Secure Remote Attestation for IoT Device Swarms under Roving Malware Threats

H. O. Abdulraheem*, I. Idris, J. A. Ojeniyi, I. A. Ibrahim

Cybersecurity Science Department, Federal University of Technology Minna, Niger State, Nigeria.

ABSTRACT: Remote attestation for IoT swarms faces critical gaps: software-based methods suffer high latency and assume tamper-proof hardware, while hardware-based solutions are costly and scale poorly. No existing approach provides both scalability and physical security simultaneously. This paper introduces a PUF-enhanced parallel attestation protocol that partitions memory across swarm nodes to achieve fast verification while embedding hardware-based security. The protocol was implemented on Raspberry Pi3 platforms as IoT provers with a dedicated 256 KB golden firmware region, excluding device-specific data to prevent false positives. Each device's memory was partitioned into blocks, with one block assigned per device for parallel attestation; each device used its PUF response to seed random bit sampling from its assigned block to generate a compact checksum. The border router aggregated individual checksums into a cumulative response for the verifier, with each device adding its MAC using a pre-shared key to prevent tampering with the router. The protocol was evaluated against three state-of-the-art techniques (SWARNA, FeSA, and JANUS) under identical hardware and memory configurations, measuring end-to-end latency, communication overhead, and energy consumption across swarm sizes of 32, 64, 128, and 256 nodes. The proposed technique detected roving malware with 95% accuracy. Experimental measurements showed end-to-end latency of 0.046 s for 32 nodes, representing reductions of 97%, 95%, and 91% compared to SWARNA (1.5 s), FeSA (1.0 s), and JANUS (0.5 s), respectively. The latency decreased as swarm size increased due to parallelism, while per-device communication overhead remained constant at 128 bytes regardless of swarm population. Energy consumption was approximately 154 μJ per device per round, which is about 17× lower than that of SWARNA and 650× lower than that of JANUS. This work provides a scalable, physically secure, and energy-efficient attestation solution for IoT swarms, suitable for smart manufacturing, industrial IoT, and defense environments, deployable on commodity hardware without expensive trusted modules.

KEYWORDS: Internet of Things (IoT), Remote Attestation, Swarms, Physically Unclonable Functions (PUFs), IoT Nodes

I. INTRODUCTION

The Internet of Things (IoT) is becoming more prevalent in several sectors e.g. smart industrial process control, smart healthcare, smart cities, military, and other domains (Orman, 2025; Song et al., 2024). Hence, it is imperative to encrypt the data and code stored on these embedded nodes. The abundance of IoT devices generating sensitive data renders them an appealing target for cybercriminals (M. A. Khan, Shalu, et al., 2024). A recent analysis revealed that firmware-related issues accounted for 95% of the vulnerabilities identified in smart devices, as reported by (Bhole et al., 2024; S. Khan et al., 2025; Wu et al., 2024.). Attestation refers to the method of verifying the legitimacy of the firmware/software installed on an embedded device, ensuring that it has not been tampered with in a malicious manner. A verifier is the reliable unit responsible for initiating an attestation process, whereas the prover is the embedded device that provides evidence of its own validity. Conventionally, the verifier dispatches a challenge (Baig et al., 2025) to the prover. The prover computes a hash digest of the items stored in its memory and then provides this digest to the verifier. The verifier utilizes this checksum to identify compromised devices. The primary limitation in traditional techniques is the increased computational difficulty and extended verification delay caused by the need to go over the entire memory many times in order to produce the hash-based checksum.

Current software-based attestation solutions are built upon two key assumptions: (1) the adversary remains inactive during the verification process, and (2) the hardware of prover is supposed to be secure in contrary to any alterations. Consequently, current software verification systems are inadequate for situations in which a malicious individual has physical access to the hardware of the prover. To address this matter, this paper suggests a software verification method that incorporates a hardware section in an arrangement of a physically unclonable function (PUF). It can be manufactured at a significantly low expense, occupying little silicon area and energy consumption. Consequently, it becomes an appealing option for software verification in the IoT. The second problem addressed in this paper is the issue of verifying the authenticity and integrity of large-scale IoT systems. The notion of device swarms has become prominent in the field of IoT systems in recent years. Swarms are extensive, autonomous networks of IoT devices. For instance, in a smart factory dedicated to vehicle assembly, every robotic arm can be linked to equipment that measure precise joint locations, stresses, torques, and the degree of corrosion, among other parameters. These devices can establish an interconnected network to create a swarm. The devices can subsequently exchange information and collaborate with one another to enhance the overall efficiency of the production process. The current methods for re mote attestation suffer from limited scalability as a result of their substantial verification burden. An effective attestation technique is necessary to verify the accurate and secure operation of large-scale networks, such as IoT device swarms. This paper proposes a multi-node

attestation system that divides the memory into blocks and assigns one block to each IoT device. Each IoT device, thus, measures different portions of the firmware in parallel. This not only leads to an efficient remote attestation technique but also leads to lower latency, thanks to the parallelism.

Existing attestation techniques can be broadly classified into three categories: software-based, hardware-based, and hybrid approaches. Each category has distinct advantages and fundamental limitations that constrain its applicability to IoT swarms. Software-based attestation techniques rely solely on cryptographic hashing and code execution to verify device integrity. Representative techniques include PROVE (Dushku et al., 2023), which provides public verifiability through a challenge-response protocol; SOTERIA (Kwon, 2025), which uses aggregate signatures for scalability; and GAROTA (Aliaj et al., 2022), which employs an active root-of-trust architecture. While these techniques avoid additional hardware costs, they share a critical limitation: they assume the adversary remains inactive during verification and that the hardware is secure against physical tampering. Consequently, software-based approaches are vulnerable to physical attacks where an adversary gains direct hardware access. Furthermore, they typically require traversing the entire memory to compute a hash digest, resulting in O(m) verification latency per device, which scales poorly for large swarms.

Hardware-based attestation embeds trusted execution environments (TEEs) or physically unclonable functions (PUFs) directly into the device hardware. Examples include the use of ARM TrustZone (Ammar et al., 2025), FPGA-based root-of-trust implementations (Ahmadi et al., 2024), and PUF-based key generation (Baig et al., 2025). These techniques provide strong protection against physical tampering by binding the attestation process to immutable hardware characteristics. However, they introduce substantial drawbacks: (i) they require specialized, often expensive hardware that is not available on commodity IoT devices, (ii) they increase silicon area and power consumption, and (iii) they do not inherently address the scalability challenge of verifying large swarms, as each device still performs independent attestation with O(m) per-device complexity.

