Collective Remote Attestation at the Internet of Things Scale: State-of-the-art and Future Challenges

M. Ambrosin, *Member, IEEE*, M. Conti, *Senior Member, IEEE*,
R. Lazzeretti, *Senior Member, IEEE*, M. Rabbani(*), S. Ranise

***Abstract*—In recent years, the booming of Internet of Things (IoT) has populated the world with billions of smart devices that implement novel services and applications. The potential for cyberattacks on IoT systems have called for new solutions from the research community. Remote attestation is a widely used technique that allows a verifier to identify software compromise on a remote platform (called prover). Traditional challenge-response remote attestation protocols between the verifier and a single prover face a severe scalability challenge when they are applied to large scale IoT systems. To tackle this issue, recently researchers have started developing attestation schemes, which we refer to as Collective Remote Attestation (CRA) schemes, that are capable of remotely performing attestation of large networks of IoT devices.**

In this paper, after providing the reader with a background on remote attestation, we survey and analyze existing CRA schemes. We present an analysis of their advantages and disadvantages, as well as of their effectiveness against a reference attacker model. We focus our attention on CRA schemes' characteristics and adversarial mitigation capabilities. We finally highlight open research issues and give possible directions for mitigating both the limitations of existing schemes, and new emerging challenges. We believe this work can help guiding the design of current and future proposals for CRA.

I. Introduction

Internet of Things (IoT) [1] devices are typically tiny embedded and low-power devices that enable a wide range of applications, from wearable applications, e.g., medical wearable devices, to smart homes, or smart factories [2]. These small devices may be employed in interconnected groups, and perform critical operations. Despite their potential in facilitating new applications and services, their low-cost nature, pervasivness, and often reduced set of security capabilities makes them an attractive target for cyber attacks. IoT devices may present vulnerabilities that adversaries could exploit [3] to mount an attack. This poses a potential danger to services' availability and, in certain applications, users' data privacy [4]. As an example, in 2016 hackers launched a Distributed Denial of Service (DDoS) attack on the website `krebsonsecurity.com`[1] by using two botnets of 980,000 and 500,000 hacked devices, mostly cameras. Furthermore, recently researchers have shown that security cameras affected by malware can receive covert signals and leak sensitive information from the very same surveillance system which is meant to protect the facility and or data [5].

Remote Attestation (RA) [6] is an effective solution to detect software compromise on a (remote) platform, and has been successfully adopted to secure low-end devices [6]. At a high-level, a RA scheme enables a verifier to assess the integrity of the configuration (e.g., software and/or data) of a remote prover device. In order to achieve this, the verifier engages in an interactive procedure with the prover, where the prover securely supplies a measurement (typically a hash) of its platform's configuration to the verifier, and the verifier matches it against a known set of measurements of "healthy" configurations.

Given the size and complexity of IoT systems, in particular swarms of resource constrained devices, current RA schemes are hard to scale [7]. In order to overcome the scalability challenges of RA, several recent research works proposed Collective Remote Attestation protocols (CRA), which allow a verifier to obtain a unique measurement from a whole network of smart devices; such measurement expresses the *collective status* of the network, and optionally the individual status of every device, effectively improving the scalability of the RA protocol. Different schemes proposed in the literature make different assumptions, e.g., they work on a hop-by-hop basis [7], or on an end-to-end basis [8], and tackle different types of attacker (e.g., software-only adversaries [7] and/or physical-adversaries [9]).

Contribution. This paper makes an effort to provide a systematic and thorough revision of the state-of-the-art CRA schemes, and to discuss their security properties w.r.t. a reference attacker model. Our broader goal is to provide the reader with the current state of the art in CRA protocols, discussing open problems and future directions. In particular, in this paper we provide the following contributions:

- We describe the system and security model for collective remote attestation, and survey the state of the art for CRA.
- We analyze the different adversary typologies and attacks that can be conducted on the IoT network.

M. Ambrosin carried out this work as an independent researcher (e-mail: ambrosin@math.unipd.it).

M. Conti is with the Department of Mathematics, University of Padua, Padua, Italy (e-mail: conti@math.unipd.it).

R. Lazzeretti is with the Department of Computer, Control, and Management Engineering "Antonio Ruberti" at Sapienza University of Rome, Rome, Italy (e-mail: lazzeretti@diag.uniroma1.it).

M. Rabbani is with the Human Inspired Technology Research Center, University of Padua, Padua, Italy (e-mail: rabbani@math.unipd.it).

S. Ranise is with Fondazione Bruno Kessler, Povo, Trento, Italy (e-mail: ranise@fbk.eu)

(*) Corresponding author.

