LIRA-V: Lightweight Remote Attestation for Constrained RISC-V Devices

Carlton Shepherd
Smart Card and IoT Security Centre
Information Security Group
Royal Holloway, University of London
Egham, Surrey, United Kingdom
carlton.shepherd@rhul.ac.uk

Konstantinos Markantonakis
Smart Card and IoT Security Centre
Information Security Group
Royal Holloway, University of London
Egham, Surrey, United Kingdom
k.markantonakis@rhul.ac.uk

Georges-Axel Jaloyan
DIENS
École Normale Supérieure
CNRS, PSL University
Paris, France

***Abstract*—This paper presents LIRA-V, a lightweight system for performing remote attestation between constrained devices using the RISC-V architecture. We propose using read-only memory and the RISC-V Physical Memory Protection (PMP) primitive to build a trust anchor for remote attestation and secure channel creation. Moreover, we show how LIRA-V can be used for trusted communication between two devices using mutual attestation. We present the design, implementation and evaluation of LIRA-V using an off-the-shelf RISC-V microcontroller and present performance results to demonstrate its suitability. To our knowledge, we present the first remote attestation mechanism suitable for constrained RISC-V devices, with applications to cyber-physical systems and Internet of Things (IoT) devices.**

I. Introduction

Constrained devices, e.g., microcontroller units (MCUs), are used ubiquitously in home and building automation, manufacturing, assistive technologies, and industrial control systems. In these situations, it is important to prevent damaging behaviour arising from malware and other unauthorised software. To this end, remote attestation (RA) protocols have been developed to assess remote devices for the presence of unauthorised software. RA is a challenge-response protocol where a remote verifier, $\mathcal{V}$, ascertains the platform operating state of a proving device, $\mathcal{P}$. RA systems rely on a trusted component—a trust anchor—for measuring and signing response reports, or *quotes*, from $\mathcal{V}$. Increasingly, constrained devices are equipped with CPU security extensions from which trust anchors can be built with hardware-assisted secure execution.

Introduced in 2010, RISC-V is a major architectural divergence from popular proprietary architectures, such as Intel and ARM. Consequently, their associated security technologies, e.g., Intel SGX and ARM TrustZone respectively, cannot be readily used for creating RA trust anchors on RISC-V devices. Meanwhile, other traditional roots of trust, namely the Trusted Platform Module (TPM) and hardware secure elements, are generally considered too cumbersome for MCU-type devices [1]–[4].

RA mechanisms have hitherto been developed for high-end RISC-V devices using privileged CPU protection modes to implement a security monitor, as used by Keystone [5] and Lebedev et al. [6]. However, current work is yet to address constrained RISC-V devices that have limited computing power, may not possess a memory management unit (MMU), and may only use a single application and CPU privilege mode.

This paper presents the first RA mechanism for constrained RISC-V devices. We leverage the Physical Memory Protection (PMP) primitive [7] and read-only memory (ROM) to build a trust anchor that does not require a fully-fledged TEE, separate CPU privilege modes, or dedicated security hardware like a TPM or secure element. Specifically, we leverage PMP to create execute-only memory as a lightweight method for protecting quote signing key secrecy, while device ROM ensures the integrity of RA measurement and reporting procedures. With this, we present the design and development of the first RISC-V RA mechanism for constrained devices, and show how it can be used for secure and trusted communication between devices using mutual attestation. To support this, we develop a bi-directional attestation protocol, which is verified using formal symbolic analysis. LIRA-V is implemented and evaluated using an off-the-shelf RISC-V MCU, which executes in $\sim$11–32s for secure and trusted channel creation when attesting 64–256KB of memory on both devices. The average case occupies $<$4KB RAM, thus demonstrating suitability for IETF Class 1 and 2 constrained devices [8].

II. RISC-V Preliminaries

RISC-V is an open-source, load-store instruction set architecture (ISA) with variable word widths (32, 64 and 128 bits) that enables vendors to pursue modular chip design. Devices must support the *base* ISA with 32-bit integer operations as a minimum requirement. Vendors may implement various extensions thereafter, such as single and double-precision floating-point, hypervisor, vector, and single-instruction multiple data instructions. Cryptographic and trusted execution environment (TEE) extensions are under development at the time of writing.

The RISC-V Privileged Architecture Specification [7] describes three CPU privilege modes: *machine* (M), *supervisor* (S) and *user* (U) mode. M mode is the highest privileged level, akin to EL3 in ARMv8, that is intended for device firmware, e.g., bootloaders. Machine mode is uninterruptible, cannot be disrupted by lower privilege levels, and is the first mode entered upon a device reset. M mode also operates only on physical addresses; it does not access memory using virtual

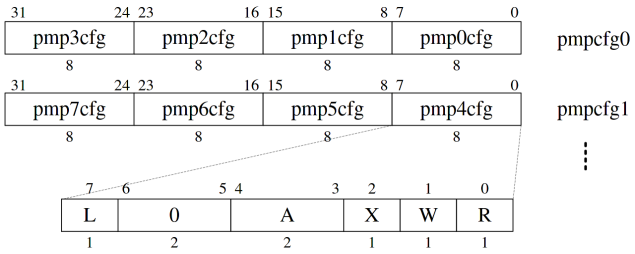

Fig. 1. RISC-V PMP configuration registers [7].

address translation. All RISC-V implementations must support M mode as a mandatory requirement. S-mode is designed for privileged OS services, such as virtual memory management, while U-mode is intended for low privilege applications. In general, embedded devices—the focus of this work—are envisaged to possess either only M or M and U modes [7].

