration of DICE for deriving identities for the host firmware and software components. At its core, the DICE-based HRoT derives dynamic device identities anchored in the PUF-generated UDS. The procedure for generating and dynamically reconstructing the device identity at each boot is described in detail in Section VI. This hardware-level integration eliminates the need for persistent secret storage, while ensuring that each boot instance securely regenerates the same identity, provided that no hardware modification has occurred. The DICE logic also supports the creation of cryptographic credentials required for attestation and authentication of subsequent components, thereby unifying secure boot, secure device identification, and system components attestation in a single trusted hardware root. During secure provisioning, the following cryptographic components are made accessible to the boot ROM:

- *Manufacturer Certificate* (optional): an X.509 certificate serving as either a self-signed Root CA or a subordinate CA certificate issued by an external Root CA;
- *DRK Certificate*: an X.509 certificate authenticating the DRK, issued and signed by the manufacturer during secure identity provisioning protocol;
- *PUF Mask* and *Helper Data*: metadata generated during the provisioning process to enable consistent PUF-based regeneration of the DRK during each boot.

Upon every power cycle, the DRK is deterministically reconstructed from the UDS derived via the PUF, guaranteeing consistent identity recovery unless hardware tampering occurs. The regeneration process, detailed in Section VI-A2 and illustrated in Fig. 12b, ensures resilience and repeatability even under minor physical variations.

After successful reconstruction of the DRK, the boot ROM derives the cryptographic material required by the subsequent firmware layer, following the sequence illustrated in Fig. 10. The steps of the DICE-based HRoT are as follows:

1) derive the UDS from the PUF module (Section VI-A2);
2) derive the DRK from the UDS using a Key Derivation Function (KDF);
3) compute the TCB Component Identifier (TCI) of Layer 0 (TCI_{L0});
4) derive the CDI for Layer 0 (CDI_{L0}) by combining the UDS and TCI_{L0};
5) generate an ECA key pair using CDI_{L0} as input to a KDF;
6) issue an X.509 ECA certificate, signed by the DRK, embedding the `DiceTcbInfo` extension, as defined in the DICE Attestation Architecture [28]. This certificate provides attestation evidence for the first mutable code (the Keystone SM), explicitly identified through a structured subject field ("CN=Security Monitor UUID=<...>");
7) securely transfer the generated cryptographic data (i.e., the ECA key pair, certificate and CDI_{L0}) to the SM for remote attestation and authentication tasks.

The hardware implementation employs the same cryptographic primitives introduced in previous subsections: the SHA-2-512 module (Section V-B) for hashing and the Ed25519 accelerator (Section V-C) for key generation and signature operations. Together, these modules ensure constant-time, high-efficiency cryptographic execution

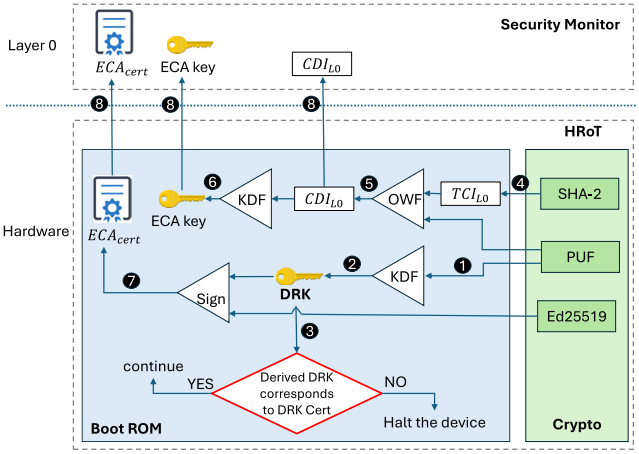

FIGURE 10. DICE Hardware RoT implementation.

suitable for embedded and IoT platforms while maintaining compliance with the DICE attestation framework. Through this design, the system establishes a verifiable, hardware-anchored identity chain that bridges the immutable boot process and higher software layers. This foundation enables the lifecycle-aware identity management mechanisms discussed in Section VI.

VI. LIFE CYCLE OF SECURE DEVICE IDENTITIES

Having detailed the technical implementation of our proposed hardware RoT, this section presents the structured lifecycle approach we designed and implemented for managing secure device identities. Our approach encompasses three critical phases, as represented in Fig. 11: (1) generation and secure provisioning of device identities, (2) secure management and continuous monitoring during operational deployment, and (3) renewal and revocation protocols designed to quickly mitigate potential identity compromises and maintain system trustworthiness.

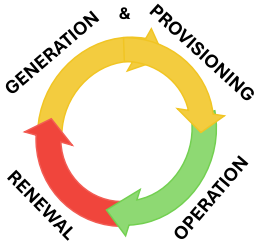

FIGURE 11. Lifecycle of Secure Device Identities.

A. STRONG IDENTITY GENERATION AND PROVISIONING

The secure and reliable generation of cryptographic identities on embedded devices is fundamental for establishing trustworthiness in hardware systems. Devices unique identities can be generated either externally, during manufacturing, and configured in the device in non-volatile and protected memory, or internally on the device itself. Internal, on-device identity generation is regarded as the most secure method, provided that generated secrets remain strictly within the device, never exposed externally ([19], [21]). In our approach, this principle is strictly followed; the UDS, from which the asymmetric DRK is derived, is generated leveraging the PUF, ensuring it remains unknown even to the manufacturer. However, on-device generation presents specific operational and practical challenges ([19]). One primary difficulty is certifying cryptographic keys generated internally by the device, without exposing sensitive information. We address these concerns through a carefully designed provisioning protocol inspired by industry-standard practices to securely generate and provision identities.

1) IDENTITY GENERATION USING PHYSICALLY UNCLONABLE FUNCTIONS

To generate a robust and device-unique identity, our methodology utilizes a PUF-based mechanism. PUFs exploits inherent manufacturing variability, offering device-specific responses to given challenges that are difficult to predict or replicate [21]. During the initial generation phase, the device produces a random seed through its TRNG facility; in our design, this random seed constitutes the UDS. Subsequently, the PUF responds to a carefully chosen set of challenges, producing a response with a size equal to the seed length multiplied by the ECC repetition factor (r), called the *ID frame*. Challenges exhibiting unstable responses are filtered out using a CM, ensuring consistency across repeated device startups. The HD is generated by XOR-ing the extended seed with the ID frame, as depicted in Fig. 12a.

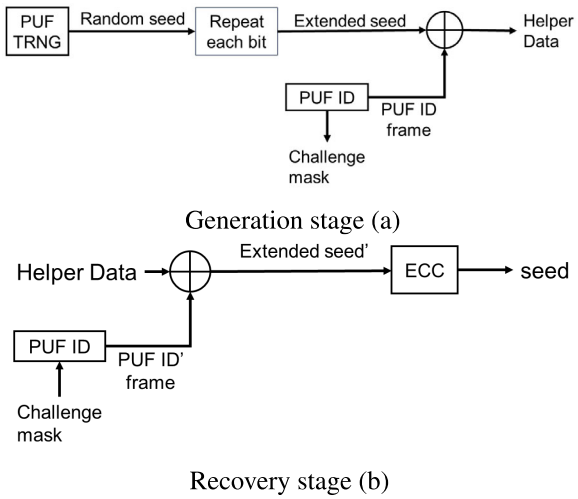

FIGURE 12. Helper Data algorithm per PUF seed generation and recovery.

2) PUF's IDENTIFICATION RETRIEVAL

During the *recovery* or reconstruction stage, at device operation, the process has to be reversed as shown in Fig. 12b, so the device regenerates the original seed using the stored HD. The ID frame is reobtained from the PUF response by applying the selected CM, then it is XOR-ed with the HD, producing the extended seed. This result is

then corrected using the ECC algorithm, which applies a majority vote to eliminate minor differences in the ID frame, thus compensating for minor variances in PUF response. The recovered seed constitutes the original UDS produced during the identity generation phase. This method ensures high reliability and repeatability of the UDS, without directly exposing the raw PUF responses or the original seed.

An additional advantage of this design is its intrinsic fail-safe behavior. If extreme environmental drift or long-term silicon aging were ever to cause the PUF response to deviate beyond the correction capability of the employed ECC scheme, the device would simply be unable to reconstruct the original UDS. As a consequence, the derived DRK would no longer match the certificate provisioned during manufacturing, causing the secure-boot process to halt (see Fig. 10). This prevents any silent divergence in device identity and ensures that deviations, whether due to tampering or aging degradation, are immediately and reliably detected.

The architecture does not rely on persistent state to record a failed reconstruction attempt. Instead, recovery is driven by the availability of identity provisioning metadata. If DRK verification fails, the secure-boot process halts and the device cannot progress to later stages. Should the certificate or identity metadata (CM/HD) be removed on the external storage, the next reboot automatically triggers the provisioning pathway described in Section VI-A3. This allows the device to regenerate a fresh, valid identity without remaining in an unrecoverable state, while still ensuring fail-secure behavior during normal operation.

3) SECURE CERTIFICATE ISSUANCE AND PROVISIONING PROCESS

The secure issuance and provisioning of cryptographic certificates is essential for creating verifiable device identities and ensuring trustworthy provenance and configuration. Since an identifier generated by a PUF cannot be predetermined or externally replicated, this step must take place only after a specific stage in device manufacturing has been completed, as it is essential to power on the board and observe the generated values during the initial boot. To securely manage this phase, we developed a robust protocol, illustrated in Fig. 13, which facilitates the delivery of all necessary data for the PUF to generate an identifier for the board and being able to securely recreate it.

