Detecting Compromised IoT Devices: Existing Techniques, Challenges, and A Way Forward

Imran Makhdoom[1], Mehran Abolhasan[2], Daniel Franklin[3], Justin Lipman[4], Christian Zimmermann[5], Massimo Piccardi[6], Negin Shariati Moghadam[7]

[1,2,3,4,6,7] *University of Technology Sydney, Ultimo NSW 2007, Australia*
[1] *Food Agility Cooperative Research Centre Ltd, Sydney NSW 2000, Australia*
[5] *Robert Bosch GmbH, 70049 Stuttgart, Germany*
Imran.Makhdoom@uts.edu.au[1], Mehran.Abolhasan@uts.edu.au[2], Daniel.Franklin@uts.edu.au[3], Justin.Lipman@uts.edu.au[4], Christian.Zimmermann3@de.bosch.com[5],Massimo.Piccardi@uts.edu.au[6],Negin.Shariati@uts.edu.au[7]

Abstract

IoT devices, whether connected to the Internet or operating in a private network, are vulnerable to cyber attacks from external or internal attackers or insiders who may succeed in physically compromising an IoT device. Once compromised, the IoT device can join a botnet to participate in large-scale distributed attacks (potentially recruiting additional nodes), exfiltrating confidential data or injecting false data into critical data sets, corrupting subsequent data analytics. Although various device attestation techniques are available to detect malicious IoT devices, these methods do not fully address all aspects of a potentially compromised node. This study explores current state-of-the-art approaches for detecting a malicious/compromised node in the network, highlights related challenges, and proposes a way forward for developing secure and economical attestation protocols.

Keywords: Internet of Things, IoT threats, IoT security, device integrity, device attestation, code integrity, memory attestation.

1. Introduction

Internet of Things (IoT) devices are employed in numerous critical infrastructures to enable remote sensing and actuation operations. The integrity of IoT devices and their data plays a crucial role in environments where user/sensor data is collected, processed, and analyzed to support data-driven objectives. There are numerous ways in which an attacker can compromise the integrity of IoT devices and data. An attacker may introduce a malicious device within the network or physically compromise and modify an existing device to inject false sensor data or launch attacks on connected network devices. A remote attacker may resort to Man-in-the-Middle (MITM) attacks and modify the IoT devices' data [1]. Moreover, due to faulty equipment/devices or degraded communication channels, "bad data" is introduced into the system. The erroneous measurements are usually detected based on a statistical analysis [2]. However, bad data detection techniques aim at detecting only data which is "bad" due to non-malicious errors [3]. However, these detection mechanisms can be bypassed by an attacker by causing correlated errors into the measurements [4].

Bad data detection approaches functioning without any mathematical model to regulate the measurements are not effective for IoT systems. In addition, bad data detection schemes are equally ineffective if uncorrelated measurements from multiple devices are processed, precluding their applicability to a wide range of IoT systems. Therefore, researchers have started focusing on specialized IoT data integrity verification techniques to protect against MITM attacks that are difficult to detect.

Security researchers in [5] present a novel approach to verify the integrity of healthcare data generated by IoT devices using blockchain technology. The proposed framework employs a blockchain-based

logging contract that allows for optimal witnessing of the data while preserving the privacy and confidentiality of the patients. The IoT data is verified by the witness nodes co-located with the node whose data is being verified. The witness nodes offer their services duly quoting the verification cost (including the service cost and the reward for the miners) and the time limit (deadline) by which they are available to submit their statement to the blockchain. The witness statements are processed in the form of Bloom Filters. Moreover, the witness nodes are selected based on their service cost and the error probability. The data logging process involves the logging contract verifying the authenticity of the data and then storing it in a tamper-evident and immutable manner on the blockchain. The logging contract also generates cryptographic proofs that can be used to verify the integrity of the data at any point in the future.

Nonetheless, the integrity of IoT devices encompasses numerous aspects, such as behavioral integrity, the authenticity of the device hardware, the validity of the software components/device configuration and the accuracy, completeness, and validity of data. Hence, depending upon the type of attack, an adversary may physically compromise an IoT device to alter its functionality or launch MITM attacks to modify IoT data. Accordingly, there are different techniques to attest the IoT device hardware and software components and verify data integrity. Therefore, considering both areas' broad scope, this study covers only the IoT device attestation techniques that aim to detect a malicious device in the network.

1.1. Threat Model

The lack of physical security makes an IoT device intrinsically vulnerable to physical compromise in an unprotected environment such as remote or vast agricultural land, livestock monitoring, or dam/river monitoring systems. According to security expert Bruce Schneier [6], "You can't defend. You can't prevent. The only thing you can do is detect and respond." This quote is substantiated by the researchers' findings in [7], as they identified many IoT devices accessible via public internet operating with outdated firmware. Similarly, [8] highlights that most commercially available IoT products are built with inadequate, incomplete, or ill-designed security mechanisms. Hence, malicious alteration of an IoT device's hardware/software modules and network/device configuration may adversely affect its behavior.

We can divide the most common attacks on IoT devices into three categories, i.e., software attacks, runtime attacks, and physical attacks [9]. In software attacks, the attacker may replace the trusted application with malicious code or inject and execute unauthorized code snippets to alter the device's behavior. On the other hand, the runtime attacks cause device misbehavior by manipulating code pointers, data variables or loop counters. The runtime attacks can be executed without injecting malicious code. Besides, in semi-intrusive and invasive physical attacks the attacker needs physical access to the device. Nonetheless, it is essential to have the ability to detect maliciously compromised IoT devices in the network and take requisite remedial measures.

An attacker's objective is to install malicious code in the executable memory of the IoT device and pass the attestation protocol without being detected. The malicious code can be installed remotely by exploiting vulnerabilities in the device software [10], non-invasive hardware attacks [11], or through a JTAG programming adapter. Correspondingly, numerous approaches are being practiced today to detect malicious IoT devices or to protect trusted applications from unauthorized alterations. These approaches may include software-based attestation techniques, hardware-based device security, and some hybrid strategies. However, no known mechanism currently provides a fair balance of security and performance efficiency. For example, hardware-based solutions are considered complex and costly, while software-based techniques are either energy-intensive, have high communication complexity, or are limited to only program memory. In this work, we present an in-depth survey of IoT device attestation techniques, identify future challenges, and suggest a way forward.

1.2. Related Work

Significant literature is available that illustrates numerous individual device and code attestation techniques. However, no existing work comprehensively discusses all aspects of attestation related to IoT devices in a single research article. For instance, [12] presents a detailed study on collective remote attestation schemes and related challenges, primarily focusing on hybrid hardware/software-based techniques. A range of strictly hardware-based approaches for remote attestation is reviewed in [13], while [14] surveys software-based strategies and open issues for IoT. Researchers consider the limitations of existing software-based remote attestation techniques in [15]. Existing remote attestation principles are examined in [16], and their functionality is compared to that of current Trusted Execution Environments (TEEs).

Researchers in [17] present a security framework that formally captures security goals, attacker models, and various system and design parameters related to software attestation techniques. Correspondingly, numerous remote attestation schemes for Wireless Sensor Networks (WSNs) are examined in [18], primarily focusing on software-based attestation schemes. In another study [9], researchers provide a comprehensive review of remote-attestation approaches only. The remote attestation approaches are devised to protect against a proposed universal adversarial model. The adversarial model, helps categorize the remote attestation techniques into five classes, i.e., attestation for software attacks, attestation for runtime attacks, attestation for physical attacks, attestation against Denial-of-Service (DoS) (verifier impersonation) attacks, and attestation against malware attacks. The authors analyse different remote attestation strategies as a defense mechanism against various attacks and also propose guidelines for developing efficient remote attestation techniques against different attacks.

Most existing studies have a narrow focus and mostly cover remote attestation approaches. In comparison, there is a need to address a broader research area, i.e., detecting a malicious or compromised node in the network, identifying related challenges and recommending guidelines for developing secure and efficient device attestation protocol.

1.3. Contributions of the Paper

To the best of our knowledge, this study is the first of its kind, encompassing most of the existing SOTA techniques to detect malicious/compromised nodes in an IoT network. The main contributions of this research include:

- Categories IoT device validation techniques.
- Provides a comprehensive survey of passive (behavioral integrity validation) and active device attestation approaches.
- Proffers a detailed gap analysis highlighting the strengths and weaknesses of each attestation technique.
- Identifies current challenges.
- Proposes a way forward for developing a unified, secure, economical, and reliable IoT device attestation protocol.

1.4. Organization of the Paper

The rest of the paper is organized as follows: Section 2 introduces the categorization of device attestation approaches and reviews the behavioral integrity verification techniques. The active attestation schemes are illustrated in Section 3. Section 4 highlights the advantages and disadvantages of each of the existing schemes and presents a detailed analysis. Section 5 identifies several future challenges, while Section 6 suggests a possible way forward. Finally, the paper is concluded in Section 7.

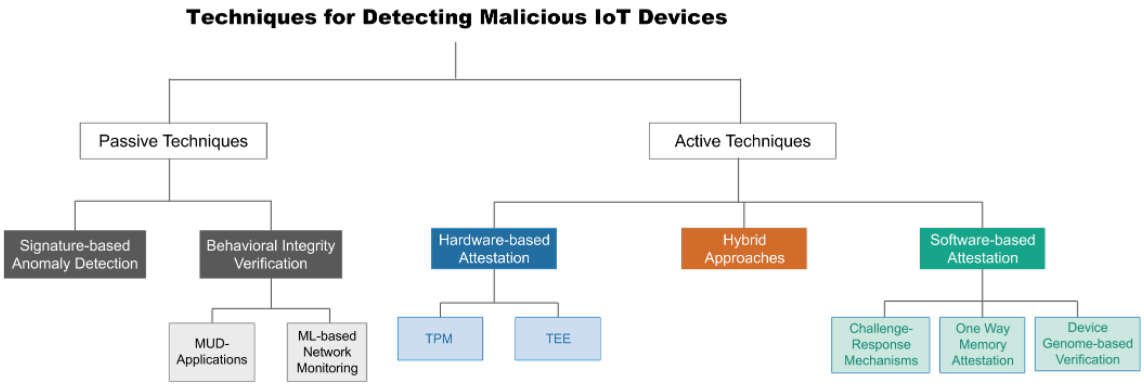


Figure 1: Categorization of IoT device validation techniques

2. State of the Art (SOTA) Techniques

2.1. Categories of IoT Device Attestation Approaches

IoT device validation techniques can be broadly divided into two main categories, i.e., Passive and active approaches. As shown in Figure 1, the passive strategies primarily include signature-based and behavior-based schemes. The signature-based techniques monitor network traffic and detect anomalies based on attack signatures. At the same time, the behavior-based methods monitor the behavior of an IoT device against a default configuration or prescribed allowable actions without necessarily challenging/probing individual devices. One of the approaches to passively detect malicious IoT devices is misbehavior verification techniques based on Manufacturer Usage Description (MUD). In addition to MUD-based methods, there are studies on developing machine learning (ML) based approaches to secure IoT networks by monitoring network traffic for any malicious action [19, 20, 21, 22]. The IoT network forensics and network security based on attack signatures and heuristic analysis is a wholesome area of research itself. Hence, signature-based and ML-based passive techniques are out of the scope of this study. On the other hand, MUD, relatively a new domain, is reviewed in this research.

The active techniques require the IoT devices to respond to certain challenges or compute and forward an integrity measurement of their memory, software components or configuration. These schemes include software-based attestation, hardware-based validation, and hybrid approaches incorporating software and hardware modules.

2.2. Behavioral Integrity Verification

A standard called MUD [23] has been approved by the Internet Engineering Task Force (IETF), which aims to formalize the expected network behavior of IoT devices for improved behavioral verification and integrity checks. This standard allows IoT device manufacturers to provide network administrators with descriptions of their devices' network behavior to enhance their security. The MUD specification includes device metadata that can be included in network traffic to define the device's intended network behavior. Network administrators can leverage this metadata to implement access control policies that permit the device's access to only the required services and block any other unauthorized or malicious traffic. Adopting MUD can reduce the attack surface of IoT devices, making it more challenging for attackers to exploit vulnerabilities or launch attacks. Moreover, MUD allows network administrators to apply granular access control policies tailored to each device's intended behavior, decreasing the risk of unauthorized access to the network. As shown in Table 1, considerable research has been done on the applications of MUD in securing IoT networks.

Authors in [24] suggest a technique to enhance the security of Industrial IoT (IIoT) devices through digital twins and software-defined networking. The proposed framework utilizes MUD metadata to create a behavioral profile of IoT devices. The digital twin of the network evaluates actions before they are used in the physical network. Updating the behavioral profiling system in real-time improves the system's overall security, and compliance with IoT deployment policies is ensured. The proposed methodology comprises three primary stages: (1) Device Profiling, (2) Rule Generation, and (3) Behavioral Profiling. MUD metadata is extracted from network traffic in the Device Profiling phase to determine the device's intended actions. Rules are generated based on the MUD metadata in the Rule Generation stage to detect and prevent malicious activity. The Behavioral Profiling stage continuously monitors the device's network activity and compares it to its expected behavior. Any deviations from desired behavior are flagged as potential security threats. The proposed approach provides an effective solution to secure IIoT devices and prevent possible attacks.

The proposed scheme has several advantages, including its ability to detect and prevent zero-day attacks, provide granular access control and monitoring, and flexibility in adapting to new IoT devices and network environments. Additionally, the proposed framework is designed to be compatible with Software-Defined Networking (SDN) architectures, which allows for centralized management and control of the network. However, there are some limitations and performance overheads. One limitation is that the framework's effectiveness depends on the accuracy and completeness of the MUD metadata, which may not always be available or up-to-date. Another area for improvement is that the proposed framework may require additional hardware resources and processing power to monitor and analyze the network traffic, which can result in performance overheads.

Similarly, [25] proposes a software-defined security-by-contract framework for blockchain-enabled MUD-aware IIoT edge networks. The proposed methodology utilizes SDN to create and enforce security policies based on MUD specifications, which outline the intended behavior of IoT devices in a network. The security-by-contract approach enables the enforcement of security policies. It ensures device compliance with those policies, while the blockchain provides a secure and decentralized platform for managing those policies. The suggested methodology comprises several steps, including creating a MUD-aware SDN controller that can dynamically configure the network based on the MUD metadata, formalizing security policies using security-by-contract, and deploying a blockchain network to manage the security policies and ensure their integrity.

The proposed framework offers a more granular approach to IoT network security that enables administrators to enforce policies specific to each device's intended behavior. Moreover, it provides a decentralized and tamper-proof platform for managing security policies, enhancing overall system security. Additionally, blockchain technology offers transparency and auditability, which helps detect and mitigate security breaches. However, this method has some limitations and performance overheads. The use of blockchain technology can increase network latency and overhead, which can impact performance. Also, the methodology requires deploying a blockchain network, which can be complex and resource-intensive. Lastly, the technique's effectiveness depends on the accuracy and completeness of the MUD metadata provided by IoT device manufacturers.

A group of researchers has proposed MUDscope [26], an approach to detecting malicious network activities targeting IoT systems in real-world environments like smart homes. MUDscope utilizes MUD profiles to capture and distinguish normal and malicious actions of IoT devices. The proposed methodology analyzes the network traffic and generates signatures for potential attacks. The attack signatures from various devices help identify emerging attack patterns. In contrast to MUD-based methods that only use static information about the expected behavior of the device, MUDscope extends the MUD specification to include contextual threat information provided by a threat intelligence service. The threat intelligence service examines the network traffic, detects possible threats, and conveys the information to the IoT device along with the MUD profile. The device uses this information to adapt its behavior and

mitigate the identified threats.

MUDscope enhances the security of IoT devices by enabling them to adjust to changing threats in real-time and providing a standardized communication protocol for threat intelligence services. Nonetheless, the proposed approach may need more computational resources to process the contextual threat information. Privacy concerns may arise regarding sharing network traffic data with a third-party threat intelligence service. Furthermore, MUDscope cannot detect network attacks that evade MUD rules, such as vendor compromise, MITM, or spoofing attacks. In addition, a MUDscope-aware attacker may bypass the signature-matching methodology by introducing noise traffic to a network attack or mimicking signatures of non-harmful or non-targeted anomalies. While such an attack will be detected, its signature will remain unknown.

The iDAM system, described in [27], leverages MUD metadata to identify the intended behavior of IoT devices, restricting network traffic to only what is necessary. To enable local enforcement of access control policies, the authors propose a distributed MUD framework architecture, which distributes MUD metadata to edge devices. iDAM is made up of three main components: 1) a distributed MUD metadata management system that manages the distribution and updates of MUD metadata, 2) a local enforcement module that enforces access control policies based on the local MUD metadata, and 3) a global enforcement module that enforces access control policies for devices without local MUD metadata. The iDAM system offers several benefits, such as a distributed solution that reduces the overhead of centralized enforcement, local enforcement of access control policies, and the ability to mitigate volumetric attacks on IoT networks by reducing unnecessary traffic. However, iDAM relies on the support of the MUD framework by IoT devices, and the system's distributed architecture may increase network traffic and processing overhead. Furthermore, the proposed approach may only be able to address some types of attacks on IoT networks.