Physical Memory Protection (PMP) is a RISC-V security enhancement proposed in 2017 that allows the designation of access control permissions—read (`R`), write (`W`), execute (`X`)—to physical memory address regions while in machine mode. PMP is configured using a bank of RISC-V control status registers (CSRs) that specify the desired permission and the associated address (`A`) region. Setting the PMP lock (`L`) bit can be optionally used to prevent M-mode from modifying that region's access control permissions once set; locked registers are released only when the CPU is reset. PMP checks are applied to all memory accesses when the current processor privilege mode is in S- or U-mode when the `L` is not set, or to all modes when `L` is set. PMP access violations are trapped at the processor level. Setting target regions and their access control parameters is performed using the PMP address (`pmpaddr`) and configuration (`pmpcfg`) registers on a supported RISC-V device, shown in Fig. 1.

III. Remote Attestation and Related Work

Traditional RA involves a remote verifier, $\mathcal{V}$, that requests the operating state of a single target device, $\mathcal{P}$. At a high level, RA systems comprise *measurement* and *reporting* stages. Historically, the TPM developed by the Trusted Computing Group (TCG) has served as a popular root of trust for storing system measurements. The TPM computes a cryptographically secure hash function over critical system component binaries and stores these measurements in its platform configuration registers (PCRs). During the reporting phase, the PCR values are signed using a hardware-bound attestation identity key (AIK) to produce the quote report that is sent $\mathcal{P} \rightarrow \mathcal{V}$.

RA protocols have been proposed using various software and hardware trust anchors, such as physically unclonable functions (PUFs) [6], [9]; TEEs, e.g., Intel SGX and ARM TrustZone [10]–[12]; and custom hardware for read-only and guaranteed execution (TyTan [2], SMART [1], VRASED [13], TrustLite [14], HYDRA [3]). Comparing known measurements at regular time intervals (ERASMUS [15]), traversing control-flow integrity (CFI) graphs [16]–[21], and using execution time as an out-of-band channel (SWATT [22] and Pioneer [23]) have also been explored.

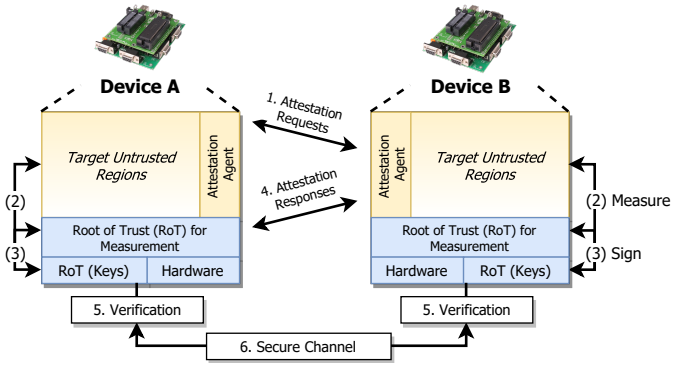

Fig. 2. Bi-directional or mutual remote attestation.

These schemes address attestation involving a single $\mathcal{P}$ and $\mathcal{V}$. Here, $\mathcal{V}$ transmits an attestation request to $\mathcal{P}$, which measures the state of a single or multiple software components at boot- or run-time. The measurement quote is signed using a private key accessible only to the trust anchor, which is returned to and verified by $\mathcal{V}$ using the corresponding public key. In contrast, an emerging paradigm is the development of proposals that attest multiple devices in a single protocol instance. *Mutual* or *bi-directional* RA has been proposed for attesting and bootstrapping secure channels between two devices—shown in Fig. 2—using ARM TrustZone [12], [24], TPMs [25], [26], and PUFs as trust anchors [9]. In other work, the Seda [27] and SANA [28] systems address RA of device swarms by constructing an efficient topological path between its constituents and aggregating the responses for $\mathcal{V}$. The reader is referred to Ambrosin et al. [29] for a recent comprehensive survey of collective attestation schemes.

IV. Proposed Measurement Procedure

This work contributes to the area of mutual remote attestation by proposing a novel mechanism using ROM and RISC-V PMP as a trust anchor. The following sections describe the threat model and system design in detail.

