POLITECNICO DI TORINO
Repository ISTITUZIONALE

SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments

Original
SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments / Ferro, L., Bravi, E., Sisinni, S., Lioy, A.. - (2024), pp. 41-50. (SaT-CPS '24: 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems Porto (Portugal) 21 June 2024) [10.1145/3643650.3658609].

Availability:
This version is available at: 11583/2988270 since: 2024-07-07T15:55:55Z

Publisher:
ACM

Published
DOI:10.1145/3643650.3658609

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

20 November 2026



SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments

Lorenzo Ferro*
lorenzo.ferro@polito.it
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy

Enrico Bravi*
enrico.bravi@polito.it
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy

Silvia Sisinni*
silvia.sisnni@polito.it
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy

Antonio Lioy
antonio.lioy@polito.it
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy

ABSTRACT

Nowadays smart contexts (such as smart cities/homes, or Industry 4.0) are rapidly gaining popularity. These new paradigms are enabled by the adoption of smart devices, that allow several programmatically driven actions. The Internet of Things (IoT) is the network built by connecting these smart devices. A critical aspect of these devices is their limited hardware support for security functions. This makes protecting IoT devices very challenging, although very important because they implement critical functions, as in Cyber Physical Systems. In this case, the protection of these systems is of paramount importance because their compromise could cause not only digital but also physical damage. Remote Attestation (RA) is a security process that permits a trusted party to remotely verify devices integrity but this becomes challenging for IoT devices due to their hardware constraints. Swarm Attestation (SA) is a generalization of RA to reduce its overhead for IoT environments. In this way, it becomes possible to attest large IoT networks. This paper introduces SAFEHIVE, a new schema for SA to maximize dynamic swarm configuration and management. This schema permits to manage heterogeneous devices in a dynamic scenario, even in the case of great variability.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; **Embedded systems security**; **Security protocols**; *Authorization.*

KEYWORDS

Remote Attestation, Swarm Attestation, IoT, Cyber-Physical System

ACM Reference Format:
Lorenzo Ferro, Enrico Bravi, Silvia Sisinni, and Antonio Lioy. 2024. SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments. In *Proceedings of the 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '24), June 21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3643650.3658609

*Authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution International 4.0 License.

SaT-CPS '24, June 21, 2024, Porto, Portugal

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0555-7/24/06
https://doi.org/10.1145/3643650.3658609

1 INTRODUCTION

The Internet of Things (IoT) [6] has surged in popularity, finding applications across diverse domains (Fig. 1) such as healthcare, industrial control, and smart home. IoT devices are often deployed in swarms, a heterogeneous group of communicating devices. For instance, in smart factory control systems, multiple interconnected IoT devices collaborate to manage crucial processes [26].

The IoT nodes are typically small, battery-powered embedded devices designed to perform specific tasks. They may be used in harsh and remote environments, presenting numerous challenges. The most important ones are trust, security, and privacy management. Those devices are less capable of detecting and blocking unauthorized activities. For this reason, they are the target of various attacks. Usually, an external trusted entity is used to monitor the system integrity status. This procedure is called Remote Attestation (RA) and consists of a remote *Verifier* asking for an integrity proof representing the current state of an *Attester* and comparing it with the expected state. If many IoT devices are involved, the standard attestation approach does not scale. Not all devices may be connected to the Internet, in some cases it is not necessary and only opens to risks. The IoT device communication process is usually a power-consuming operation, especially if directed to a distant receiver. Not all the devices connected in the swarm have the same needs: Some devices may necessitate more integrity guarantees because they perform more critical operations. Other devices may have to guarantee short delays so can not spend periodically many resources on attestation procedures. For those reasons, the IoT device swarms are usually attested as a single entity, this process is called *Swarm Attestation.* The Swarm Attestation idea is that the Verifier contacts a single node which is in charge of obtaining the integrity proof of the whole system and generating a quote for the whole system. The integrity proof of each device may have a different form given the heterogeneous identity of the swarm.

Besides, there are some solutions for Swarm Attestation in the literature, but they miss some core features or lack adaptability to

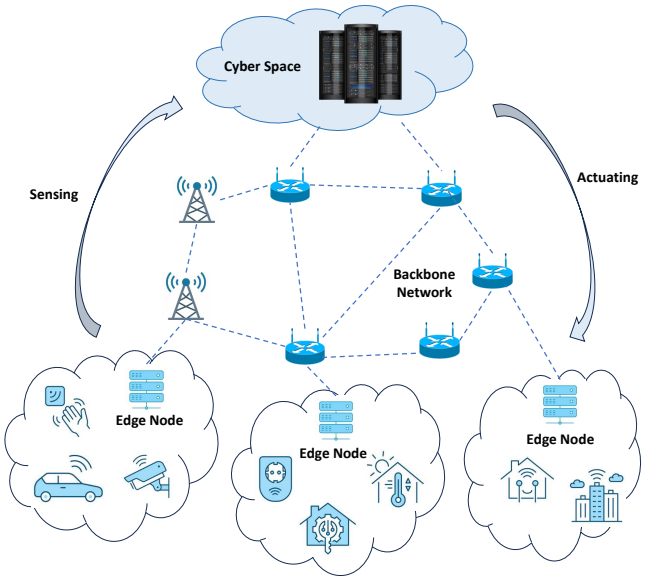


Figure 1: High-level view of a Cyber-Physical System.

various contexts. An important feature that is often not considered is to handle dynamic join and leave of swarm nodes during the attestation process. This feature is important in various fields to allow modification to the network topology without interrupting the verification process. Another feature not well established in the existing works is an isolation method for compromised nodes. Usually, when the Verifier finds anomalies in the network it marks the whole storm as untrusted, even if was one node compromised.

Contributions. This paper introduces SAFEHIVE, a new schema for Swarm Attestation. The solution aims to maximize configurability and better adapt to different swarm configurations. Another important feature is the support for dynamic swarm, where a node may connect or disconnect during the attestation procedure. SAFEHIVE introduces a protocol for the isolation of compromised nodes, not allowing them to communicate in the network. The schema considers the heterogeneity of the swarms and allows custom configuration for each node given its peculiarities. In addition, the solution is compared with other works in the Swarm Attestation field considering different scenarios.

Paper structure. This paper is organized as follows. *Section* 2 presents a brief explanation of the main concepts needed for the proposal, providing an overview of Remote Attestation and Swarm Attestation. *Section* 3 overviews the most related work to the proposed schema underlining the differences. *Section* 4 explain the assumption made in terms of minimal device requirements and the threat model defined. *Section* 5 explains in detail the SAFEHIVE architectural design explaining all the protocols defined. *Section* 6 underlines the security guarantees provided by the proposed schema. *Section* 7 provides some conclusion.