During the first boot of the device, conducted within a secure and controlled manufacturing environment, the device initiates the internal generation of an asymmetric key pair derived from the PUF-generated seed, ensuring this seed remains confidential and never externally exposed. Given the absence of a certificate at this stage, the device enters a personalization mode, designed explicitly for initial provisioning. In this mode, the device establishes a secure Transport Layer Security (TLS)-protected communication channel over a dedicated point-to-point Ethernet connection with the manufacturer's provisioning appliance. The authentication of this communication channel relies solely on the

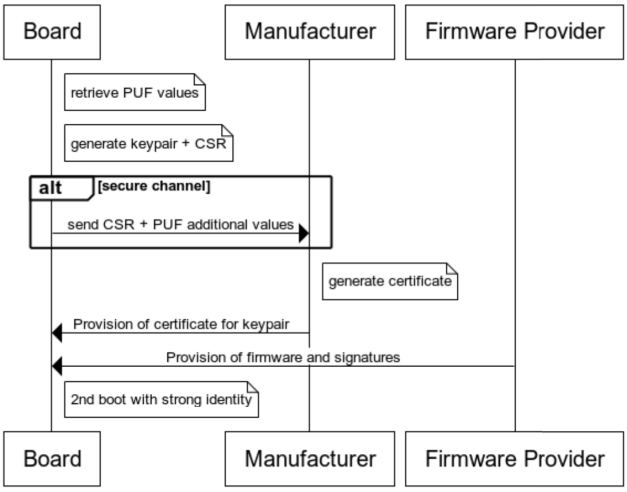

FIGURE 13. Schema on the protocol interaction.

manufacturer's pre-embedded public key, securely stored within the device's boot ROM. The device subsequently generates and transmits a Certificate Signing Request (CSR) alongside necessary PUF-related metadata, i.e., the CM and HD. The receipt of this data allows the manufacturer to verify device authenticity and generate the corresponding certificate without ever accessing the private key or the raw PUF-derived seed directly. Once the manufacturer signs and issues the device-specific certificate, it is provisioned in the flash memory of the device, along with the additional identity metadata (i.e., CM and HD) and firmware security metadata (i.e., firmware signatures for allowing secure boot), ensuring secure operational deployment.

During subsequent boots, the device leverages the previously stored CM and HD within the secure boot ROM to regenerate the original key pair. Since the regeneration of this identity depends strictly on an unaltered hardware configuration, any unauthorized hardware modifications will result in key regeneration failure.

This protocol ensures the platform to have a PUF-generated keypair accompanied by a valid certificate issued by the manufacturer, providing a strong identity for the board. Furthermore, if there are any attempts to alter the configuration, the board will be unable to regenerate the same keypair. This regeneration failure will prevent the board from having a valid certificate for the keypair, ensuring that any tampering is immediately detected.

The provisioning process also defines how the device can securely re-enter this personalization mode when needed. Since the HRoT does not maintain persistent error flags, the transition is governed solely by the presence or absence of valid identity metadata on external storage. If, during a future boot, the DRK cannot be reconstructed or validated against the provisioned certificate, the secure-boot mechanism halts execution (as described in Section VI-A2). Should the certificate or associated metadata then be removed or replaced (e.g., by an operator initiating re-provisioning),

the next boot will detect the absence of valid credentials and automatically enter the provisioning mode illustrated in Fig. 13. This provides a controlled and secure recovery pathway without requiring additional non-volatile state.

An important aspect of the proposed identity mechanism is that it does not require perfect PUF repeatability. Instead, it relies on three complementary safeguards: (i) stability filtering through the CM during enrollment, (ii) robust reconstruction using ECC-assisted HDA, and (iii) certificate-based verification of the DRK at boot. These combined measures guarantee that identity regeneration always succeeds correctly or fails securely, never producing inconsistent or ambiguous device identities.

B. SECURE DEVICE IDENTITIES IN OPERATION

To participate securely in a system, a device must have a trusted identity that allows it to authenticate and securely communicate with other devices and management infrastructure. The risk of physical tampering or targeted attacks against embedded devices is significant, given they are typically deployed in remote, uncontrolled, or vulnerable environments. Because of this, relying solely on a strong hardware-based identity established during the provisioning phase, although critical, remains insufficient for comprehensive security. A broader identity management and continuous monitoring system is essential for monitoring the devices' behavior and integrity throughout their operational lifecycle and promptly detecting anomalies indicative of identity compromises.

Operational monitoring involves regular checks and analyses of device behavior during authentication and interactions with management systems. By employing advanced anomaly detection techniques, it becomes possible to identify irregularities such as duplicated identities, unusual geographic location changes, or abnormal communication patterns. Upon detecting suspicious patterns, the identity management system initiates automated incident responses, isolating the affected device from the network pending further investigation. Moreover, robust identity management must also continuously validate the integrity and authenticity of the device's software environment, as any deviation from the expected software configuration may be used by attackers seeking to exploit residual vulnerabilities in the system to compromise the device's identity, or software component's identities derived from it. Integrating robust integrity and operational monitoring enhances the efficacy of secure identity management systems by rapidly identifying and mitigating threats in real time.

C. DELETION AND RENEWAL OF DEVICE IDENTITIES

In real-world systems, secure device identities are not immutable and inevitably reach the end of their validity even when the hardware may continue to be used in the field. For instance, web-server certificates commonly expire every 90 days, and emerging standards aim to limit TLS certs to as little as 47 days by 2029 [94]. Although industrial devices typically still carry long-lived X.509 certificates, often spanning decades, this longevity is misleading. The rapid evolution of cryptography, operational environments, and compliance requirements means that a certificate issued today may lack trustworthiness 15 or 20 years later.

An interesting example of weak renewal mechanisms comes from the 2022 German health-telemedicine gateway case [95]. The manufacturers of special security gateways, required to connect to the telematics health data network, claimed that devices required physical replacement following the expiration of their five-year certificates. The Chaos Computer Club demonstrated that it was possible to avoid the expensive hardware replacement by applying a patch to the open source components of the device firmware for having it to use an additional set of renewed certificates in addition to the expired ones, so averting what would have been a €400 million cost to the German healthcare system. Such case underscores how deficient renewal mechanisms can lead to unnecessary replacement and financial loss, highlighting the need for flexible and robust certificate renewal processes. Other domains present similar risks: in critical infrastructure and vehicular networks, expired or non-updated certificates have resulted in temporary service disruptions or forced legacy decommissioning, highlighting gaps in machine-identity lifecycle management [96].

Our HRoT and key generation mechanism support seamless identity renewal without requiring physical removal or reprogramming of key material in non-volatile memory. Since the asymmetric DRK is deterministically derived from the UDS, defined as a random seed protected by the PUF-generated ID-frame, identity regeneration becomes a software-driven operation. The re-execution of the enrollment protocol allows full cryptographic regeneration under a fresh certificate, preserving continuity of identity trustworthiness without touching the hardware. However, while the first generation process can be executed in the protected manufacturer environment, the renewal protocol has to be executed remotely, as the procedure in the manufacturer factory would be impractical for embedded and IoT devices deployed in the field.

To align with emerging standards for automated IoT certificate lifecycle, our renewal protocol adheres to Enrollment over Secure Transport (EST) (RFC 7030 [97]). EST defines RESTful, TLS-authenticated Application Programming Interfaces (APIs) for certificate provisioning (`/simpleenroll`) and renewal (`/simplereenroll`). While EST was originally tailored for enterprise networking products, it lacks widespread adoption in IoT. To our knowledge, there are no commonly adopted IoT device renewal protocols beyond some proposals, e.g., Bootstrapping Remote Secure Key Infrastructure (BRSKI), RFC-8995 [98]. But industry efforts should point to automated procedures to realize scalable identity management systems [99]. The proposed procedure comprises two main phases, whose steps are represented in Fig. 14.

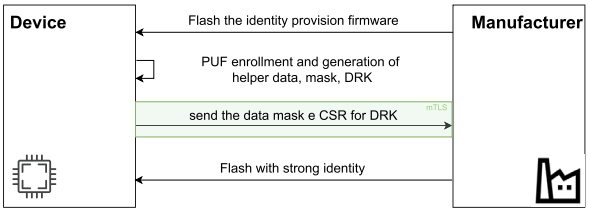

FIGURE 14. Remote Strong identity provision.

Initially, the manufacturer remotely provisions the device with a secure minimal firmware (''bootstrap and enrollment bitstream''), cryptographically signed to be accepted by the boot ROM secure boot mechanism. This firmware is equipped with a minimal Hypertext Transfer Protocol Secure (HTTPS)/TLS client stack, a trust anchor for the manufacturer's EST CA, and a temporary client certificate for initial mutual TLS authentication. The core of the bootstrap firmware is the PUF enrollment routine. Upon booting, the device executes this routine, retrieving the CM and the PUF ID-frame. Then, it deterministically derives a new DRK from a randomly-generated UDS, and generates the HD following the procedure represented in Fig. 12a. A CSR is generated from this DRK, preparing the data for certificate issuance.

Then, the device uses the standard EST re-enrollment protocol (`/simplereenroll`) to update its certificate. It initiates a mutual TLS session, authenticated by its current DRK certificate, to securely deliver its PKCS#10 CSR to the manufacturer's EST server. The server validates and returns a signed certificate encapsulated in a PKCS#7 message, establishing a new trustworthy device identity. This certificate replaces the old DRK certificate, allowing the device to reboot with its normal production bitstream with a valid, updated DRK certificate. Revocation mechanisms, including Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP), are integrated directly into the issued certificates via standard distribution Uniform Resource Locators (URLs), ensuring immediate revocation of compromised or outdated keys. To prevent attackers from revoking and obtaining valid certificates, the manufacturer must apply all necessary controls to ensure that identity reprovisioning firmware is provided only to legitimate customers.

When a certificate is revoked, the old DRK and all the certificates derived from it become invalid. This ensures that compromised devices can no longer use those certificates, effectively blocking them from the system. Moreover, our architecture ensures device identities are independent of any previously compromised credentials, allowing manufacturers to securely re-provision affected devices without any hardware modification.

D. END-TO-END IDENTITY AND ATTESTATION FLOW

This subsection describes a complete execution path that links the mechanisms presented in Sections V, VI, and VII, highlighting how a single hardware-rooted identity, derived at boot from a silicon PUF, is propagated through DICE layering to the SM firmware and enclaves, and how it supports secure boot, device authentication, attestation, and lifecycle operations under a unified trust model.