As part of the ongoing efforts to identify malicious activities in IoT devices, [28] has introduced a new approach known as OAT, which facilitates remote Operation Execution Integrity attestation for embedded devices based on the ARM architecture. The primary objective of OAT is to establish a mechanism that enables remote parties to verify the operational status of a device and ensure that malicious attackers have not compromised it. OAT is based on the principle of "operational attestation," which entails measuring the device's runtime operation and comparing it against an expected behavior reference model. To measure the device's operation, a set of sensors is used to monitor various performance aspects, such as power consumption, memory usage, and execution time. Subsequently, these measured values are compared to the expected behavior reference model to determine if the device is functioning correctly.

The proposed method effectively detects malicious attacks on IoT devices and ensures that the device operates as intended. This can help prevent unauthorized access and malware attacks. However, implementing OAT may require additional hardware and software, which could increase the complexity and cost of the device and may result in a slight increase in power consumption and execution time. Additionally, the proposed technique may not detect certain types of hardware-level attacks.

Table 1: Behavioral Integrity Verification

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
MUD Profiling of IIoT [24] (Application area - IIoT)	Uses MUD metadata to create a behavioral profile of IoT devices	Detect and prevent zero-day attacks, provide granular access control and monitoring, adapt to new IoT devices and network environments, compatible with SDN architectures	Dependency on the accuracy and completeness of the MUD metadata, may require additional hardware resources and processing power
Software-Defined Security-by-Contract Framework [25] (Application area - IIoT)	Employs a software-defined security-by-contract based approach to define and enforce security policies based on the MUD specification, which describes the intended network behavior of IoT devices	Fine-grained approach allowing administrators to define and enforce policies,ensures a high level of transparency and auditability	Blockchain may introduce additional latency, network overheads, and complexity, effectiveness of the technique depends on the accuracy and completeness of the MUD metadata
MUDscope [26] (Application area - Consumer IoT environment such as smart homes)	Leverages MUD profiles and contextual threat information to detect anomalous activities of smart devices	Increased adaptability of IoT devices to changing threats in real-time, may require additional computational resources or have privacy concerns in relation to third-party threat intelligence service	Ineffective against vendor compromise, MITM, or spoofing attacks
iDAM [27] (Application area - Resource-constrained IoT devices, e.g., WSN)	Performs local enforcement of access control policies by distributing the MUD metadata to the network edge devices	Reduced network overheads as compared to the centralized systems, local enforcement of access control policies, mitigation of volumetric attacks on IoT networks	May introduce additional overheads in terms of network traffic and processing, may not mitigate all types of attacks on IoT networks

Continued on next page

Table 1 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
OAT [28] (Application area - ARM-based bare-metal embedded devices integrated into autonomous systems)	Allows remote parties to attest the Operation Execution Integrity for ARM-based embedded devices	Detect maliciously compromised IoT devices, prevent malware infections or unauthorized access to the device	May require additional hardware and software, increased power consumption, execution time cost and complexity, inability to detect hardware-level attacks

3. Active Device Attestation Schemes

3.1. Software-based Attestation Techniques

A purely software-based attestation approach is best suited to legacy, and ultra low-cost devices, where hardware-based schemes cannot be used due to the lack of specialized hardware on the device [29]. Some of the most prominent software-based device attestation techniques are enumerated in Table 2.

SWATT [30] is a software-based challenge-response scheme designed to validate the integrity of an embedded device using an external verifier. The protocol is illustrated in Figure 2; it allows the configuration settings, code, and static data of embedded devices such as smartphones, eVoting machines, or smart cards to be remotely attested. Being purely software-based, this method is very cost-effective compared to an approach requiring dedicated hardware. It is also suitable for the verification of legacy devices. The basic idea is that the verifier sends a challenge to the device being attested; the end device computes a response to this challenge over its memory contents according to a pre-defined verification procedure that is already stored in the memory of the embedded device. The challenge sent by the verifier acts as a seed to a pseudo-random number generator (PRNG). Subsequently, the output of the PRNG serves as the memory addresses to be traversed during the response calculation. Since an attacker does not know in advance which memory addresses are to be accessed to compute the response, the scheme is safe against pre-computation or replay attacks. Moreover, the procedure for calculating memory checksums is made non-parallelizable by making each byte of the checksum dependent upon the previous byte of the checksum.

However, there are some limitations to this technique. The verifier must be a trusted party and have a complete model of the end device's hardware architecture and contents in its memory. This scheme cannot detect changes made by an attacker to the hardware of the targeted device, such as changing the processor's clock speed or memory configuration. Furthermore, the verifier initiates the attestation procedure upon sensing any misbehavior. Therefore, this scheme does not protect every sensor data/response the end device sends. In addition, since the challenge and response messages are transmitted over the network, the protocol is vulnerable to rainbow [31] and interference attacks [32].

The Rainbow attack is a password-cracking method. The attacker first captures the encrypted password hashes used to authenticate users on the wireless network. Then he uses a precomputed table of possible password combinations, known as a rainbow table, to match the captured hashes with the

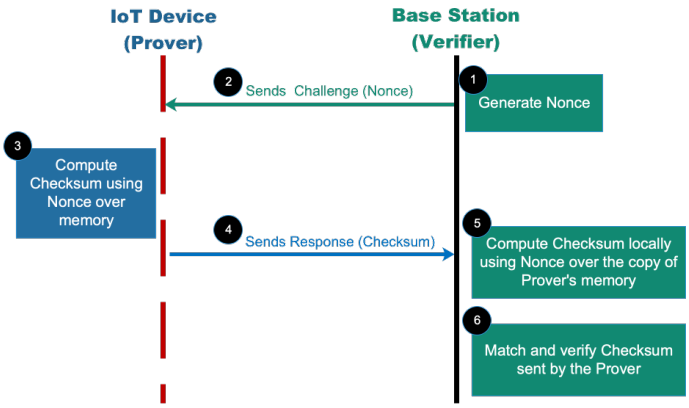
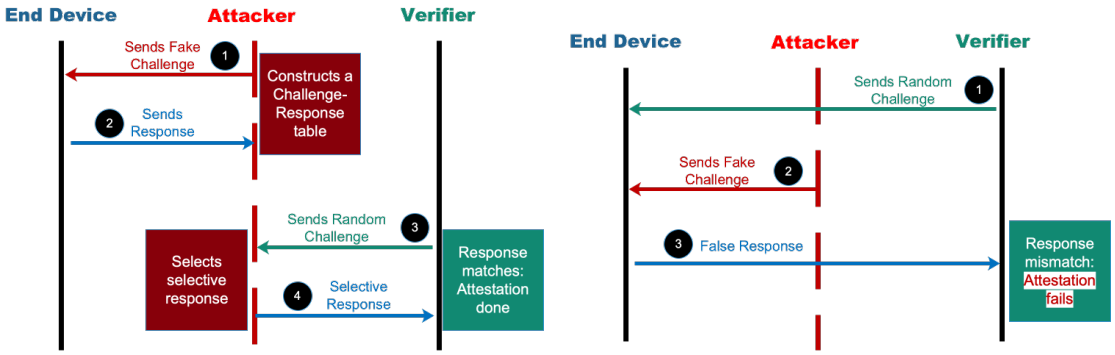


Figure 2: A simple challenge-response protocol



(a) Rainbow attack

(b) Interference attack

Figure 3: Two modes of attack against software attestation protocols

corresponding plaintext passwords. Once the password is obtained, the attacker can use it to gain access to the network. Moreover, as shown in Figure 3(a), the attacker may intercept the challenge and response messages between the end device and the verifier. The intercepted messages enable the attacker to infer the attestation information, such as the length of the challenge. The attacker later sends fake challenges to the end devices and builds up a lookup table consisting of challenges and respective responses. These responses are then used for false attestation.

As per Figure 3(b), in an interference attack, the attacker's objective is to manipulate the end node to send miscalculated responses or disrupt the communication between the attesting device and the verifier. One of the motives behind this attack is to prevent the attesting device from successfully completing the attestation protocol and thus avoid the detection of any malicious behavior. The rainbow and interference attacks apply to both wireless and wired networks. However, these attacks are more commonly used against wireless networks due to their vulnerabilities and the ease of capturing network traffic in wireless environments.

SWATT protocol follows the Coupon Collector's Problem [33] to calculate the number of accesses to the device memory to access each address (location) at least once. The Coupon Collector's problem is a well-known mathematical problem that deals with the number of trials required to collect a complete set of items from a set of available items. Each item is equally likely to be selected at each trial. Accordingly, SWATT requires $O(m \log(m))$ accesses to memory to access each address at least once, where m is the memory size. As per [34, 35], SWATT needs to access the memory for atleast 50,000 times to identify

a malicious device, resulting in long execution times and complexity. Another limitation of SWATT is the difference between the time of attestation and the time the IoT device sends sensor data. The difference in attestation time and device operation time is termed as the difference in the Time-of-Check-to-Time-of-Use (TOCTTOU). Exploiting the gap in TOCTTOU, the contents in the memory of the sensor node can be altered by the attacker after verification but before the device sends a sensor data.

Table 2: Software-based attestation techniques

Technique	**Basic Idea**	**Advantages**	**Limitations/ Performance Overheads**
SWATT [30] (Application area - Embedded IoT devices (WSN))	Remotely attests the configuration settings, code and static data of the embedded device	Low cost, suitable for legacy devices, safe against pre-computation and replay attacks, non-parallelizable	The verifier has to be trustworthy, cannot accommodate hardware changes, vulnerable to rainbow and interference attacks, TOCTTOU is different, vulnerable to ROP attack
SCUBA [36] (Application area - Embedded IoT devices (WSN))	A challenge-response based secure attestation technique	Small memory traversal required for attestation, low energy consumption, safe against impersonation, checksum forgery and speeding up checksum computation attacks	Vulnerable to ROP attack, does not consider attacker as part of the network during the verification process, requires an efficient key exchange and management scheme between the base station and end nodes
Distributed Software-based Attestation [37] (Application area - Embedded IoT devices (WSN))			
a. Basic Threshold Secret Sharing Scheme (BTSS)	A secret-sharing based challenge-response protocol	Safe against pre-computation attacks	Cluster head is a trusted party, resource intensive, TOCTTOU is different, vulnerable to ROP, rainbow and interference attacks
b. Majority Voting-based Scheme (MVBS)	A majority voting-based scheme, in which more than half of the network nodes attest an end device using a challenge-response protocol	Does not rely on a single trusted party for the attestation process	Resource intensive compared to BTSS, TOCTTOU varies, vulnerable to ROP, good mouth, bad mouth, rainbow and interference attacks

Continued on next page

Table 2 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
OMAP [38] (Application area - Embedded IoT devices, e.g., Smart Meters)	A one-way memory attestation protocol	Avoids network and message forging attacks, prevent parallel execution of the protocol iterations	Depends on efficient clock synchronization between network devices, does not consider hardware modifications, protocol initiation procedure is very subtle
OWCAP [34] (Application area - Embedded IoT devices (WSN))	One way remote attestation protocol	Implicit code attestation, eliminates TOCTTOU gap, prevents network attacks, infers minimal communications and energy overheads, avoids replay and impersonation attacks	Memory intensive, does not consider hardware, network configuration and modified ambient conditions
SIMPLE [39] (Application area - Class 1 IoT devices integrated into smart systems)	A hypervisor-based remote attestation technique for IoT devices	Does not require hardware modifications such as MPU or ROM	Considers only remote software-based attacks, the verifier has to be a trusted party, vulnerable to physical compromise attacks involving memory manipulation
RealSWATT [29] (Application area - Embedded devices integrated into smart systems)	A continuous remote attestation scheme for real-time embedded systems that uses a dedicated processor core for the attestation process	Detects malware infections and malicious changes by remotely verifying code and data sections, provides timing guarantees without additional hardware, does not require trusted computing components, resolves TOCTTOU problem	The embedded device should have a multi-core processor, the verifier and the gateway devices are trusted entities, assumes that attacker cannot modify the hardware of the end devices
D2Gen [40] (Application area - General purpose microprocessors/mini computers with Linux OS)	Verifies integrity of devices by computing digital genome at runtime and comparing it with genome computed earlier at default device settings	Detects a compromised device based on changes made to the default software, hardware and network configuration	Currently, works for static parameters only; thus further research is required to incorporate dynamic parameters as well

Similarly, (SCUBA) [36] is a challenge-response based security technique that detects a compromised node in a WSN using code attestation. Once identified, the victim node is recovered through a secure code update procedure. However, the researchers assume that the process of code update cannot be hindered by the attacker. The proposed technique assumes that the verifier/base station knows the specific hardware in each sensor node. Also, messages between the verifier and the end node are authenticated and secured using PKI and symmetric encryption, respectively. There is a strong assumption that the verifier is a trustworthy device that cannot be compromised and that the sensor device identity (ID) is stored in read-only memory (ROM) which is immutable. Hence, it prevents spoofing attacks.

SCUBA employs an Indisputable Code Execution (ICE) technique that facilitates uninterrupted code execution on embedded devices. The ICE attestation method sets up a secure execution environment before it computes the checksum by disabling CPU interrupts. The checksum is computed over parts of memory holding the ICE attestation method, the SCUBA application code, end device's identity (ID), gateway device's public key stored in the ROM, and challenge code. Hence, if the CPU state is modified by any means, either the produced checksum would be incorrect, or it will have different computation time. The proposed scheme protects against checksum forgery and speeding up checksum computation attacks. It also prevents impersonation attacks. However, using a simple checksum rather than a cryptographic hash is a weakness of this approach, particularly given the power of modern computing hardware.

After returning the checksum, the SCUBA protocol computes the end device's memory hash and forwards it to the verifying node. The verifier compares both the hash values to detect a compromised device. To precisely identify the modified memory locations of a device, the verifier may ask the end device to compute hashes of specific memory contents.

Distributed software-based attestation [37], a challenge-response based approach, comprises two sub-techniques: a basic threshold secret sharing scheme (BTSS) and a majority voting-based attestation scheme (MVBS). In the BTSS scheme, an end device's unused memory space is initially filled with pseudo-random content. Then, the end device splits the pseudo-random number generation seed into multiple parts and forwards each part, along with the cryptographic hash of the seed, to a specific neighbor (peer node). The seed is then removed from the node's memory. The attestation protocol is triggered if more than half of the peers suspect malicious behavior. The peers select a cluster head, which sends an authentication challenge to the suspect node. While the node computes a response, the cluster head collects the seed fragments from other nodes and recovers the pseudo-random number generation seed. If the hash of the recovered seed does not match the cryptographic hash sent by the end device earlier, the cluster head collects another set of fragments and recalculates the seed. However, if the seeds match, the cluster head calculates and compares the expected checksum with the value sent by the end device being attested. If the responses match, the end device is authenticated. It is difficult for an attacker to determine the responses beforehand because the attestation protocol utilizes random challenges. Moreover, to retrieve the seed, the adversary must get hold of $>= k$ fragments.

BTSS has several specific limitations. For instance, the cluster head is reliable (trusted). Similarly, suppose a compromised peer contributes a false fragment. In that case, the cluster head must select another k fragment from the other network devices to retrieve the desired seed, thus consuming more network resources and time for attestation. As an alternative, every end device should also store a copy of each fragment's hash - however, this will consume additional memory resources for every end device in the network. In addition, the TOCTTOU is also different, giving a window of opportunity for malicious activity.

The second sub-technique, MVBS, is also a challenge-response scheme. As for BTSS, in MVBS, unused memory in end devices is filled with a pseudo-random sequence before deployment. The devices are also pre-loaded with tuples of challenge-response pairs. After deployment, every end device shares

a specific tuple with its neighbors and removes (overwrites) all the tuples from its memory. Suppose that at least half of the network devices/nodes suspect malicious behavior of an end device and agree to perform a validation test on it. They send challenges from the challenge-response tuple set earlier received from the respective end device. Subsequently, based on the responses, the respective end device is identified as a compromised node if more than half of the nodes detect a mismatch.

Like other voting-based schemes, MVBS is also prone to good and bad mouth attacks [41]. These attacks can arise when there are already some malicious peers near the node being attested. The malicious peers can collude and maliciously vote in favor of a compromised node or against an uncompromised node. Moreover, the TOCTTOU is also different. Since the device attestation protocol is initialized upon detection of skeptical performance, a node can behave maliciously but below this sensitivity threshold, allowing some malicious activity to go undetected. MVBS is notably more computationally intensive than the BTSS scheme, as the attested node has to compute R_n responses to C_n challenges. In addition, being challenge-response based, both BTSS and MVBS are vulnerable to rainbow and interference attacks.