Hybrid approaches combine software and hardware elements to balance cost and security. Notable examples include DuATT (Athar et al., 2025), which employs a dual-layer attestation scheme for PLC-based industrial IoT, and HA-CAAP (Kuang et al., 2023), which integrates hardware anchors with software-based verification. While these techniques improve resilience against physical attacks compared to pure software methods, they still incur significant overhead. DuATT relies on public-key cryptography for inter-layer communication, which is computationally expensive for resource-constrained nodes. HA-CAAP requires trusted hardware modules on all devices, increasing deployment costs. Moreover, neither technique explicitly addresses the challenge of verifying a large swarm in parallel, leaving the O(n·m) complexity issue unresolved.

The notion of device swarms has become prominent in the field of IoT systems in recent years. Swarms are extensive, autonomous networks of IoT devices. For instance, in a smart factory dedicated to vehicle assembly, every robotic arm can be linked to equipment that measures precise joint locations, stresses, torques, and the degree of corrosion, among other parameters. These devices can establish an interconnected network to create a swarm. The devices can subsequently exchange information and collaborate with one another to enhance the overall efficiency of the production process.

For IoT swarms specifically, a limited number of attestation techniques have been proposed. These include swarm attestation protocols by Ammar et al. (n.d.), Dushku et al. (2023), and Kassem et al. (2025), as well as PUF-based swarm attestation by M. A. Khan, Aman, et al. (2024) and Usman & Asplund (2024). A critical analysis of these techniques reveals three persistent limitations:

First, most swarm attestation protocols require each device to compute a full memory hash, leading to O(n·m) total verification complexity. For example, the protocol by Dushku et al. (2023) verifies each device independently, resulting in linear growth in verification time with swarm size. This fundamentally limits scalability for swarms exceeding 100 nodes. Second, techniques that support parallel attestation (e.g., Kassem et al., 2025; Kuang et al., 2023) either rely on expensive trusted hardware modules (such as TPMs) or require public-key infrastructure for secure communication. Both approaches impose prohibitive computational and energy overhead on resource-constrained IoT devices, making them impractical for large-scale deployment. Third, no existing swarm attestation method simultaneously addresses physical tampering attacks while maintaining low latency. Software-based approaches (e.g., Kumar et al., 2022; Zhang et al., 2024) assume the hardware is secure against physical attacks. Hardware-based approaches (e.g., Ahmadi et al., 2024; Akhtar et al., 2025) provide physical security but are cost-prohibitive for large swarms. Hybrid approaches (e.g., Athar et al., 2025; Kuang et al., 2023) attempt to balance both but introduce significant overhead through public-key cryptography or trusted hardware requirements.

Consequently, there is a clear and specific gap: a lightweight, scalable, and physically secure attestation technique for IoT swarms that (i) avoids full-memory hashing, (ii) operates without expensive trusted hardware or public-key cryptography, and (iii) remains robust against physical attacks. This paper fills that gap by proposing a PUF-enhanced parallel attestation protocol that partitions memory blocks across swarm nodes, achieving O(1) per-node verification latency while embedding physical security into the attestation process itself.

To address these challenges, this paper proposes a software-based attestation method that incorporates a hardware component in the form of a Physically Unclonable Function (PUF). PUFs can be manufactured at a significantly low expense, occupying little silicon area and energy consumption, making them an appealing option for IoT security. Furthermore, this paper proposes a multi-node attestation system that divides the memory into blocks and assigns one block to each IoT device. Each IoT device, thus,

measures different portions of the firmware in parallel. This not only leads to an efficient remote attestation technique but also leads to lower latency, thanks to parallelism. This approach directly fills the identified gap by providing scalability without expensive hardware while maintaining physical security through PUF integration.

The primary contributions of this paper can be succinctly described as:

i. A novel remote attestation technique for IoT device swarms using PUFs, achieving O(1) per-node verification latency.
ii. Exploiting parallelism by attesting multiple memory blocks simultaneously across different devices in the swarm.
iii. Embedding physical security directly into the attestation process through PUF integration, eliminating the need for separate trusted hardware.
iv. A thorough formal security analysis demonstrating up to 95% detection accuracy for roving malware.
v. Experimental validation on actual hardware showing significantly lower latency compared to existing techniques.

The remaining structure is organised as follows: Section II details the system model and assumptions considered in this paper. The threat model is described in Section III while the suggested technique is presented in Section IV.The security of the suggested validating technique is analysed in Section V and the experimental validation is presented in Section VI. The paper is concluded in Section VII.

II. SYSTEM MODEL AND ASSUMPTIONS

An overview of the system model is provided in Figure 1. A set of IoT nodes forming a swarm are linked to a remote verifier through a border router (e.g. 6LoWPAN routers) connected to the Internet.

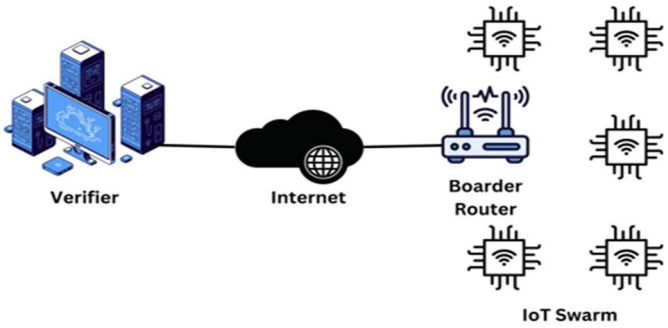


Figure 1 System model

The assumptions made in this paper are as follows:

i. The IoT devices in a swarm share an identical attestable firmware region, specifically, the code section that is identical across all devices. Device-specific variations (e.g., calibration data, MAC addresses, unique IDs) are stored in a separate excluded region that is not subject to attestation. The attestation protocol operates exclusively on the golden firmware region, ensuring that benign deviations do not trigger false alarms. Minor hardware or version differences are tolerated through a one-time capability discovery mechanism.
ii. Each IoT node possesses a hardware root of trust, either a Physically Unclonable Function (PUF), a one-time-programmable (OTP) secret, or a device-unique key derived from SRAM startup behavior. For nodes lacking a PUF, the protocol uses a software-based secret with equivalent security properties under the threat model.
iii. The hardware root of trust is bound to the device such that physical removal or tampering renders it non-functional. Internal communication between the root of trust and the main processor is assumed secure against bus-level snooping.
iv. IoT devices operate under strict constraints in terms of computational capability and available memory.
v. The verifier is fully trusted and possesses sufficient computational and storage resources without practical limitations.
vi. The attestation routine is stored in a write-protected memory region (e.g., flash with secure boot). Its integrity is verified before each attestation round via a stored hash. Firmware updates are permitted but trigger re-initialization of attestation parameters.
vii. The execution of the attestation procedure for an individual memory segment is effectively atomic, interrupts are masked during the critical section, and a watchdog timer enforces a maximum execution window. Block-level attestation completes in microseconds, making interrupt masking feasible on typical IoT hardware.
viii. The device memory is partitioned into two logical regions: (a) the golden firmware region, which contains executable code that is identical across all devices in the swarm, and (b) the device-specific region, which stores configuration data, calibration constants, unique identifiers, and other benign variations. The attestation protocol only measures the golden firmware region; the device-specific region is excluded from attestation. This ensures that legitimate device-to-device variations do not produce false positives during verification.
ix. PUF aging effects can be managed through standard enrollment and periodic recalibration procedures, which are outside the scope of this work.