Upon power-on, execution starts in the immutable boot ROM, which constitutes the HRoT (see Fig. 5). The boot ROM (see Fig. 10) evaluates the RO-PUF to reconstruct the UDS using the stored CM and HD (see Fig. 12). No device-unique secret is stored in non-volatile memory. From the UDS, the DRK is deterministically derived and verified against the provisioned DRK certificate. If this verification fails, the boot process halts, enforcing fail-secure behavior. After successful reconstruction of the DRK, the boot ROM performs secure boot by verifying the integrity and authenticity of the next software stage. In parallel, it initiates the DICE process by computing the measurement of the first mutable code (corresponding to the Keystone SM in our architecture) and deriving the Layer-0 CDI (CDI_{L0}). From CDI_{L0}, the boot ROM generates a DICE alias key pair and issues the corresponding certificate, which binds the device identity to the measured firmware state. Control is then transferred to the Keystone SM, which intentionally receives CDI_{L0} but never gains access to the UDS or the DRK. Before this transfer, the boot ROM clears sensitive intermediate values and configures locked PMP entries, preventing any later access to PUF logic or root identity material. The SM uses CDI_{L0} as the root for runtime identity derivation.

When a Keystone enclave is created, the SM measures the enclave code and derives an enclave-specific CDI. From this CDI, it internally generates cryptographic keys used for enclave authentication and attestation. These keys and intermediate CDIs are never exposed to enclave software. Enclaves interact with them only through SM-mediated services, such as key generation, certificate retrieval, and cryptographic operations. Secure authentication can rely on these enclave-derived keys to authenticate enclaves to external entities. Remote attestation allows external verifiers to check the chain of trust. The enclave or platform presents DICE certificates, which contain the measurements of the software component whose identity they represent and are rooted in the original DRK, allowing a verifier to validate both the device identity and the integrity of the executing software stack. The security experiments reported in Section VIII-B validate that this chain enforces correct boot behaviour, prevents unauthorized provisioning, and preserves key isolation across all stages.

VII. IMPLEMENTATION ON RISC-V SoC PLATFORM

This section describes the implementation and validation of our HRoT architecture on a RISC-V SoC platform. Section VII-A introduces the SPIRS platform, highlighting its high-level hardware architecture and key features used for our experiments. Section VII-B details the integration of our HRoT within the platform, including necessary modifications to the default boot ROM provided by the Keystone framework and the device identity provisioning

protocol. Finally, Section VII-C describes the modifications to the firmware level required to implement the DICE Layering Architecture on top of our DICE-based HRoT, specifically within the Keystone SM.

A. SPIRS PLATFORM

We integrated and tested our HRoT architecture for IoT and embedded devices on the SPIRS platform, developed as part of the EU-funded Horizon 2020 SPIRS project [23]. The SPIRS platform was specifically designed to serve as an open, secure, and extensible RISC-V–based environment for experimenting with HRoTs–secured protocols. It provides an integrated hardware-software stack that includes a RISC-V SoC with custom HRoT cores, a SM, and a Keystone-based TEE. Unlike generic commercial boards, SPIRS offers full control over the TCB and its measurement chain, enabling the reproducible evaluation of secure-boot, lightweight identity, and attestation mechanisms aligned with the DICE framework.

This platform includes a CVA6 processor core [100], a boot ROM implementing Secure Boot and DICE RoT functionalities (discussed in Sections V-D and V-E), and cryptographic cores (e.g., SHA-2 and EdDSA, discussed in Sections V-B and V-C), and two PUFs, one for the secure device identification and another for the random number generation (see Section V-A).

Additionally, the CVA6 processor is equipped with sixteen PMP registers managed by Keystone's SM to enforce strong isolation among platform components (see Section II-D). The platform runs a Linux-based software stack built with Buildroot [101], which facilitates its use and enables reproducible, rigorous testing. Communication between the CVA6 processor and cryptographic hardware components is efficiently managed through memory-mapped I/O.

The SPIRS platform was validated on a Digilent Genesys 2 FPGA development board, featuring a Kintex-7 FPGA from Xilinx [102]. A block diagram of the final SPIRS platform is shown in Fig. 15.

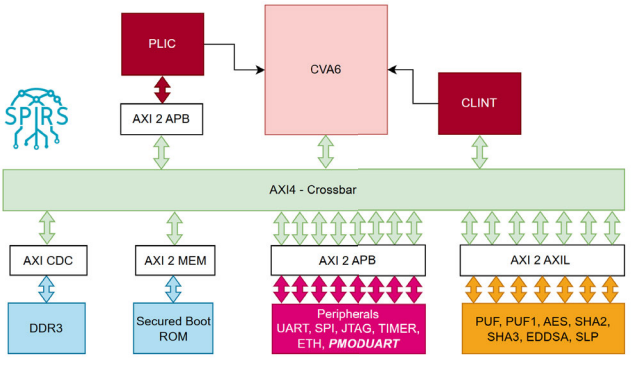

FIGURE 15. Block diagram of the final version of the SPIRS platform [103].

B. HRoT INTEGRATION AND BOOT ROM MODIFICATIONS

We integrated our proposed HRoT and associated identity life cycle management protocol into the SPIRS platform. Upon integration of the cryptographic cores into the hardware, the boot ROM was developed to support secure device identity provisioning and manage the boot sequences based on the data availability of provisioning data from an external SD card.

Algorithm 1 Board Device Personalization

```
if certificate ≠ NULL and additional_data ≠ NULL then
    seed, add_data ← get_from_PUF(add_data)
    p_key, s_key ← HW_gen_ed25519(seed)
    return certificate, keypair
else
    // start personalization process
    seed, add_data ← get_from_PUF()
    p_key, s_key ← HW_gen_ed25519(seed)
    CSR ← HW_compute_CSR(p_key, s_key)
    Secure channel:
        send CSR
        send add_data
    WAIT FOR REBOOT…
```

At startup, the boot ROM initially verifies the presence of provisioning data on the SD card, i.e., software signatures, identity certificate, and associated metadata. If these data are absent, the platform enters a restricted state, initiating the device identity provisioning process. Conversely, when the SD card provides the required provisioning data, the boot ROM directly interacts with the PUF through memory-mapped I/O to regenerate the UDS, utilizing additional data as input. This seed allows to derive the DRK matching the stored certificate, securely associating the platform with its provisioned identity.

Two software applications were developed to implement the identity provisioning protocol: one executed on the device and the other running on the manufacturer's machine. The on-board application generates the seed by relying on the PUF, derives an Ed25519 key pair using the hardware cryptographic modules, and creates a CSR without exposing any sensitive information outside the device. Communication between the device and manufacturer's machine is secured via a TLS channel authenticated by a public key embedded within the boot ROM, ensuring secure identity provisioning (Algorithm 1).

The manufacturer's application receives the CSR and additional metadata via Ethernet, verifies CSR validity, generates the corresponding certificate, and prepares an SD card containing provisioning and security data for subsequent device boot (Algorithm 2). We defined a USB drive with three partitions: the first for the board's software, the second for a filesystem if needed, and the third for security-related data (including the certificate, additional information, and any necessary signatures). The manufacturer's application will flash the last partition with the security data, after which the firmware provider will flash the firmware and filesystem partitions. Once the SD card has been prepared, the board can

Algorithm 2 Manufacturer Device Personalization

1: **Secure channel:**
2: *receive CSR*
3: *receive add_data*
4: *verify CSR validity*
5: *generate certificate*
6: **FLASH SD CARD**

be booted into operational mode, becoming fully functional with all security features enabled.

Upon reboot, the device regenerates the DRK linked to the provisioned certificate using the seed from the PUF and the additional data stored in the SD card. The boot ROM then executes DICE-compliant logic to generate a new key pair and corresponding certificate for subsequent firmware layers, reinforcing a robust device identity linked to firmware trustworthiness and enabling advanced security services.

To prevent unauthorized access or leakage of the private part of the DRK, the boot ROM explicitly clears memory locations containing the DRK before transferring control to the Keystone SM. It also employs two PMP configurations to enforce strict isolation and prevent any other subsequent code to access the PUF ID frame and recreate the data directly derived from it (i.e., UDS and DRK). The first PMP entry restricts memory-mapped I/O access to the PUF identity functionality (PUF2 / ID frame in Fig. 16) by setting a locked memory region with no read, write, or execute permissions, even for code running in M-mode. This ensures that subsequent firmware (even the Keystone SM) cannot regenerate or read the UDS seed or its corresponding asymmetric alias (DRK). The second PMP entry disables execution privileges for the boot ROM region containing the DICE HRoT logic. This measure ensures that any other code (even the Keystone SM operating in M-mode) cannot re-execute the DRK generation logic, effectively sealing off sensitive code segments post-boot. The PMP registers are configured using standard RISC-V assembly mechanisms, as detailed in [59]. With these PMP protections, the integrity of device identity and cryptographic keys is robustly preserved throughout the device lifecycle.

C. FIRMWARE AND SOFTWARE COMPONENTS

Our firmware and software modifications primarily focused on integrating the DICE Layering Architecture [22] within the Keystone SM. In particular, we integrated a DICE scheme where the Keystone SM serves as the DICE Layer 0, responsible for securely generating, managing, and providing cryptographic credentials for individual enclaves.

The boot ROM enforces a strict separation between the device root identity and runtime identities. The UDS and the derived DRK represent the root identity of the device and are never exposed outside the boot ROM. Before transferring control to subsequent stages, the boot ROM

Base	Description	
0x0000_0000	Debug Module	
...	...	
0x0001_0000	Boot ROM	HRoT
0x4100_0000	AES	HRoT
0x4200_0000	PUF1 / TRNG	HRoT
0x4300_0000	SHA2	HRoT
0x4400_0000	SHA3	HRoT
0X4500_0000	EdDSA	HRoT
0x4700_0000	SLP	HRoT
0x4800_0000	PUF2 / ID frame	HRoT
...	...	
0x8000_0000	Keystone SM	DRAM
0x8020_0000	OpenSBI	DRAM
...	...	DRAM

FIGURE 16. CPU memory space of the SPIRS platform.

explicitly clears their private material and configures locked PMP entries that prevent any later software from accessing the PUF or reconstructing the DRK. This prevents device impersonation and ensures that the device root identity cannot be extracted or reused by software executing after the boot transition. By contrast, the Keystone SM intentionally receives the Layer 0 CDI, which is a derived identity bound to the measured firmware state. This CDI does not allow reconstruction of the UDS or the DRK, but is required by the SM to derive further identities for enclaves and to bind them to a specific platform. The SM was extended to compute an enclave-specific CDI derived from a combination of the Layer 0 CDI and the enclave's own integrity measurement (TCI).