In [15], researchers evaluated SWATT [30], SCUBA [36], and distributed software-based attestation techniques [37] for their vulnerability to Return Oriented Programming (ROP) attacks. The researchers identified that the leveraging ROP technique, an adversary can move the malicious code to the unused parts of the data memory or kept it in compressed form in the program memory to remain undetected during attestation. The distributed software-based attestation technique, fills the unused program memory with a pseudo-random sequence to prevent storing of malicious code in the free space. The researchers also note that the distributed software-based technique and SWATT both attest the program space only. At the same time, a malicious prover may store malicious code in the data memory or external memory and restore it after the attestation using ROP.

By contrast, OMAP [38] does not send a challenge to verify the integrity of a smart meter's code. However, OMAP makes several unrealistic assumptions about the verifier and the attacker. E.g., authors assume that the gateway device knows the end node's hardware specifications and memory configuration, maintains a precise copy of the remote device's memory, and the attacker cannot compromise the verifier. Similarly, the assumptions about the attacker's capabilities include the following: The attacker can alter the memory contents and intercept network traffic; however, the attacker is not capable of altering the hardware, i.e., changing the firmware, adding memory or modifying memory access timing, and increasing processor's clock speed. In contrast, all these hardware-level attacks are possible in the real world [42].

Concerning smart meters, OMAP assumes that every smart meter/end device utilizes its unique serial number for the attestation protocol, which the attacker cannot modify. This assumption can be realized using a Physically Unclonable Function (PUF) generated unique device identifier. A PUF is a hardware component that produces a unique key for every IC by exploiting the inherent imperfections of the integrated circuit (IC) induced during manufacturing [43]. Hence, every IC is physically different than other ICs. The imperfections or variations can be measured differently, e.g., voltage gains, path delays, transistor threshold voltages, etc. Although these variations differ from IC to IC, they are deterministic and will always yield the same output given a similar input. However, the authors have not explained the assumption concerning the secrecy of the serial number, that how they intend to keep the device serial number secret. On the contrary, relying on this assumption, the researchers claim that the attacker cannot forge a message because of the use of a unique and secret serial number and the synchronized timestamp added by the end device.

Since, OMAP utilized the timestamp while computing checksum, it prevents the attacker from parallelizing the iterations of the protocol. As OMAP is a one-way attestation scheme, it is not vulnerable to rainbow and interference attacks. However, despite its computational efficiency compared to other attestation techniques, OMAP does not explicitly have a procedure to initiate the code attestation process.

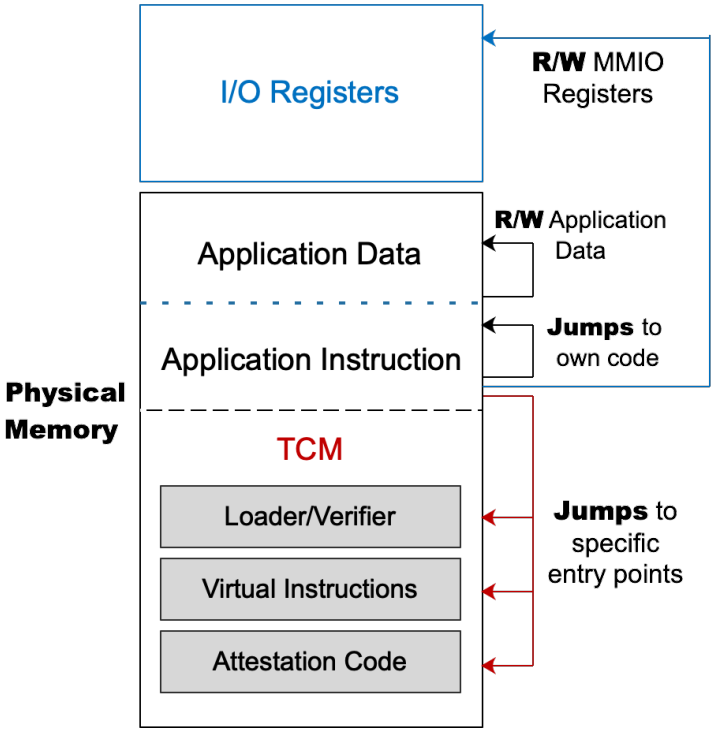


Figure 4: Isolated memory map

Therefore, a compromised device may continue to maliciously contribute to the respective network until it is suspected and an attestation protocol is triggered. Moreover, the time stamp mechanism requires an efficient clock synchronization protocol to maintain consistency between the end node and verifier devices.

To avoid several of the weaknesses of OMAP, OWCAP [34] extends it in a number of important ways [38]. The primary objectives of OWCAP include implicit code attestation, eliminating the TOCTTOU gap, preventing network attacks, and imposing minimal communications and energy overheads on IoT devices. As for OMAP, OWCAP assumes that the serial number of an end device is unique and secret, the verifier node has knowledge of the end device's memory space and processing capabilities, and it keeps memory images of all the nodes. The OWCAP protocol attests a remote IoT device with a routine sensor update message. Therefore, unlike other attestation schemes, OWCAP avoids a rootkit-based ROP attack [15]. The end device uses the timestamp and the device's secret serial number to compute the PRNG seed. The protocol avoids replay attacks by using network-synchronized system time as a nonce. Impersonation attacks are prevented by using the device's secret serial number. The authors suggest that like in some of the digital rights management (DRM) schemes, white-box cryptography can help secure the serial number. Nonetheless, while verifying code integrity, OWCAP cannot prevent malicious hardware changes or modifications to network configuration or other metadata.

Researchers in [44] proposed a combination of the delta updates mechanism and a private blockchain-based solution to validate the legitimacy of IoT devices' firmware and delta updates. The researchers demonstrate that delta updates alone fail to efficiently verify the integrity of outdated firmware and guarantee the proper application of the requisite updates. The proposed scheme stores the firmware versions, firmware and delta files' metadata, and checksum in respect of the end devices on a private blockchain server. Later, when the firmware manager detects any anomaly in the device's behavior, it sends a query to the device asking for its firmware version information. The blockchain server receives the query response and verifies its correctness.

Most of the traditional software-based approaches are susceptible to runtime manipulation of the attestation protocol. In addition, most IoT devices being used today do not meet the minimal hardware requirements to deploy hardware-based or hybrid attestation schemes. [39] presents a SIMPLE, a software-based attestation protocol that leverages micro visor (SμV) to isolate parts of device memory [45]. SIMPLE protects against remote-only attacks without additional hardware requirements such as MPU or ROM. The proposed approach segregates the unsafe application code from a software-based Trusted Computing Module by employing software virtualization and code verification at the assembly-level. SIMPLE leverages single-threaded micro-controllers with sufficient flash or ROM, lacking MPUs, and supporting global-interrupt disabling.

The main idea behind SIMPLE is that it ensures control-flow integrity [46] by protecting various sensitive parts of the device's memory. For instance, as shown in Figure 4, SIMPLE reserves a memory portion exclusively for the TCM. The remaining memory is used for the application executable and related data. Although, TCM memory has no operational restrictions, the data memory stores only data and cannot execute instructions. The application code can only be executed from instruction memory with access to data memory and memory-mapped IO (MMIO) registers. Application can access addresses within the instruction memory, and execute instructions in its logical domain or from specific points in the TCM memory.

In SIMPLE, the application code is restricted at the instruction level through a) deployment of the application occurring only after verification by security micro visor TCM at load time, and b) mapping of unsafe instructions to their virtualized equivalents in TCM memory. Correspondingly, if an instruction attempts to compromise the device's memory, it is regarded as unsafe by the secure micro visor and therefore rejected from the outset. However, comparing SIMPLE with other hybrid remote attestation schemes, after physically compromising a device, the attacker can re-flash (remove) the secure micro visor, which is essential for ensuring dynamic root of trust.

In contrast to the approach taken in SIMPLE, researchers in [29] argue that the control-flow integrity mechanism is unsuitable for real-time IoT systems since it impacts the execution time of the tasks [47]. This is a serious problem as the real-time systems have strict timing requirements. Hence, the performance of a device may be affected due to any alteration in the execution flow while performing control-flow attestation. Therefore, [29] proposed a unique remote attestation mechanism for real-time embedded systems without specialized hardware or trusted computing components. The proposed solution, RealSWATT, utilizes a dedicated processor core to isolate the untrusted application from the attestation protocol.

RealSWATT primarily implements a continuous attestation mechanism in which a verifier sends an attestation request to the end device before receiving the previous response. Thus, the end device starts working on the next response after finishing the previous one. The authors argue that the continuous attestation approach does not provide a window of opportunity for a potential attacker in which no attestation is being performed. Subsequently, RealSWATT also resolves the TOCTTOU problem. However, the gateway/verifier nodes are considered to be trusted entities that cannot be compromised.

Similarly, researchers in [40] introduced a method for verifying the integrity of devices in collaborative intrusion detection systems (CIDS). The proposed methodology is inspired by the mechanism of identifying living things, especially humans, based on their unique genetic code [48]. Based on the concept of the Genome, which comprises the complete set of DNA of a living thing, including all the genes, the researchers verify the integrity of a device by computing its digital Genome named "D2Gen." The digital Genome is a cryptographic hash that is computed over selected software and hardware features, information about the device and network configuration. Subsequently, a malicious modification in any of the above-mentioned default features/settings will result in a different D2Gen. However, the proof of concept of the proposed technique only incorporated some static parameters; further research is required to identify anomalies in the normal operating profile of the devices, including dynamic parameters such as

current drain at specific ports, power consumption, memory usage, and CPU load, in case of an attack.

3.2. Hardware/Co-Processor based Device Validation Techniques

In any IoT system, guaranteeing the integrity of operating systems (OS) and applications running on an end device is essential. This requirement becomes necessary if these devices operate in a trustless environment where they may be subject to various attacks. Moreover, depending on the type of industry, the applications may require stronger trust guarantees. The trust is established by ensuring that the communicating parties are mutually attested. E.g.,, when an IoT device sends a sensor reading to a gateway device, both devices mutually authenticate each other. Later, the gateway device verifies that the IoT device has not been tampered with.

The hardware-based approaches require specialized security hardware modules alongside commodity processors, such as Intel SGX, ARM TrustZone-A/M Trusted Execution Environments (TEEs), AMD SEV, and RISC-V PMP to ensure the integrity of device OS and mission-critical applications such as the attestation code. The security hardware also assures secure authentication of the verifier and the end device. TEEs do not rely on a trusted OS; they execute application code in a hardware-protected environment with robust security guarantees. Similarly, RISC-V, an open-source ISA, offers Physical Memory Protection (PMP) [49]. A finite number of PMP regions can be configured to enforce access permissions to a range of addresses in memory.

Table 3: Hardware-based validation techniques

Technique	**Basic Idea**	**Advantages**	**Limitations/ Performance Overheads**
TEE and Remote Attestation [16] (Application area - Trusted/secure computing in any type of IT/IoT system)	TEE provides evidence for the attestation of software and hardware component	Establishes a root of trust between the host (user) and the verifier	Requires special hardware
Intel SGX [50] (Application area -All use cases that require secure execution and storage of sensitive data and code.)	Creates and utilizes encrypted regions of memory and EPC to store data and code at boot time	EPC is inaccessible to other programs, including OS and hypervisor running on the same machine, verify that RAM associated with an EPC was not modified by any external application	Require special hardware
Arm trustZone [51] (Application area - Use cases vary from mobile devices to cloud computing)	Divides a processor into two regions, a secure TEE and a normal untrusted one, where each region has its own user and kernel space	Trusted applications run as an isolated process on trusted OS	Lacks built-in attestation mechanism, require additional hardware capabilities

Continued on next page

Table 3 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
AMD SEV [52] (Application area - Virtual Environments (Virtual Machines))	SEV isolates VMs and containers from trusted hypervisors. It protects code and data from unauthorized access inside the processor by tagging each VM and hypervisor with an ASID and restricting the usage of code and data to only the owner with the same ASID	Code and data are encrypted using AES-128	Require a secure co-processor
Co-pilot [53] (Application area - Integrity monitoring for commodity systems (with specific series of Linux Kernel))	It is a co-processor-based mechanism to verify the integrity of the host system's kernel	Ability to detect some of the common rootkits at the time of its development	It cannot guarantee that a malicious code has not executed on the host system, the attacker can stealthily launch attacks including timing attacks, and advanced relocation attacks
TCG-based Integrity Measurement Architecture [54] (Application area - Integrity Measurement Architecture for Linux Systems)	Measures the integrity of the executable content loaded onto the respective Linux system before its execution, provides a mechanism for the attestation of software stack without special CPU modes or OS	Protects the integrity of the in-kernel lists	Vulnerable to DoS attack due to increase in the size of measurement list amid frequent changes in loaded executable files

Several well-known secure hardware technologies and associated attestation schemes are listed in Table 3. TEE-based remote attestation [16] provides evidence for the attestation of software and hardware components in the form of a cryptographic value such as a hash or a message authentication code (MAC). The procedure of deployment and subsequent attestation using a TEE is illustrated in Figure 5. The developer first tests the application; once functionally validated, a cryptographic checksum of the application is computed. Later, when the application is deployed on a user's untrusted system, the application is executed inside the TEE to establish a root of trust between the host (user) and the verifier. The TEE generates evidence that can be passed to a trusted application and then to a verifier over a secure channel. The verifier compares the evidence with pre-computed reference values to determine if the trusted application is legitimate.

The Skylake CPU architecture from Intel Software Guard Extension (SGX) includes a TEE [50]. The SGX instruction set creates encrypted enclaves in memory protected by a special execution mode of the CPU. A reserved Enclave Page Cache (EPC) stores code and data of secure enclaves, and a Memory Encryption Engine (MEE) protects the CPU and system memory traffic. The EPC also keeps verification codes to prevent modification by external software. A local attestation can be established between trusted applications in different enclaves on the same device.

On the Intel SGX platform, reports containing identities, attributes, trustworthiness information, and metadata are created and signed using the EREPORT instruction. Intel introduced the concept of a quoting enclave, which provides a cryptographically signed attestation secured by PKI. The quote binds a legitimate Intel SGX processor to a specific application code measurement. This mapping can be remotely attested using the Intel attestation service or dedicated PKI.

According to [51], ARM TrustZone creates a secure and an untrusted execution environment on a processor. The two regions are separated by a secure monitor instruction (SMC), each with its user and kernel space. Trusted applications run in a TEE, while the normal region uses a traditional OS like Linux. Despite its widespread use, TrustZone does not have a built-in attestation mechanism, making it impossible to verify the TEE's state remotely. To overcome this limitation, various one-way remote attestation [55, 56] and mutual remote attestation protocols [57, 58] have been proposed, which rely on additional hardware, a secure TEE, key management, and boot mechanism. Devices without built-in attestation mechanisms may build a root of trust by deriving cryptographic materials using a secret built into the processor. The integrity of boot stages must be ensured with a secure boot process, and attesters' trustworthiness can be checked by verifying the evidence issued by the TEE.

On the other hand, AMD Secure Encrypted Virtualization (SEV) [52] secures virtualized environments, such as virtual machines (VMs) and containers, from trusted hypervisors. SEV encrypts memory using an embedded hardware AES engine and has an Arm Cortex-v5 secure co-processor that generates cryptographic primitives. The access to the processor is controlled using unique keys and Address Space Identifiers (ASID) assigned to each VM and hypervisor. The code and data are encrypted using the AES-128 cipher for additional security. Cross-TEE attacks are prevented using ASID by restricting access only to authorized owners.

In an updated version of the original SEV architecture, SEV-ES (SEV encrypted state) was introduced to address a flaw in the original design that made it susceptible to exposing sensitive information during guest register interrupts to the hypervisor [59]. With SEV-ES, register states are encrypted, and the guest OS grants hypervisor access to specific registers. SEV uses a Chip Endorsement Key (CEK) embedded into the processor for its attestation mechanism, similar to SGX enclaves. The VM is initiated in an unencrypted state, and remote attestation queries are used to provide cryptographic secrets and confidential data. The secure co-processor measures the integrity checksum of the VM, including the initial guest memory layout in the digest. The metadata of the memory pages is measured through AMD-SNP (Secure Nested Paging). SEV and SEV-ES support remote attestation only during the guest OS's launch. Whereas, in SEV-SNP, the guest VM may request attestation reports at any time and also obtain the cryptographic details to ensure secure storage of data.

Several TEE designs have been proposed for the open-source RISC-V processor architecture. These designs support remote attestation and physical memory protection (PMP) instructions. Some of these RISC-V TEE architectures are listed below:

a. Sanctum [60] helps verify trusted applications. It proposes a remote attestation protocol based on a root of trust model. Like Intel SGX, it also ensures appropriate software/application isolation through enclaves. Moreover, to provide verifiable protection, it has two open-source components; a secure monitor and a measurement root called *mroot*. Once the device is booted, necessary keys are generated by the mroot and handed over to the secure monitor. Sanctum has a signing

enclave that produces the evidence signed with a secret key received from the secure monitor. In addition, the protocol establishes a secret session key between the verifier and the attester. Later, based on multiple claims, a regular enclave asks for a piece of evidence from the signing enclave. The evidence is computed as a cryptographic hash over the code of requesting enclave and key exchange messages. The hash is shared with the verifier through a secure communication channel established for attestation. As an alternative to the remote attestation scheme requiring a root of trust, the researchers suggest the use of a PUF to generate a unique key and secure boot mechanism.

b. TIMBER-V [61] provides secure execution for low-end embedded devices using hardware-assisted memory tagging, including the transparent association of additional metadata with the tagged memory. As in TrustZone, TIMBER-V has separate supervisor and user modes split into normal and secure spaces. A trust manager called *TagRoot* controls access to the memory in the supervisor mode.