A. Threat Model and Design Goals

We consider a privileged software adversary that has read, write and execute access to memory that is not explicitly protected by hardware-assisted access control, i.e. PMP and read-only memory. Such an adversary may exploit an existing software vulnerability—a vulnerable network service or connected I/O device—that provides access unprotected memory regions. The attacker then establishes a persistent presence on the device by overwriting untrusted memory. The attacker may operate in M mode, which may be the only CPU protection mode supported by a constrained device. However, in line with other hardware-based RA proposals, we *do* trust the RISC-V PMP primitive and a ROM unit for preserving the integrity of signing keys and the core root of trust for measurement (CRTM) code. As such, sophisticated attacks that bypass the protections afforded by PMP, like physical fault injections,

are considered beyond the scope of this work. Additionally, we assume that the measuring procedure executes atomically. Numerous proposals have been made for enforcing this [30], [31] using a small hardware extension for validating program counter (PC) bounds to enforce code entry points. Unlike existing proprietary processor architectures, this is significantly eased by the open-source nature of many RISC-V cores.

The following security goals are addressed in this work:

[G1] **Measurement Procedure Integrity**: the proposal shall resist privileged attacks that attempt to modify the code for measuring the device's current operating state.

[G2] **Signing Key Secrecy**: the key used for signing response quotes for entity authentication shall remain secret against privileged adversaries under the threat model.

[G3] **Secure Channel Creation**: the proposed mechanism shall bootstrap a secure channel for bi-directional data transfer between $\mathcal{P}$ and $\mathcal{V}$ with mutual entity authentication, replay protection, and forward secrecy.

[G4] **Bi-Directional Attestation**: secure channel creation shall be predicated on $\mathcal{P}$ and $\mathcal{V}$ having known and authorised platform configuration states in a single protocol instance.

The following deployability goals are also considered:

[D1] **Constrained Device Suitability**: the proposed system shall be suitable for constrained devices, e.g., MCUs. We assume the devices can execute standard public-key cryptographic algorithms, but can benefit from lightweight mechanisms due to computational constraints. This corresponds to IETF Class 1 and 2 devices (10–50KB RAM, 100–250KB storage, and a CPU in the MHz range) [8].

[D2] **Cost and Flexibility**: the proposal shall avoid relying upon external security hardware or CPU extensions, such as privilege modes unlikely to exist on embedded devices, e.g., hypervisor modes, and security co-processors.

B. Proposed Measurement Procedure

The proposed trust anchor is underpinned by two trusted components: the RISC-V PMP primitive for configuring physical memory regions with access control permissions (see §II) and ROM code for configuring PMP registers at start-up. The target is a standard, single-core embedded system with its components deployed in a system-on-chip (SoC) that executes basic configuration routines in ROM before transferring control to the primary program in persistent memory. LIRA-V comprises the following stages, shown in Fig. 3:

1) PMP Configuration: Following a reset signal and basic SoC setup, ROM code is used to assign PMP access control permissions to physical memory regions. A single PMP entry is configured to denote execute-only (X-only) permissions to the region that encompasses the quote signing key (QSK) embedded in its associated signing method. For measurement integrity, ROM is used to host the CRTM and its dependent cryptographic algorithms.

2) Attestation Request: The attestation agent (AA) is untrusted code that implements the services for transmitting/receiving attestation messages to/from $\mathcal{V}$ over a network medium, e.g., Wi-Fi, Bluetooth, or LPWAN. AA invokes the ROM CRTM for executing the measurement procedure after receiving an attestation request.

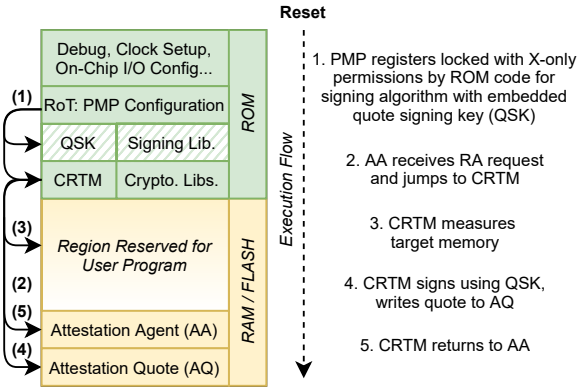

Fig. 3. Remote attestation measurement procedure. Trusted components shown in green, untrusted in yellow, and PMP-configured regions are hatched.

3) CRTM: The ROM CRTM computes an aggregated hash of the device physical memory region of the form $H(\{m_i, m_{i+1}, \ldots, m_{i+b}\}, H(\{m_{i+1+b}, m_{i+2+b}, \ldots, m_{i+j+b}\}, H(\ldots)))$, where $H(\cdot)$ is a cryptographic hash function, m_i is the contents of the i^{th} address, and b is the address block size to be hashed. The attested region is pre-programmed in ROM CRTM that encompasses all or part of the memory map. We evaluate the performance of varying regions in §VII.

4) Quote Signing: The aggregated measurement is signed using the ROM QSK for entity authentication. The corresponding verification key is held by $\mathcal{V}$ before deployment. As per [G2] (§IV-A), the QSK must remain confidential to prevent an adversary forging valid quotes while unauthorised software persists on the device. We leverage RISC-V PMP to set X-only memory to a signing procedure with an embedded QSK as a lightweight method for achieving this. Lastly, the signed attestation quote is placed into AA-readable memory.