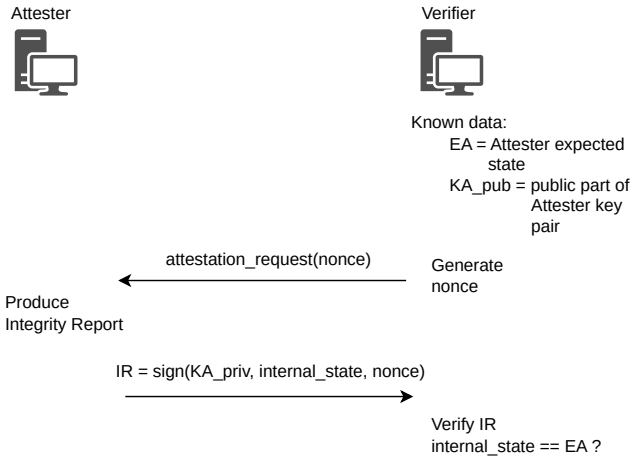


Figure 2: Basic Remote Attestation schema.

2 BACKGROUND

2.1 Remote Attestation

Remote Attestation (RA) [9] is a procedure that permits a trusted party (*Verifier*) to remotely verify the software and hardware configuration of a platform (*Attester*). The basic structure, as shown in Fig. 2, is the one of a challenge-response protocol. It begins with the Verfier sending an attestation request to the Attester, generating a *nonce* to avoid replay attacks. Once the Attester receives the attestation request, it produces an Integrity Report (IR), which contains an integrity proof of its internal state and the nonce, and it is signed with its private key (*KA_priv*) to provide authentication. The Verifier, after having received the IR, first of all, verifies the signature of the IR with the public part of the Attester key pair (*KA_pub*) and then verifies that the Attester's internal state is equal to the expected one (EA). If this check is successful, the Attester can be evaluated as trusted, otherwise, it is considered untrusted.

The key component necessary to perform RA is the Root of Trust (RoT) [8]. The peculiarity of this component is that it is considered trusted apriori because there is no possibility of detecting misbehaviour at runtime. The RoT is composed of three elements:

(1) *Root of Trust for Measurement* (RTM): it has the purpose of collecting integrity measurement of the software and hardware configuration;
(2) *Root of Trust for Storage* (RTS): it has the purpose of storing the integrity measurements calculated by the RTM;
(3) *Root of Trust for Reporting* (RTR): it is the element that permits to externally provide the measurement securely stored in the RTS.

Based on these concepts, the Trusted Computing Group[1] (TCG), proposed the concept of *Trusted Platform* (TP). This is a platform that can measure all its software and hardware components, to be able to provide them to another entity to verify the TP integrity. The TCG proposed the Trusted Platform Module (TPM) [25] which is a possible way to implement a TP. The TPM is a secure element, that implements a RTS and RTR. It is implemented as a hardware

[1] https://www.trustedcomputinggroup.org

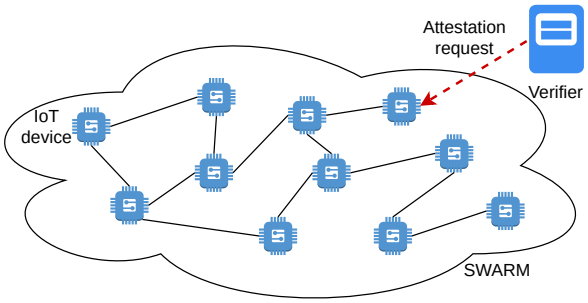


Figure 3: Basic Swarm Attestation schema.

component, built to provide security capabilities such as tamper resistance. The TPM possesses some registers, called Platform Configuration Registers (PCRs), that are used to store the measurements computed by the RTM. A PCR supports only the *extend* operation for updating its value. This operation permits to keep the history of all values assigned to a PCR. The extension is performed by taking the current value of the PCR, concatenating it with the new measurement to store, and then calculating the hash of this concatenation. The hash calculated is the new value that is stored in the PCR:

$$PCR_{new} = \mathbf{Hash}_{Algo}(PCR_{old} \mid\mid measurement)$$

Hardware-based Attestation. These techniques rely on dedicated hardware, typically a cryptographic chip, present on the Prover. One of the most common hardware devices for RA is the TPM. The TCG proposed the TPM specification, with version 1.2 being the first [24]. Subsequently, an improved version, TPM 2.0, was proposed and is the current standard [24]. Alternative RA approaches leverage Trusted Execution Environments (TEEs) [20], such as Intel SGX [10] or ARM TrustZone [19]. These techniques typically rely on specific hardware extensions that provide enhanced security features.

Software-based Attestation. Although hardware-based attestation offers a highly effective solution for RA, it might not always be practical due to hardware and software limitations, particularly in embedded devices. To overcome this challenge, software-only RA approaches have emerged to minimize hardware overhead. Pioneer [21] exemplifies such a software-based primitive, operating without dependence on CPU architecture extensions or secure coprocessors. This method's core concept involves the *dispatcher* (Verifier) utilizing egeer to establish a *dynamic root of trust* [17] on the *untrusted platform* (Prover), guaranteeing the integrity of all contained code.

Hybrid Attestation. While software-based approaches for RA may not be sufficient in certain networked settings due to potential adversarial capabilities [12], a hybrid approach that incorporates both software and hardware has been developed to address this issue. One example is SMART [11], which is based on a minimal hardware modification of embedded Micro Controller Units (MCUs) and represents the first minimal hardware solution for establishing a dynamic root of trust in such devices.

While purely software-based approaches for RA might be insufficient in specific networked environments due to potential adversarial capabilities [12], a hybrid approach combining software and hardware elements has been proposed to address this challenge. One proposed mechanism is SMART [11], which leverages minimal hardware modifications to embedded MCUs. This approach represents the first instance of a minimal hardware solution for establishing a dynamic RoT on such devices.

2.2 Swarm Attestation

Swarm Attestation [7] schemas have been proposed because the single attester-verifier schema started to be inefficient for IoT infrastructures. For this reason, new methods have been developed to be able to manage the RA of potentially vast IoT networks. A *swarm* is a network of interconnected IoT devices that can communicate with each other and it represents the system on which to perform RA to verify the integrity of the devices. The goal of swarm attestation schemas (Fig. 3) is to be able to collect attestation evidence by contacting only one node of the swarm. This permits to be unaware of IoT network topology and even the number and identity of the devices that compose the swarm. This is typically not required because the purpose is to verify the integrity of the whole swarm, and not of the single device. This can be done by defining an *aggregation pattern* which permits to collect and aggregate all the swarm integrity reports, sending to the verifier only the final aggregate. The main aggregation patterns are: *spanning tree*, *broadcast*, and *hierarchical*.