   Unlike ARM's TrustZone, the secure mode in TIMBER-V handles multiple isolated enclaves. These enclaves support numerous processes by combining tagged memory with an MPU. Enclaves can retrieve evidence from an API provided by the trust manager based using a secret key (the root of trust), enclave ID, and a random identifier provided by the trusted application. Concerning remote attestation, the verifier sends a challenge to the attester (trusted application). Based on the same challenge, the trust manager issues evidence that is authenticated using a MAC. Currently, TIMBER-V leverages a symmetric encryption key to share the evidence securely.

c. Keystone [62] is a modular framework that has been developed to create TEEs in RISC-V systems. This modular framework allows for the implementation of a secure monitor in machine mode, without requiring any hardware changes. It offers various security features, including dynamic memory management, encryption, and cache partitioning. Keystone securely measures the image of the secure monitor and generate a response for attestation. The response is signed with a root of trust (hardware-visible secret). The enclaves communicate through a Supervisor System Interface (SBI) the secure monitor provides. Moreover, a part of the SBI issues signed pieces of evidence (using keys provided and endorsed by the verifier) computed using the secure monitor measurement, runtime, and enclave's trusted application. Accordingly, during remote attestation, the verifier challenges the trusted application being attested. The application responds with an evidence in the form of challenge reply and its public key. The evidence is then check for the legitimate public key and correct measurements.

d. LIRA-V [63] offers an attestation framework for resource-constraint IoT devices. Importantly, LIRA-V does *not* perform code execution inside the TEE. Despite this, it presents a complete remote attestation mechanism based on user and machine modes. Hence, a program in ROM computes claims over selected addresses of device's physical memory. The mutual attestation protocol depends on secret keys for root of trust. During attestation, the verifier sends a challenge with a public session key. The attester then sends a response containing the challenge, public session key, and evidence for the respective device. The response is encrypted using a pre-established session key; subsequently, if the verifier affirms the evidence, it becomes the attester and issues evidence to the other device (previous attester) that has now become the verifier.

An extension of a file system auditing tool, Co-pilot [53] is a co-processor-based mechanism to verify the integrity of the host system's kernel. Its efficacy had been tested based on its ability to detect some common rootkits at its development. It was a landmark achievement at that time because the most sophisticated rootkits modify the OS kernel of the compromised host and remain undetected to work on

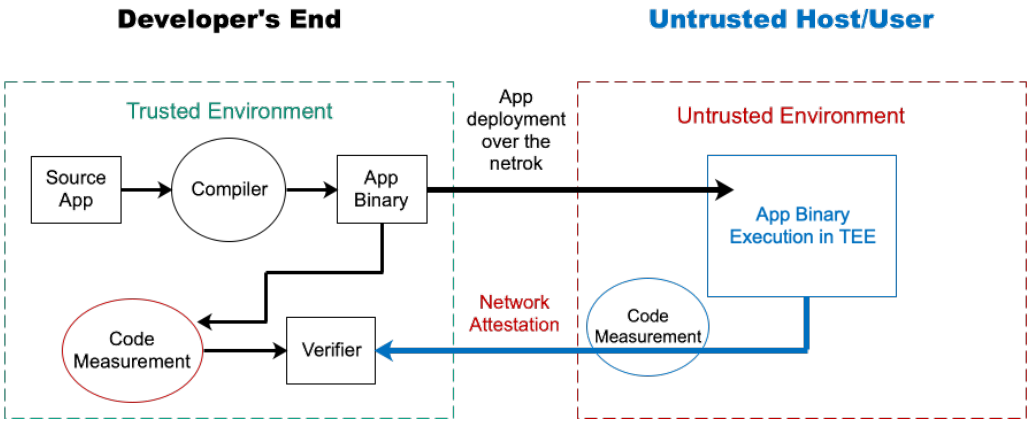


Figure 5: Deployment and attestation of applications in a TEE

behalf of the attacker secretly. Consequently, once the adversary has changed the kernel's functionality, no program running on the host system can be trusted to give correct output/results - a problem that also can compromise the programs used for detecting rootkits. However, the main limitation of a co-processor-based kernel monitor is its inability to guarantee that an invalid piece of code has not been executed. Even so, since the Co-pilot can monitor a system's main memory, the attacker has limited options to launch a successful attack while remaining undetected. Such attacks may include timing attacks and extremely advanced relocation attacks.

In [54], researchers describe a system for measuring the integrity of executable content on Linux using Trusted Computing Group (TCG) technology. Before execution, the integrity of the content is measured, and the resulting measurements are stored in an ordered list within the kernel. To protect the list's integrity, the trusted platform module (TPM) is used. However, instead of holding the measurements directly, the TPM protects the list's integrity. The proposed system uses TCG attestation mechanisms to present the TPM state and ordered list to a remote party, which can then verify the integrity of the measurements and determine the level of trust associated with them. To minimize performance overhead, the measurement results are cached, and no further computations are performed as long as the executable content is not modified. However, frequent changes to the content can cause the measurement list to grow beyond practical limits, leading to a DoS attack. To prevent this, a maximum length for the measurement list can be configured. Any additional measurements are aggregated into the TPM-protected PCR register, but they are not stored in the kernel. Thus, a system exceeding the maximum number of measurements will be unable to prove its integrity because the measurement list will not validate against the aggregate.

3.3. Hybrid Attestation Approaches

Researchers in [64] presented the proof of concept of a scalable embedded device attestation (SEDA) scheme. SEDA is intended for device swarms of up to one million devices, with dynamic network topologies such as VANETs, robot swarms, and fluid sensor networks, where the device swarm consists of heterogeneous devices with a variety of different hardware and software configurations. The proposed method aims to distribute the communication and computational workload across all the devices in the swarm, with the end objective of verifying that all the devices have an authentic software. SEDA assumes that no physical attacks on network devices are possible and that it is infeasible for an adversary to extract cryptographic secrets used for attestation.

Once the network is bootstrapped, every device equipped with a secure boot feature connects to its neighbors and obtains a copy of their authentication certificate (as issued by the swarm operator). The

same certificate is used to authenticate all attestation reports signed by the respective neighbor. Every device shares an attestation key with every neighbor, with the key establishment being performed via an authenticated key agreement protocol. The network has an online as well as offline phase. In the offline phase, the swarm operator initializes all the embedded devices. Later, in the online phase, the verifier triggers the swarm attestation protocol through a network device acting as the attestation tree's root. The root device progressively connects to all the network nodes in a Spanning Tree Protocol (STP) topology. It returns a cumulative attestation response for all the connected devices to the verifier.

SMART [65] is a hybrid remote attestation scheme designed for embedded devices comprising a ROM to store the attestation application and the attestation key. It also has a memory protection unit (MPU) that manages access to the part of the ROM, holding attestation key. Similarly, the TrustLite [66] provides an attestation mechanism for devices equipped with Intel's Siskiyou Peak Research platform. TrustLite's secure architecture enables secure execution of the program code by incorporating an execution-aware memory protection unit (EA-MPU). The EA-MPU controls access to data depending on the code being executed at that time.

However, a version of SEDA with a compromised device identification feature infers high message complexity. Moreover, to protect against physical compromise and device cloning by an adversary, the researchers recommend using PUFs, which is a type of hardware security primitive. In addition, in SEDA, all intermediaries have to be trusted parties. Moreover, SEDA is ineffective if an attacker can physically tamper with devices.

Improving upon the security and scalability in remote attestation, researchers in [67] proposed SANA, a secure and scalable network aggregate attestation scheme. SANA is designed to attest IoT swarms using an optimistic aggregate signature scheme that facilitates public verification of collective attestation without any trusted aggregators/intermediaries. It can attest around one million IoT devices in just 2.5 seconds. However, all the end devices must meet minimum hardware security features, including the provision of an MPU, and the network owner must be a trusted party. The overhead of SANA is a linear function of the number of compromised devices/bad provers. Hence, when the number of bad provers exceeds a threshold defined by the network owner, the ultimate aggregated response becomes unverifiable.

Most of the software-based attestation schemes discussed in Section 3.1 (with the exception of OWCAP [34] and OMAP [38]), and some of the hybrid techniques discussed earlier are challenge-response based. According to the researchers in [68], challenge-response-based attestation protocols are intrinsically vulnerable to DoS attacks. The adversary can overwhelm an end device with fake attestation challenges/requests in these attacks. Hence, to prevent such a DoS attack, [68] proposed SeED, a secure non-interactive attestation for embedded devices in which an end device initiates the attestation protocol. Though SANA addresses the issue of a DoS attack by using an authentication token (issued by a third party) in the challenge, it still relies on a trusted third party. Moreover, SANA employs expensive public-key cryptography that infers considerable computational overhead to the attestation process.

Table 4: Hybrid attestation techniques

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
SEDA [64] (Application area - Low-end embedded systems)	A remote attestation scheme with a single verifier and multiple provers for device swarms with dynamic network topologies	Scalable for up to one million embedded devices	Assumes no physical attacks on devices, requires MPUs or PUFs for secure code execution and tamper resistance, all intermediaries are trusted parties
SANA [67] (Application area - Low-end embedded systems)	An aggregate network attestation scheme	Scalable to large IoT networks, publicly verifiable, accounts for heterogeneous IoT devices	All the end devices have to meet minimum hardware security features, network owner has to be a trusted party, the protocol overhead is a linear function of the number of bad provers
SeED [68] (Application area - Low-end embedded systems)	A non-interactive remote attestation protocol	Prevents DoS attacks with low communications and computational overhead	Requires secure hardware (MPU, RTC), depends upon very accurate clock synchronization among all the network nodes
Low Power Data Integrity in IoT Systems [69] (Application area - WSN)	Employs a random time hopping sequence and permutations-based approach to verify data integrity	Low energy consumption, safe against data modification, data injection, replay, physical compromise and cloning attacks	
DARPA [70] (Application area - WSN)	Employs heartbeat protocol to detect the absence of devices and mark these devices as physically compromised	Requires minimal security hardware, i.e., ROM to store attestation code and an MPU to control access to ROM and RAM	Only detects the absence of nodes based on their failure to send timestamped heartbeat message, prone to false positives, poor scalability with high computation and communication costs

Continued on next page

Table 4 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
Lightweight Swarm Attestation [71] (Application area - Low-end embedded systems)	A remote attestation scheme for IoT swarms which verifies both code and data memory against remote malware attacks	Considers limited mobility of IoT network during attestation, minimal security hardware requirements for IoT devices	Does not focus on hardware or software modification attacks, vulnerable to attestation timeout-related DoS attacks, the verifier is assumed to be a trusted party that cannot be compromised, not implemented on practical IoT swarms
US-AID [72] (Application area - Large, autonomous and dynamic-topology networks of embedded devices)	Combines attestation of network peers and heartbeats to detect compromised nodes using a key exchange mechanism	Caters to the dynamic network topology, infers constant computation and communication costs with the increase in network size	Does not account for a network scenario where a device remains offline for a long period of time, each device needs to store a number of lists for continuous attestation, high energy costs over and above routine device operation
ESDRA [73] (Application area - Low-end embedded systems (WSN))	A many-to-one distributed remote attestation scheme in which neighboring nodes verify the prover, prevents single point of failure verifier	Low attestation run-time, reports compromised nodes	Does not cater to the hardware compromise and DoS attacks
HEALED [74] (Application area - Low-end embedded systems)	Focuses on the detection of compromised software on low-end embedded devices and restoring the devices to the benign state in a secure manner	Provides a healing mechanism to restore compromised nodes	Only considers malware attacks, the verifier devices must know about the software configuration of all the prover devices, all the devices in a group need to share pair-wise symmetric keys which is not scalable to large groups

Continued on next page

Table 4 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
RADIS [75] (Application area - Control-flow integrity of distributed IoT services)	Instead of software integrity, it verifies the integrity of distributed IoT services' control-flow	Augments security of single-device control-flow attestation mechanisms, caters to heterogeneous IoT devices	Does not consider a physical compromise of end-devices and DoS attacks, requires further research on the optimization of hash computation for resource-constrained IoT devices
HAtt [35] (Application area - Low-end embedded systems)	A passive, lightweight hybrid attestation scheme that uses a randomized approach to attest different parts of an IoT device's memory	Uses PUFs to safeguard against physical compromise, ensures high availability of IoT devices during the execution of security protocol	The verifier needs to store memory images of all the dependant prover nodes, the verifier needs to be a trusted party
SCAPI [76] (Application area - Low-end embedded systems)	Detects a physically compromised device by its failure to authenticate to a leader using a previous time interval's secret session key	Improves on DARPA and reports the ID of compromised devices, has low communication complexity, energy consumption and runtime	Not very effective for highly dynamic networks
ERASMUS [77] (Application area - Low-end embedded systems)	Let provers self-attest at specific intervals, stores m consecutive measurements and later hands over k measurements to the verifier for verification	Helps detect a mobile adversary, low computation complexity for the provers	Provers have to store m consecutive measurements which can be memory intensive
SALAD [78] (Application area - Large and dynamic-topology networks of embedded devices)	A challenge-response based attestation scheme that focuses on response aggregation	Minimizes the storage requirements by storing only the aggregated attestation responses on the network nodes, a verifier collectively attests the complete network	Merely focuses on response attestation along with traditional challenge-response protocol
PADS [79] (Application area - Large and dynamic-topology networks of embedded devices)	A collective remote attestation scheme	A verifier can attest the integrity of the complete network, resilient to network topology changes	Vulnerable to hardware intrusion attacks

Continued on next page

Table 4 – *Continued from previous page*

Technique	Basic Idea	Advantages	Limitations/ Performance Overheads
WISE [80] (Application area - Low-end embedded systems)	Verifies the integrity of a subset of network devices in each attestation round	Aims to reduce the complexity of the attestation protocol, protects against roving malware	Prone to communication errors in highly dynamic networks
SlimIoT [81] (Application area - Large and dynamic-topology networks of embedded devices)	Organizes devices in clusters, and attests each cluster at a predefined time interval (epoch)	Detects and accurately identifies compromised devices until there is at least one confirmed uncompromised device in the network, allows devices to move during the attestation period	Declares a device as compromised if that device is offline during attestation epoch
EAPA [82] (Application area - Low-end embedded systems)	Detects compromised devices based on heartbeat mechanism similar to DARPA	Has reduced runtime and network overhead as compared to SCAPI, every device acts as a verifier for its neighbors, avoids the potential single point of failure of a unique trusted verifier	Detects a compromised device only if it fails to send a heartbeat message in a given time interval
SARA [83] (Application area - Low-end embedded systems (WSN))	Verifies the integrity of not only the IoT devices but also the distributed services provided by these devices	Avoids suspension of routine device operation during attestation interval	The size of the attestation evidence is directly proportional to the number of services that are included in the attestation evidence

The SeED attestation scheme [68] avoids DoS attacks, and since it does not require a challenge message, the communication overhead is at least 50% lower than challenge-response based schemes. Consequently, it offers an energy consumption advantage, particularly important for low-end battery-powered embedded devices. Moreover, being non-interactive, the attestation protocol is not significantly degraded by network latency, resulting in lower network congestion than challenge-response protocols. However, it requires robust clock synchronization between the prover and the verifier to ensure the attestation response arrives within a pre-defined time window. A real-time write-protected clock (RTC) is required to ensure the integrity of the attestation time. The RTC prohibits the generation of responses over old software measurements, and it has been used to prevent DoS attacks [84] and to detect physical attacks [70]. Finally, SeED depends upon secure hardware to ensure the integrity of the attestation procedure (ROM) and confidentiality of the attestation key and the PRNG seed (MPU).