5) Attestation Response: AA returns the attestation quote (AQ) to $\mathcal{V}$ as part of a secure remote attestation protocol (§V), who verifies the signature and whether the quote measurement conforms to expectations.

V. Proposed Protocol

This section develops the *reporting* phase of LIRA-V for secure quote transmission, which formalises a novel secure channel protocol with bi-directional attestation.

A. Assumptions and Threat Model

From [G3] and [G4] (§IV-A), the aim is to bootstrap a secure channel following mutual attestation—see Fig. 2—where two constrained devices serve as $\mathcal{P}$ and $\mathcal{V}$. Each device is assumed to implement the measurement procedure outlined in §IV. During the protocol, we address network-based adversaries that attempt to forge attestation responses and intercept, replay, redirect and otherwise manipulate protocol messages permitted under the Dolev-Yao adversarial model. We assume that the proving devices can access a cryptographically secure

Protocol 1 Proposed Mutual Attestation Protocol

1: $A \rightarrow B : n_A \mid\mid q_A \mid\mid AR$

2: $B \rightarrow A : n_B \mid\mid q_B \mid\mid AR \mid\mid AE_K(Q_B \mid\mid \sigma_B(H(Q_B \mid\mid n_A \mid\mid n_B \mid\mid q_B \mid\mid q_A)))$

3: $A \rightarrow B : AE_K(Q_A \mid\mid \sigma_A(H(Q_A \mid\mid n_A \mid\mid n_B \mid\mid q_A \mid\mid q_B))$

TABLE I
NOTATION FOR PROTOCOL 1.

Notation	Description
D_{ID}	Identifier of device D, e.g, IP address or UUID.
n_D	Cryptographically secure nonce generated by D.
q_D	Public ECDH value generated by D (scalar multiplication of an agreed base and D's randomly generated secret point.)
AR	Attestation request operation.
$H(\cdot)$	Cryptographically secure one-way hash function.
$X \mid\mid Y$	Concatenation of X and Y.
$AE_K(\cdot)$	Authenticated symmetric encryption algorithm keyed with K.
Q_D	Measurement response quote signed by D.
$\sigma_D(\cdot)$	Message signed by D using crytographic signature algorithm.

entropy source for random number generation and can execute standard public key cryptography algorithms ([D1], §IV-A).

B. Offline Phase

Each device is provisioned a unique QSK used during the measurement phase for entity authentication. The corresponding public key and expected measurements for verifying responses are provisioned into the opposing devices in PMP-protected ROM by an administrator. For performance, a compact signature scheme is suggested, e.g., Ed25519 (256-bit key sizes). Devices may also be enrolled in a group signature scheme with multiple QSKs mapped to a single group verification key for scalability purposes. Elliptic Curve Diffie-Hellmann (ECDH) key exchange is proposed as an efficient basis for mutual key agreement. As such, common domain parameters, $\Delta = (p, a, b, G, \eta, h)$ (in the prime case), must be standardised across devices. The domain parameters, signature verification keys, and expected measurement hashes are provisioned in trusted ROM prior to deployment.

C. Online Phase

The proposed mutual RA protocol is presented in Protocol 1, comprising three messages between devices A and B:

1) M1: A initiates the protocol with B by transmitting a cryptographically secure nonce, n_A; its public ECDH point, $q_A = d_A \cdot G$, where d_A is the randomly generated private key and G is the base using domain parameters, Δ; and an attestation request flag, AR, of $A \rightarrow B$.

2) M2: B executes the measurement process in §IV to produce its signed attestation quote, Q_B. B generates its secure nonce, n_B, and ECDH public point, $q_B = d_B \cdot G$. Using q_A from A, B derives the shared ECDH secret using $d_B \cdot q_A \cdot G$, and applies a key derivation function to derive an ephemeral session key, K. This key is used with an authenticated encryption algorithm, AE, e.g., AES-GCM, to encrypt its quote response, Q_B, and a signed message of $H(Q_B||n_A||n_B||q_B||q_A)$ signed using B's QSK.

3) M3: A computes the shared ECDH secret ($d_A \cdot q_B \cdot G$) and derives K. A decrypts M2 the sent by B and verifies the signature and measurements of Q_B therein. A aborts the protocol if Q_B fails A's signature verification, the measurements deviate from the expected value, or AE contains an invalid authentication tag. Next, A executes its measurement procedure, after which AE_K is used to encrypt the measured quote, Q_A, and $H(Q_A||n_A||n_B||q_A||q_B)$ under A's QSK, which are sent to B. B then validates the signature and measurement contents of A's quote using its public key. □