In the spanning tree schema [1, 5] works with the Verifier sending an attestation request to a node of the swarm (Fig. 4). This node that receives the first attestation request, sends it to its neighbours, following a spanning tree defined on the network. Each node sends its attestation report to the node that forwarded the attestation request. Each node aggregates also the attestation report received and then passes it to the previous node. In this way, the verifier receives an aggregate report that can be used to evaluate the swarm's trustworthiness.

The broadcast schema [3, 15] works a bit differently from the spanning tree. It begins with an attestation request sent to the verifier to a node of the swarm, but the propagation is performed by sending this attestation request is sent in broadcast to all the devices in the communication range (Fig. 5). The aggregation is

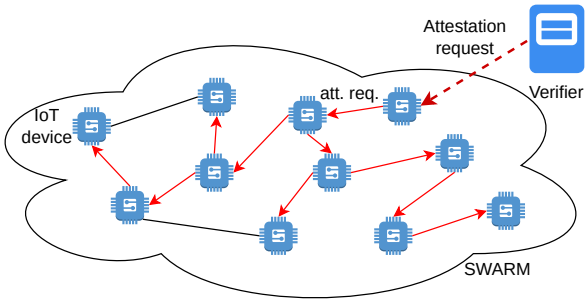


Figure 4: Spanning tree aggregating pattern in Swarm Attestation.

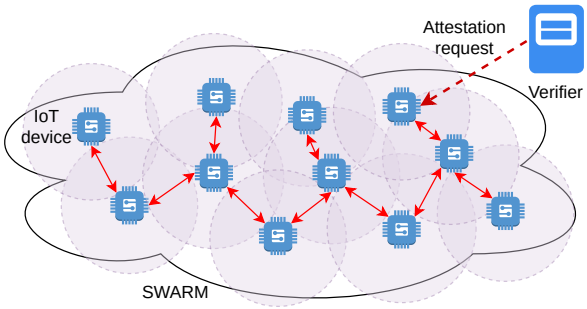


Figure 5: Broadcast aggregating pattern in Swarm Attestation.

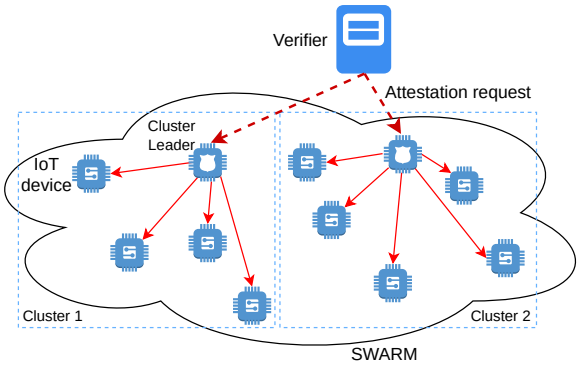


Figure 6: Hierarchical aggregating pattern in Swarm Attestation.

performed in broadcast as well, so the integrity report is sent in broadcast and the node that receives it, will aggregate it with the others received.

The hierarchical schema [13, 16] is quite different from the previous ones. The swarm is divided into clusters of devices, with one cluster leader for each cluster. In this case, the verifier sends the attestation request to all the cluster leaders, and then they send it to all the nodes that belong to the cluster (Fig. 6). The aggregation is performed by all the cluster leaders, and then they send the aggregate to the verifier.

3 RELATED WORK

In the last few years, many security schemas have been presented to address the attestation of computationally lightweight device swarms. Firstly embedded devices were attested singularly principally using software-based techniques [22], later also hybrid [18] and, hardware-based ones [27]. SEDA (Scalable Embedded Device Attestation) [5] introduced the concept of Swarm Attestation. This schema allows the attestation of the whole device network as a single entity. Each component of the swarm is based on an architecture compliant with SMART [11] or TrustLite [14], to set minimum hardware requirements. The protocol is divided into two phases. The offline phase is initializing and registering each device to neighbouring devices. The online phase consists of the attestation of the system. A Verifier contacts an arbitrary element of the swarm (initiator) that will recursively propagate the request reaching the whole system. Every node can attest its neighbours and transmit the results to other nodes. The accumulated attestation result will be sent to the Verifier by the initiator. Besides giving the basis for swarm attestation, this protocol has limitations. SEDA lacks a method for the identification and isolation of compromised devices. When one component within the swarm is compromised, the entire swarm is marked as untrusted. Swarm components are usually heterogeneous, with each device having unique priorities, capabilities, and requirements. Additionally, each component has varying security needs, necessitating different attestation times. SEDA forces strict attestation times, for each attestation cycle every device is attested. This leads to attesting too frequently on some devices while others may not attest as often as required. Another concern regards dynamic swarms, as SEDA lacks support for adding or removing devices during the attestation process.

After SEDA many other solutions were proposed to attest swarms, the first one that introduces a more dynamic topology proposed by Ambrosin et al. [2]. In this schema, each node will perform self-attestation and communicate in broadcast the result to its neighbors. A Verifier can contact any node and obtain information regarding the status of the device and its neighbours. PADS [3] propose a similar approach using a more sophisticated method for converging into a "network view". Periodically each node broadcasts its status and the knowledge about its neighbours. In this case, every node knows the whole network status. Also, SALAD [15] is a schema for dynamic swarm attestation. The concept is similar to the other solution, propagating the self-attestation results but in a more secure way. However, SALAD necessitated that every pair of devices within the swarm possess a unique symmetric key, leading to significant resource consumption. These solutions tackle the challenge of provisioning reference values in a dynamic system by storing values directly on the devices. This enhances the dynamism of the verification process but also introduces security concerns regarding self-attestation. Additionally, updating these devices becomes more challenging.

Alternative solutions, like WISE [4], concentrate on adapting to the diverse requirements of the devices. During each attestation round, only a portion of devices are queried for integrity proofs. This solution is cluster-based, some nodes will be elected as cluster leaders and have to propagate to the cluster members the attestation requests. The cluster leaders must be configured statically before initiating the attestation process. Leaders establish a virtual spanning tree for communication and forwarding attestation requests. Dividing the swarm into clusters reduces the communication overhead but this solution requires no topological changes during the attestation process. Another solution cluster-based is MTRA [23] which requires a heterogeneous fixed network. MTRA assumes the presence in the swarm of devices that mount a TPM that will be elected cluster leaders. Also in this design, there is no possibility for dynamic insertion or removal of nodes. The network is configured, sharing secrets and discovering neighbours in the offline phase, once the attestation starts it is not possible to perform changes to the topology.

ESDRA [16] introduced the concept of untrusted node accusation, employing a many-to-one attestation scheme. It employs adjacent devices to assess the prover's integrity and promptly notifies the

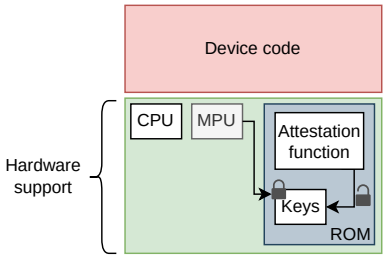


Figure 7: Minimal device architecture for supporting the security requirements.

network owner of any compromised nodes. This schema introduces also the reputation mechanism if a node has been trusted for various attestation routines it will play more important roles in the system. The division in clusters is still static and delegated to the network owner. Besides introducing a way to identify compromised nodes, it does not dynamically isolate them but relies on the intervention of the network owner.