From this enclave-specific CDI, the SM securely generates cryptographic keys, such as the Local Attestation Key (LAK) for attestation and additional credentials suitable for authentication (LDevIDs), without delegating direct key management responsibilities to the enclaves themselves. Enclave-specific CDIs are generated and kept entirely inside the SM and are never exposed to enclave software. Enclaves interact with CDI-derived keys only through SM-mediated services, mitigating potential security risks arising from vulnerabilities in the enclave application code or from attempts to migrate enclave identities across devices. In particular, to support secure enclave operations within the Keystone framework, we defined custom Supervisor Binary Interface (SBI) calls, including `create_keypair()`, `get_cert_chain()`, and `do_crypto_op()`, enabling enclaves to securely generate and use cryptographic credentials under the SM's strict control. PMP-based isolation enforced by the SM ensures that enclave code cannot access SM memory, effectively making the SM a runtime RoT for enclaves, anchored in the boot ROM HRoT

Our integration of the HRoT and firmware extensions to support cryptographic keys derived from the secure device identity significantly strengthened the security properties of the Keystone-based platform while balancing performance, usability, and cryptographic isolation.

VIII. EVALUATION AND RESULTS

This section rigorously evaluates our proposed robust device identity architecture and secure identity management procedures implemented on the RISC-V-based SPIRS platform. The evaluation focuses on two main aspects:

- *Security Effectiveness*, by assessing the resilience of our implementation against representative attack scenarios;
- *Performance Efficiency*, by measuring the overhead introduced by secure boot, identity provisioning, and cryptographic operations.

A. TESTBED SETUP AND METHODOLOGY

Our evaluation was conducted on the the Genesys2 Kintex-7 FPGA (XC7K325T-2FFG900C) [102], equipped with 1 GB of DDR3 RAM and 256 Mbit Quad Serial Flash. It features a single-core CVA6-RISCV, a 6-stage, single-issue, in-order CPU that implements the 64-bit RISC-V instruction set. The board supports a fully programmable boot process, which is essential for correctly implementing the protocol and accurately measuring performance times. Additionally, it includes a hardware-implemented PUF module and all necessary functions for interaction. All the cryptographic operations are also implemented in hardware to optimize execution time.

B. SECURITY EFFECTIVENESS EVALUATION

In this section, we evaluate the security effectiveness of the proposed robust device identity architecture. The security evaluation focuses on representative attack classes aligned with the threat model defined in Section IV. Rather than attempting exhaustive coverage of all possible attacks, the experiments validate that the implemented mechanisms correctly enforce their intended security properties under realistic adversarial capabilities, including firmware tampering, unauthorized identity provisioning, and key-extraction attempts from untrusted software contexts.

All cryptographic operations involved in secure boot, identity derivation, and attestation are implemented in dedicated hardware with constant-time execution to reduce timing-based side-channel leakage. While this design choice mitigates a broad class of timing attacks, no dedicated power or electromagnetic side-channel measurements were performed in this work.

1) SECURE BOOT VALIDATION

Ensuring firmware integrity from the initial boot stages is paramount for device security. To assess the robustness of our secure boot implementation, we conducted a set of controlled attacks aimed at tampering with firmware signatures and public keys stored on the SD card. Table 8 summarizes the conducted experiments and their outcomes.

Our secure boot implementation reliably rejected any modified, invalid, or missing cryptographic signatures. Only firmware with valid signatures successfully executed, validating our secure boot chain and robust RTD capability.

TABLE 8. Secure boot integrity validation results.

Scenario	Expected Outcome	Observed Outcome
Valid signature	Successful boot	✅ Passed
Invalid signature	Boot failure (firmware rejected)	✅ Passed
Modified public key	Boot failure (firmware rejected)	✅ Passed
Missing signature	Boot failure (firmware rejected)	✅ Passed

2) IDENTITY PROVISIONING SECURITY

Secure device identity provisioning ensures that cryptographic identities remain unique, unclonable, and protected from unauthorized extraction or replay attacks. To evaluate this, we simulated several provisioning attack scenarios, focusing on potential replay attacks and the manipulation of provisioning metadata (CM and HD). Table 9 presents the outcomes of these tests.

TABLE 9. Identity provisioning security test results.

Attack Scenario	Expected Outcome	Observed Outcome
Valid initial provisioning	Device authenticated successfully	✅ Passed
Replay of previously used CSR	Provisioning rejected	✅ Passed
CSR with manipulated metadata	Provisioning rejected	✅ Passed
Unapproved manufacturer's key	Device authentication failed	✅ Passed

Our protocol successfully prevented unauthorized provisioning attempts, rejecting replayed and manipulated provisioning requests, ensuring the integrity and uniqueness of provisioned cryptographic identities.

3) KEY ISOLATION AND CONFIDENTIALITY

The following tests specifically assess the robustness of key isolation and confidentiality mechanisms, which are critical for ensuring that cryptographic keys cannot be extracted or manipulated by unauthorized software or hardware adversaries. Each scenario directly corresponds to threats described in the threat model (Section IV) and the implemented mitigations (Section IV-B). Table 10 summarizes the validation scenarios performed.

TABLE 10. Key isolation and confidentiality validation results.

Scenario	Expected Outcome	Observed Outcome
Unauthorized Enclave Attempt to Access Another Enclave's Key	Access denied	✅ Passed
Unauthorized REE Attempt to Access SM-managed Keys	Access denied	✅ Passed
Simulated Hardware Modification (PUF replaced)	Key derivation failed	✅ Passed
Unauthorized Access to PUF or boot ROM via Enclave Software	Access denied	✅ Passed

Our tests confirm a rigorous isolation of cryptographic keys and seeds, effectively preventing unauthorized access at both the hardware (PUF and boot ROM) and software (REE, enclaves, and SM) levels.

C. PERFORMANCE EVALUATION

This section evaluates the performance efficiency of our proposed robust device identity architecture, specifically

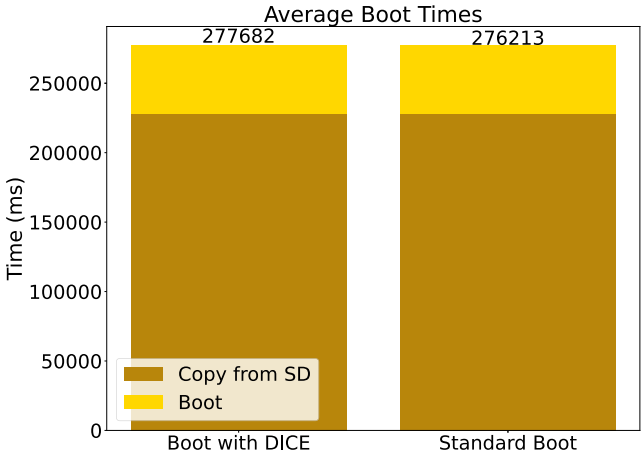

FIGURE 17. Comparison between boot times.

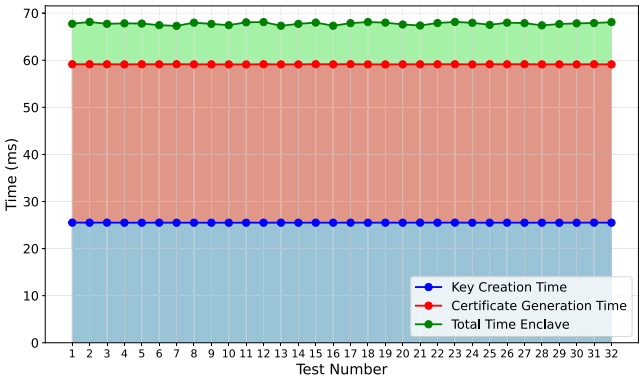

FIGURE 18. Total time for creation of a keypair and its certificate.

analyzing the overhead introduced by integrating the proposed HRoT, secure boot process, and identity provisioning procedures. Our evaluation covers four critical performance aspects:

- *Boot Time Overhead*
- *Key Generation and Certificate Issuance Efficiency*
- *Identity Provisioning Protocol Latency*
- *Energy Considerations*

1) BOOT TIME OVERHEAD

We first quantified the boot overhead introduced by the additional security measures. Specifically, we measured the boot time under two distinct scenarios to evaluate the impact of the security enhancements:

1) *Baseline (No Security)*: Platform boot without any security mechanisms enabled.
2) *Secure Boot and full DICE Layering Initialization*: Complete boot process including firmware authentication, PUF-based device identity regeneration, and initialization of the DICE Layering Architecture.

As shown in Fig. 17, the introduction of the PUF-based identity derivation and secure boot adds only minimal overhead, amounting to less than 1% of the total boot duration. The largest contributor to the boot time (approximately 82%) is data transfer from the SD card into DDR3 memory, which varies depending on the selected Linux configuration. These results indicate that, despite introducing robust hardware-level security features, our architecture maintains a negligible impact on overall boot efficiency.

2) KEY GENERATION AND CERTIFICATE ISSUANCE EFFICIENCY

We next evaluated the performance of keypair generation and the corresponding certificate issuance within the SM, as well as the overhead introduced by enclave requests for SM's cryptographic services. This evaluation is critical for assessing the practicality of performing secure cryptographic operations with DICE identity keys in various scenarios.

Fig. 18 shows the measured execution times for each cryptographic operation (keypair generation and certificate issuance), along with the overhead associated with enclave context-switch (highlighted by the green section of the chart).

Our results demonstrate that keypair generation and certificate issuance within the secure environment (SM) are highly efficient, requiring an average of only 60 ms, of which 26 ms for key generation and 34 ms for certificate issuance. The overall enclave request overhead, encompassing secure context switches between the enclave (Trusted Application (TA)) and the SM, amounts to approximately 10%. This overhead is considered minimal and acceptable, especially considering the security it provides, confirming that the implemented security mechanisms do not compromise system responsiveness, even in performance-constrained IoT contexts.

3) IDENTITY PROVISIONING PROTOCOL LATENCY

Finally, we assessed the end-to-end latency of our secure identity provisioning protocol, executed via a point-to-point Ethernet-based TLS communication channel. Although this process is performed only once for each device during its lifetime (typically at manufacturing or initial commissioning), its efficiency remains important for production scalability and operational usability.

Table 11 presents the most time-consuming steps, averaged over five runs with stable results. The PUF enrollment phase dominates the overall execution, requiring approximately 29 min on average, while the remaining phases (e.g., memory mapping, TLS channel creation, and data transfer) complete within the millisecond-to-second range.