D. Formal Verification

We subject the proposed bi-directional attestation protocol to formal verification using the SCYTHER analyser [32], which operates under the symbolic model using the perfect cryptography assumption. Previously, SCYTHER has been used to formally verify Internet Key Exchange (IKEv1 and IKEv2) [33], ISO/IEC 9798 [34] and 11770 [35] protocol families, and WiMAX (IEEE 802.16) [36]. A given protocol is specified using the Security Protocol Description Language (SPDL) defining the communicating entities (*roles*); the protocol messages using built-in primitives, e.g., nonces, hash functions, symmetric and public-key signature algorithms, and user-defined types; and the security properties to assess (*claims*). SCYTHER analyses whether the claims hold against all possible behaviours of a Dolev-Yao adversary (*traces*) using the protocol specification. No attacks were discovered on the proposed protocol; the claims of quote and session key secrecy (confidentiality), reachability, aliveness (weak authentication) and non-injective agreement and synchronisation (replay attack protection) were successfully maintained. The full protocol specification is released freely.[1]

VI. PROTOCOL IMPLEMENTATION

A. Platform Specifications

We implemented a proof-of-concept of our proposal using the SiFive HiFive1 Rev B—a 32-bit RISC-V single-board computer with a low-powered FE310-G002 MCU SoC with 16KB SRAM and 4MB off-chip SPI flash memory. The SoC hosts a single-core E31 CPU at 320MHz; a 16KB L1 instruction cache; multiplication, atomic, and compressed instruction extensions (RV32IMAC); and support for the Privileged Architecture specification with 8 configurable PMP registers [7].

B. Measurement Phase

The CRTM (§IV-B) enumerates a memory range in contiguous blocks between two pre-programmed addresses. On request, the CRTM function computes an aggregated hash of the memory contents in this range to create the raw measurement. SHA-3 was used as the hash function using

[1]Protocol SCYTHER source URL: https://cs.gl/extra/ecc-btp-scyther.spdl

TABLE II
MEAN PROTOCOL WALL-CLOCK TIMES (SECONDS) BY DEVICE FOR VARIOUS ATTESTED MEMORY RANGES (1KB BLOCKS, 100 ITRS.; 4.S.F).

	Total CRTM Attested Memory								
	64KB			*128KB*			*256KB*		
Message	*A*	*B*	**Total**	*A*	*B*	**Total**	*A*	*B*	**Total**
M1: $A \rightarrow B$	0.342	0.049	0.391	0.346	0.052	0.398	0.341	0.048	0.389
M2: $B \rightarrow A$	1.073	4.698	5.771	1.069	8.289	9.358	1.067	14.93	16.00
M3: $A \rightarrow B$	4.701	0.097	4.798	8.302	0.103	8.405	14.92	0.090	15.01
Grand Total	6.116	4.844	10.96	9.717	8.444	18.16	16.33	15.07	31.40

`tiny_sha3`, a portable implementation of FIPS-202/SHA-3 [37]. The Freedom Metal library of the Freedom E SDK—a hardware abstraction layer for providing portability between Freedom E RISC-V targets—was used for PMP configuration. The SDK provides C interfaces for configuring `pmpcfg` and `pmpaddr` registers to assign access control permissions, e.g., `R`/`W`/`X`/`L` (Fig. 1), to a contiguous memory address region.

C. Quote Reporting

The protocol was implemented using the RISC-V port of the Network and Cryptography Library (NaCl) for ECC mutual key agreement using X25519 and symmetric authenticated encryption using the XSalsa20-Poly1305 construction [38], [39]. SHA-3 served as the CRTM hash function using `tiny_sha3`, an embedded implementation of FIPS-202/SHA-3 [37]. For QSK, Ed25519 was employed using an adapted implementation from the Keystone project [5]. This adaption replaced using a byte buffer for defining the signing key with an assembly routine that loaded immediate values from PMP-protected static memory for X-only execution. Both devices were connected to a Windows workstation over USB serial for inter-device connectivity and protocol orchestration. A Python script was developed for benchmarking runs using the Time module with microsecond precision, and used the PySerial module for initiating the protocol on device A over the serial interface. The timer was terminated by the script after reading an acknowledgement serial message from device B after it received and validated [M3] (§V-C).

VII. EVALUATION

A. Performance Analysis

Our implementation was cross-compiled for 32-bit RISC-V platforms using the GNU C Compiler with the `-Os` optimisation flag, and deployed on both boards using the Freedom E SDK [40]. The stack and heap were globally limited to use 6KB and 1KB respectively per device. We first benchmarked the total protocol execution time and its distribution by message and entity. Here, the protocol was executed 100 times using the setup in §VI-C for total attested memory ranges between 64KB–256KB. The mean execution times were computed for each message and by device. The results are presented in Table II and Fig. 4. The measurements show that [M2] and [M3] dominate the total execution time, accounting for ∼48% per message of the total protocol time.