The proposed solution will fill those gaps, aiming to be complete and highly configurable. The solution will handle dynamic swarms without forcing strict attestation times on all devices.

4 SYSTEM MODEL

4.1 Device Requirements

To perform hardware-based RA, a device needs basic hardware requirements (Fig. 7) that permit it to have a hardware RoT. These basic requirements are necessary for having the possibility to secure store *identity* and *attestation* keys needed by the device. The secure storage must provide constraints on the possibility of modifying those keys. For these reasons, a ROM must be present on the device, to protect those data from manipulation. In addition, also a memory protection mechanism is required, such as a Memory Protection Unit (MPU), to have the possibility of setting permissions on specific memory regions. This permits to protect some portions of code from the untrusted environment and can perform some critical operations like measuring the device code that must be attested and the report function which has access to the private part of the attestation key. In the must be present an *attestation function* which performs all the operations needed by RA. The MPU assert that the keys can be accessed only by the memory region of the attestation function, and protect the attestation function from external manipulations. Further enhancements may be present depending on the application, such as cryptographic accelerators, display adapters, and communication interfaces.

4.2 Threat Model

Only software adversaries are considered, who can perform remote attacks. Adversaries can fully access the network and perform passive (e.g. eavesdrop on communication) or active (e.g. malware injection) attacks. Coherently with all the other swarm attestation techniques we didn't consider DoS and physical attacks. Physical ones are expensive and difficult to detect and DoS are almost impossible to completely resist. We consider that the adversaries are

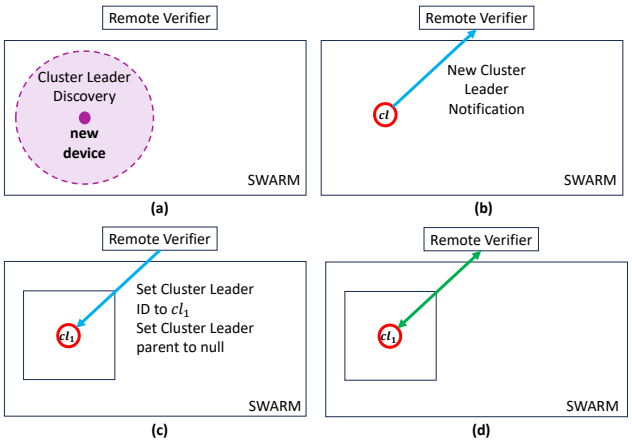


Figure 8: Swarm initialization.

capable of modifying every packet sent in the network and evading the attestation protocol via various methods, such as forging, substituting, replaying, and eavesdropping.

5 SAFEHIVE: ARCHITECTURAL DESIGN

The proposed design is cluster-based, where each cluster has a dynamically elected leader that coordinates integrity proofs aggregation. The attestation process starts with the Remote Verifier sending a broadcast attestation request to the cluster leaders. The cluster leaders forward the attestation request to the cluster members. Each member may take part in the attestation process, collect its integrity proofs and send them to the cluster leader. The cluster leader will gather integrity proofs and, along with its integrity proof, transmit them to the next cluster leader according to a spanning tree structure, ultimately reaching the Verifier. When the verifier receives the integrity report will perform the verification of the whole system's integrity proofs and release an authentication token for each entity. These tokens are distributed to each device and are used to prove the state when communicating with another node.

5.1 Cluster Configuration

The cluster structure is designed to be dynamic and automatically adapt to different topologies. It is assumed that the heterogeneous structure of the swarm comprehends devices with low capabilities but also nodes with more computational power and internet connection. Each device is configured with a value that reflects its predisposition to become a cluster leader. The value considers the device's hardware capabilities, its connection to the Internet and, the possibility of delaying its common operations. If the device cannot become a leader will have a value of 0 (zero). All the devices with a predisposition to become leaders have the Verifier IP and information configured and a crescent value of up to 10.

The initialization of the swarm starts with the first device turned on as represented in Fig. 8. The first device turns on the network and broadcasts a Cluster Leader Discovery packet (a). It receives no response, so it sets itself as Cluster Leader and notifies the Remote Verifier of the presence of a new Cluster Leader in the network (b). The Remote Verifier notifies the new Cluster Leader of its identity

and of its Parent Cluster Leader, which is *null*, so it becomes the root of the tree (c). cl_1 will send its integrity reports directly to the Remote Verifier (d). The second device turned on will start the *leader discovery protocol*. As shown in Fig. 9, this involves broadcasting its identity and value in a cluster leader discovery packet (a). If a cluster leader is present in the communication range will respond with its value. The node will join the cluster leader's network in case has a lower or equal value, and in case of a higher value will become the cluster leader and inglobate the other cluster leader network (b).

If no other node responds, it will become a cluster leader, given it possesses the necessary characteristics (Fig. 10 (a)). The new cluster leader, to be recognized by the system, notifies the Remote Verifier about the presence of a new cluster leader (b). Then the Remote Verifier responds with the new Cluster Leader identity (cl_2) and its Parent Cluster Leader in the spanning tree, which in this case is cl_1 (c). After this procedure, the cl_2 will send its integrity reports to cl_1 (d). Otherwise, if the node does not have the capabilities, it will attend to the connection of a cluster leader performing this protocol on loop.

A cluster leader keeps track of the number of cluster members, to monitor their quantity. Each cluster leader has a maximum amount of devices that can be accepted into its cluster. When the cluster reaches 70% of the max dimension will start to accept only devices not able to become cluster leaders (the ones with values lower than 3). This happens to encourage the creation of new clusters to avoid large clusters. It may still happen that a cluster reaches the maximum dimension, in this case is necessary to scatter the cluster into smaller ones. The cluster rebalancing procedure, shown in Fig. 11, happens when a new device broadcasts a Cluster Discovery packet to join the network (a). If the cluster leader *cl3* reaches the maximum number of child nodes (b), it will *cl3* broadcast a Cluster Leader Election packet indicating the average leadership score (e.g. 5) of its child nodes (c). All nodes with a leadership score higher or equal to the score indicated by the leader re-execute the Cluster Leader Discovery protocol. The device with the highest score assumes the leadership of the cluster and notifies the Remote Verifier (d). The Remote Verifier notifies the new Cluster Leader of its ID and its Cluster Leader parent (e). The network automatically converged to a new clusters configuration (f). After that, those nodes will start the leader discovery protocol and create new clusters. The cluster leader will not accept them back when reaches 70% of its maximum capacity.