III. THREAT MODEL

The adversary $\mathcal{A}$ resides persistently in the IoT device's memory and evade exposure. The proposed technique splits the memory of an IoT device into blocks as shown in Figure 2. Figure 2 shows n blocks $B_1, B_2, \cdots, B_n$ with m words $W_1, W_2, \cdots, W_m$ per each block. The adversary $\mathcal{A}$ resides in at least one segment of the device memory and is capable of modifying any memory segment during the intervals between successive executions of the attestation routine on individual blocks.

It is assumed that the adversary remains persistently present on the device and is unable to remove itself entirely. Furthermore, the adversary is assumed to be aware of: (i) the initiation time of the attestation process and (ii) the duration

required to verify a single memory block. Using this information, the adversary can infer which portions of memory have not yet been subjected to attestation.

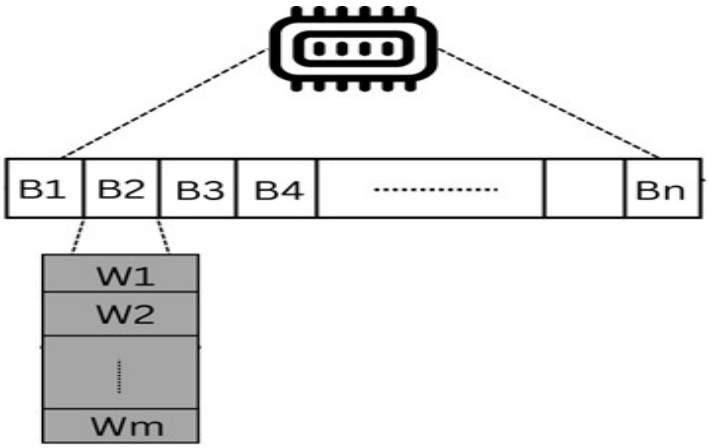


Figure 2 IoT device memory model

The standard Dolev Yao threat model is considered, i.e., $\mathcal{A}$ has the capability to inject, tamper, replay, and eavesdrop on the packets sent/received by the IoT devices. In addition, the adversary $\mathcal{A}$ may also obtain physical access to an IoT device and perform hardware-based attacks to extract sensitive information stored on the device. The adversary's capabilities are formalized using the following set of queries:

i. Send *S* (*S*, m0, r0, m1): This query represents a scenario in which the adversary impersonates a legitimate IoT device, transmits a message m0 to entity S, receives a response r0, and subsequently sends a follow-up message m1 to S.
ii. $\mathcal{A}$ acts like a legitimate IoT device and sends a message m0 to *S* and receives r0. $\mathcal{A}$ then replies to *S* with m1.
iii. Send *ID* (*ID*, m0, r0): This query captures the adversary acting as a trusted server, initiating communication by sending message m0 to an IoT node identified by *ID* and receiving the corresponding response r0.
iv. Monitor (*ID*, *S*): This query models the adversary's ability to passively intercept and continuously observe communications exchanged over the wireless channel between IoT device *ID* and *S*.
v. Drop ($\mathcal{A}$): This query reflects the adversary's capability to selectively discard packets transmitted between the IoT node and S, anabling denial-of-service attacks by disrupting message delivery and desynchronizing protocol execution.
vi. Compromised Border Router**:** This query considers the scenario where the border router itself is compromised by the adversary. In this case, the adversary could attempt to manipulate the aggregation of individual responses σ_i into the cumulative response Ψ. To defend against this, each IoT device computes a Message Authentication Code (MAC) over its response σ_i using a device-specific pre-shared key K_i (established during provisioning and shared with the verifier). The device transmits $\{\sigma_i, \text{MAC}(K_i, \sigma_i)\}$ to the border router. The verifier independently verifies each MAC upon receipt; any tampering or forgery by the border router is immediately detected. This countermeasure adds minimal overhead (32 bytes per device using HMAC-SHA256) and does not affect the core attestation protocol.
vii. Reveal (*ID*): This query represents the adversary physically compromising an IoT device *ID* in order to extract confidential data or secrets stored in its memory.

It is noted that while the adversary knows the attestation timing parameters (observable via local side-channels), it cannot meaningfully tamper with the challenge distribution from the border router, as the verifier-router channel is cryptographically secured. Any attempted modification would be detected, and the adversary's stealth objective would be compromised.

IV. PROPOSED TECHNIQUE

The overall functioning of the proposed technique is presented in Figure 3. A set of IoT device creating a swarm is connected to a border router. The verifier sends a cumulative challenge to the border router. This border router breaks up cumulative challenge into individual challenges and sends one individual challenge to each IoT device in the swarm. Each IoT device then uses its individual challenge to compute the individual response and sends it back to the border router. The border router then creates a cumulative response to send back to the verifier. The verifier can then use this cumulative response to attest the swarm of IoT devices. Next, the proposed protocol is described in detail.

Let us assume a set of *N* IoT devices, $\mathfrak{T} = [I_1, I_2, \cdots, I_N]$. It is assumed that the verifier and the border router have a pre-established cryptographically secure communication channel, this can be done using a pre-shared key between the two entities. The detailed steps of the proposed protocol demonstrated in Figure 4 are as mentioned below:

i. The verifier generates a arbitrary permutation of n numbers representing the number of blocks in each IoT device's memory, i.e.,

$$\rho = [\rho_1, \rho_2, \ldots\ldots\ldots \rho_n] \quad (1)$$

Prior to attestation, each IoT device's memory is partitioned into two logical regions:

a) Golden Firmware Region (G): Contains the executable firmware and attestation routine. This region is identical across all devices in the swarm and is fully attestable.

b) Device-Specific Region (D): Contains calibration data, unique identifiers (e.g., MAC addresses), configuration parameters, and other benign variations. This region is excluded from attestation to prevent false alarms.

The attestation protocol measures only blocks within the golden firmware region. The memory block count *n* in Algorithm 1 and Algorithm 2 refers exclusively to blocks within this golden region, not the entire physical memory.