From this, we further characterised CRTM measurement time against the attested memory size and the block size (the number of contiguous memory addresses that are hashed). The default block size was tested at values between 1KB—4KB, the maximum our test devices could accommodate,[2] while the total attested memory ranged from 1KB to the maximum size of flash memory (4MB). These results are shown in Table III and Fig. 5. Expectedly, the measurement time is directly proportional to the total attested memory. At worst, attesting 4MB required 207.2s (1KB blocks) and 201.1s (4KB). In general, the rate of increase is ∼1.6s for every 32KB of attested memory, while the total time doubles for each doubling of attesting memory. A 1KB–4KB increase in block size produces a small reduction (∼3%) of the measurement time, thus exhibiting a memory-time trade-off given the greater required stack allocation for 4KB blocks.

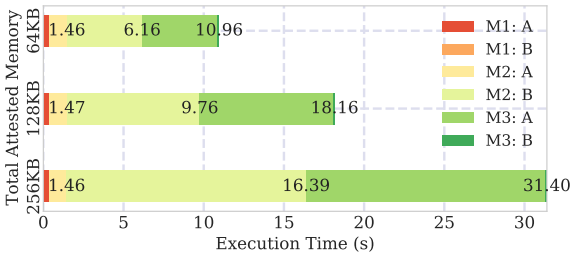

Fig. 4. Mean cumulative protocol execution time.

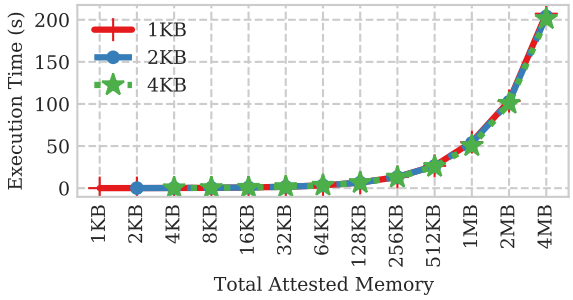

Fig. 5. CRTM execution time for varying attested memory and block sizes.

B. Security Evaluation, Attacks, and Countermeasures

The threat model in §IV-A describes a privileged adversary that can access memory that is not protected by hardware security. In §IV, we proposed moving the root of trust to ROM, which executes immediately following a device reset and before the execution of untrusted memory. This configures PMP regions to secure the integrity of the CRTM, thus satisfying [G1]. The confidentiality of QSKs is preserved by embedding the key into an X-only signing routine configured in start-up ROM using PMP with the `L`-bit set, thus meeting [G2]. We also proposed a novel protocol that bootstraps a secure channel using ephemeral keys and the mutual attestation of both devices for meeting [G3] and [G4]. We now analyse some specific attacks under the threat model in §IV-A:

[A1] *Modifying QSK's PMP address range or its access control permissions*. The QSK PMP entry is locked

[2]The global stack required 6KB when using 4KB blocks, while only a 4KB stack was needed to support 1KB blocks in our implementation.

TABLE III
MEAN WALL-CLOCK CRTM EXECUTION TIME (SECONDS) FOR VARYING MEMORY BLOCK SIZES AND TOTAL MEMORY (100 ITRS.; 4 S.F.).

Block	1KB	2KB	4KB	8KB	16KB	32KB	64KB	128KB	256KB	512KB	1MB	2MB	4MB
1KB	0.050	0.102	0.206	0.411	0.826	1.646	3.311	6.625	13.25	26.53	54.76	103.5	207.2
2KB	—	0.099	0.202	0.409	0.823	1.650	3.309	6.621	13.24	26.41	53.71	102.6	204.0
4KB	—	—	0.191	0.398	0.802	1.609	3.212	6.431	12.86	25.53	50.27	100.5	201.1

after being set by start-up ROM, which prevents writes to its configuration (`pmpcfg`) and address (`pmpaddr`) registers in M mode. Locked entries are released only after a device reset; since PMP registers are configured *before* untrusted memory is afforded control, the attacker is unable to modify QSK's PMP registers after boot-time.

[A2] *Transferring removable flash memory.* Many embedded devices rely on removable memory, e.g., SD cards. The trust anchor—QSK and PMP configuration code—resides in ROM, and so replacing removable memory would affect attestation agent and signed quote availability. The device would be unable to respond to RA requests, akin to disabling the device. Such attacks are difficult to address without non-removable memory or tamper-resistant casing.

[A3] *Adversary overwrites the attestation quote (AQ) in untrusted memory.* This yields effects akin to [A2]: the device will fail to transmit a valid quote to $\mathcal{V}$, therefore failing the attestation process and notifying $\mathcal{V}$ of an issue.

[A4] *Leveraging side-effects of X-only code.* We use RISC-V PMP to assign an X-only region to protect the confidentiality of quote signing. A general limitation of X-only code is that, while direct reads and writes are prohibited, CPU caches and registers still exhibit the effects of executed code. Consequently, these must be correctly sanitised to mitigate potential information leakage before returning control flow to untrusted memory.