Another aspect to consider is when clusters become too small. In this case, there would be a high number of clusters that can be merged, avoiding unnecessary overhead. For this reason, we defined a *cluster merging protocol*, depicted in Fig. 12. When clusters are too small, periodically Cluster Leaders can try to re-balance clusters in the network (a). When the Merging Protocol is triggered, the Leader of the small cluster sends a Cluster Leader Election packet to its child nodes (b). Nodes perform a Cluster Leader Discovery protocol, in which the Leaders of non-maximal clusters, that fall within the reception range of the short-range network, participate (c). The new Cluster Leader is elected (d). If a Cluster Leader has been "downgraded" to a normal node, it notifies the Remote Verifier of its removal as a Cluster Leader (e). The network converged to a more balanced topology (f).

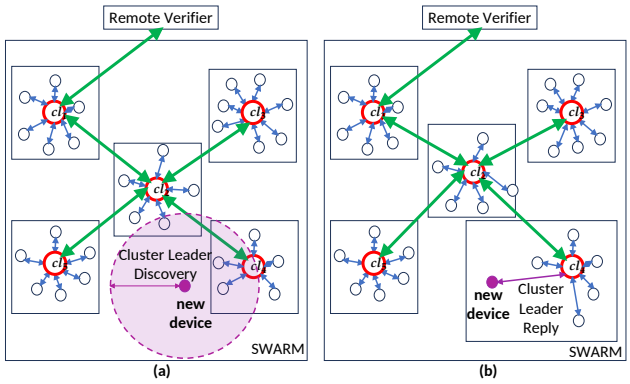


Figure 9: Joining Protocol.

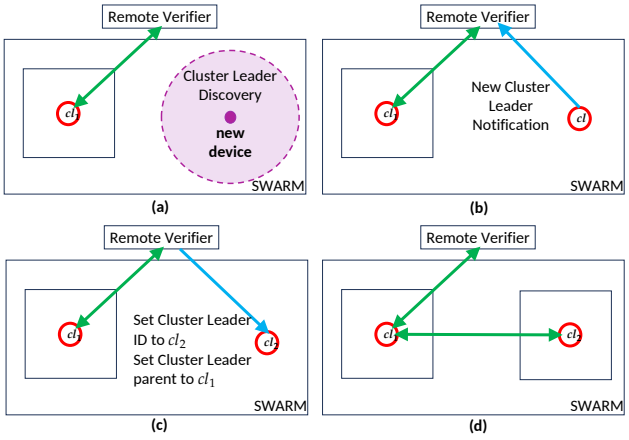


Figure 10: Dynamic Configuration of the Spanning Tree of Cluster Leaders.

All the cluster leaders in the network are connected with a spanning tree. The spanning tree facilitates parallel computing while also reducing transmission costs. In such scenarios, the energy cost of nodes increases significantly with greater communication distances. Utilizing multi-hop communication helps mitigate energy costs. Each member of the spanning tree has information regarding its parent in the tree and the number of nodes that it has connected. When a node becomes cluster leader it necessitates to be registered at the Verifier. The node sends a request to become cluster leader, giving its information, including also its position in the swarm. The Verifier accepts the new cluster leader sending him the information regarding its parent in the spanning tree. After that, the Verifier also notifies the parent node about the insertion of a new node between its child. The Verifier knows the best point where insert the new cluster leader in the spanning tree given its position in the swarm and the tree's balance.

Another scenario to be considered is a device leaving the swarm. In case of graceful leave, the device will communicate to its cluster leader that he is leaving. A cluster leader may communicate its intent to leave to the members, prompting them to search for a new leader. Simultaneously, it communicates this decision to the

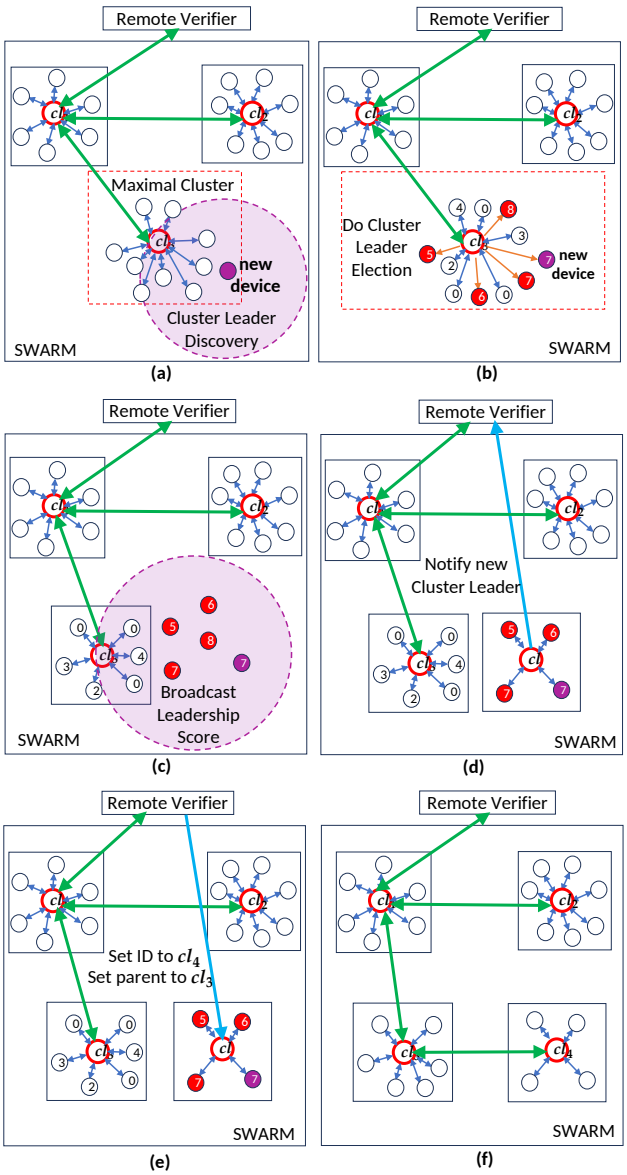


Figure 11: Cluster Re-balancing Protocol.