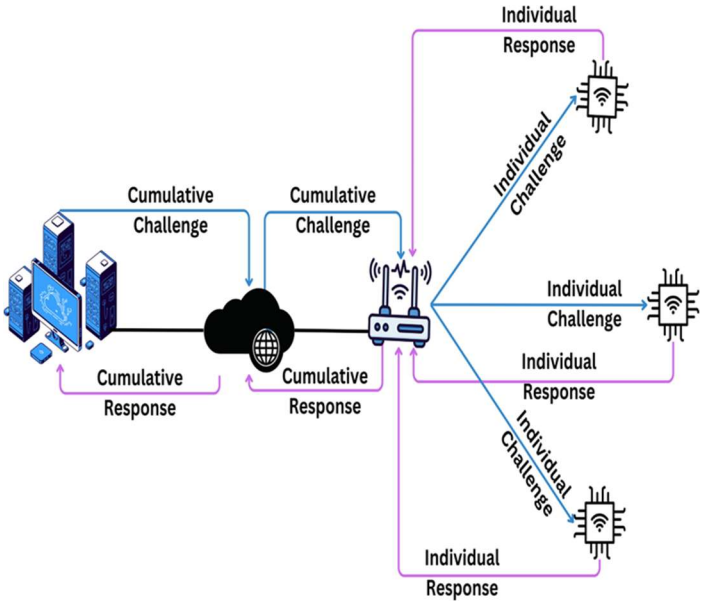


Figure 3 Overall operation of the proposed technique

Note that the number of blocks in memory should be equal to the total count of IoT-enabled nodes in the swarm i.e., $n = N$.

ii. The verifier then translates the random permutations into a set of challenges, i.e., $C = [\rho_1, \rho_2, \cdots, \rho]^T$

iii. The verifier then creates a random number S_b which is used as the seed for sampling individual bits within the memory. The verifier sends C and S_b to the border router.

iv. The border router breaks down the cumulative challenge C into individual challenges $C = [c_1\, c_2 \cdots c_n]$ and the verifier sends an individual challenge to each IoT device along with S_b.

v. Each IoT device after receiving the individual challenge c_i and S_b uses Algorithm 1 to compute the individual response σ_1 as follows:

a) The IoT device uses its PUF to generate the response R_i.

b) The IoT device calls the atomic function AttIoT with R_i as the input. This routine cannot be interrupted in the middle of measuring a single memory block. The IoT device derives a seed by computing $R_i \oplus \rho_i$ which is then used to randomly select β bits position for each element of the allocated storage block. Specifically for every one of the m words in a block, the attestation routine randomly chooses β bits. To accomplish this, the device generates a sequence of random indices Ω, drawn uniformly from the range 1 to ω, where ω denotes the number of bits in a word. F Normally, Ω consists of $m\beta$ values sampled from U [1, ω] using every allocated storage block. Based on these indices, a total of mβ bits are extracted from the memory block $\boldsymbol{G}_{\rho i}$ and the concatenation of these bits forms the response σ_i.

c) The IoT device responds to the border router by sending back the computed σ_i.

In Step v(c), each IoT device additionally computes a Message Authentication Code (MAC) over its individual response σ_i using a pre-shared key K_i. The device transmits $\{\sigma_i, \text{MAC}(K_i, \sigma_i)\}$ to the border router. The border router aggregates the responses and forwards $\{\Psi, \{\text{MAC}_i\}\}$ to the verifier. The verifier then verifies each MAC using the stored keys. This ensures that even if the border router is compromised, it cannot manipulate individual responses without detection.

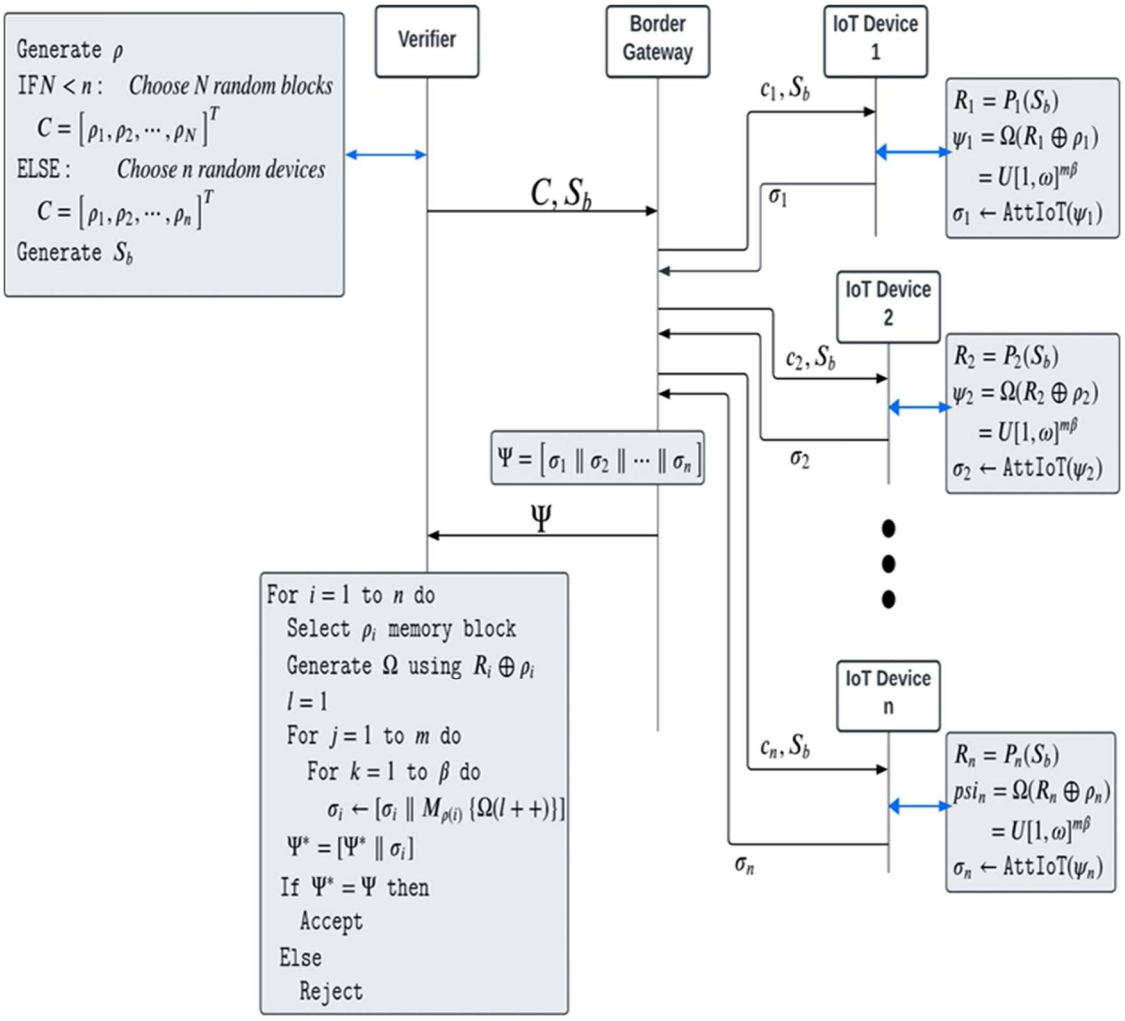


Figure 4 Proposed Attestation Protocol

vi. After the border router receives the response σi from each IoT device, it concatenates them to form the cumulative response $\Psi = [\sigma_1\ \sigma_2 \quad \cdots \quad \sigma_N]^T$.

vii. The border router sends the cumulative response Ψ to the verifier.

viii. Verifier operates the retained replica of the IoT device's storage and stored PUF responses (for each IoT device) to compute the cumulative response Ψ^* using Algorithm 2.

ix. The verifier compares the locally computed cumulative response Ψ^* with the cumulative response received from the IoT swarm Ψ. If both response match, the attestation is accepted. Otherwise, the attestation is rejected, and the swarm is considered malicious.