The results show an average total latency of about 30 min for completing the secure provisioning protocol, including both hardware and cryptographic processes. It is important to note that the most time-consuming operation (i.e., the PUF enrollment and corresponding CM retrieval) is a one-time procedure per device, carried out only during its initial personalization. Once completed, subsequent operations rely on the stored identity metadata (i.e., CM and HD) and do

TABLE 11. Average identity provisioning protocol latency.

Provisioning Step	Average Latency (ms)
Memory Mapping	58.167
PUF Enrollment	1764963.667 (~29 min)
TLS Channel Establishment	1821.0
Data Transfer	232.667
Total Provisioning Time	**~29** min

not repeat this lengthy phase. Considering the infrequent and non-recurring nature of this step, the observed latency is practically acceptable, especially in light of the strong security assurances it provides.

The reported provisioning time of approximately 30 min refers to a single device. This latency is dominated by the PUF enrollment phase, which is executed entirely on the device and does not require intensive processing on the manufacturer side. In a manufacturing or deployment setting, multiple devices can therefore be provisioned in parallel through independent provisioning sessions or services, with the overall throughput scaling linearly with the number of devices processed concurrently. Since the manufacturer appliance performs only lightweight tasks (CSR verification, certificate issuance, and data storage), it does not constitute a bottleneck for large-scale provisioning.

4) ENERGY CONSIDERATIONS

While we do not report absolute energy measurements, the energy cost of the proposed HRoT is bounded by hardware-accelerated cryptographic operations executed during secure boot and by the cryptographic services invoked at runtime through the SM (e.g., key generation, certificate issuance, and signing for attestation). All primitives (PUF evaluation, hashing, signature verification, and Ed25519 operations) are implemented in dedicated hardware, reducing the number of CPU cycles compared to RoTs which embed microcontrollers or soft-core processors, where security primitives are executed in software. The overall energy impact therefore depends mainly on the frequency of key-management and attestation requests, whereas the boot-time cost is incurred only on reset and the provisioning cost is incurred only during enrollment or renewal.

IX. SECURITY ANALYSIS

This section provides a structured analysis of the security properties of the mechanisms adopted in the proposed HRoT architecture. Each subsection explicitly links a security mechanism to the corresponding threat assumptions and clarifies the classes of attacks that are mitigated or considered out of scope. The analysis complements the experimental evaluation presented in Section VIII by explaining how secure boot, DICE-based identity derivation, PUF-based root secrets, and hardware isolation jointly contribute to device identity protection and system integrity.

A. SECURE BOOT AND FIRMWARE INTEGRITY

The secure boot mechanism ensures that only authenticated firmware components are executed, starting from the immutable boot ROM. Digital signature verification prevents unauthorized modification of the bootloader and, when enabled, of subsequent software layers stored on external memory. This mechanism mitigates attacks based on firmware tampering, signature stripping or manipulation, rollback to unsigned images, and execution of arbitrary code injected through removable storage. Secure boot does not prevent vulnerabilities in correctly signed firmware from being exploited at runtime. Its role is limited to enforcing software authenticity and integrity at load time, which is consistent with the defined threat model and with standard secure boot architectures.

B. PUF-BASED ROOT SECRET AND HELPER DATA

The use of a weak RO-PUF avoids persistent storage of the device root secret and reduces exposure to key extraction through memory disclosure attacks. Because the PUF is evaluated only during early boot and its raw responses are not exposed, the design is not vulnerable to modeling attacks typically associated with strong PUFs that allow repeated challenge–response access. HDAs are used to reconstruct stable secrets from noisy PUF outputs. As reported in prior work [74], HDAs are not necessarily secure against certain classes of physical attacks and may leak information if an adversary can observe internal reconstruction values. In this work, leakage resistance of HDA is not claimed. Security relies on the assumption that PUF evaluation, reconstruction, and intermediate values remain confined to protected hardware during boot, and that raw PUF responses are not observable by a physical attacker. This assumption is explicitly stated in the threat model and reflects the intended deployment context.

C. ISOLATION, KEY CONFIDENTIALITY, AND SIDE-CHANNEL CONSIDERATIONS

After boot, access to the PUF, the UDS, and the derived DRK is blocked through RISC-V PMP configuration. Sensitive material is cleared from memory before control is transferred to later stages, and execution of identity derivation logic is permanently disabled until next boot. These measures prevent software executing after boot, including privileged firmware and enclaves, from accessing or reconstructing device root secrets.

All cryptographic operations involved in secure boot, identity derivation, and attestation are implemented in dedicated hardware with constant-time execution, reducing timing-based side-channel leakage. In addition, sensitive operations are confined to the early boot phase and isolated through hardware mechanisms, limiting exposure to software-driven leakage vectors. This work does not claim resistance against advanced power or electromagnetic side-channel attacks requiring invasive access or specialized laboratory

equipment. Such attacks are considered out of scope and are not aligned with the assumed deployment model, where devices are provisioned and deployed in operational settings without continuous physical attacker access. Within this model, the adopted design choices mitigate timing-based and software-assisted side-channel attacks, while invasive physical analysis remains outside the evaluated threat surface.

D. DICE-BASED IDENTITY DERIVATION AND ATTESTATION

DICE binds cryptographic identities to measurements of the software state executed during boot, chaining them starting from a HRoT. Any modification of the measured firmware results in a different derived identity, preventing use of trusted credentials across altered platform configurations. This property mitigates attacks aimed at using valid credentials on modified firmware images or migrating identities on other platforms. The security of this mechanism relies on the correctness of the measurement process and on the immutability of the boot ROM logic implementing it. Since identity derivation is completed before control is transferred to mutable software, attackers operating at the REE or enclave level cannot influence the derivation process within the assumed threat model.

E. IDENTITY LIFECYCLE AND FAIL-SECURE BEHAVIOR

Identity provisioning, renewal, and revocation are directly tied to the ability to reconstruct the PUF-derived root secret. If reconstruction fails due to tampering of identity metadata, excessive drift, or aging of the PUF circuitry beyond the correction capability of the employed scheme, the derived key no longer matches the provisioned certificate and the secure boot process halts. This fail-secure behavior prevents silent identity divergence and ensures that devices cannot continue operation with inconsistent or partially compromised credentials.

Identity renewal procedures preserve the same security properties, as new certificates are issued only after successful reconstruction of the current PUF-derived ID frame under the same protection and isolation guarantees enforced at boot. Minor variations in the PUF response caused by aging are tolerated by the proposed renewal mechanism, while larger deviations due to environmental drift prevent identity recovery and trigger a fail-secure halt. As a result, renewal rebinds the device identity to its current physical state without requiring persistent storage of sensitive key material or having to replace the device due to slight changes in the PUF ID frame due to aging.

X. DISCUSSION

A. COMPARATIVE POSITIONING

The comparative analysis in Table 12 highlights that our proposal occupies a design space not fully addressed by prior work. Solutions such as OpenTitan [13] and Caliptra [14] deliver comprehensive RoT subsystems, but their complexity and resource requirements make them unsuitable for deeply embedded or low-power IoT devices. At the other extreme, software-only frameworks like Rolling DICE [24] demonstrate deployability on constrained platforms but lack hardware anchoring, leaving them more exposed to physical or lifecycle attacks.

Our design introduces a lightweight yet integrated HRoT, directly embedded in the host RISC-V SoC. By combining a PUF-protected root secret with full DICE-based layering extended to the host system, it offers strong, unclonable device identities and lifecycle-aware provisioning while keeping hardware overhead minimal. This makes the architecture particularly well-suited for constrained environments, where integrating a heavyweight RoT would be impractical, but where robust security assurances remain mandatory. In doing so, our work complements existing proposals: instead of replicating datacenter-grade subsystems at the edge, it provides a tailored hardware–software co-design that preserves end-to-end trust with a fraction of the cost and complexity. This positions our solution as a novel contribution aimed at securing the ''in-between'' class of devices: those that are too constrained for full-featured RoTs, yet too critical to rely solely on software-based mechanisms.

B. DESIGN TRADEOFFS AND PRACTICAL CONSIDERATIONS

The use of a PUF-protected root secret provides strong resistance to cloning and physical tampering, but it shifts reliability onto the PUF's stability. Concretely, achieving a highly stable reference ID frame requires an *enrollment* step that selects only the most reliable response bits and derives helper data; this step dominates the provisioning latency in our evaluation (on average $\sim$29 minutes for PUF enrollment within an otherwise millisecond workflow). By contrast, stored-key approaches using OTP or flash offer fast and deterministic retrieval but expose a greater risk of key extraction if memory protections are bypassed. Once enrollment is complete the first time the DRK is provisioned, however, the operational cost is negligible: boot-time identity regeneration and DICE initialization add $< 1\%$ to the total boot time on our platform, and the secure identity remains confined to hardware-protected contexts (boot ROM HRoT).

Furthermore, our lifecycle support (renewal, revocation, and zeroization) is achieved *without* additional hardware complexity: identities can be renewed by relying on the PUF TRNG functionality, without requiring the PUF enrollment step, while remote renewal leverages standard EST re-enrollment over TLS (Fig. 14) for certificate updates. This keeps the hardware/software co-design simple, yet enables long-lived fleets to evolve securely in the field.

C. INTEROPERABILITY AND DEPLOYMENT SCENARIOS

The security mechanisms introduced in our design (i.e., a hardware-rooted device identity based on silicon PUF, lifecycle-aware credential management, and full DICE-based attestation) translate into practical advantages in real deployment contexts. In the IoT domain, a representative

TABLE 12. Comparison with state-of-the-art approaches in device identity provisioning.