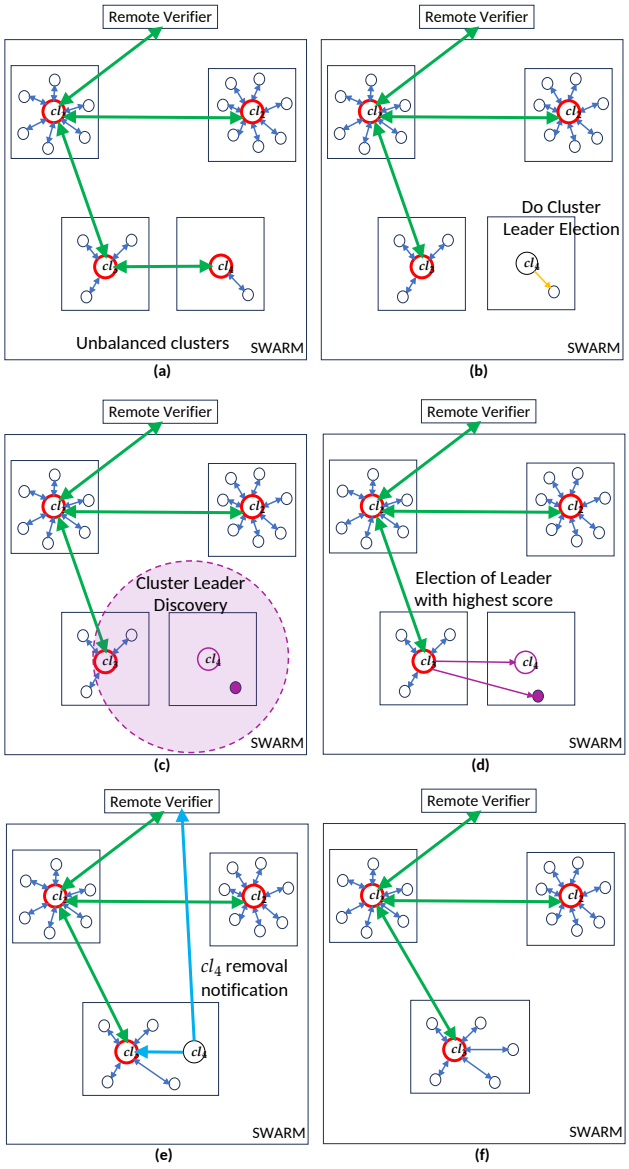


Figure 12: Cluster Merging Protocol.

Verifier, which will readjust the spanning tree accordingly. In the case of ungraceful leave, if it is a cluster leader and fails to respond during an attestation cycle, the Verifier will label it as untrusted and disband its cluster. Common nodes that ungracefully leave the swarm, will be set as unreachable.

5.2 Attestation Schema

The attestation in this solution allows the attestation of devices only when necessary, without forcing attestation time for the whole system. The network is composed of different devices, and forcing all to attest with the same frequency will lead to critical devices being under-attested and low-capability devices spending too much time in the attestation procedure. This solution allows custom attestation times and isolates compromised devices or those with not fresh enough attestation reports. The idea is not to have the cluster components not always in listen mode for attestation requests, but only when their token is expiring they participate in the attestation procedure.

The attestation procedure is still driven by the Verifier, which has high computation capabilities (Fig. 13). The Verifier knows the cluster leader's topology, it can start the verification phase by sending an attestation request in broadcast to all the cluster leaders (a). The request will include a nonce to avoid both replay attacks and duplicate request messages received within the network. The cluster leader will forward the attestation request in-broadcast to

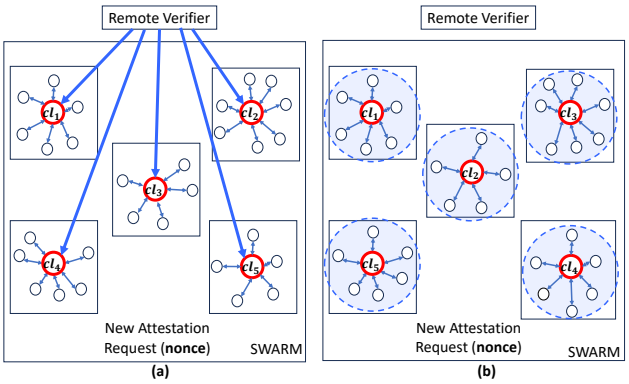


Figure 13: New RA request by the Verifier, forwarded to all the nodes.

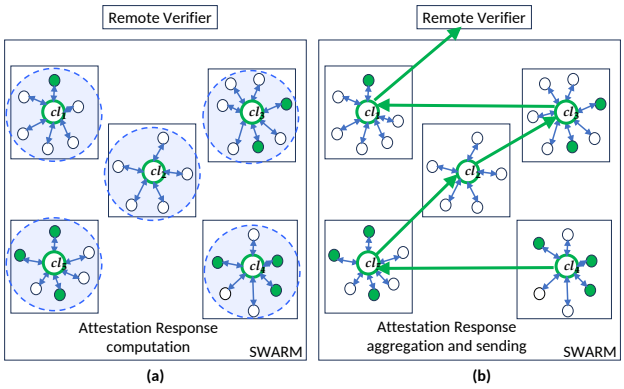


Figure 14: The gathering, aggregation, and transmission of integrity proofs back to the Verifier.

all the cluster members (b). Only a subset of the cluster members participate in the attestation: all Cluster Leaders and child devices with expiring authentication tokens (Fig. 14). They will collect the integrity proofs and send them to the cluster leader (a). The cluster leaders will aggregate all the proofs received together with its evidence and send them to its parent in the spanning tree (b). The last cluster leader will send the response to the Verifier. At this point, the Verifier checks the validity of the integrity proofs and emits tokens for each entity attested. The tokens are sent back to the cluster leaders who will distribute them in the cluster. If a node does not respond even if its token is expired it is not possible to mark it as compromised. These devices may go into power-safe mode if has no tasks to address and in this mode ignore the requests. In this case, the device shouldn't communicate with others, in case it happens its token is expired and the communication request will be rejected. Instead, if it is a cluster leader that does not respond to attestation requests, it will be marked as untrusted and removed from the network. Each device in the swarm may require different times for integrity-proof generation. The cluster leader will wait for the response of the cluster members but may happen that a device sends the integrity proofs when the cluster leader has already passed the quote to its parent. In this case, the integrity proof of this

Field	Description
status	result of last verification
last-verification	timestamp of the last verification
token-duration	duration of the token validity
device-class	class of the device
device-pkey	public key of the device
value	score of the device predisposition to become a cluster leader
max-dimension	max number of devices that could handle in its cluster if became cluster leader

Table 1: Token's fields.

device will be inserted in the next verification round. The Verifier detect that those proofs belong to the previous attestation cycle but still verifies them and releases a token with lower time validity for this device.

Given the heterogeneous network, the Verifier could not possess the reference values for each kind of device. This problem is addressed by including information regarding the device in the response, together with the integrity proof. There will be specified the device type, its public key, its endorser and, a pointer to download its reference values from the endorser. With this information, the Verifier, if trusts the endorser, can download the reference values and use them for the attestation.

5.3 Authentication Token

The communication between devices in the swarm is regulated by an authentication token, which is released and signed by the Verifier. The token is released after a correct attestation of the device, it has a duration dependent on the device's characteristics. The Verifier sends the tokens to the cluster leaders who will distribute them to the cluster members. The token contains information related to the device and its status (Tab. 1).