The proposed attestation technique should be run multiple times by the verifier to enhance the detection probability as discussed in Section 5. In the event of a transient device failure (e.g., packet loss or temporary power glitch), the border router implements a timeout-and-retry mechanism for the non-responding device. After a configurable number of failed retries, the device is temporarily quarantined and excluded from the current attestation round, allowing verification of the remaining swarm to proceed. Corrupted responses, however, are treated as potential compromises and trigger an immediate security alert. This ensures practical fault-tolerance without compromising the security guarantees of the protocol.

Algorithm 1 Computing individual checksum at an IoT device.

Require: $[\, c_i \quad S_b]^T$

Ensure: σ_i

$R_1 \leftarrow P_i(c_i)$ ▷ Generate PUF response

Function AttIoT(R_1)

$\psi_i \leftarrow \Omega(R_i \oplus \rho_i)$ ▷ Generate $U[1, \omega]^{m\beta}$ using ψ_i as the seed

$ind \leftarrow 1$

while $j < m$ **do while** $k < \beta$ **do**

$\sigma_i \leftarrow [\sigma_i \,||\, \boldsymbol{G}_{\boldsymbol{\rho(i)}}\{\Omega(ind++)\}]$

end while

end while End Function

Return σ_i

Algorithm 2 Computing cumulative response at the verifier.

Require: S, ρ, S_b, list of CRPs

Ensure: Ψ^*

$i \leftarrow 1$ ▷ Initialize the IoT device counter

textttRead ρ_i

$\Psi_i = \Omega(R_i \oplus S_b)$ $l \leftarrow 1$

while $j < m$ **do while** $k < \beta$ **do**

$\sigma_i \leftarrow [\sigma_i \,||\, \boldsymbol{G}_{\boldsymbol{\rho(i)}}\{\Omega(ind++)\}]$

end while

end while

$\Psi^* \leftarrow [\Psi^* \,||\, \sigma_i]$

Return Ψ^*

V. SECURITY ANALYSIS

The security analysis presented in this section assumes the adversary operates within the Dolev-Yao query model defined in Section 3, including the capabilities represented by Send, Reveal, Monitor, and Drop. A structured security analysis of the suggested verification method was carried out in this sectional part. As mentioned in the Section 3, the adversary $\boldsymbol{\mathcal{A}}$ occupies a single block of memory. Although $\boldsymbol{\mathcal{A}}$ can interrupt the execution of the IoT device, it cannot do so while the AttIoT routine is running. Therefore, the adversary cannot modify anything while a single block of memory is being measured.

As the adversary's aim is to evade detection while being in memory, It needs to relocate and reinstate the block's contents where it is stored before any measurement occurs. Moreover, it is assumed that $\boldsymbol{\mathcal{A}}$ knows the memory size M yet to be measured. Note that continuous presence of $\boldsymbol{\mathcal{A}}$ in a single block lead to eventual detection. Therefore, $\boldsymbol{\mathcal{A}}$ needs to change its location at least once.

Lemma 1. An adversary cannot predict a PUF's response.

Proof: A PUF with a challenge and response having lengths l_C and l_R, respectively, can be modelled as $\{0, 1\}^{l_C} \rightarrow \{0, 1\}^{l_R}$. Representing a cryptographic contest between a challenger $\boldsymbol{\mathcal{C}}$ and adversary $\boldsymbol{\mathcal{A}}$ using $\mathrm{Exp}^{Sec}_{PUF,\mathcal{A}}$in the following manner:

(i) The adversary $\boldsymbol{\mathcal{A}}$ first submits a randomly selected challenge C^i to the challenger $\boldsymbol{\mathcal{C}}$. The Challenger evaluates this challenge using the PUF and returns the corresponding response R^i to $\boldsymbol{\mathcal{A}}$.

(ii) The challenger $\boldsymbol{\mathcal{C}}$ independently selects a fresh challenge C^x which has not been queried previously, and computes its response using the PUF, i.e., $R^x = PUC\ (C^x)$.

The adversary $\boldsymbol{\mathcal{A}}$ is then permitted to interact with the PUF a polynomial number of times using challenges other than C^x. After completing these interactions, $\boldsymbol{\mathcal{A}}$ outputs a guessed response R^x for the challenge C^x. the adversary $\boldsymbol{\mathcal{A}}$ is considered successful if the guessed respons matches the actual response, i.e. $R^{x'} = R^x$.

The success probability of the adversary $\boldsymbol{\mathcal{A}}$ here is defined as $\alpha^{PUF}_{\mathcal{A}} = \Pr\,[R^{x'}=R^x]$. Since each PUF produces a unique and

unclonable response, the adversary has no better strategy than random guessing when attempting to predict the response to an unseen challenge. As a result, the adversary's advantage is bounded by $\alpha_{\mathcal{A}}^{PUF} = \frac{1}{2^{lR}}$, where lR denotes the length of the PUF response.

Lemma 2. The IoT node does not retain any confidential information in non-volatile or accessible memory. As a result, an adversary is unable to extract the secret response R^i through the Reveal oracle.

Proof: IoT node does not retain any confidential data in its memory. Consequently, even if the adversary invokes the *Reveal* oracle, it cannot recover the secret response R^i. Moreover, under the system-on-chip (SoC) assumption, the challenge C^i alone does not enable the adversary to derive R^i, since access to the internal PUF evaluation process is not possible.