Work	Scope	Identity Root	Attestation Model	Lifecycle Support
OpenTitan [13]	Standalone RISC-V–based RoT subsystem (Ibex core)	Stored keys in RoT	Secure boot + DICE (internal to RoT, not extended to host system)	Limited
Caliptra [14]	RISC-V–based RoT subsystem for SoCs (datacenter/edge)	Stored keys in RoT	Secure boot + DICE (internal to RoT, not extended to host system)	Limited
Sancus [63]	Hardware security extensions for MSP430 microcontrollers	Random OTP secret	Symmetric module attestation	None
Sanctum [26]	System-level TEE (based on RISC-V custom ISA)	PUF-derived key	Enclave attestation	None
Rolling DICE [24]	Lightweight software-only DICE framework for constrained IoT devices	Software-generated keys	Lightweight DICE chain	None
Our Proposal	Integrated HRoT in host RISC-V SoC	PUF-protected key	Secure boot + full DICE layering (extended to host system)	Generation & provisioning, operation, renewal

case is a sensor node publishing data to a cloud service over the Message Queuing Telemetry Transport (MQTT) protocol. While conventional solutions often rely on pre-shared keys or software-stored asymmetric keys, which are susceptible to leakage and cloning, our approach ensures that the device presents a credential derived from its unique hardware-rooted identity, regenerated at each boot, and certified during the provisioning process. The MQTT broker can therefore verify not only the device's authenticity, but also the integrity of the firmware stack executing on it, resulting in a secure communication channel anchored in provable device trust. Similar benefits emerge in industrial environments, where embedded controllers must continuously exchange commands and telemetry with Manufacturing Execution Systems (MES) or Supervisory Control and Data Acquisition (SCADA) servers. By extending the DICE-based attestation chain to system components and enclaves, our design allows the MES to remotely validate both the provenance of the device and the correctness of the control software running on it, mitigating the risk of compromised nodes injecting falsified data or executing unauthorized logic.

In long-lived industrial deployments, lifecycle support for credential renewal and revocation further guarantees that devices can remain trustworthy without requiring costly physical replacement or rekeying. The same mechanisms also strengthen supply-chain assurance, since each manufactured device carries a PUF-bound identity that cannot be cloned or forged. As a result, integrators and operators can verify that each board originates from a legitimate production process, ensuring authenticity throughout the distribution chain.

Finally, our proposal emphasizes interoperability with existing standards: the device credentials are provisioned as X.509 certificates, directly compatible with TLS-based communication stacks and industrial security standards such as IEC 62443. This avoids the need for proprietary trust anchors or protocol extensions, facilitating integration into existing infrastructures while enhancing their overall security posture.

D. LIMITATIONS AND FUTURE DIRECTIONS

While the proposed architecture demonstrates strong guarantees in secure device identity, lifecycle management, and DICE-based attestation, several limitations remain that open opportunities for future research.

A first limitation is the dependency on the quality and long-term stability of the PUF. Although our enrollment procedure (selection of stable bits and HD derivation) yields highly reliable reconstruction, it incurs a significant latency during first provisioning, and reconstruction may still be sensitive to environmental conditions (temperature/voltage) or silicon aging. In our system, the failure mode is immediately detectable: if the UDS cannot be reconstructed, the board cannot recover the certified key (DRK) and halts boot, immediately surfacing the anomaly.

Future research should therefore focus on methods to distinguish between environmental effects, natural device aging, and genuine tampering with identity metadata. One possible direction is to authenticate the helper data and challenge mask with a manufacturer's signature, so that any alteration can be immediately attributed to malicious modification. In addition, the enrollment process could be extended to cover multiple environmental corners, allowing the device to adapt its reconstruction logic depending on operating conditions and thus reduce false failures caused by benign variations. Finally, a drift-tolerant helper-data refresh mechanism could be introduced, enabling the device to regenerate stable metadata under controlled conditions and with strong verifier approval, while always preserving continuity with the originally certified identity.

A second limitation concerns crypto-agility in the immutable boot ROM. In the current prototype, the boot ROM does not support algorithm agility. Cryptographic algorithms for hashing, signature verification, and key derivation are fixed at design time and are not selected through a manifest or policy descriptor. This design choice simplifies the implementation and consequent verification phase of the boot ROM code and reflects common constraints of embedded platforms, where early boot code is expected to

remain small, stable, and easy to validate. At the same time, the proposed architecture does not preclude future support for crypto-agility. Algorithm agility can be introduced by extending the secure boot flow with a signed boot policy or manifest that is verified by the boot ROM and that specifies the allowed cryptographic suites, minimum security levels, and downgrade constraints. The same approach can be applied consistently to DICE operations, enabling future DICE profiles or post-quantum algorithms to be adopted without modifying the immutable boot ROM. Designing and validating such a crypto-agile secure boot and DICE framework is left as future work.

Beyond PUF-specific issues and crypto-agility, another important direction for future research is the integration of attestation evidence into higher level protocols. While our work establishes a DICE-based trust chain from the boot ROM to the host and enclaves, extending this evidence directly into application-layer sessions would enable end-to-end assurance of both device identity and runtime integrity. For instance, DICE-derived claims could be embedded into MQTT/TLS client authentication, allowing back-end services in IoT or industrial environments to verify not only the device identity but also its measured state before granting access. Such contextual attestation would let authorization policies depend on the current integrity of the device rather than on static credentials alone. This approach would strengthen interoperability with existing security infrastructures and ensure that the trust established at boot is maintained throughout the entire communication stack.

XI. CONCLUSION

This work presented a lightweight yet robust integration of silicon PUFs with the DICE to establish a hardware RoT directly within the boot ROM of RISC-V systems. Our approach defines and implements a secure device identity provisioning protocol anchored in unclonable silicon variability, ensures identity regeneration at every boot, and extends the DICE layering architecture to enclave-based TEEs. The resulting architecture provides strong resistance to physical tampering and identity theft, while maintaining minimal hardware overhead and negligible performance impact. The main contributions of this research include: (i) the definition of a secure identity generation and provisioning process leveraging silicon PUFs, (ii) the seamless embedding of DICE-based hardware-rooted identities into the early boot process, and (iii) the demonstration of lifecycle-aware identity management, including renewal, revocation, and zeroization, on a practical RISC-V SoC platform. Together, these contributions address the pressing need for secure, scalable, and interoperable device identity solutions for constrained IoT and embedded systems.

At the same time, this research opens several paths for future investigation. A key direction concerns the long-term stability and reliability of PUF responses under environmental variations and device aging, which remain critical for practical deployments. Enhancements in drift-tolerant helper data, adaptive reconstruction techniques, and authenticated metadata may further strengthen robustness. Additionally, extending attestation evidence into application-layer protocols would enable end-to-end assurance of both device identity and runtime integrity, paving the way for secure, context-aware IoT ecosystems.

REFERENCES

[1] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019.

[2] J. Granjal, E. Monteiro, and J. Sá Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 3rd Quart., 2015.

[3] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[4] B. Chen, T. Ignatenko, F. M. J. Willems, R. Maes, E. van der Sluis, and G. Selimis, "A robust SRAM-PUF key generation scheme based on polar codes," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–6.

[5] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.

[6] M. Fornero, N. Maunero, P. Prinetto, and A. Varriale, "SEkey: A distributed hardware-based key management system," in *Proc. IEEE East-West Design Test Symp. (EWDTS)*, Bulgaria, Sep. 2020, pp. 1–7.

[7] N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul, and S. Katzenbeisser, "A lightweight architecture for hardware-based security in the emerging era of systems of systems," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 3, pp. 1–25, Jun. 2021.

[8] J. G. Rajan and R. S. Ganesh, "Hardware based data security techniques in IoT: A review," in *Proc. 3rd Int. Conf. Smart Electron. Commun. (ICOSEC)*, India, Oct. 2022, pp. 408–413.

[9] U. Ali, H. Omar, C. Ma, V. Garg, and O. Khan, "Hardware root-of-trust implementations in trusted execution environments," Cryptol. ePrint Archive, Tech. Paper 2023/251, Feb. 2023. [Online]. Available: https://eprint.iacr.org/2023/251

[10] L. Deutschmann, J. Müller, M. R. Fadiheh, D. Stoffel, and W. Kunz, "Towards a formally verified hardware root-of-trust for data-oblivious computing," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 727–732.

[11] O. Demigha and R. Larguet, "Hardware-based solutions for trusted cloud computing," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102117.

[12] E. T. Michailidis and D. Vouyioukas, "A review on software-based and hardware-based authentication mechanisms for the Internet of Drones," *Drones*, vol. 6, no. 2, p. 41, Feb. 2022.

[13] *OpenTitan*. Accessed: Apr. 7, 2026. [Online]. Available: https://opentitan.org/

[14] B. Kelly, A. Lagar-Cavilla, J. Andersen, P. Jayana, P. Kwidzinski, R. Strong, J. Traver, L. Ferraro, I. Agarwal, A. Parthasarathy, B. Pillilli, V. Soni, M. Schilder, S. Mathane, N. Nadarajah, and K. Nielsen. (Jul. 2022). *Caliptra: A Datacenter System on a Chip (SoC) Root of Trust (RoT)*. [Online]. Available: https://www.opencompute.org/documents/caliptra-silicon-rot-services-09012022-pdf

[15] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanovic, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proc. 15th Eur. Conf. Comput. Syst.*, Greece, Apr. 2020, pp. 1–16.

[16] A. Ehret, E. Del Rosario, K. Gettings, and M. A. Kinsy, "A hardware root-of-trust design for low-power SoC edge devices," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2020, pp. 1–6.

[17] K. Yoshida, K. Suzaki, and T. Fujino, "Towards trusted IoT sensing systems: Implementing PUF as secure key generator for root of trust and message authentication code," in *Proc. Workshop Hardw. Architectural Support Secur. Privacy*, Oct. 2021, pp. 1–8.

[18] M.-Y. Wu, "Hardware root-of-trust design based on on-chip PUF for AIoT applications," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2022, p. 1.

[19] R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications* (Heidelberg), 1st ed., Cham, Switzerland: Springer, Dec. 2013.

[20] M. Tehranipoor, N. Pundir, N. Vashistha, and F. Farahmandi, *Hardware Security Primitives*, 1st ed., Cham, Switzerland: Springer, Dec. 2022.

[21] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.

[22] (2020). *DICE Layering Architecture*. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19.pdf

[23] *SPIRS: Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process (SPIRS)*. Accessed: Apr. 7, 2026. [Online]. Available: https://www.spirs-project.eu/

[24] L. Jäger, R. Petri, and A. Fuchs, "Rolling DICE: Lightweight remote attestation for COTS IoT hardware," in *Proc. 12th Int. Conf. Availability, Rel. Secur.*, Italy, Aug. 2017, pp. 1–8.