C. Limitations

Our proposal focused on single-core MCUs that possess a single application and CPU protection mode. Attesting dynamically loaded tasks or applications using a real-time OS (RTOS) poses greater challenges, which we defer to future research. Secondly, by relaxing the atomicity requirement, we are able to construct a lightweight and portable system that requires no additional security hardware, e.g., TPMs, or CPU protection modes. As such, our proposal is not best suited against sophisticated adaptive malware that transiently and reactively relocates during the attestation process to avoid detection. Addressing this challenge remains an open challenge in the literature without resorting to additional hardware [1], [13], [30], which contravenes [D1] (§IV-A). Both Nunes et al. [13] and Francillon et al. [30] propose countermeasures using dedicated hardware extensions that enforce controlled invocation and PC validation. In future work, we aim to investigate the use of PMP and trusted ROM to provide these properties. Furthermore, we consider our proof-of-concept performance measurements baseline results:

TABLE IV
COMPARISON OF RELATED RA PROPOSALS USING CRITERIA FROM §IV-A.

Root of Trust	Work	G1	G2	G3	G4	D1	D2
TPM	Gasmi et al. [26]	●	●	●	●	◐	○
	Greveler et al. [25]	●	●	○	●	◐	○
	TPM 2.0 DAA [41]	●	●	○	○	◐	○
PUF	Lebedev et al. [6]	●	●	○	○	○	◐
Intel SGX	EPID [42]	●	●	●	○	○	◐
HW+SW Co-Design	SMART [1]	●	●	○	○	●	◐
	VRASED [13]	●	●	○	○	●	◐
EA-MPU	TyTAN [2]	●	●	○	○	●	◐
	TrustLite [14]	●	●	○	○	●	◐
Microkernel	HYDRA [3]	●	●	○	○	◐	◐
GP TEE	Shepherd et al. [12]	●	●	●	●	◐	◐
ARM TrustZone	Schulz et al. [10]	●	●	○	○	●	●
RISC-V PMP	Keystone [5]	●	●	○	○	○	●
	LIRA-V	●	●	●	●	●	●

EA-MPU: Execution-aware memory protection unit, GP: GlobalPlatform.
●: Satisfies, ◐: Partially satisfies, ○: Unsatisfactory.

optimisation was not the primary objective, and it is likely that significant benefits could be made through using, for example, a hardware-accelerated hash function for the CRTM. Lastly, a compromised device may refuse to participate in RA protocols by dropping messages to/from $\mathcal{V}$. Such non-participation attacks are generally beyond the scope of RA systems. A countermeasure is fixing RA timeouts and black-listing or investigating unresponsive devices.

VIII. CONCLUSION

This paper presented LIRA-V, a lightweight attestation system for RISC-V constrained devices. Our proposal uses on-board ROM and PMP to build X-only memory for preserving the integrity and confidentiality of attestation measurement and reporting. To the best of our knowledge, LIRA-V is the first remote attestation mechanism for RISC-V constrained devices that does not require dedicated security hardware, TEEs, or separate CPU privilege modes to build a trust anchor. We also went beyond related work (Table IV) and showed how LIRA-V can support secure device-to-device communication with bi-directional attestation. We presented a multi-part evaluation with performance measurements from an implementation on an off-the-shelf RISC-V MCU, alongside a security analysis and a limitations discussion. Future work includes, once standardised, migrating and evaluating post-quantum cryptography for constrained device attestation.

ACKNOWLEDGMENTS

The authors would like to thank Blake Loring and the IEEE SafeThings reviewers for their helpful reviews towards improving this paper. This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 88315 (EXFILES).

REFERENCES

[1] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust." in *Network and Distributed Systems Security*, vol. 12, 2012, pp. 1–15.

[2] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: tiny trust anchor for tiny devices," in *52nd Design Automation Conference*, 2015, pp. 1–6.

[3] K. Eldefrawy, N. Rattanavipanon, and G. Tsudik, "HYDRA: Hybrid design for remote attestation (using a formally verified microkernel)," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2017, pp. 99–110.

[4] C. Shepherd, R. N. Akram, and K. Markantonakis, "EmLog: tamper-resistant system logging for constrained devices with TEEs," in *11th IFIP International Conference on Information Security Theory and Practice*. Springer, 2017, pp. 75–92.

[5] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the 15th European Conference on Computer Systems*, 2020, pp. 1–16.

[6] I. Lebedev, K. Hogan, and S. Devadas, "Secure boot and remote attestation in the sanctum processor," in *31st Computer Security Foundations Symposium*. IEEE, 2018, pp. 46–60.

[7] RISC-V Foundation, "The RISC-V Instruction Set Manual Volume II: Privileged Architecture," June 2019, https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMFDQC-and-Priv-v1.11/riscv-privileged-20190608.pdf.

[8] C. Bormann, M. Ersue, and A. Keranen, "RFC 7228: Terminology for constrained-node networks," *Internet Engineering Task Force*, 2014.

[9] R. N. Akram, K. Markantonakis, and K. Mayes, "A privacy preserving application acquisition protocol," in *11th Int'l Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012.

[10] S. Schulz, A. Schaller, F. Kohnhäuser, and S. Katzenbeisser, "Boot attestation: Secure remote reporting with off-the-shelf IoT sensors," in *European Symposium on Research in Computer Security*, ser. ESORICS. Springer, 2017, pp. 437–455.