Lemma 3. The probability of detection for an adversary in each iteration of the suggested validation method at any IoT node is given by:

$$\mathrm{P}_\zeta = \frac{1}{n}\left[1 - \left(\frac{1}{2}\right)^{m\beta}\right] \qquad (2)$$

Proof: Consider n number of blocks in memory, each IoT device is asked to attest one of the randomly selected blocks. Therefore, if a memory block selected for attestation is denoted in time interval i with M_{Δ_i} & the occurrence of being trapped by ζ, then the detecting probability of $\boldsymbol{\mathcal{A}}$ in any iteration of suggested validating technique is given by:

$$\mathrm{P}_\zeta = \Pr[M_{\Delta i}]\Pr[\zeta \mid M_{\Delta i}] \qquad (3)$$

$$\mathrm{P}_\zeta = \frac{1}{n}\left(\left(1 - \frac{1}{2}\right)^{m\beta}\right) \qquad (4)$$

The detection probability analysis assumes that the adversary resides within the golden firmware region G. Since the device-specific region D is excluded from attestation, the adversary cannot evade detection by hiding in D. Any compromise of the golden region will be detected with the probability stated in Theorem 4.

Theorem 4. The probability of detection can be improved by running the proposed attestation technique multiple times.

Proof: If an adversary is not caught during the first $i-1$ iterations of the suggested procedure, then the evading detection probability at the i-th iteration is given by:

$$\mathrm{P}_d = (1-\mathrm{P}_\zeta)^{i-1} \times \left(1-\mathrm{P}_\zeta\right) = (1-\mathrm{P}_\zeta)^i \qquad (5)$$

$$= \left[1 - \frac{1}{n}\left(1 - \left(\frac{1}{2}\right)^{m\beta}\right)\right]^i \qquad (6)$$

Note that $\left(1 - \left(\frac{1}{2}\right)^{m\beta}\right) \approx 1$, if $m\beta \geq 8$. Therefore:

$$\mathrm{P}_\zeta = \left(1 - \frac{1}{n}\right)^i \qquad (7)$$

A critical observation regarding Equation (7) is that it completely removes the dependence on m (number of words per block) and β (number of bits sampled per word). This is a deliberate approximation valid for mβ ≥ 8, where (1/2)^(mβ) ≤ 1/256 ≈ 0.0039, making the term negligible. For mβ < 8, the exact Equation (6) must be used. This approximation reveals an important practical insight: once mβ exceeds 8, increasing the bit sampling depth yields diminishing returns in detection probability. Thus, practitioners can choose mβ = 8 as a cost-effective operating point that balances security and efficiency. Figure 5 illustrates this saturation behaviour, where curves for m = 4, 8, and 16 (all with mβ ≥ 4 for β = 1) overlap almost perfectly, while only m = 1 (where mβ = 1 < 8) shows a distinctly lower detection probability.

Figure 5 shows the Adversary detection rate probability for the given the number of iterations for the suggested technique. It is observed that P_d gets close to 1 after $i \geq n$, i.e., the count for iterations of the suggested attestation method should be at least equal to the number of blocks in memory.

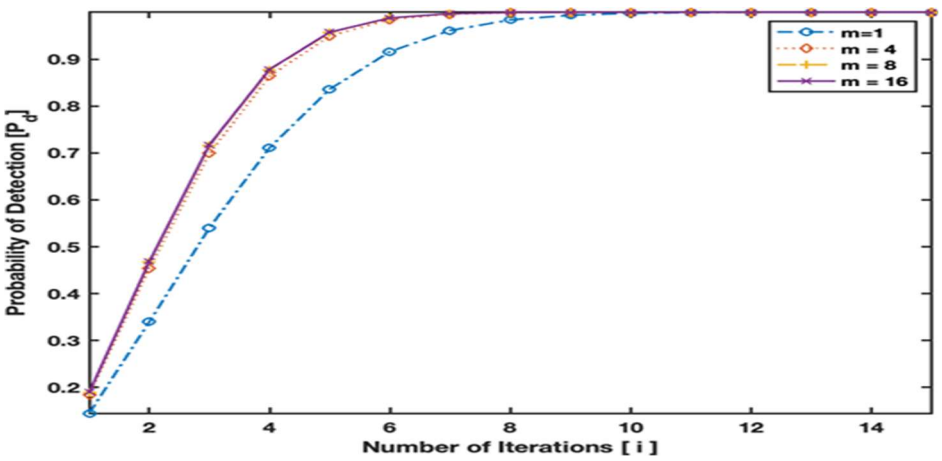


Figure 5: Detection probability per iteration (P_ζ) versus number of iterations (i) for $n = 10$, $\beta = 1$, and variable m (1, 4, 8, 16). Due to Equation (4), for $m\beta \geq 4$ the term $(1/2)^{m\beta}$ becomes negligible (≤ 0.0625), making $P_\zeta \approx 1/n = 0.1$. Consequently, the curves for $m = 4$, 8, and 16 overlap almost perfectly and are graphically indistinguishable. Only $m = 1$ shows a distinctly lower detection probability because $(1/2)^1 = 0.5$ is not negligible. Thus, increasing m beyond 4 yields no meaningful improvement in single-iteration detection probability.

Lemma 4. A compromised border router cannot manipulate the cumulative response Ψ without detection.

Proof: Each IoT device computes MAC (K_i, σ_i) and transmits it alongside σ_i. The border router does not possess K_i. Any modification to σ_i by the border router would require the adversary to forge a valid MAC without knowledge of K_i, which is computationally infeasible under standard cryptographic assumptions. The verifier detects the tampering upon MAC verification failure.

VI. EXPERIMENTAL VALIDATION

To assess the effectiveness of the suggested attestation technique, experimental tests were performed using a Raspberry Pi 3 (Rpi3) platform interfaced with external sensors through a serial communication link. The Rpi3 is connected as the IoT prover and a laptop as the verifier. The Rpi3 communicated with Arduino Duo sensors with measurements collected at a sampling interval of one reading every 60 seconds. A memory region of 256K was allotted for the attestation process. The Rpi3s is linked to the verifier (a laptop in this case) through a wireless connection.

A. Statistical Analysis and Assumptions

Latency is the amount of time it takes for the attesting device to complete one attestation round. A safety feature that makes it hard for an IoT device to work normally is called" low availability." Because an IoT device might do things that need to be done in a timely manner, the attestation methods designed for them should ensure high availability, and low latency is one way that this can be achieved. Table 1 illustrates the latency associated with the suggested attestation method. For comparison purposes, the latency of latest available techniques is hereby presented such as in SWARNA (Kumar et al., 2022), FeSA (Kuang et al., 2023), and JANUS (Zhang et al., 2024). The latency values presented here are for a single run of each technique such that the $P_d \geq 95\%$. The memory is partitioned into 32 blocks, with each block consisting of 1024 words of 64 bits.