The issue of data tampering or modification in IoT is a significant concern, as attackers may tamper with data to disrupt system operations, leading to incorrect control decisions. For instance, in an

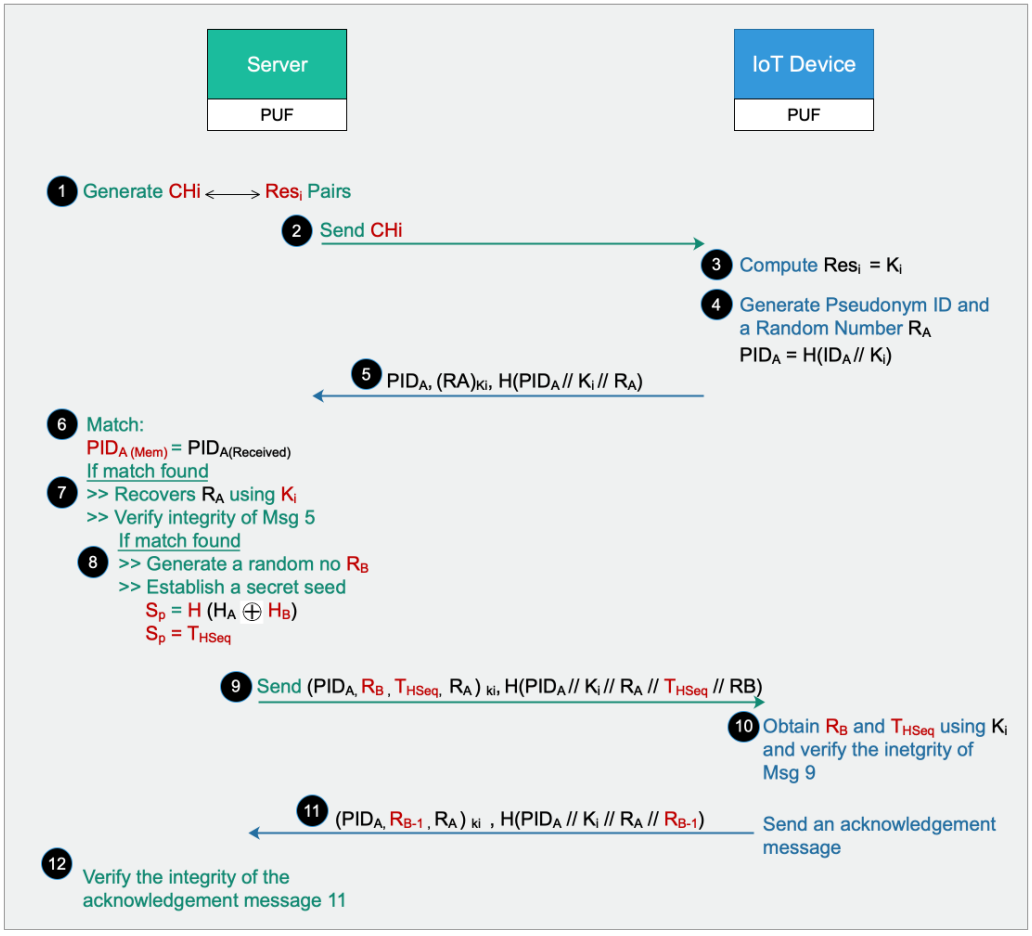


Figure 6: Sequence Sharing Phase - Message Exchange

Industrial Control System (ICS), false temperature sensor data can cause damage to automated control systems or machinery. To address this, existing techniques require IoT devices to compute a cryptographic hash or MAC over their program memory to confirm their integrity. However, this approach may only be appropriate for low-energy consumption applications where data transmission is infrequent. Conventional integrity verification mechanisms can lead to significant energy overheads for high data rate applications like VANETs and ICS process sensors.

To overcome this issue in high data rate IoT systems, the authors in [69] proposed a method based on a random time hopping sequence and random permutation to conceal validation information. This method verifies multiple packets rather than checking one packet simultaneously. In this approach, the server sends several challenges to a PUF and records their responses to detect data tampering. Later, the PUF is installed in an IoT device. Whenever the IoT device needs to communicate with the server securely, it uses a challenge to generate a unique symmetric secret key by the PUF. Hence, the PUF eliminates the requirement of keeping the symmetric key on the device. Instead, the device only stores the challenge bits. An attacker cannot generate the secret key even if the challenge is known [85]. Thus, this technique protects end devices against physical attacks involving node cloning.

The protocol executes in two phases: the sequence sharing phase and the data transfer phase. At the beginning of the sequence sharing phase, the server challenges the PUF and generates the required responses. These steps are illustrated in Figure 6. Following this, the PUF is installed in IoT device IDA, and a challenge is sent to the device by the server. The device stores the challenge in memory and computes the response using PUF. This response includes a secret symmetric key, Ki, used for secure communication between the device and the server. Ki is not saved in the device's memory. The device then encrypts and sends its pseudonymous ID (PIDA) and a random number (RA) using Ki to the server. The server checks the received PIDA with the IDs stored in its memory and cancels the request if it does not find a match. Otherwise, the server recovers RA using Ki and validates the integrity and freshness of the message by constructing and matching the H(PIDA, Ki, RA) with the value sent by the device. If the hashes match, the server generates a random number RB, which is used to establish a secret seed SP. SP generates a secret time hopping sequence T$HSeq$. The server sends T$HSeq$ along with PIDA, RB, and RA encrypted with Ki to the device. The device then decrypts and obtains RB and T$HSeq$ using Ki and validates the message sent by the server. Finally, the device sends an acknowledgment message to the server, followed by the legitimacy check of the acknowledgment message by the server.

To utilize T$HSeq$, both the device and the server keeps track of the number of packets sent and received. T$HSeq$ is a sequence of random numbers, denoted by $\mathcal{T}HSeq = T1, T2, T3, ...Tn$, where each T$i$ has a range between 15 and 25. When the number of data packets the device sends reaches Ti, the device resets the counter P_{Ctr} after inserting validation information in the last packet. The device also includes garbage over and above sensed data in non-validation packets. The maximum tolerable delay for verification is 25, which is set to a maximum of one second for high data rate IoT systems. Furthermore, the proposed method employs random permutations on buffered packets while constructing validation packets, making them statistically unpredictable and practically indistinguishable from non-validation packets, making it impossible for attackers to differentiate between them.

DARPA [70] is an extension of SEDA [64] that aims to detect physical attacks in an IoT network where the adversary can extract confidential data. During such an attack, the victim device is disconnected from the network for some time; this downtime can therefore be used to indicate physical compromise. Hence, researchers in [70] proposed a device attestation mechanism that detects the absence of nodes using a heartbeat protocol. Each device in the IoT network periodically sends a heartbeat containing a timestamp to the rest of the network devices. If its peers do not receive a device's heartbeat within the expected time, it is deemed physically compromised. It requires minimal security hardware - a ROM to store attestation code and an MPU to control access to ROM and RAM storing secret keys and private protocol data. However, as the proposed technique detects the absence of nodes based on their failure

to send timestamped heartbeat messages, it is prone to false positives (for example, as a result of packet loss due to transient network congestion). It also scales poorly, incurring rapidly increasing computation and communication overheads as the network size grows.

Unattended Scalable Attestation of IoT Devices (US-AID) [72] aims to improve the scalability of DARPA [70] while detecting physically compromised devices in large, unattended, and dynamic-topology networks. The proposed technique detects software and physical attacks by employing a key exchange mechanism to combine attestation of network peers and periodic heartbeats with minimal computation and communication costs [86]. However, US-AID does not account for a network environment where a device remains offline for an extended time. Nonetheless, this approach incurs a constant computation and communication cost with increased network size. However, many messages are exchanged between network devices during connect, attest and beat phases. Similarly, each device needs to store and maintain benign and secure device lists and also a list of neighboring devices present during a specific heartbeat interval. Moreover, the network devices are equipped with ROM and MPU to be protected against software attacks and require a loosely synchronized, reliable read-only clock (RROC). Lastly, energy consumption/costs increase linearly with the number of heartbeat intervals, and this cost is over and above the routine operation of the devices.

SCAPI [76], a modified version of DARPA, aims to detect compromised devices by periodically distributing a session key among uncompromised devices. The leader node generates a secret session key for the subsequent period, which network devices authenticate using the old session key. However, SCAPI's security relies on the assumption that physically intrusive attacks cannot capture a device without turning it off for a detectable period, allowing compromised devices to miss the updated session key and be detected. Although SCAPI improves upon DARPA in terms of communication complexity, energy consumption, and runtime, it is ineffective in highly dynamic networks or in situations where nodes have intermittent connectivity, as it relies on other remote attestation protocols like SEDA and SANA for scaling.

Researchers in [71] propose an advanced lightweight hybrid remote attestation scheme for IoT swarms. The method is built upon a single-prover remote attestation system with minimal secure hardware requirements for network devices. The proposed scheme assumes that IoT devices have limited mobility; a heartbeat-based absence detection is used to detect physically compromised nodes. The proposed method accounts for remote malware and DoS attacks that force an IoT device to consume excessive CPU time or network bandwidth. The scheme attests only the code and data segments of the device memory; it also does not consider the possibility of physical attacks involving modification of hardware or software components of network devices. In addition, strong synchronization between network devices is needed to effectively manage attestation timeouts, which may introduce a vulnerability to DoS attacks.

The authors in [73] present a distributed remote attestation scheme (ESDRA) for IoT swarms. It supports peer-to-peer verification thus avoiding a single trusted verifier. It also employs an accusation mechanism to report the compromised nodes. Every prover is attested by its neighbors, thus limiting the required attestation run-time. However, the proposed scheme considers only software attacks including attestation result tampering and bypassing of the attestation protocol. The attacker may achieve the desires objectives by replaying or forging messages, and eavesdropping on the network traffic. The scheme is also susceptible to DoS attacks.

HEALED [74], a recovery and attestation scheme for embedded devices, measures the software state of the nodes by employing Merkle Hash Trees (MHT). It not only detects the malware but also securely restores the device to its original state. The MHT enables the verifier to correctly identify the modified software parts on the attester. However, the proposed technique considers only malware attacks and requires the verifier to keep a copy of the default configuration of the prover nodes. All the devices in a group need to share pairwise symmetric keys; HEALED cannot always guarantee successful

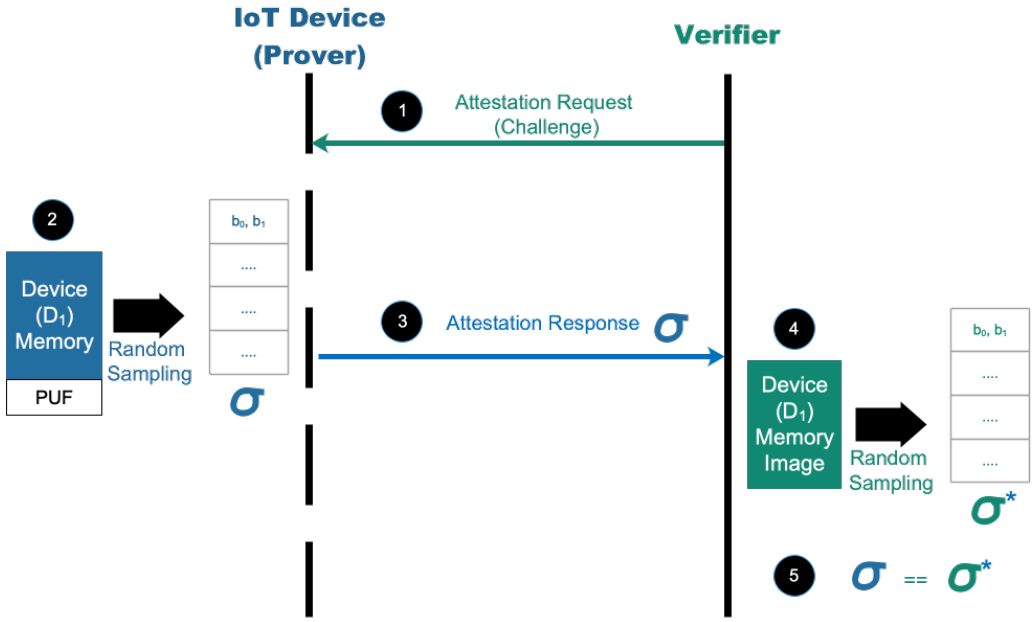


Figure 7: HAtt Protocol

node restoration, and it cannot prevent subsequent compromise of devices.

The authors in [75] introduce RADIS, an attestation scheme designed for distributed IoT systems. They argue that a malicious service can adversely affect the control-flow of other legitimate services, resulting in unauthorized actions. To illustrate this type of threat, consider a critical IoT system where the opening of a security door relies on detecting movement by a security camera and processing the image for identifying authorized individuals by the security controller. The door will unlock only when an individual is positively identified as a legitimate worker. Even if the control-flow of the security door application is unaltered, an attacker can eavesdrop, interrupt, and modify the communication between the security controller and the door or compromise the security controller to produce malicious software executions, leading to an unauthorized control action of the security door (opening of the door). Since a flawed input caused this anomaly, it cannot be identified by applying a control-flow attestation scheme for the services with genuine software. To solve this issue, the authors propose control-flow attestation technique for services offering unique functionality. RADIS can secure an IoT system even if there is a single device with a legitimate control-flow [87].

In [35], researchers propose *HAtt*, a passive lightweight hybrid attestation scheme. As shown in Figure 7, instead of computing the checksum of the complete memory, HAtt uses a randomized approach to attest different parts of an IoT device's memory. It also employs PUFs to safeguard against physical compromise. Furthermore, it ensures the high availability of IoT devices during the execution of security protocols. To preserve the privacy of IoT devices, HAtt uses pseudonym identities for end devices. It detects roving malware with a high probability of success; however, the verifier must store memory images of all the dependent prover nodes, and the verifier must be a trusted party.

Researchers in [77] developed a hybrid remote attestation protocol, *ERASMUS*, which aims to reduce computation time for the provers by letting them self-attest at specific time intervals. The provers then locally store up to m consecutive measurements and hand over a subset k of these measurements to the verifier for verification. This technique helps the verifier detect a mobile adversary between two successive attestation instances. This makes ERASMUS a suitable protocol for highly mobile use cases.

SALAD [78] is a lightweight, secure attestation scheme for highly dynamic networks. Instead of pro-

viding novelty in remote attestation, SALAD aims to offer a lightweight message aggregation scheme for dynamic networks with intermittent connectivity. Message aggregation is done in a distributed manner, whereby the verifier sends a signed challenge to the prover. The prover computes its integrity measurement and shares the challenge request with its neighbors. All of these neighbors then compute and share their responses; subsequently, every node has the aggregated response of the entire network. Finally, the verifier can connect to any device/node in the network and verify the attestation result for the complete network. Importantly, SALAD ensures no duplication in the aggregated attestation results, minimizing storage requirements for resource-constrained devices. Nonetheless, the scheme is limited to measurement aggregation with a traditional challenge-response attestation protocol.

Practical Attestation for Highly Dynamic Swarm Topologies (PADS) is an attestation protocol introduced in [79]. PADS is similar to SALAD because it focuses on collective remote attestation in dynamic networks. Furthermore, like SeED, every prover in PADS performs self-attestation using a non-interactive attestation scheme and then broadcasts the result to the entire network. Subsequently, all the devices/nodes form their view of the network by updating the attestation responses of other devices. Hence, a verifier needs to connect to any device or a subset of devices to access the collective remote attestation result. The proposed scheme is resilient to network topology changes; however, in PADS, all the network devices share a single symmetric encryption key for secure communication, which makes the protocol vulnerable to hardware intrusion attacks.

WISE [80] is a proposed attestation scheme that aims to reduce IoT devices' energy and memory consumption. WISE reduces the complexity of the attestation procedure by verifying the integrity of one subset of the provers at a time. The proposed protocol operates in two phases. In phase 1, the network owner configures the devices with the required firmware and cryptographic material and then groups them into clusters based on their location. In the second phase, the verifier initiates the collective remote attestation protocol over a subset of devices. The first round of attestation is for the complete network in which the attestation request/challenge is propagated through Spanning Tree Protocol (STP). Accordingly, the responses are also aggregated through STP. Subsequently, in later rounds, devices are selected based on their attestation history, the number of compromised devices in their neighborhood, and the time interval between successive attestations. Due to the attestation of different subsets of devices in each round and at varying time intervals, WISE protects against roving malware attacks. Nonetheless, the use of STP for propagation and aggregation of attestation requests and requisite responses makes WISE prone to communication errors in highly dynamic networks.

SlimIoT [81] is proposed as an improvement of SCAPI, which forms clusters of devices and attests each cluster at a predefined epoch (time interval). The attestation epoch is selected considering the time a device may remain offline once physically compromised by an adversary. SlimIoT can detect and precisely identify compromised devices as long as there is at least one healthy device in the network and supports device mobility during the attestation period. SlimIoT employs a one-way key chain (composed of n symmetric keys) to authenticate messages the verifier propagates. It considers a device to be compromised if it is offline during its attestation epoch.

Researchers in [82] introduce an attestation protocol called Efficient Attestation Resilient to Physical Attacks (EAPA), which aims to prevent physical attacks efficiently. EAPA leverages the heartbeat mechanism introduced in DARPA and SCAPI to attest network devices. Based on a secure internal clock, every device securely shares its heartbeat with its immediate neighbor using a secret symmetric key. Neighbors authenticate the message and update the device ID in their respective lists of current devices. Every device maintains two lists for its neighbors: one for present and one for absent devices. During attestation, the verifier requests the nodes, acting as a verifier for their neighbors, for the list of active and absent devices. Like DARPA, EAPA also detects compromised devices based on failure to send a heartbeat message in an expected time interval. The authors claim EAPA improves SCAPI by reducing runtime and network overhead.