[11] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *2nd Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, 2013.

[12] C. Shepherd, R. N. Akram, and K. Markantonakis, "Establishing mutually trusted channels for remote sensing devices with trusted execution environments," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES, 2017, pp. 1–10.

[13] I. D. O. Nunes, K. Eldefrawy, N. Rattanavipanon, M. Steiner, and G. Tsudik, "VRASED: A verified hardware/software co-design for remote attestation," in *USENIX Security*, 2019, pp. 1429–1446.

[14] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *Proceedings of the 9th European Conference on Computer Systems*, 2014, pp. 1–14.

[15] X. Carpent, G. Tsudik, and N. Rattanavipanon, "ERASMUS: Efficient remote attestation via self-measurement for unattended settings," in *Design, Automation and Test in Europe*. IEEE, 2018, pp. 1191–1194.

[16] I. D. O. Nunes, S. Jakkamsetti, and G. Tsudik, "Tiny-CFA: A minimalistic approach for control-flow attestation using verified proofs of execution," *arXiv preprint arXiv:2011.07400*, 2020.

[17] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-FLAT: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 743–754.

[18] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N. Asokan, and A.-R. Sadeghi, "LO-FAT: Low-overhead control flow attestation in hardware," in *54th Design Automation Conference*, 2017.

[19] G. Dessouky, T. Abera, A. Ibrahim, and A.-R. Sadeghi, "Litehax: lightweight hardware-assisted attestation of program execution," in *Int'l Conference on Computer-Aided Design*. IEEE, 2018.

[20] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A.-R. Sadeghi, "Atrium: Runtime attestation resilient under memory attacks," in *Int'l Conference on Computer-Aided Design*. IEEE, 2017.

[21] Z. Sun, B. Feng, L. Lu, and S. Jha, "OAT: Attesting operation integrity of embedded devices," in *IEEE Symposium on Security and Privacy*. IEEE, 2020, pp. 1433–1449.

[22] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy*. IEEE, 2004, pp. 272–282.

[23] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. Van Doorn, and P. Khosla, "Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems," in *Proceedings of the 20th ACM Symposium on Operating Systems Principles*, 2005, pp. 1–16.

[24] C. Shepherd, R. N. Akram, and K. Markantonakis, "Remote credential management with mutual attestation for trusted execution environments," in *12th IFIP International Conference on Information Security Theory and Practice*. Springer, 2018, pp. 157–173.

[25] U. Greveler, B. Justus, and D. Loehr, "Mutual remote attestation: enabling system cloning for TPM based platforms," in *International Workshop on Security and Trust Management*. Springer, 2011.

[26] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan, "Beyond secure channels," in *Scalable Trusted Computing*. ACM, 2007.

[27] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "Seda: Scalable embedded device attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 964–975.

[28] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "SANA: secure and scalable aggregate network attestation," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 731–742.

[29] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, "Collective remote attestation at the Internet of Things scale: State-of-the-art and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2447–2461, 2020.

[30] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, "A minimalist approach to remote attestation," in *Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2014, pp. 1–6.

[31] X. Carpent, K. Eldefrawy, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik, "Reconciling remote attestation and safety-critical operation on simple IoT devices," in *55th Design Automation Conference*. IEEE, 2018, pp. 1–6.

[32] C. J. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols," in *International Conference on Computer Aided Verification*. Springer, 2008, pp. 414–418.

[33] C. Cremers, "Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2," in *European Symposium on Research in Computer Security*. Springer, 2011, pp. 315–334.

[34] D. Basin, C. Cremers, and S. Meier, "Provably repairing the ISO/IEC 9798 standard for entity authentication," *Journal of Computer Security*, vol. 21, no. 6, pp. 817–846, 2013.

[35] C. Cremers and M. Horvat, "Improving the ISO/IEC 11770 standard for key management techniques," in *International Conference on Research in Security Standardisation*. Springer, 2014, pp. 215–235.

[36] A. M. Taha, A. T. Abdel-Hamid, and S. Tahar, "Formal verification of IEEE 802.16 security sublayer using scyther tool," in *International Conference on Network and Service Security*. IEEE, 2009, pp. 1–5.

[37] M.-J. O. Saarinen, "tiny_sha3: Very small, readable implementation of the FIPS 202 and SHA3 hash function," 2021, https://github.com/mjosaarinen/tiny_sha3.

[38] S. Van den Berg, "NaCl in RISC-V," 2021, https://github.com/stefanberg96/NaCl-RISC-V.

[39] D. J. Bernstein, T. Lange, and P. Schwabe, "NaCl: Networking and cryptography library," 2011, http://nacl.cr.yp.to.

[40] SiFive, "RISC-V Freedom E SDK," 2021, https://github.com/sifive/freedom-e-sdk.

[41] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, 2004, pp. 132–145.

[42] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel Software Guard Extensions: EPID provisioning and attestation services," *Intel White Paper*, vol. 1, no. 1-10, p. 119, 2016.