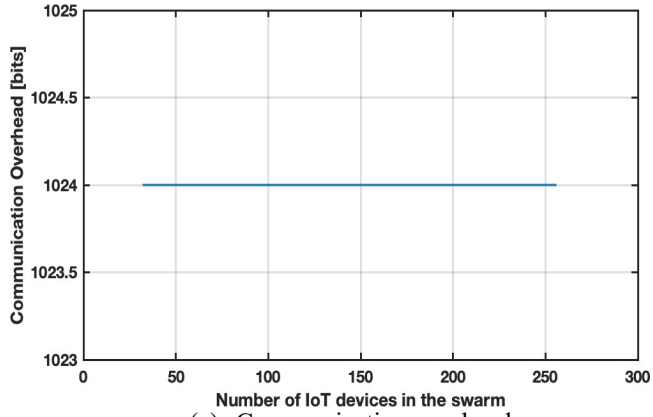
Table 1: End-to-end latency of the proposed attestation technique

Technique	Latency (sec)
(Kumar et al., 2022)	1.5
(Kuang et al., 2023)	1.0
(Zhang et al., 2024)	0.5
Proposed Technique	0.046

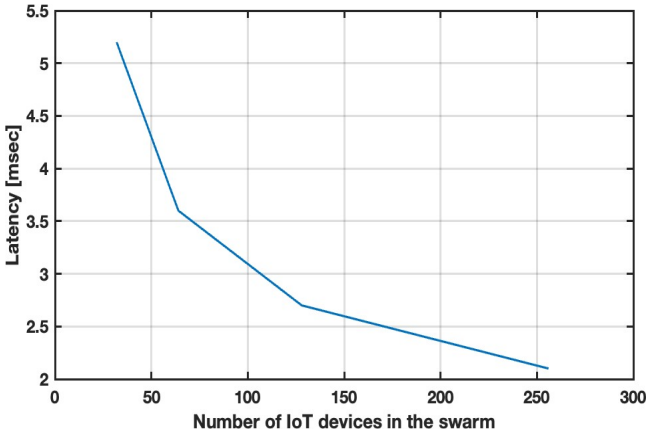
It is observed that the proposed technique leads to significantly lower latency value thanks to the parallelism of running the proposed attestation technique, i.e., in each iteration of the proposed technique one block of the memory is measured by each device.

B. Scalability

To evaluate the scaling capability of the introduced approach, observations show that communication network overhead is the key restricting factor when it comes to an increasing number of IoT devices. For each iteration of the proposed attestation technique, each IoT device needs to send mβ bits and from Theorem 4, $m\beta \geq 8$. If the memory size to 256 - Kbytes and $\beta = 1$ is fixed, the communication overhead and latency is shown in Figure 6 for n = 32, 64, 128, 256.



(a) Communication overhead.



(b) Latency.

Figure 6 Evaluation of scalability of the proposed attestation technique. (a) Total communication overhead at the border router (N × mβ bits), which scales linearly with swarm size N. Per-device overhead remains constant at mβ bits. (b) End-to-end latency, which decreases with N due to increased parallelism.

From Figure 6(a), it is observed that the total communication overhead at the border router increases linearly with the number of IoT devices N, as expected, since the router aggregates N individual responses of mβ bits each. However, the per-device communication overhead remains constant at mβ bits per device, independent of swarm size. This is because each device transmits only its individually computed response, regardless of how many other devices are in the swarm. Thus, the proposed attestation technique scales efficiently: while the border router experiences linear growth in total overhead, each individual device experiences no additional communication burden as the swarm expands. Figure 6(b) shows that the latency of the proposed technique reduces as the quantity of IoT nodes with respect to swarm $N = n$ grows. This is thanks to the increased level of parallelism as the number of IoT devices is increased. Communication efficiency of the suggested attestation approach not only remains constant, but the latency reduces with the number of IoT nodes, speeding up the overall verification procedure.

In the prototype implementation, the 256K memory region allocated for attestation was restricted to the golden firmware region (executable code). Device-specific data, including calibration constants and unique identifiers, were stored in a separate memory region that was excluded from attestation. This ensured that benign inter-device variations did not produce false positives during verification.

Also, with m = 1024, even β = 1 gives mβ = 1024 >> 8, placing the protocol well into the saturated detection regime as established in Section 5. Thus, the choice of β = 1 minimizes communication overhead without compromising detection probability. Increasing β would linearly increase overhead with no measurable security benefit.

For fair comparison, all baseline techniques (SWARNA, FeSA, JANUS) were re-implemented from their published specifications and evaluated on the identical Rpi3 hardware platform, using the same 256KB memory allocation and configured to achieve P_d ≥ 95%. The latency values presented are from our own measurements, not cited from literature.

B. Energy Consumption Analysis

For resource-constrained IoT devices, energy efficiency is often more critical than latency. An approximate energy estimates per device per attestation round using established literature values is provided.

The total energy per device is modelled as given in Equation (8).

$$E_{total} = E_{PUF} + E_{memory} + E_{tx} + E_{rx} \quad (8)$$

with E_{PUF} being the energy required for PUF generation, E_{memory}, the memory sampling energy, E_{tx}, the transmission energy and E_{rx} the receiver energy.

A ring oscillator PUF consumes approximately 5.16 μW per challenge-response pair. At ~10 μs evaluation time, this is 0.052 μJ per device. For the E_{memory}, each device reads mβ = 1024 bits from SRAM. At 32.8 fJ/bit, this is <0.001 μJ — negligible. For the communication related energy ($E_{tx} + E_{rx}$), each device transmits 128 bytes (1024 bits) per round. At typical 2.4 GHz radio parameters (0 dBm, 250 kbps), the transmission energy is ~1.2 μJ per byte, giving approximately 154 μJ per device. Therefore, the total energy is approximately 154 μJ per device per attestation round.

For comparison, SWARNA requires full 256 KB memory hashing (~2.6 mJ) which is 17× higher than our technique. FeSA employs public-key operations (~500 μJ) translating to about 3× higher. JANUS uses TPM-based signatures (~100 mJ) and is about 650× higher. Thus, the proposed technique is significantly more energy-efficient while maintaining the same detection probability threshold of ≥ 95%.

VII. CONCLUSION

This study introduces a remote attestation approach aimed at a swarm of IoT nodes. The developed technique divides the memory into blocks and assigns each IoT device in the swarm one block for attestation. Every IoT node is loaded using PUF & the attestation technique embeds the PUF response for each IoT device into the attestation routine of each IoT device. A formal security investigation showed that the projected

attestation technique achieves up to 95% accuracy in detecting roving malware. Experimental justification on real hardware showed that the proposed technique has pointedly low verification latency as compared to existing techniques. Moreover, the proposed attestation technique is not affected by the quantity of IoT nodes, in fact the latency reduces with the increase in the quantity of IoT nodes. Thanks for parallelism inherent in the proposed technique. This demonstrates that the proposed verification method provides no scalability issues.