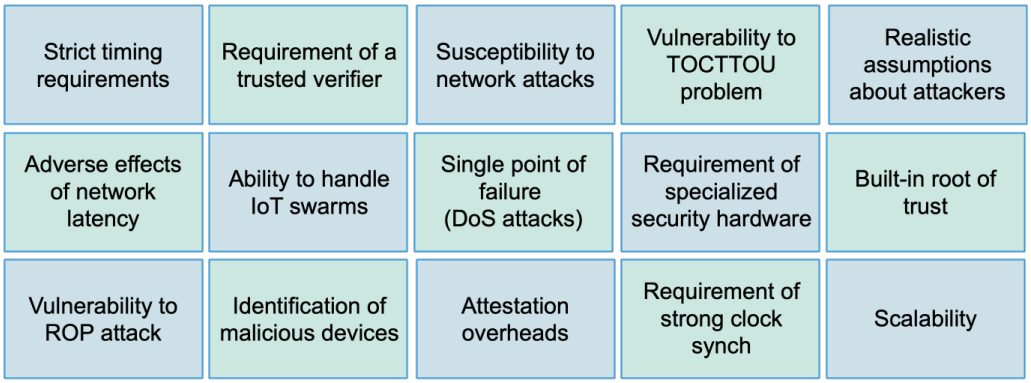


Figure 8: Evaluation Criteria

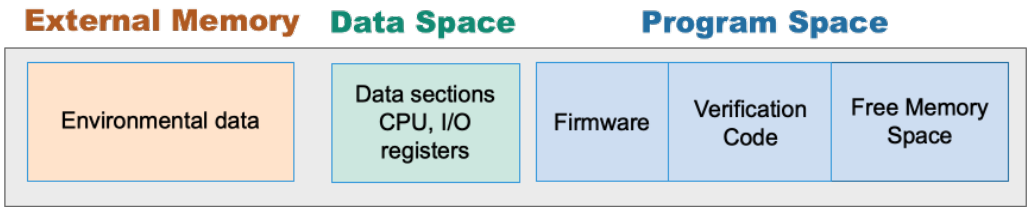


Figure 9: A Typical IoT Device Memory Setup

SARA [83] leverages asynchronous communication between IoT devices to verify the integrity of services rendered by these devices. Moreover, SARA confirms if an IoT device is compromised and the legitimacy of its operations. SARA ensures that every service maintains complete information about its interactions. The historical data is then asynchronously transmitted to other interacting services. The proposed mechanism also endeavors to avoid suspension of routine operation by a device during uninterrupted attestation intervals.

4. Discussion and Gap Analysis

The preceding sections discussed numerous behavioral integrity verification and device attestation schemes classified as software-based, hardware-based, and hybrid techniques that combine software with minimal secure hardware. Subsequently, this section analyses these attestation schemes based on the evaluation criteria shown in Figure 8. For instance, the MUD specification is focused on providing a standardized way for IoT device manufacturers to describe the intended behavior of their devices in terms of network communication protocols and services. The specification does not protect the device against hardware or software modifications. Hence, IoT device manufacturers need to implement secure design and development practices to minimize the risk of hardware or software modifications. This can include measures such as using secure boot processes, implementing hardware and software protections against tampering, and regularly updating the firmware to address security vulnerabilities.

The software-based device verification can be based on a challenge-response or a one-way remote attestation protocol. Accordingly, most software-based techniques typically imply strict timing requirements on the network; the verifier requires a valid prover (end device) response within a permitted time interval/window for it to be considered uncompromised. However, achieving a sufficient level of clock synchronization may not always be feasible in distributed IoT systems [12].

The challenge-response based remote attestation techniques involve using cryptographic challenges

and responses to verify the identity and integrity of IoT devices. This approach can effectively detect compromised devices, requiring the device to prove its identity by correctly responding to a cryptographic challenge. However, this approach can also lead to network congestion, especially when implemented at scale. Each attestation request requires the device to perform a cryptographic calculation, which can be resource-intensive for low-powered IoT devices. If the attestation process is performed too frequently or intensely, it can cause the device to slow down or even crash, which can disrupt production and cause downtime. Additionally, frequent attestation requests can lead to congestion, especially in networks with many IoT devices. Each attestation request adds to the network traffic, which can result in congestion and latency issues. This can be particularly problematic in mission-critical IoT environments, where real-time data processing is essential.

To mitigate the risk of network congestion, IoT device attestation schemes can be optimized to minimize the frequency and intensity of attestation requests. This can be achieved by using adaptive attestation algorithms that adjust the frequency of attestation requests based on the device's behavior and network conditions. Additionally, attestation requests can be scheduled during off-peak hours to reduce the impact on production and network performance.

Moreover, the challenge-response-based remote attestation techniques are susceptible to network attacks, including eavesdropping, message forgery, rainbow, and interference attacks. Some other potential weaknesses include the difference in TOCTTOU, and DoS attacks based on frequent triggering of the attestation protocol by the attacker. Most of the remote attestation schemes may be ineffective if they do not address the issue of TOCTTOU. In this context, an attacker exploiting the time gap between device attestation and its legitimate operation (e.g., sending a sensor reading) can modify the code before sending its scheduled updates to the other network devices. Furthermore, the exclusive execution of the code attestation protocol results in high communications, computation, or memory overheads.

A software-based attestation approach without any secure hardware is a suitable option for resource-constrained devices. However, the security of this approach has been questioned [88] due to the impractical assumptions it is based on [17]. For instance, software-based attestation assumes that while the attestation protocol executes, the adversary is passive and cannot interfere with the process. Additionally, most of these techniques do not accommodate hardware modification, network configuration alterations, or other device metadata changes. Consequently, these schemes only apply to traditional IoT devices without secure hardware.

Most software-based schemes have strict timing constraints, which means they are suitable for one-hop networks with low network latency [71]. Therefore, the behavior of the attestation protocol may fluctuate based on the network size and topology. Due to variations in network latency, the time difference in challenge and reception of requisite response messages can vary substantially. Since it is necessary to measure the computation time of the verification checksum accurately, it may be essential to either severely limit network usage to minimize latency and jitter. In addition, a strict quality-of-service traffic policing mechanism may be enforced to ensure that the attestation protocol packets are delivered with the highest priority. However, due to the challenges involved, the software-based attestation schemes are mostly appropriate for one-hop network settings [17].

Almost, all the software-based attestation methods authenticate the whole device memory contents, including data that should be kept hidden from the verifier. However, it may not be feasible to review the complete memory space since certain sections of the memory could contain confidential information, such as dynamic data or cryptographic keys. One of the primary goals of memory attestation schemes is to ensure that no malicious code has been added to the memory space. This objective can only be achieved by considering a constant memory size. For reference, a typical IoT device memory layout is displayed in Figure 9.

The majority of current remote attestation methods [30, 37, 65, 66] are susceptible to a single point of failure - the verifier node [73]. To address this issue, secure hardware-based or co-processor-based

attestation approaches have been developed that alleviate many of the vulnerabilities of software-based methods. However, hardware-based TEEs, which simplify the deployment and operation of trusted applications across a range of devices and cloud providers, are yet to be widely explored, and they pose an industrial challenge. Intel's SGX has an embedded attestation mechanism for numerous trusted applications. Whereas AMD's SEV is specifically designed to safeguard VMs from untrusted hypervisors in virtualized environments. However, neither the ARM TrustZone scheme nor native RISC-V supports the attestation of software code being executed in a secure environment. Rather, they rely on software tools, e.g., TrustZone-M offers a security primitive, which facilitates the creation of an adequately trusted software-based attestation mechanism. Furthermore, some RISC-V versions provide hybrid extensions supporting root of trust for multiple trusted applications.

Remote attestation is critical to designing trusted computing solutions. However, current solutions vary significantly in terms of security and maturity. Some TEEs developed by leading processor manufacturers have embedded attestation mechanisms, while others still need more hardware or software modules for secure attestation. Purely hardware-based approaches are considered too costly and complex for IoT devices, and are deemed appropriate for general-purpose computing platforms only [64].

Security researchers have developed hybrid approaches that incorporate a root of trust, such as TPM functionality within the prover, to address the weaknesses of software-based remote attestation techniques. The hybrid approaches use minimal security hardware to ensure security of the attestation protocol. The least level of security can be achieved by using ROM or an MPU. SEDA [64], and SANA [67] use minimum hardware to realize collective attestation. However, SEDA primarily focuses on functionality with respect to embedded devices as compared to the security. SEDA considers software attacks only. To achieve high efficiency and scalability all the network nodes participate in the attestation process. Nonetheless, SEDA cannot identify the malicious devices that fail the attestation. It is also ineffective against hardware-level attacks.

In addition, hybrid schemes such as SEDA are impractical for several reasons [71]. Firstly, such techniques do not clearly mention the specifications of hardware security features that may be insufficient for more complex swarm attestation schemes as compare to single-verifier approaches. Secondly, these techniques do not specify any timing requirements for the verifier. Thirdly, there is a question about when and who will trigger the attestation process. Fourthly, most hybrid remote attestation schemes cannot handle dynamic networks with intermittent node connectivity. Finally, schemes like SANA have high computation and attestation overhead due to the use of public-key cryptography, and they are also dependent on trusted third-parties [68].

Most of the techniques discussed above are susceptible to DoS attacks. These attacks can drain a device's battery or interfere with its intended function by flooding it with fake attestation requests. To ensure data integrity, most networks rely on message MAC. However, many MACs are considered infeasible for high data rate IoT systems as they are computationally expensive with a complexity of $O(nlogn)$ [69]. Similarly, PKI-based approaches [89] are too computationally complex for many low-power IoT devices [90, 91]. Besides, physical and side-channel attacks can be prevented by masking power consumption [92], using dedicated hardware [93], or employing dynamic software diversity [94]. Hence, these solutions are not feasible for resource-constraint devices due to increased costs and high-end system requirements.

Collective remote attestation is an attractive proposition for large IoT swarms that cover a wide geographical area and contain thousands of devices. Such techniques may be easily adapted for static networks. However, network topology discovery, key management, and routing becomes more challenging in mobile networks with heterogeneous devices [71]. Swarm attestation schemes such as SANA [67], SEDA and SeED [68] also require strong clock synchronization among all the devices, which is difficult to achieve in distributed networks.

Runtime state validation is a crucial aspect of device validation that protects against runtime attacks.

Many existing protocols for remote attestation, including software-based and hybrid approaches, only verify the integrity of the software when it is initially uploaded, making them vulnerable to runtime attacks. Runtime attacks may target data memory of a device without affecting its program memory [75], by utilizing ROP and Data-Oriented Programming [95, 96]. The two types of runtime attacks are control-flow and non-control data vulnerabilities [97, 98]. In a runtime attack, an adversary manipulates the dynamic parts of an IoT device's memory, which makes it challenging to attest the dynamic data.

Dynamic remote attestation techniques such as ReDAS [99] aim to verify the dynamic integrity of IoT devices at runtime. ReDAS attests the dynamic data based on structural and global data integrity. Structural integrity involves stack constraints and return data constraints that the application binary must satisfy at runtime. In comparison, global data integrity requires all data variables and the relationships between them to be satisfied. However, existing dynamic integrity attestation schemes are deemed ineffective [9]. On the contrary, Control-flow integrity (CFI) prevents runtime attacks that modify the control-flow of software to cause unwanted program behavior [100]. Various CFI schemes have been introduced for ensuring control-flow integrity of applications [87, 101]. CFI schemes complement static remote attestation techniques in that they attest runtime behavior, which is orthogonal to the attestation of software binaries. However, the impact of CFI schemes on execution time makes them unsuitable for real-time IoT systems [29, 47].

It is essential to secure resource-constrained embedded devices connected to the Internet as they play a crucial role in many applications including Industry 4.0. Most of the IoT devices are Class-1 devices with just 10kB of RAM and 100kB of flash memory [102]. Despite their ubiquitous use, Class-1 IoT devices may be considered insecure due to their limited hardware features and inability to support remote attestation schemes.

According to [29], none of the existing hardware-based solutions using trusted computing, hybrid techniques utilizing secure hardware extensions or software-based schemes such as SWATT [30], which demand strict timing assumptions, is suitable for legacy real-time embedded devices. In WSNs deployed for IoT or Cyber Physical Systems (CPS), there may be substantial variations in transmission times, which conflicts with the strict timing requirements for software-based attestation [17]. Due to their strict timing requirements and the potential vulnerability to compressing attacks [15], software-based attestation schemes are considered insecure or impractical; however, they remain the only appropriate solution for resource-constrained legacy devices. Hence, [29] proposed RealSWATT, a software-based remote attestation scheme that overcomes the challenges of TOCTTOU problem and strict timing requirements (as in SWATT [30]) by implementing a continuous attestation mechanism. In this case, the verifier can safely assume that the prover has been running the attestation task continuously and there has been no gap in the attestation process. However, RealSWATT assumes that the verifier and the gateway devices are trusted and cannot be compromised, and the adversary cannot modify the hardware of the end devices. In addition, RealSWATT assumes that attackers can compromise embedded devices using software attacks only.

In summary, the five primary concerns regarding current device attestation schemes are:

a. Network-level data integrity protection techniques often have high energy and computational requirements, making them unsuitable for resource-constrained devices.

b. The verifier is assumed to be a trusted party that cannot be compromised by the attacker.

c. The possibility of physical attacks is usually not considered in current methods [69].

d. Perfect clock synchronization is required across all network devices, which can be difficult to achieve.

e. Security is a low priority for manufacturers, especially for low-cost Class-1 IoT devices [10, 103].

5. Future Challenges

a. **A Decentralized Device Integrity Check Scheme:** Most of the existing software-based attestation schemes rely on a trusted base station or a cluster head to perform the end device memory attestation procedure. Although the base station can have hardware protection in the form of a TPM, and it is assumed that it cannot be compromised by an attacker, a single verifier in the context of a many-to-one attestation environment is likely to create both a bottleneck and a single point of failure. Hence, there is a requirement to develop a decentralized attestation scheme that should avoid network attacks such as rainbow, interference, DoS, and clock variations.

b. **Designing an Economical Attestation Protocol:** In addition to the security requirements, an ideal attestation scheme should not require the verifier to store complete memory images of all the connected nodes. Moreover, the data integrity verification mechanism should not have high computational or communication complexity and energy consumption. It should be suitable for low-end resource-constraint IoT devices.

c. **Key Management:** Key management encompasses the mechanisms used to generate, distribute, refresh and revoke cryptographic keys. It can be performed either in a centralized or decentralized manner. Most existing device/code attestation schemes employ centralized key management protocols. However, centralized methods can become a severe bottleneck in the context of large networks. In contrast, decentralized schemes are more fault-tolerant and scalable. A majority of current attestation schemes use symmetric keys only, where only a single cryptographic key is employed, with security maintained using a TEE. However, such practices are vulnerable to security breaches if the adversary launches physical or hardware intrusion attacks. The alternative approach, based on public-key infrastructure, implies increased computational complexity and cost compared to symmetric-key cryptography. Hence, there is a need to develop a secure and efficient decentralized key management protocol that offers a fair balance of security and usability for resource-constrained devices.

d. **TOCTTOU:** Time of Check to Time of Use (TOCTTOU) is a significant problem for attestation protocols. In this context, it is difficult to determine whether a prover was malicious before the attestation time and whether its integrity will remain intact immediately after attestation. Accordingly, an adversary can evade the attestation process by using the TOCTTOU gap. Researchers in [104] note that, with the exception of the ERASMUS and MTRA schemes, none of the collective remote attestation protocols even consider TOCTTOU as an issue. Recently [105] studied the TOCTTOU problem and proposed a solution for low-end devices; however, clearly, much more research is required to overcome this problem in all IoT applications.

e. **Developing Software-based Secure Attestation Schemes:** Most of the existing swarm attestation schemes employ specialized hardware or software and minimal hardware architecture to secure the attestation protocol. However, the complexity and cost incurred by pure hardware and hybrid attestation techniques make security less desirable than serviceability. Therefore, future research may focus on practical software-only solutions, even for existing deployments, by balancing the consideration of security and usability. There exists a clear need to develop a secure and economical integrity verification technique for low-end IoT devices. The challenge here is protection against an adversary which can potentially physically compromise a device and tamper with its hardware. Moreover, an ideal remote attestation scheme should not rely on any trusted entity for endorsement of verification.

6. A Way Forward

Most hardware-based attestation approaches are built on top of the TPM, which reports the current software and hardware configuration details concerning an IoT device to the verifier/base station. The verifier then determines if an end device is compromised or not based on the received state response. The Trusted Computing Group [106] specifies these TPM modules as the trust anchors, which work in close synchronization with the verifier to ensure that memory is not tampered with. Although these TPMs are also built-in to many modern general-purpose CPUs and offer functionalities like secure generation and storage of cryptographic material, memory attestation, etc., it remains difficult to equip resource-constrained Class-1 IoT devices with security hardware.