At each attestation cycle, the Verifier releases a token for each trustworthy node that participated in the attestation (Fig. 15). Those certificates are sent to the cluster leaders (a) who will broadcast them into their subnets (b). The compromised devices in the Cluster will not acquire a new valid Authorization Token (c). If the Cluster Leader is compromised, the Remote Verifier also considers the devices connected to it to be compromised and does not issue Authorization Tokens for them (d).

When a cluster member receives a new token, it activates a timer, and before its expiration, the device must participate in an attestation round to get a fresh one. Each time a device wants to communicate with another one, it must present its token and request the other node's token. The token is a way to introduce itself to others, giving secure information regarding its status and identity. Firstly, when receiving another token, it is essential to check that it's signed by the same authority that signed mine. Then it is necessary to check that the tokens have not expired. A device can verify the validity of a token by comparing the timestamp with its one. It must calculate the difference (Δt) between the two timestamps representing the issue time of the tokens. Once calculated this delta, it must read the current value of its timer, to know how much time

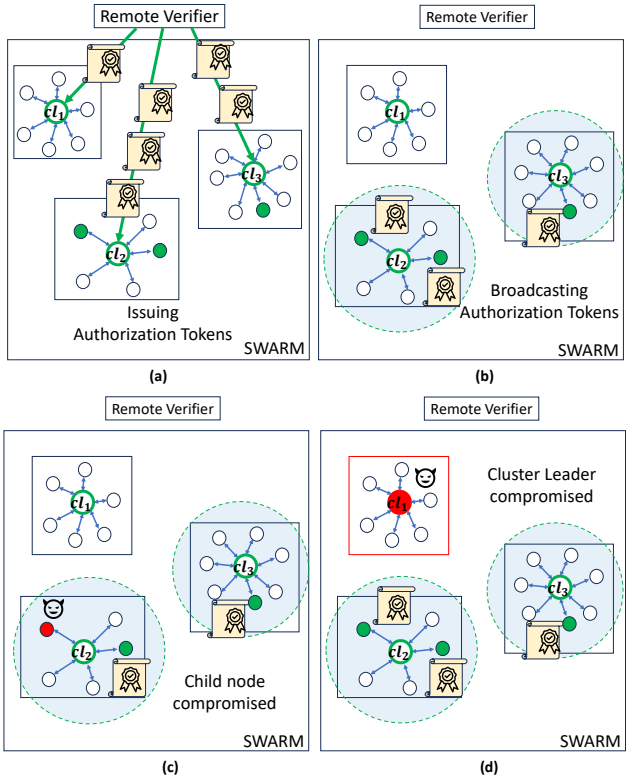


Figure 15: Integrity verification and issuing of authorization tokens.

has passed since the last valid token has been received. Adding the value of the timer to the Δt makes it possible to know the time passed since the issue time of the token to verify. If this value is less than the expiration time, then the token is still valid and the device that sent it is trustworthy. If the value obtained is positive it means the token is still valid.

$$\Delta t = \textbf{Issue}_{time}(MyToken) - \textbf{Issue}_{time}(ReceivedToken) \quad (1)$$

$$Validity = \textbf{Expire}_{time}(ReceivedToken) - (\Delta t + MyTimer) \quad (2)$$

$$ReceivedToken\ is\ \textbf{valid} \iff Validity > 0 \quad (3)$$

This procedure enables token validity verification without requiring time synchronization, which is difficult to obtain in swarm systems.

This schema isolates automatically untrusted nodes, not allowing them to communicate in the swarm and compromise others (Fig. 16). With this mechanism, when a node is compromised, there is no need for a timely action by the network administrator as the compromised node is automatically and seamlessly ring-fenced. In case a cluster leader is compromised, all its nodes will be considered untrusted, since their attestation procedure is based on the cluster leader. Since the token is released and signed by the Verifier, it's feasible to use it for intra-cluster communications, since a single entity releases all the tokens.

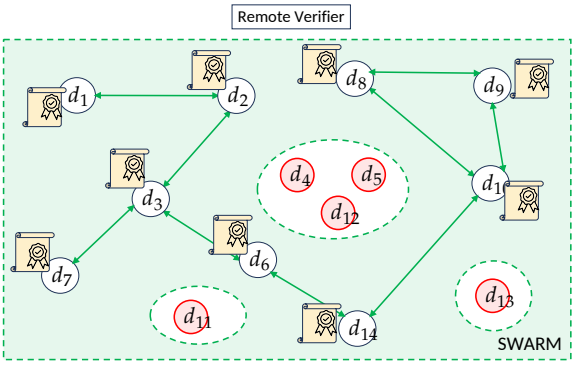


Figure 16: Trusted Communications in the swarm.

6 SECURITY DISCUSSION

We will now examine how SAFEHIVE's security holds up against the adversary model introduced in Section 4.2. In an attestation scheme, the goal of an adversary, whether local or remote, is to compromise the firmware and/or configuration data of a device without the Verifier being able to detect the compromise and manage to alter/poison the correct functioning of the network.

In general, an attestation protocol is considered secure if it is computationally impossible for a polynomial attacker to trick the verifier into accepting a forged attestation result as valid. We assume that the implementation of the attestation protocol in SAFEHIVE uses cryptographic primitives that guarantee security strength compliant with the recommendations of the TCG. Furthermore, we assume that the Remote Verifier adopts a True Random Number Generator (TRNG) which guarantees that the nonce issued by the Verifier at each attestation cycle is unpredictable. If an attacker could predict a small enough set of values into which the nonce would most likely fall, he could collect attestation reports generated using the predicted nonces and subsequently respond to the Remote Verifier with them.

The adjustable window between two attestations on a device, tailored to its capabilities, enables strict attestation intervals for security-critical devices without forcing constrained devices to spend excessive time in the attestation process. Utilizing tokens to mediate interactions with other devices can effectively isolate compromised devices and mitigate the risk of infecting the entire network.

7 CONCLUSION

IoT is widely gaining interest and protecting them is becoming crucial, because they can manage sensitive data that need secure processing. Typically Remote Attestation (RA) is adopted to assert the trustworthiness of network components. In the case of IoT networks, traditional RA procedures are of difficult application, due to the large number of connected devices. For this reason, Swarm Attestation has been proposed, to provide integrity information of an IoT network while avoiding the necessity to contact directly each node of the network. This paper introduced SAFEHIVE, a

swarm attestation protocol for heterogeneous groups of resource-constrained devices. It leverages dynamic clusters and isolates compromised nodes with a token mechanism. SAFEHIVE is scalable and highly efficient, making it suitable for dynamic networks encompassing thousands of devices. Its minimized communication overhead depends on the acceptable latency for detecting compromised devices, rendering it well-suited for application in various domains, particularly time-sensitive and safety-critical environments.

ACKNOWLEDGMENTS

This work was partially supported by the project SERICS (PE00000014) under the NRRP MUR program, funded by the European Union - NextGenerationEU. This publication is also part of the project PNRR-NGEU which has received funding from the MUR - DM 352/2022.