REFERENCES

Ahmadi, S., Le-Papin, J., Chen, L., Dongol, B., Radomirovic**, S., & Treharne, H. (2024).** *On the Design and Security of Collective Remote Attestation Protocols*. http://arxiv.org/abs/2407.09203.

Akhtar, M. H., Jabeen, T., Aziz, R., Amin, M. N., Rizwan, S. M., & Hamid, K. (2024). *Intelligence based Self-Healing Network Design: An Automated Incident Response System for Troubleshooting of IoT Security Breaches*. Retrieved: http://amresearchreview.com/index.php/Journal/about

Alam, I., Samiullah, M., Asaduzzaman, S. M., Kabir, U., Aahad, A. M., & Woo, S. S. (2025). MIRACLE: Malware image recognition and classification by layered extraction. *Data Mining and Knowledge Discovery*, *39*(1). https://doi.org/10.1007/s10618-024-01078-z

Aliaj, E., De, I., Nunes, O., De Oliveira Nunes, I., & Tsudik, G. (2022). *GAROTA: Generalized Active Root-Of-Trust Architecture (for Tiny Embedded Devices)*. Retrieved https://www.usenix.org/conference/usenixsecurity22/presentation/aliaj

Ammar, M., Caulfield, A., De, I., & Nunes, O. (2025). *SoK: Integrity, Attestation, and Auditing of Program Execution*.

Athar, S. O., Aman, M. N., & Sikdar, B. (2025). DuAtt: A Dual-Layer Attestation Scheme for PLC-Based Industrial Internet of Things. *IEEE Internet of Things Journal*, *12*(20). https://doi.org/10.1109/JIOT.2025.3589940

Baig, M. A., Iqbal, A., Aman, M. N., & Sikdar, B. (2025). Leveraging AI to Compromise IoT Device Privacy by Exploiting Hardware Imperfections. *IEEE Transactions on Artificial Intelligence*, *6*(6). https://doi.org/10.1109/TAI.2025.3526139

Bhole, M., Kastner, W., & Sauter, T. (2024). *From Manual to Semi-Automated Safety and Security Requirements Engineering: Ensuring Compliance in Industry 4.0*. https://standards.ieee.org/

Dushku, E., Rabbani, M. M., Vliegen, J., Braeken, A., & Mentens, N. (2023). PROVE: Provable remote attestation for public verifiability. *Journal of Information Security and Applications*, *75*. https://doi.org/10.1016/j.jisa.2023.103448

Ferro, L., Bravi, E., Sisinni, S., & Lioy, A. (2024). SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments. *SaT-CPS 2024 - Proceedings of the 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, 41–50. https://doi.org/10.1145/3643650.3658609

Kassem, N. El, Hellemans, W., Siachos, I., Dushku, E., Vasileiadis, S., Karas, D. S., Chen, L., Patsakis, C., & Giannetsos, T. (2025). PRIVE: Towards Privacy-Preserving Swarm Attestation. *Proceedings of the International Conference on Security and Cryptography*, *1*, 247–262. https://doi.org/10.5220/0013629000003979

Khan, M. A., Aman, M. N., & Sikdar, B. (2024). Soteria: A Quantum-Based Device Attestation Technique for Internet of Things. *IEEE Internet of Things Journal*, *11*(9). https://doi.org/10.1109/JIOT.2023.3346397

Khan, M. A., Shalu, Naveed, Q. N., Lasisi, A., Kaushik, S., & Kumar, S. (2024). A Multi-Layered Assessment System for Trustworthiness Enhancement and Reliability for Industrial Wireless Sensor Networks. *Wireless Personal Communications*, *137*(4). https://doi.org/10.1007/s11277-024-11391-x

Khan, S., Martins, P. A. F. L., Sousa, B., & Pereira, V. (2025). A Comprehensive Review on Lightweight Cryptographic Mechanisms for Industrial Internet of Things Systems. In *ACM Computing Surveys* (Vol. 58, Number 1). Association for Computing Machinery. https://doi.org/10.1145/3757734

Kibret, S. W. (2022). Property-based attestation in device swarms: A machine learning approach. In *Machine Learning for Cyber Security*. https://doi.org/10.1515/9783110766745-004

Kuang, B., Fu, A., Gao, Y., Zhang, Y., Zhou, J., & Deng, R. H. (2023). FeSA: Automatic Federated Swarm Attestation on Dynamic Large-Scale IoT Devices. *IEEE Transactions on Dependable and Secure Computing*, *20*(4), 2954–2969. https://doi.org/10.1109/TDSC.2022.3193106

Kumar, S., Eugster, P., & Santini, S. (2022). Software-Based Remote Network Attestation. *IEEE Transactions on Dependable and Secure Computing*, *19*(5). https://doi.org/10.1109/TDSC.2021.3077993

Kwon, H. (2025). Secure and Scalable Device Attestation Protocol with Aggregate Signature. *Symmetry*, *17*(5). https://doi.org/10.3390/sym17050698

Mehjabin, S. S., & Younis, M. (2025). PAMA: PUF-based Aggregated Multi-hop Attestation Protocol for IoT. *IEEE International Conference on Communications*. https://doi.org/10.1109/ICC52391.2025.11161539

Orman, A. (2025). Cyberattack Detection Systems in Industrial Internet of Things (IIoT) Networks in Big Data Environments. *Applied Sciences (Switzerland)*, *15*(6). https://doi.org/10.3390/app15063121

Song, H., Yuan, Y., Wang, Y., Yang, J., Luo, H., & Li, S. (2024). A Security Posture Assessment of Industrial Control Systems Based on Evidential Reasoning and Belief Rule Base. *Sensors*, *24*(22). https://doi.org/10.3390/s24227135

Usman, A. B., & Asplund, M. (2024). Remote Attestation with Software Updates in Embedded Systems. *2024 IEEE Conference on Communications and Network Security, CNS 2024*. https://doi.org/10.1109/CNS62487.2024.10735526

Vuseghesa, F. K., Messai, M. L., & Bentayeb, F. (2025). SHIELD: Self-Healing IoT Networks with Automated Response and AI-Driven Detection of Node Compromising

Attacks. *21st International Wireless Communications and Mobile Computing Conference, IWCMC 2025*. https://doi.org/10.1109/IWCMC65282.2025.11059566

Wu, Y., Wang, J., Wang, Y., Zhai, S., Li, Z., He, Y., Sun, K., Li, Q., & Zhang, N. (2024). *Your Firmware Has Arrived: A Study of Firmware Update Vulnerabilities*. Retrieved https://www.usenix.org/conference/usenixsecurity24/presentation/wu-yuhao

Zhang, X., Qin, K., Qu, S., Wang, T., Zhang, C., & Gu, D. (2024). *Teamwork Makes TEE Work: Open and Resilient Remote Attestation on Decentralized Trust*. http://arxiv.org/abs/2402.08908