Hence, due to the resource-constrained nature of the sensor networks, it becomes increasingly important to protect device memory using secure and economical protocols involving minimal security hardware components. Accordingly, keeping in view the IoT threat environment and also the advantages and disadvantages of the discussed attestation approaches, we identify some essential requirements of an ideal attestation protocol.

6.1. Requirements of an Ideal Device Attestation Protocol

Developing an ideal attestation protocol meeting all the undermentioned requirements may not be practically feasible. Yet, an ideal IoT device attestation scheme must qualify most of these prerequisites.

a. **Scalability:** If a centralized verifier (e.g., a cloud server) attests one device at a time, the attestation of a large IoT swarm will take a long time. Hence, an attestation protocol should scale well for a large network of IoT devices.

b. **Identify Bad Devices:** The attestation scheme should detect and identify malicious nodes in the network so that corrective action can be taken for the compromised devices.

c. **Public Verifiability:** Any party should be able to verify the collective attestation response in a public key domain.

d. **Prover Integrity Window:** The prover device's integrity should not be limited to the time window between the generation of the challenge and the receipt of the response by the verifier.

e. **Heterogeneous Network:** The attestation protocol should account for a diversity of IoT devices comprising different hardware and software configurations.

f. **Prevention of DoS Attacks:** The device verification mechanism should be resilient against DoS attacks launched through unauthenticated/fake challenges. This can cause battery drain or prevent the IoT device from performing routine operations.

g. **Low Overheads:** An ideal attestation protocol should have low computation, communication, and energy costs.

h. **Integrity of Attestation Process:** The process of measuring a device's attributes and verifying its integrity should be immutable.

i. **Confidentiality of Cryptographic Primitives on a Device:** Access to cryptographic primitives stored on a device and executable code must be protected through minimal hardware assistance such as MPUs and ROM.

j. **Unpredictable Attestation Window:** The attestation process should occur at random time intervals so that an adversary cannot predict and eavesdrop on an attestation response.

k. **Size of Attestation Response:** The attestation response transmitted over the network by an IoT device should not be more than 20 bytes since the energy consumed to send 1 bit by an IoT device is equivalent to executing 800-1000 instructions [107].

l. **High Data Rate Systems:** The proposed technique should cater to the resource-constrained (especially energy-limited) IoT applications/devices that need to send data/sensor readings at high rates.

m. **Security of Session Keys:** Secret keys to be used for secure communication between a device and a server should not be saved in device memory. Instead, a PUF can be used to generate a key at runtime.

n. **No Trusted Party:** The attestation process should be completely decentralized without the need for a trusted party.

o. **Security Evaluation:** The security of an attestation scheme should be evaluated against memory copy attacks, replay, pre-computation, and parallel computation attacks. It should also avoid collusion attacks and provide an increased detection rate of modified memory. Moreover, it should provide message authentication and integrity, minimize the TOCTTOU gap, and avoid network attacks such as rainbow, interference, message modification, and impersonation attacks.

p. **Performance Efficiency:** The factors that are likely to reflect the performance efficiency of the attestation protocol may include a tangible start of code attestation procedure, implicit code attestation, efficient and reliable seed generation, the minimal execution time for checksum computation, minimal transmission overhead, and minimal energy consumption.

In addition, while designing and developing an attestation scheme, it is essential to consider the attacker's capabilities - in particular, the question of whether the adversary can only utilize software-based remote attacks or can physically access and compromise the IoT devices as well. Accordingly, an attacker can physically tamper a device using invasive, or semi-invasive techniques. Using invasive approach [108] the attacker can access a device's internal data (e.g., via access to a JTAG port, ROM or flash memory). This attack requires specialized equipment and the opportunity to access (or potentially even remove) the physical device. Semi-invasive attacks [109] also require direct access (albeit typically not via an electronic interface) to the hardware and include methods such as thermal, UV or other optical imaging techniques. The attacker needs physical access to the victim device for an extended period varying from a few hours to weeks to execute invasive and semi-invasive attacks [110, 108]. There is another technique in which an adversary can extract secure device information through non-invasive side-channel attacks [111]. Hence, we may conclude that a secure IoT device attestation protocol must consider a strong adversary with the ability to:

a. Deploy software attacks, including remote malware attacks.

b. Physically compromise end devices and launch hardware attacks, including modification of hardware components and accessing cryptographic primitives.

c. Launch DoS attacks on challenge-response based protocols.

d. Delay network packets and cause network interruptions, thus causing DoS for strongly synchronized network devices that require a response in highly restricted time windows.

e. Block attestation response packets that indicate a malicious state (hence attestation response for a malicious or a benign device should be indistinguishable);

However, we may also reasonably assume that:

a. A remote software attacker is not capable of modifying the contents of ROM or an MPU.

b. The attestation code is unforgeable.

c. The execution of the attestation protocol cannot be interrupted.

d. IoT devices have limited resources, including small memory and processing power.

e. Heterogeneous provers will generate different attestation responses.

Based on our review of the state-of-the-art device attestation techniques - especially SIMPLE [39] and D2Gen [40] - a simplistic hybrid solution can be developed which can maintain security under these assumptions. Such a technique can leverage the software-based memory isolation mechanism from SIMPLE to ensure the integrity of application control-flow. Correspondingly, the methodology of D2Gen would confer strong protection against software modifications, hardware misuse/tampering, and changes to ambient conditions of IoT devices. Moreover, using blockchain technology to store default/base genome values for verification at runtime would eliminate the threat of single verifier failure, related trust problems, and communications/energy overheads.

In addition, IETF vide internet-draft titled "Discovering and Retrieving Software Transparency and Vulnerability Information" proposed a standard method for discovering and accessing Software Bills of Materials (SBOM) information for software applications [112]. SBOMs are descriptions of what software, including versioning and dependencies, a device contains. Subject internet-draft helps answer two fundamental questions about a particular system/device: Is this system vulnerable to a specific vulnerability? Which devices in a particular environment contain vulnerabilities that require some action? The proposed mechanism can enable a network management tool to discover what software components a system comprises, and it can also determine known vulnerabilities by searching a national vulnerability database. A detailed vulnerability report can be generated based on retrieved information about software components and related vulnerabilities.

The proposed approach can be applied to a network-layer management system retrieving information from an IoT device using HTTP, CoAP or other interfaces. If the IoT device does not have an appropriate retrieval interface, the data can be acquired from the manufacturer using interfaces such as MUD. Similarly, an application-layer management system can leverage the proposed technique to retrieve vulnerability or SBOM information for an application server. However, certain limitations necessitate requisite security measures. E.g., SBOM provides details about the software components of a system. An attacker can exploit the system through known vulnerabilities if he acquires the same information. Hence, if the information about software components resides on the device itself, it must not provide unrestricted access to the well-known URL by default. Moreover, if the SBOM information is acquired from a server, then there should be a strong authentication mechanism between authorized clients and the server. Nonetheless, SBOM information can be integrated into an existing technique to enhance the device attestation mechanism.

7. Conclusions

In this study, we have presented a detailed survey of active and some passive IoT device attestation approaches. These techniques include behavioral integrity verification, software and hardware-based, and hybrid approaches. However, there are some baseline differences between these schemes. For instance, software-based attestation protocols are not safe against hardware attackers that physically compromise a device. Similarly, co-processor-based or hardware-only solutions are considered to be complex and very costly for use with low-end IoT applications. Moreover, the hybrid approaches are

feasible but also have some intrinsic vulnerabilities and limitations, including the TOCTTOU problem and the requirement of strict clock synchronization between network devices. However, based on a thorough gap analysis, we present a possible way forward to help develop a secure and efficient IoT device attestation protocol.

Acknowledgments

The authors acknowledge the support of Food Agility CRC Ltd, funded under the Commonwealth Government CRC Program. The CRC Program supports industry-led collaborations between industry, researchers and the community. The financial and in-kind support of Robert Bosch (Australia) Pty Ltd & Robert Bosch GmbH in completing this work is also acknowledged.

References

[1] M. Siddiqi, V. Sivaraman, S. Jha, Timestamp integrity in wearable healthcare devices, in: 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2016, pp. 1–6. doi:10.1109/ANTS.2016.7947783.

[2] E. Handschin, F. Schweppe, J. Kohlas, A. Fiechter, Bad data analysis for power system state estimation, IEEE Transactions on Power Apparatus and Systems 94 (2) (1975) 329–337. doi:10.1109/T-PAS.1975.31858.

[3] M. Esmalifalak, H. Nguyen, R. Zheng, Z. Han, Stealth false data injection using independent component analysis in smart grid, in: Proceedings of the IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 244–248. doi:10.1109/SmartGridComm.2011.6102326.

[4] A. Giani, E. Bitar, M. Garcia, M. McQueen, P. Khargonekar, K. Poolla, Smart grid data integrity attacks: characterizations and countermeasures, in: Proceedings of the IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 232–237. doi:10.1109/SmartGridComm.2011.6102324.

[5] M. H. Chinaei, H. Habibi Gharakheili, V. Sivaraman, Optimal witnessing of healthcare iot data using blockchain logging contract, IEEE Internet of Things Journal 8 (12) (2021) 10117–10130. doi:10.1109/JIOT.2021.3051433.

[6] B. Schneier, Schneier on security, John Wiley & Sons, 2009.

[7] F. Ebbers, A large-scale analysis of iot firmware version distribution in the wild (2020). doi:10.48550/ARXIV.2010.11026.
URL https://arxiv.org/abs/2010.11026

[8] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices, IEEE Internet of Things Journal 6 (5) (2019) 8182–8201. doi:10.1109/JIOT.2019.2935189.

[9] B. Kuang, A. Fu, W. Susilo, S. Yu, Y. Gao, A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects, Computers & Security 112 (2022) 102498. doi:https://doi.org/10.1016/j.cose.2021.102498.
URL https://www.sciencedirect.com/science/article/pii/S0167404821003229

[10] A. Francillon, C. Castelluccia, Code injection attacks on harvard-architecture devices, in: Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 15–26. doi:10.1145/1455770.1455775.
URL https://doi.org/10.1145/1455770.1455775

[11] R. Anderson, M. Kuhn, Tamper resistance: A cautionary note, in: Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2, WOEC'96, USENIX Association, USA, 1996, p. 1.

[12] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, S. Ranise, Collective remote attestation at the internet of things scale: State-of-the-art and future challenges, IEEE Communications Surveys Tutorials 22 (4) (2020) 2447–2461. doi:10.1109/COMST.2020.3008879.

[13] I. Sfyrakis, T. Gross, A survey on hardware approaches for remote attestation in network infrastructures, arXiv preprint arXiv:2005.12453 (2020).

[14] S. F. J. J. Ankergård, E. Dushku, N. Dragoni, State-of-the-art software-based remote attestation: Opportunities and open issues for internet of things, Sensors 21 (5) (2021). doi:10.3390/s21051598.
URL https://www.mdpi.com/1424-8220/21/5/1598

[15] C. Castelluccia, A. Francillon, D. Perito, C. Soriente, On the difficulty of software-based attestation of embedded devices, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 400–409. doi:10.1145/1653662.1653711.
URL https://doi.org/10.1145/1653662.1653711

[16] J. Ménétrey, C. Göttel, M. Pasin, P. Felber, V. Schiavoni, An exploratory study of attestation mechanisms for trusted execution environments, arXiv preprint arXiv:2204.06790 (2022).

[17] F. Armknecht, A.-R. Sadeghi, S. Schulz, C. Wachsmann, A security framework for the analysis and design of software attestation, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 1–12. doi:10.1145/2508859.2516650.
URL https://doi.org/10.1145/2508859.2516650

[18] R. V. Steiner, E. Lupu, Attestation in wireless sensor networks: A survey, ACM Comput. Surv. 49 (3) (sep 2016). doi:10.1145/2988546.
URL https://doi.org/10.1145/2988546

[19] Y. Otoum, A. Nayak, On securing iot from deep learning perspective, in: 2020 IEEE Symposium on Computers and Communications (ISCC), 2020, pp. 1–7. doi:10.1109/ISCC50000.2020.9219671.

[20] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, M. Guizani, A survey of machine and deep learning methods for internet of things (iot) security, IEEE Communications Surveys & Tutorials 22 (3) (2020) 1646–1685. doi:10.1109/COMST.2020.2988293.

[21] N. Koroniotis, N. Moustafa, E. Sitnikova, A new network forensic framework based on deep learning for internet of things networks: A particle deep framework, Future Generation Computer Systems 110 (2020) 91–106. doi:https://doi.org/10.1016/j.future.2020.03.042.
URL https://www.sciencedirect.com/science/article/pii/S0167739X19325105

[22] H. Jagruthi, C. Kavitha, Analysis and evaluation of machine learning classifiers for iot attack dataset, in: J. I.-Z. Chen, H. Wang, K.-L. Du, V. Suma (Eds.), Machine Learning and Autonomous Systems, Springer Nature Singapore, Singapore, 2022, pp. 471–482.

[23] L. Eliot, R. Dan, D. Ralph, Manufacturer Usage Description Specification, RFC 8520, IETF (Mar 2019).
URL https://www.rfc-editor.org/rfc/rfc8520.html

[24] P. Krishnan, K. Jain, R. Buyya, P. Vijayakumar, A. Nayyar, M. Bilal, H. Song, Mud-based behavioral profiling security framework for software-defined iot networks, IEEE Internet of Things Journal 9 (9) (2022) 6611–6622. doi:10.1109/JIOT.2021.3113577.

[25] P. Krishnan, K. Jain, K. Achuthan, R. Buyya, Software-defined security-by-contract for blockchain-enabled mud-aware industrial iot edge networks, IEEE Transactions on Industrial Informatics 18 (10) (2022) 7068–7076. doi:10.1109/TII.2021.3084341.

[26] L. Morgese Zangrandi, T. Van Ede, T. Booij, S. Sciancalepore, L. Allodi, A. Continella, Stepping out of the mud: Contextual threat information for iot devices with manufacturer-provided behavior profiles, in: Proceedings of the 38th Annual Computer Security Applications Conference, ACSAC '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 467–480. doi:10.1145/3564625.3564644.
URL https://doi.org/10.1145/3564625.3564644

[27] S. Datta, A. Bhattacharya, R. Rana, U. Venkanna, idam: A distributed mud framework for mitigation of volumetric attacks in iot networks, in: 2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2022, pp. 326–331. doi:10.1109/CSNDSP54353.2022.9908058.

[28] Z. Sun, B. Feng, L. Lu, S. Jha, Oat: Attesting operation integrity of embedded devices, in: Proceedings of the IEEE Symposium on Security and Privacy (SP), 2020, pp. 1433–1449. doi:10.1109/SP40000.2020.00042.

[29] S. Surminski, C. Niesler, F. Brasser, L. Davi, A.-R. Sadeghi, Realswatt: Remote software-based attestation for embedded devices under realtime constraints, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2890–2905. doi:10.1145/3460120.3484788.
URL https://doi.org/10.1145/3460120.3484788

[30] A. Seshadri, A. Perrig, L. van Doorn, P. Khosla, Swatt: software-based attestation for embedded devices, in: Proceedings of the IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004, 2004, pp. 272–282. doi:10.1109/SECPRI.2004.1301329.

[31] P. Oechslin, Making a faster cryptanalytic time-memory trade-off, in: Proceedings of the Annual International Cryptology Conference, Springer, 2003, pp. 617–630.

[32] J. Newsome, E. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis amp; defenses, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004, 2004, pp. 259–268. doi:10.1109/IPSN.2004.239019.

[33] I. Adler, S. Oren, S. M. Ross, The coupon-collector's problem revisited, Journal of Applied Probability 40 (2) (2003) 513–518. doi:10.1239/jap/1053003560.

[34] I. Makhdoom, M. Afzal, I. Rashid, A novel code attestation scheme against sybil attack in wireless sensor networks, in: Proceedings of the National Software Engineering Conference, 2014, pp. 1–6. doi:10.1109/NSEC.2014.6998232.

[35] M. N. Aman, M. H. Basheer, S. Dash, J. W. Wong, J. Xu, H. W. Lim, B. Sikdar, Hatt: Hybrid remote attestation for the internet of things with high availability, IEEE Internet of Things Journal 7 (8) (2020) 7220–7233. doi:10.1109/JIOT.2020.2983655.

[36] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, P. Khosla, Scuba: Secure code update by attestation in sensor networks, in: Proceedings of the 5th ACM Workshop on Wireless Security, WiSe '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 85–94. doi:10.1145/1161289.1161306.
URL https://doi.org/10.1145/1161289.1161306

[37] Y. Yang, X. Wang, S. Zhu, G. Cao, Distributed software-based attestation for node compromise detection in sensor networks, in: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), 2007, pp. 219–230. doi:10.1109/SRDS.2007.31.

[38] K. Song, D. Seo, H. Park, H. Lee, A. Perrig, Omap: One-way memory attestation protocol for smart meters, in: Proceedings of the IEEE 9th International Symposium on Parallel and Distributed Processing with Applications Workshops, 2011, pp. 111–118. doi:10.1109/ISPAW.2011.37.