[25] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. V. Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens, "Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base," in *Proc. 22nd USENIX Secur. Symp.*, Aug. 2013, pp. 479–494. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/noorman

[26] V. Costan, I. A. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 857–874. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[27] *IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity*, IEEE Standard 802.1AR, IEEE Standards Association. [Online]. Available: https://standards.ieee.org/standard/8021AR-2018.html

[28] (2024). *DICE Attestation Architecture*. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.1-Revision-18.pdf

[29] *Security for Industrial Automation and Control Systems–Part 4-2: Technical Security Requirements for IACS Components*, document 62443-4, International Electrotechnical Commission, Feb. 2019. [Online]. Available: https://webstore.iec.ch/en/publication/34421

[30] P. J. Leach, R. Salz, and M. H. Mealling, *A Universally Unique IDentifier (UUID) URN Namespace*, document RFC 4122, Jul. 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4122

[31] A. Buhr, D. Lindskog, P. Zavarsky, and R. Ruhl, "Media access control address spoofing attacks against port security," in *Proc. 5th USENIX Conf. Offensive Technol.*, Aug. 2011, p. 1. [Online]. Available: https://www.usenix.org/conference/woot11/media-access-control-address-spoofing-attacks-against-port-security

[32] F. Khan, A. A. Al-Atawi, A. Alomari, A. Alsirhani, M. M. Alshahrani, J. Khan, and Y. Lee, "Development of a model for spoofing attacks in Internet of Things," *Mathematics*, vol. 10, no. 19, p. 3686, Oct. 2022.

[33] C. Zuo, H. Wen, Z. Lin, and Y. Zhang, "Automatic fingerprinting of vulnerable BLE IoT devices with static UUIDs from mobile apps," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1469–1483.

[34] D. Merli, *Engineering Secure Devices: A Practical Guide for Embedded System Architects and Developers*. San Francisco, CA, USA: No Starch Press, 2024.

[35] C. Li, L. Guan, J. Lin, B. Luo, Q. Cai, J. Jing, and J. Wang, "Mimosa: Protecting private keys against memory disclosure attacks using hardware transactional memory," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1196–1213, May 2021.

[36] H. Böck, J. Somorovsky, and C. Young, "Return of bleichenbacher's Oracle threat (ROBOT)," in *Proc. 27th USENIX Secur. Symp.*, pp. 817–849. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/bock

[37] T. Zhang, J. Jiang, and Y. Zhang, "Revisiting and evaluating software side-channel vulnerabilities and countermeasures in cryptographic applications," 2019, *arXiv:1911.09312*.

[38] G. Doychev and B. Köpf, "Rigorous analysis of software countermeasures against cache attacks," in *Proc. 38th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Spain, Jun. 2017, pp. 406–421.

[39] *EdgeLock SE050: Plug and Trust Secure Element Family–Enhanced IoT Security With High Flexibility*. Accessed: Apr. 7, 2026. [Online]. Available: https://www.nxp.com/products/SE050

[40] *CryptoAuthentication Family*. Accessed: Apr. 7, 2026. [Online]. Available: https://www.microchip.com/en-us/products/security/security-ics/cryptoauthentication-family

[41] S. Akter, K. Khalil, and M. Bayoumi, "Hardware security in the Internet of Things: A survey," in *Proc. IEEE 36th Int. Syst.-Chip Conf. (SOCC)*, Sep. 2023, pp. 1–6.

[42] M. Khan, M. Ilyas, and O. Bayat, "Enhancing IoT security through hardware security modules (HSMs)," in *Proc. Int. Conf. Intell. Comput., Commun., Netw. Services (ICCNS)*, Sep. 2024, pp. 278–282.

[43] D. Challener, K. Yoder, R. Catherman, D. Safford, and L. Van Doorn, *A Practical Guide to Trusted Computing*. Indianapolis, IN, USA: IBM Press, Dec. 2007.

[44] (2011). *TPM Main–Part 1: Design Principles*. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles.pdf

[45] R. Wang and Y. Yan, "A survey of secure boot schemes for embedded devices," in *Proc. 24th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2022, pp. 224–227.

[46] (2024). *Hardware Requirements for a Device Identifier Composition Engine*. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-a-Device-Identifier-Composition-Engine-Version-1.0-Revision-0.91pub.pdf

[47] S. Chen, B. Li, Z. Chen, Y. Zhang, C. Wang, and C. Tao, "Novel strong-PUF-based authentication protocols leveraging Shamir's secret sharing," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14408–14425, Aug. 2022.

[48] J. Wen, M. Huang, Z. Chen, L. Zhu, S. Chen, and B. Li, "A multi-line arbiter PUF with improved reliability and uniqueness," in *Proc. IEEE 4th Int. Conf. Signal Image Process. (ICSIP)*, China, Jul. 2019, pp. 641–648.

[49] B. Li, S. Chen, and F. Dan, "Design and implementation of an improved MA–APUF with higher uniqueness and security," *ETRI J.*, vol. 42, no. 2, pp. 205–216, Apr. 2020.

[50] U. Ruhrmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013.

[51] C. Yehoshuva, R. R. Adhithan, and N. N. Anandakumar, "A survey of security attacks on silicon based weak PUF architectures," in *Proc. Secur. Comput. Commun. (SSCC)*, Oct. 2021, pp. 107–122.

[52] N. Saadvikaa, K. J. Saketi, A. Gopishetti, B. Degala, and K. K. Anumandla, "PUF modeling attacks using deep learning and machine learning algorithms," *Eng. Proc.*, vol. 56, no. 1, p. 187, Nov. 2023.

[53] M. C. Martínez-Rodríguez, L. F. Rojas-Muñoz, E. Camacho-Ruiz, S. Sánchez-Solano, and P. Brox, "Efficient RO-PUF for generation of identifiers and keys in resource-constrained embedded systems," *Cryptography*, vol. 6, no. 4, p. 51, Oct. 2022.

[54] Y. Cui, J. Li, Y. Chen, C. Wang, C. Gu, M. O'neill, and W. Liu, "An efficient ring oscillator PUF using programmable delay units on FPGA," *ACM Trans. Design Autom. Electron. Syst.*, vol. 29, no. 1, pp. 1–20, Nov. 2023.

[55] T. Kaya, "A true random number generator based on a Chua and RO-PUF: Design, implementation and statistical analysis," *Anal. Integr. Circuits Signal Process.*, vol. 102, no. 2, pp. 415–426, Jun. 2019.

[56] L. F. Rojas-Muñoz, S. Sánchez-Solano, M. C. Martínez-Rodríguez, and P. Brox, "On-line evaluation and monitoring of security features of an RO-based PUF/TRNG for IoT devices," *Sensors*, vol. 23, no. 8, p. 4070, Apr. 2023.

[57] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Finland, Aug. 2015, pp. 57–64.

[58] *TrustZone for Cortex-A*. [Online]. Available: https://www.arm.com/technologies/trustzone-for-cortex-a

[59] A. Waterman, K. Asanovìc, and J. Hause. (2024). *The RISC-V Instruction Set Manual: Volume II*. [Online]. Available: https://drive.google.com/file/d/17GeetSnT5wW3xNuAHI95-SI1gPGd5sJVview

[60] C. Göttel, M. Kabir-Querrec, D. Kozhaya, T. Sivanthi, and O. Vukovic, "Qualitative analysis for validating IEC 62443–4–2 requirements in DevSecOps," in *Proc. IEEE 28th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Romania, Sep. 2023, pp. 1–8.

[61] *Caliptra IP and Firmware for Integrated Root of Trust Block*. Accessed: Apr. 7, 2026. [Online]. Available: https://github.com/chipsalliance/Caliptra

[62] A. Lagar-Cavilla, P. Jayanna, and B. Kelly. *Caliptra—An Open Source, Reusable Silicon IP Block for a Root of Trust for Measurement (RTM)*. Accessed: Apr. 7, 2026. [Online]. Available: https://146a55aca6f00848c565-a7635525d40ac1c70300198708936b4e.ssl.cf1.rackcdn.com/images/6dadf83e9f93ca89efaf3b93ab076cea8f9ac747.pdf

[63] J. Noorman, J. V. Bulck, J. T. Muhlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. M´uller, and F. Freiling, "Sancus 2.0: A low-cost security architecture for IoT devices," *ACM Trans. Privacy Secur.*, vol. 20, no. 3, pp. 1–33, Jul. 2017.

[64] E. Bravi, S. Sisinni, and A. Lioy, "Exploiting the DICE specification to ensure strong identity and integrity of IoT devices," in *Proc. 8th Int. Conf. Smart Sustain. Technol. (SpliTech)*, Croatia, Jun. 2023, pp. 1–6.

[65] E. Lear, R. Droms, and D. Romascanu, *Manufacturer Usage Description Specification*, document RFC 8520, Mar. 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8520

[66] R. Román, R. Arjona, and I. Baturone, "A quantum-safe authentication scheme for IoT devices using homomorphic encryption and weak physical unclonable functions with no helper data," *Internet Things*, vol. 28, Dec. 2024, Art. no. 101389.

[67] K. Lounis and M. Zulkernine, "Lessons learned: Analysis of PUF-based authentication protocols for IoT," *Digit. Threats: Res. Pract.*, vol. 4, no. 2, pp. 1–33, Feb. 2022.

[68] Y. Zheng, W. Liu, C. Gu, and C.-H. Chang, "PUF-based mutual authentication and key exchange protocol for peer-to-peer IoT applications," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 3299–3316, Jul. 2023.

[69] H. Men, L. Cao, G. Zheng, and L. Chen, "A PUF-based lightweight identity authentication protocol for Internet of Vehicles," *Comput. Electr. Eng.*, vol. 123, Apr. 2025, Art. no. 110210.

[70] S. Chen, B. Li, and Y. Cao, "Intrinsic physical unclonable function (PUF) sensors in commodity devices," *Sensors*, vol. 19, no. 11, p. 2428, May 2019.

[71] B. Han, B. Li, R. Jurdak, P. Zhang, H. Zhang, P. Feng, and C. Yuen, "PBFL: A privacy-preserving blockchain-based federated learning framework with homomorphic encryption and single masking," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 14229–14243, May 2025.