REFERENCES

[1] Moreno Ambrosin, Mauro Conti, Ahmad Ibrahim, Gregory Neven, Ahmad-Reza Sadeghi, and Matthias Schunter. 2016. SANA: Secure and Scalable Aggregate Network Attestation. In *2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, Vienna (Austria), 731–742. https://doi.org/10.1145/2976749.2978335

[2] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani, and Silvio Ranise. 2017. Toward secure and efficient attestation for highly dynamic swarms: poster. In *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Boston (MA, USA), 281–282. https://doi.org/10.1145/3098243.3106026

[3] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani, and Silvio Ranise. 2018. PADS: Practical Attestation for Highly Dynamic Swarm Topologies. In *2018 International Workshop on Secure Internet of Things (SIoT)*. IEEE, Barcelona (Spain), 18–27. https://doi.org/10.1109/SIoT.2018.00009

[4] Mahmoud Ammar, Mahdi Washha, and Bruno Crispo. 2018. WISE: Lightweight Intelligent Swarm Attestation Scheme for IoT (The Verifier's Perspective). In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, Limassol (Cyprus), 1–8. https://doi.org/10.1109/WiMOB.2018.8589107

[5] N. Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. 2015. SEDA: Scalable Embedded Device Attestation. In *22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, Denver (CO, USA), 964–975. https://doi.org/10.1145/2810103.2813670

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (June 2010), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

[7] Alexander Sprogø Banks, Marek Kisiel, and Philip Korsholm. 2021. Remote Attestation: A Literature Review. *CoRR* abs/2105.02466 (May 2021), 34 pages. https://doi.org/10.48550/arXiv.2105.02466

[8] Michael Bartock, Murugiah Souppaya, Ryan Savino, Tim Knoll, Uttam Shetty, Mourad Cherfaoui, Raghu Yeluri, Akash Malhotra, Don Banks, Michael Jordan, Dimitrios Pendarakis, J. R. Rao, Peter Romness, and Karen Scarfone. 2022. Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases. https://doi.org/10.6028/NIST.IR.8320

[9] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of remote attestation. *International Journal of Information Security* 10, 2 (01 June 2011), 63–81. https://doi.org/10.1007/s10207-011-0124-7

[10] Victor Costan and Srinivas Devadas. 2016. Intel SGX explained. https://eprint.iacr.org/2016/086.pdf

[11] Karim Eldefrawy, Aurélien Francillon, Daniele Perito, and Gene Tsudik. 2012. SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust. In *Network and Distributed System Security Symposium*. San Diego (CA, USA), 462–473.

[12] Karim Eldefrawy, Norrathep Rattanavipanon, and Gene Tsudik. 2017. HYDRA: Hybrid Design for Remote Attestation (Using a Formally Verified Microkernel). In *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '17)*. ACM, Boston (MA, USA), 99–110. https://doi.org/10.1145/3098243.3098261

[13] Bei Gong, Yu Zhang, and Yubo Wang. 2018. A remote attestation mechanism for the sensing layer nodes of the Internet of Things. *Future Generation Computer Systems* 78 (2018), 867–886. https://doi.org/10.1016/j.future.2017.07.034

[14] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. 2014. TrustLite: a security architecture for tiny embedded devices. In *EuroSys '14: Ninth European Conference on Computer Systems*. ACM, Amsterdam (The Netherlands), 14 pages. https://doi.org/10.1145/2592798.2592824

[15] Florian Kohnhäuser, Niklas Büscher, and Stefan Katzenbeisser. 2018. SALAD: Secure and Lightweight Attestation of Highly Dynamic and Disruptive Networks. In *2018 Asia Conference on Computer and Communications Security*. Association for Computing Machinery, Incheon (Korea), 329–342. https://doi.org/10.1145/3196494.3196544

[16] Boyu Kuang, Anmin Fu, Shui Yu, Guomin Yang, Mang Su, and Yuqing Zhang. 2019. ESDRA: An Efficient and Secure Distributed Remote Attestation Scheme for IoT Swarms. *IEEE Internet of Things Journal* 6, 5 (May 2019), 8372–8383. https://doi.org/10.1109/JIOT.2019.2917223

[17] Cong Nie. 2007. Dynamic root of trust in trusted computing. TKK T1105290 Seminar on Network Security. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a446b7b281c55d4e94bb79c034cf3f2d86bd3267

[18] Daniele Perito and Gene Tsudik. 2010. Secure code update for embedded devices via proofs of secure erasure. In *15th European Conference on Research in Computer Security*. Springer Berlin Heidelberg, Athens (Greece), 643–662. https://doi.org/10.1007/978-3-642-15497-3_39

[19] Sandro Pinto and Nuno Santos. 2019. Demystifying ARM TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* 51, 6, Article 130 (Jan. 2019), 36 pages. https://doi.org/10.1145/3291047

[20] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted Execution Environment: What It is, and What It is Not. In *IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. IEEE, Helsinki (Finland), 57–64. https://doi.org/10.1109/Trustcom.2015.357

[21] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. 2005. Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems. In *20th ACM Symposium on Operating Systems Principles (SOSP '05)*. ACM, Brighton (UK), 1–16. https://doi.org/10.1145/1095810.1095812

[22] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. 2004. SWATT: softWare-based attestation for embedded devices. In *IEEE Symposium on Security and Privacy, 2004*. IEEE, Oakland (CA, USA), 272–282. https://doi.org/10.1109/SECPRI.2004.1301329

[23] Hailun Tan, Gene Tsudik, and Sanjay Jha. 2017. MTRA: Multiple-tier remote attestation in IoT networks. In *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, Austin (TX, USA), 1–9. https://doi.org/10.1109/CNS.2017.8228638

[24] TCG. 2014. Trusted Computing Group Protection Profile PC Client Specific Trusted Platform Module TPM Family 1.2. https://www.commoncriteriaportal.org/files/ppfiles/pp0030b.pdf

[25] Allan Tomlinson. 2017. *Introduction to the TPM*. Springer International Publishing, Cham, 173–191. https://doi.org/10.1007/978-3-319-50500-8_7

[26] Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. 2016. Towards Smart Factory for Industry 4.0: A Self-organized Multi-agent System with Big Data Based Feedback and Coordination. *Computer Networks* 101 (June 2016), 158–168. https://doi.org/10.1016/j.comnet.2015.12.017

[27] Wenjuan Xu, Xinwen Zhang, Hongxin Hu, Gail-Joon Ahn, and Jean-Pierre Seifert. 2011. Remote Attestation with Domain-Based Integrity Model and Policy Analysis. *IEEE Transactions on Dependable and Secure Computing* 9, 3 (Dec. 2011), 429–442. https://doi.org/10.1109/TDSC.2011.61