[39] M. Ammar, B. Crispo, G. Tsudik, Simple: A remote attestation approach for resource-constrained iot devices, in: Proceedings of the ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS), 2020, pp. 247–258. doi:10.1109/ICCPS48487.2020.00036.

[40] I. Makhdoom, K. Hayawi, M. Kaosar, S. S. Mathew, P.-H. Ho, D2gen: A decentralized device genome based integrity verification mechanism for collaborative intrusion detection systems, IEEE Access 9 (2021) 137260–137280. doi:10.1109/ACCESS.2021.3117938.

[41] I.-R. Chen, Y. Wang, Reliability analysis of wireless sensor networks with distributed code attestation, IEEE Communications Letters 16 (10) (2012) 1640–1643. doi:10.1109/LCOMM.2012.091212.121454.

[42] M. Bettayeb, Q. Nasir, M. A. Talib, Firmware update attacks and security for iot devices: Survey, in: Proceedings of the ArabWIC 6th Annual International Conference Research Track, ArabWIC 2019, Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3333165.3333169. URL https://doi.org/10.1145/3333165.3333169

[43] C. Herder, M.-D. Yu, F. Koushanfar, S. Devadas, Physical unclonable functions and applications: A tutorial, Proceedings of the IEEE 102 (8) (2014) 1126–1141. doi:10.1109/JPROC.2014.2320516.

[44] S. Dhakal, F. Jaafar, P. Zavarsky, Private blockchain network for iot device firmware integrity verification and update, in: Proceedings of the IEEE 19th International Symposium on High Assurance Systems Engineering (HASE), 2019, pp. 164–170. doi:10.1109/HASE.2019.00033.

[45] W. Daniels, D. Hughes, M. Ammar, B. Crispo, N. Matthys, W. Joosen, Sμv - the security microvisor: A virtualisation-based security middleware for the internet of things, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track, Middleware '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 36–42. doi:10.1145/3154448.3154454. URL https://doi.org/10.1145/3154448.3154454

[46] M. Abadi, M. Budiu, U. Erlingsson, J. Ligatti, Control-flow integrity principles, implementations, and applications, ACM Trans. Inf. Syst. Secur. 13 (1) (nov 2009). doi:10.1145/1609956.1609960. URL https://doi.org/10.1145/1609956.1609960

[47] S. Das, W. Zhang, Y. Liu, A fine-grained control flow integrity approach against runtime memory attacks for embedded systems, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24 (11) (2016) 3193–3207. doi:10.1109/TVLSI.2016.2548561.

[48] I. Makhdoom, F. Tofigh, M. Abolhasan, A method of electronic device integrity check based on device digital genome (d2igen) (Jan. 2020). URL http://pericles.ipaustralia.gov.au/ols/auspat/applicationDetails.do?applicationNo=2018204784

[49] K. Cheang, C. Rasmussen, D. Lee, D. W. Kohlbrenner, K. Asanovic, S. A. Seshia, Verifying risc-v physical memory protection, in: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) Workshop on Secure RISC-V Architecture Design, 2020.

[50] V. Costan, S. Devadas, Intel sgx explained, Cryptology ePrint Archive (2016).

[51] S. Pinto, N. Santos, Demystifying arm trustzone: A comprehensive survey, ACM computing surveys (CSUR) 51 (6) (2019) 1–36.

[52] Secure encrypted virtualization api, Tech. Rep. 55766, Advanced Micro Devices, Inc. (Apr. 2020).

[53] N. L. Petroni Jr, T. Fraser, J. Molina, W. A. Arbaugh, Copilot-a coprocessor-based kernel runtime integrity monitor., in: Proceedings of the USENIX security symposium, San Diego, USA, 2004, pp. 179–194.

[54] R. Sailer, X. Zhang, T. Jaeger, L. Van Doorn, Design and implementation of a tcg-based integrity measurement architecture., in: Proceedings of the USENIX Security symposium, Vol. 13, 2004, pp. 223–238.

[55] W. Li, H. Li, H. Chen, Y. Xia, Adattester: Secure online mobile advertisement attestation using trustzone, in: Proceedings of the 13th annual international conference on mobile systems, applications, and services, 2015, pp. 75–88.

[56] S. Zhao, Q. Zhang, Y. Qin, W. Feng, D. Feng, Sectee: A software-based approach to secure enclave architecture using tee, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1723–1740.

[57] J. Ahn, I.-G. Lee, M. Kim, Design and implementation of hardware-based remote attestation for a secure internet of things, Wireless personal communications 114 (1) (2020) 295–327.

[58] C. Shepherd, R. N. Akram, 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, 2017, pp. 1–10.

[59] K. David, Protecting vm register state with sev-es - amd, Tech. rep., Advanced Micro Devices (2017). URL https://www.amd.com/system/files/TechDocs/Protecting%20VM%20Register%20State%20with%20SEV-ES.pdf

[60] V. Costan, I. Lebedev, S. Devadas, Sanctum: Minimal hardware extensions for strong software isolation, in: 25th USENIX Security Symposium (USENIX Security 16), 2016, pp. 857–874.

[61] S. Weiser, M. Werner, F. Brasser, M. Malenko, S. Mangard, A.-R. Sadeghi, Timber-v: Tag-isolated memory bringing fine-grained enclaves to risc-v, in: 26th Annual Network & Distributed System Security Symposium (NDSS), 2019.

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

[63] C. Shepherd, K. Markantonakis, G.-A. Jaloyan, Lira-v: Lightweight remote attestation for constrained risc-v devices, in: 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 221–227. doi:10.1109/SPW53761.2021.00036.

[64] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, C. Wachsmann, Seda: Scalable embedded device attestation, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 964–975. doi:10.1145/2810103.2813670.
URL https://doi.org/10.1145/2810103.2813670

[65] K. El Defrawy, G. Tsudik, A. Francillon, D. Perito, Smart: Secure and minimal architecture for (establishing dynamic) root of trust., in: Proceedings of the Network and Distributed System Security Symposium, Vol. 12, 2012, pp. 1–15.

[66] P. Koeberl, S. Schulz, A.-R. Sadeghi, V. Varadharajan, Trustlite: A security architecture for tiny embedded devices, in: Proceedings of the 9th European Conference on Computer Systems, EuroSys ́14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 1–14. doi:10.1145/2592798.2592824.
URL https://doi.org/10.1145/2592798.2592824

[67] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, M. Schunter, Sana: Secure and scalable aggregate network attestation, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 731–742. doi:10.1145/2976749.2978335.
URL https://doi.org/10.1145/2976749.2978335

[68] A. Ibrahim, A.-R. Sadeghi, S. Zeitouni, Seed: Secure non-interactive attestation for embedded devices, in: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 64–74. doi:10.1145/3098243.3098260.
URL https://doi.org/10.1145/3098243.3098260

[69] M. N. Aman, B. Sikdar, K. C. Chua, A. Ali, Low power data integrity in iot systems, IEEE Internet of Things Journal 5 (4) (2018) 3102–3113. doi:10.1109/JIOT.2018.2833206.

[70] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, S. Zeitouni, Darpa: Device attestation resilient to physical attacks, in: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 171–182. doi:10.1145/2939918.2939938.
URL https://doi.org/10.1145/2939918.2939938

[71] X. Carpent, K. ElDefrawy, N. Rattanavipanon, G. Tsudik, Lightweight swarm attestation: A tale of two lisa-s, in: Proceedings of the ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 86–100. doi:10.1145/3052973.3053010.
URL https://doi.org/10.1145/3052973.3053010

[72] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, Us-aid: Unattended scalable attestation of iot devices, in: Proceedings of the IEEE 37th Symposium on Reliable Distributed Systems (SRDS), 2018, pp. 21–30. doi:10.1109/SRDS.2018.00013.

[73] B. Kuang, A. Fu, S. Yu, G. Yang, M. Su, Y. Zhang, Esdra: An efficient and secure distributed remote attestation scheme for iot swarms, IEEE Internet of Things Journal 6 (5) (2019) 8372–8383. doi:10.1109/JIOT.2019.2917223.

[74] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, Healed: Healing & attestation for low-end embedded devices, in: I. Goldberg, T. Moore (Eds.), Proceedings of the Financial Cryptography and Data Security, Springer International Publishing, Cham, 2019, pp. 627–645.

[75] M. Conti, E. Dushku, L. V. Mancini, Radis: Remote attestation of distributed iot services, in: Proceedings of the 6th International Conference on Software Defined Systems (SDS), 2019, pp. 25–32. doi:10.1109/SDS.2019.8768670.

[76] F. Kohnhäuser, N. Büscher, S. Gabmeyer, S. Katzenbeisser, Scapi: A scalable attestation protocol to detect software and physical attacks, in: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 75–86. doi:10.1145/3098243.3098255.
URL https://doi.org/10.1145/3098243.3098255

[77] X. Carpent, G. Tsudik, N. Rattanavipanon, Erasmus: Efficient remote attestation via self-measurement for unattended settings, in: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018, pp. 1191–1194. doi:10.23919/DATE.2018.8342195.

[78] F. Kohnhäuser, N. Büscher, S. Katzenbeisser, Salad: Secure and lightweight attestation of highly dynamic and disruptive networks, in: Proceedings of the Asia Conference on Computer and Communications Security, ASIACCS '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 329–342. doi:10.1145/3196494.3196544.
URL https://doi.org/10.1145/3196494.3196544

[79] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, S. Ranise, Pads: Practical attestation for highly dynamic swarm topologies, in: Proceedings of the International Workshop on Secure Internet of Things (SIoT), Barcelona, Spain, 2018, pp. 18–27. doi:10.1109/SIoT.2018.00009.

[80] M. Ammar, M. Washha, B. Crispo, Wise: Lightweight intelligent swarm attestation scheme for iot (the verifier's perspective), in: Proceedings of the 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Limassol, Cyprus, 2018, pp. 1–8. doi:10.1109/WiMOB.2018.8589107.

[81] M. Ammar, M. Washha, G. S. Ramabhadran, B. Crispo, Slimiot: Scalable lightweight attestation protocol for the internet of things, in: Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC), Kaohsiung, Taiwan, 2018, pp. 1–8. doi:10.1109/DESEC.2018.8625142.

[82] W. Yan, A. Fu, Y. Mu, X. Zhe, S. Yu, B. Kuang, Eapa: Efficient attestation resilient to physical attacks for iot devices, in: Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things, IoT S&P'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2–7. doi:10.1145/3338507.3358614.
URL https://doi.org/10.1145/3338507.3358614

[83] E. Dushku, M. M. Rabbani, M. Conti, L. V. Mancini, S. Ranise, Sara: Secure asynchronous remote attestation for iot systems, IEEE Transactions on Information Forensics and Security 15 (2020) 3123–3136. doi:10.1109/TIFS.2020.2983282.

[84] F. Brasser, K. B. Rasmussen, A.-R. Sadeghi, G. Tsudik, Remote attestation for low-end embedded devices: The prover's perspective, in: Proceedings of the 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016, pp. 1–6. doi:10.1145/2897937.2898083.

[85] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser, Physically uncloneable functions in the universal composition framework, in: P. Rogaway (Ed.), Proceedings of the Advances in Cryptology - CRYPTO, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 51–70.

[86] A. Ibrahim, Collective attestation: for a stronger security in embedded networks, in: Proceedings of the IEEE 37th Symposium on Reliable Distributed Systems (SRDS), 2018, pp. 267–268. doi:10.1109/SRDS.2018.00039.

[87] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, G. Tsudik, C-flat: Control-flow attestation for embedded systems software, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 743–754. doi:10.1145/2976749.2978358.
URL https://doi.org/10.1145/2976749.2978358

[88] G. Wurster, P. van Oorschot, A. Somayaji, A generic attack on checksumming-based software tamper resistance, in: Proceedings of the IEEE Symposium on Security and Privacy (S P'05), 2005, pp. 127–138. doi:10.1109/SP.2005.2.

[89] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, G. Vassilacopoulos, Enabling data protection through pki encryption in iot m-health devices, in: Proceedings of the IEEE 12th International Conference on Bioinformatics Bioengineering (BIBE), 2012, pp. 25–29. doi:10.1109/BIBE.2012.6399701.

[90] A. W. Atamli, A. Martin, Threat-based security analysis for the internet of things, in: Proceedings of the International Workshop on Secure Internet of Things, 2014, pp. 35–43. doi:10.1109/SIoT.2014.10.

[91] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, S. Shieh, Iot security: Ongoing challenges and research opportunities, in: Proceedings of the IEEE 7th International Conference on Service-Oriented Computing and Applications, 2014, pp. 230–234. doi:10.1109/SOCA.2014.58.

[92] D. McCann, K. Eder, E. Oswald, Characterising and comparing the energy consumption of side channel attack countermeasures and lightweight cryptography on embedded devices, in: Proceedings of the International Workshop on Secure Internet of Things (SIoT), 2015, pp. 65–71. doi:10.1109/SIOT.2015.11.

[93] K. Tiri, I. Verbauwhede, A vlsi design flow for secure side-channel attack resistant ics, in: Proceedings of the Design, Automation and Test in Europe, Vol. 3, 2005, pp. 58–63. doi:10.1109/DATE.2005.44.

[94] S. Crane, A. Homescu, S. Brunthaler, P. Larsen, M. Franz, Thwarting cache side-channel attacks through dynamic software diversity., in: Proceedings of the NDSS, 2015, pp. 8–11.

[95] H. Shacham, The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86), in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 552–561. doi:10.1145/1315245.1315313.
URL https://doi.org/10.1145/1315245.1315313

[96] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, Z. Liang, Data-oriented programming: On the expressiveness of non-control data attacks, in: Proceedings of the IEEE Symposium on Security and Privacy (SP), 2016, pp. 969–986. doi:10.1109/SP.2016.62.

[97] K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen, A.-R. Sadeghi, Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization, in: 2013 IEEE Symposium on Security and Privacy, 2013, pp. 574–588. doi:10.1109/SP.2013.45.

[98] S. Chen, J. Xu, E. C. Sezer, P. Gauriar, R. K. Iyer, Non-control-data attacks are realistic threats., in: USENIX security symposium, Vol. 5, 2005, p. 146.

[99] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, X. Zhang, Remote attestation to dynamic system properties: Towards providing complete system integrity evidence, in: 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, 2009, pp. 115–124. doi:10.1109/DSN.2009.5270348.

[100] R. Roemer, E. Buchanan, H. Shacham, S. Savage, Return-oriented programming: Systems, languages, and applications, ACM Trans. Inf. Syst. Secur. 15 (1) (mar 2012). doi:10.1145/2133375.2133377.
URL https://doi.org/10.1145/2133375.2133377

[101] G. Dessouky, T. Abera, A. Ibrahim, A.-R. Sadeghi, Litehax: Lightweight hardware-assisted attestation of program execution, in: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 2018, pp. 1–8. doi:10.1145/3240765.3240821.

[102] C. Bormann, M. Ersue, A. Keranen, Terminology for constrained-node networks, Tech. Rep. rfc7228, IETF (2014).

[103] G. Polat, Security issues in iot: Challenges and countermeasures, Isaca Journal (2019) 1–7.

[104] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, A.-R. Sadeghi, Atrium: Runtime attestation resilient under memory attacks, in: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, USA, 2017, pp. 384–391. doi:10.1109/ICCAD.2017.8203803.

[105] I. D. O. Nunes, S. Jakkamsetti, N. Rattanavipanon, G. Tsudik, On the TOCTOU problem in remote attestation, CoRR abs/2005.03873 (2020). arXiv:2005.03873.
URL https://arxiv.org/abs/2005.03873

[106] Trusted Computing Group, Accessed on: Oct 19, 2022 (2022).
URL https://trustedcomputinggroup.org

[107] C. Lee, L. Zappaterra, K. Choi, H.-A. Choi, Securing smart home: Technologies, security challenges, and security requirements, in: IEEE Conference on Communications and Network Security, 2014, pp. 67–72. doi:10.1109/CNS.2014.6997467.

[108] S. Skorobogatov, Physical Attacks and Tamper Resistance, Springer, New York, NY, 2012, Ch. 7, pp. 143–173. doi:10.1007/978-1-4419-8080-9_7.
URL https://doi.org/10.1007/978-1-4419-8080-9_7

[109] S. P. Skorobogatov, Semi-invasive attacks: a new approach to hardware security analysis, Ph.D. thesis, University of Cambridge (2005).

[110] S. Skorobogatov, Physical attacks on tamper resistance: progress and lessons, in: Proceedings of the 2nd ARO Special Workshop on Hardware Assurance, Washington DC, USA, 2011.

[111] Y. Zhou, D. Feng, Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing, Cryptology ePrint Archive (2005).

[112] L. Eliot, R. Scott, Discovering and Retrieving Software Transparency and Vulnerability Information, Memo, IETF (Apr 2023).
URL https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-sbom-access#EO2021