[72] B. Han, B. Li, Y. Zhang, P. Feng, K. Wolter, H. Zhang, Y. Li, R. Jurdak, and C. Yuen, "Repeated game-based long-term incentive mechanism for blockchain-enabled reliable federated learning in IIoT," *IEEE Internet Things J.*, vol. 12, no. 21, pp. 45567–45582, Nov. 2025.

[73] Z. Lu, D. Li, H. Liu, M. Gong, and Z. Liu, "An anti-electromagnetic attack PUF based on a configurable ring oscillator for wireless sensor networks," *Sensors*, vol. 17, no. 9, p. 2118, Sep. 2017.

[74] S. Sánchez-Solano, L. F. Rojas-Muñoz, M. C. Martínez-Rodríguez, and P. Brox, "Hardware-efficient configurable ring-oscillator-based physical unclonable function/true random number generator module for secure key management," *Sensors*, vol. 24, no. 17, p. 5674, Aug. 2024.

[75] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, J. Dray, S. Vo, and L. Bassham, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-22 Rev. 1a, Apr. 2010, doi: 10.6028/NIST.SP.800-22r1a.

[76] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Jan. 2018, Tech. Rep. NIST Special Publication 800-90B, doi: 10.6028/NIST.SP.800-90B.

[77] *Information Security, Cybersecurity and Privacy Protection–Physically Unclonable Functions, Part 1: Security Requirements*, Standard ISO/IEC 20897-1, Dec. 2020. [Online]. Available: https://www.iso.org/standard/76353.html

[78] *Information Security, Cybersecurity and Privacy Protection–Physically Unclonable Functions, Part 2: Test and Evaluation Methods*, Standard ISO/IEC 20897-2, May 2022. [Online]. Available: https://www.iso.org/standard/76354.html

[79] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 889–902, Jun. 2015.

[80] E. Camacho-Ruiz, "Design of a hardware root-of-trust on embedded systems," Ph.D. thesis, Electrónica y Electromagnetismo, Univ. Seville, Seville, Spain, Mar. 2024. [Online]. Available: https://idus.us.es/handle/11441/158566

[81] *Secure Hash Standard (SHS)*, document FIPS 180-4, National Institute of Standards and Technology (NIST), Aug. 2015.

[82] H. Mestiri, F. Kahri, B. Bouallegue, and M. Machhout, "Efficient FPGA hardware implementation of secure hash function SHA-2," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 1, pp. 9–15, Dec. 2014.

[83] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *Proc. 7th Int. Symp. Embedded Comput. Syst. Design (ISED)*, India, Dec. 2017, pp. 1–4.

[84] V. D. Phan, H. L. Pham, T. H. Tran, and Y. Nakashima, "High performance multicore SHA-256 accelerator using fully parallel computation and local memory," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Japan, Apr. 2021, pp. 1–3.

[85] S.-H. Lee and K.-W. Shin, "An efficient implementation of SHA processor including three hash algorithms (SHA-512, SHA-512/224, SHA-512/256)," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2018, pp. 1–4.

[86] S. Josefsson and I. Liusvaara, *Edwards-Curve Digital Signature Algorithm (EdDSA)*, document RFC 8032, Jan. 2017.

[87] *Digital Signature Standard (DSS)*, document FIPS 186-5, National Institute of Standards and Technology (NIST), Feb. 2023.

[88] P. Navarro-Torrero, E. Camacho-Ruiz, M. C. Martínez-Rodríguez, and P. Brox, "Design of a Karatsuba multiplier to accelerate digital signature schemes on embedded systems," in *Proc. IEEE Nordic Circuits Syst. Conf. (NorCAS)*, Oct. 2024, pp. 1–7.

[89] H. Fujii and D. F. Aranha, "Curve25519 for the cortex-M4 and beyond," in *Proc. Prog. Cryptol. – LATINCRYPT*, Cuba, Sep. 2019, pp. 109–127.

[90] M. A. Mehrabi and C. Doche, "Low-cost, low-power FPGA implementation of ED25519 and CURVE25519 point multiplication," *Information*, vol. 10, no. 9, p. 285, Sep. 2019.

[91] F. Turan and I. Verbauwhede, "Compact and flexible FPGA implementation of Ed25519 and X25519," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 3, pp. 1–21, Apr. 2019.

[92] S. K. Saha, A. N. Butka, M. K. Ahmed, and C. Bobda, "OpenTitan based multi-level security in FPGA system-on-chips," in *Proc. Int. Conf. Field Program. Technol. (ICFPT)*, Japan, Dec. 2023, pp. 302–303.

[93] *RISC-V Open Source Supervisor Binary Interface (OpenSBI)*. Accessed: Apr. 7, 2026. [Online]. Available: https://github.com/riscv-software-src/opensbi

[94] S. Davidson. (2025). *TLS Certificate Lifetimes Will Officially Reduce to 47 Days*. [Online]. Available: https://www.digicert.com/blog/tls-certificate-lifetimes-will-officially-reduce-to-47-days

[95] (2022). *Chaos Computer Club Saves the Health System 400 Million Euros*. [Online]. Available: https://www.ccc.de/updates/2022/konnektoren-400-millionen-geschenk

[96] A. Tesei, D. Lattuca, M. Luise, P. Pagano, J. Ferreira, and P. C. Bartolomeu, "A transparent distributed ledger-based certificate revocation scheme for VANETs," *J. Netw. Comput. Appl.*, vol. 212, Mar. 2023, Art. no. 103569.

[97] M. Pritikin, P. Yee, and D. Harkins, *Enrollment Over Secure Transport*, document RFC 7030, Oct. 2013.

[98] M. Pritikin, M. Richardson, T. Eckert, and K. Watsen, *Bootstrapping Remote Secure Key Infrastructure (BRSKI)*, document RFC 8995, May 2021.

[99] K. Varia. (2025). *IoT PKI+Certificate Management*. [Online]. Available: https://accutivesecurity.com/iot-pki-clm-identity-security/

[100] *CVA6: An Application Class RISC-V CPU Core*. Accessed: Apr. 7, 2026. [Online]. Available: https://docs.openhwgroup.org/projects/cva6-user-manual/

[101] *Buildroot User Manual*. Accessed: Apr. 7, 2026. [Online]. Available: https://buildroot.org/downloads/manual/manual.pdf

[102] (2017). *Genesys 2 FPGA Board Reference Manual*. [Online]. Available: https://digilent.com/reference/reference/programmable-logic/genesys-2/genesys2 pdf

[103] C. Andriamisaina, F. Thabet, G. Chauvon, J. R. Coulon, P. Brox, M. C. M. Rodriguez, A. C. Aldaya, and N. Tuveri. (2023). *Preliminary FPGA Implementation of the SPIRS Platform*. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e503a6aa04&app=PPGMS

SILVIA SISINNI received the M.Sc. and Ph.D. degrees in computer engineering from Politecnico di Torino. Her current research interests include trusted execution environments, trusted computing, trusted channels, and confidential computing. She is a member of the TORSEC Cybersecurity Research Group.

LORENZO FERRO received the M.Sc. degree in computer engineering (cybersecurity) from Politecnico di Torino, where he is currently pursuing the Ph.D. degree in computer engineering. He is also a member of the TORSEC Cybersecurity Research Group, Politecnico di Torino. His research interests include trusted computing, trusted execution environments, and remote attestation.

ENRICO BRAVI received the M.Sc. degree in computer engineering (cybersecurity) from Politecnico di Torino. He is currently pursuing the Ph.D. degree in computer engineering. His current research interests include trusted computing, trusted execution environments, and remote attestation. He is a member of the TORSEC Cybersecurity Research Group, Politecnico di Torino.

PABLO NAVARRO-TORRERO received the B.Sc. degree in physics and the M.Sc. degree in microelectronics from the University of Seville, Spain, in 2022 and 2024, respectively, where he is currently pursuing the Ph.D. degree in science and physical technologies program. In November 2022, he joined Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC/University of Seville, where he is a Technical Researcher. His primary interests include cryptography, hardware security, HW/SW co-design, open hardware, and processor and microcontroller design.

EROS CAMACHO-RUIZ received the B.Sc. degree in physics from Universidad de Córdoba, Spain, in 2017, the M.Sc. degree in microelectronics from Universidad de Sevilla, Spain, in 2020, and the Ph.D. degree (Hons.) in science and physical technologies program from Universidad de Sevilla, in March 2024. He is currently a Postdoctoral Researcher with Instituto de Microelectrónica de Sevilla (IMSE), Centro Nacional de Microelectrónica (CNM), CSIC/University of Seville. His main research interest is the field of hardware security. He has studied and developed physical unclonable functions (PUFs) in analog and digital devices. Apart from that, he has been working on the development of hardware accelerators for classical and post-quantum algorithms in embedded systems and IoT devices.

MACARENA C. MARTÍNEZ-RODRÍGUEZ received the Ph.D. degree (Hons.) in science and physical technologies program from Universidad de Sevilla (USE), in July 2018. Since 2010, she has been a Researcher with Instituto de Microelectrónica de Sevilla (IMSE-CNM), and she became a Tenured Scientist, in 2024. Her main research focus is on the development of efficient digital circuits, and particularly dedicated hardware design of security cryptographic primitives, ring oscillators, and SRAMS physical unclonable functions, and acceleration of post-quantum cryptographic algorithms.

PIEDAD BROX was born in 1979. She received the Doctor of Science degree (Hons.) in microelectronics from the University of Seville, Spain, in 2009. She obtained two postdoctoral fellowships funded by Spanish Government and the University of Seville. She became a Tenured Scientist with the IMSE (CSIC/University of Seville), in 2018. Her background in the design of digital circuits and systems for IA-based image and video processing, and for non-linear controllers, has been applied in the area of security during the last few years. In the design of hardware for security, it has covered two major lines of research: 1) the design of cryptographic primitives based on non-clonable physical functions (PUFs); and 2) the hardware implementation of cryptographic algorithms for encryption and digital signature (both classical and post-quantum cryptography approaches), and message authentication (hash functions and HMAC).

ANTONIO LIOY received the M.Sc. degree (summa cum laude) in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Torino. He is currently a Full Professor with Politecnico di Torino, where he leads the TORSEC Cybersecurity Research Group. His current research interests include network security, policy-based system protection, trusted computing, and electronic identity.