[1] https://motherboard.vice.com/en_us/article/8q8dab/15-million-connected-cameras-DDoS-botnet-brian-krebs

- We perform a comparison between the protocols proposed in the literature in terms of their characteristics, adversarial mitigation capabilities and defensive capabilities against different attacks.
- We discuss open problems and future research directions in the field of CRA.

Organization. The rest of the paper is organized as follows. In Section II, we discuss the common system and adversarial model for CRA schemes. Section III discusses the different types of device attestation mechanisms along with their pros and cons. In Section IV, we present state-of-the-art for CRA schemes which includes their features and comparative studies. Security analysis of different CRA protocols is provided in Section V. In Section VI, we highlight open problems regarding CRA and their wide adaptability. Finally, in Section VII and Section VIII, we present possible directions and future work, and conclude the paper.

II. System and Security Model for Collective Remote Attestation

A. System Model

In this section, we outline the general system model for CRA schemes. Typically, a system-model in CRA considers a large network of low-end, embedded devices, e.g., IoT devices in smart environments, cyber-physical systems in industrial settings. These devices are heterogeneous in terms of underlying software and hardware configurations and act as provers ($\mathbf{P}$). Along them, the other major stakeholders are the network owner ($\mathbf{O}$), the verifier ($\mathbf{V}$), and aggregators ($\mathbf{A}$). In line with [7], [8], [10], we assume that devices in a large network are able to communicate and identify their direct neighbours.

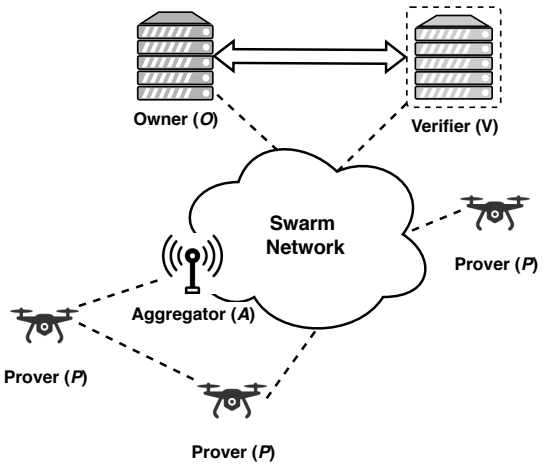


Fig. 1. Example of Swarm Network.

As shown in Figure 1, a brief description of the CRA system model is as follows:

- *Owner or Operator* ($\mathbf{O}$): The network owner ($\mathbf{O}$) is responsible for: (1) network setup and maintenance; (2) provisioning of necessary cryptographic material and credentials for attestation; and (3) (optionally) delegate a third party entity to carry out periodic attestation rounds.
- *Verifier* ($\mathbf{V}$): Throughout the CRA literature, the attestation process is usually carried out by a third-party entity that acts as a verifier (V) on behalf of the owner. $\mathbf{V}$ carries out the attestation process usually by sending an attestation request to the network, and collecting the (global) attestation result in the form of an aggregated proof. As an alternative, the attestation protocol can be triggered by provers.
- *Prover* ($\mathbf{P}$): Each device that has to be checked by the verifier is referred to as prover ($\mathbf{P}$). Provers in the network are assumed to be heterogeneous w.r.t. the underlying software and/or hardware. As a result of the attestation procedure, $\mathbf{P}$ can be considered "healty"[2] or "compromised".
- *Aggregator* ($\mathbf{A}$): The main purpose of an aggregator ($\mathbf{A}$) is to relay messages among entities in a network and, when possible, aggregate inputs from neighbors in the topology. This entity is first explicitly introduced in [8]. In a decentralized environment, within a network each prover can act as an aggregator.

B. Reference Attacker Model for CRA

Adversaries are interested in compromising IoT network services to perform attacks on authentication, network availability, and service integrity[3]. In order to succeed, they are hence interested in compromising devices and evade detection by CRA schemes. In what follows, we summarize the attacker models for CRA considered in the literature. The CRA and RA literature, typically deal with the following types of attacker:

- *Software Adversary* (Adv_{SW}). This type of adversary, also regarded as Remote Adversary in [12], has the ability to run malicious code or firmware on a device.
- *Mobile Software Adversary* (Adv_{MSW}). This adversary [13] is capable of compromising the software configuration of a device and then to eliminate any trace of its presence from the device (e.g., he is capable of erasing the malware used to compromise the device).
- *Physical Non-Intrusive Adversary* (Adv_{PNI}). The attacker [12] is in the proximity of the device and may infer information from the devices, e.g., using side-channel attacks.
- *Stealthy Physical Intrusive Adversary* (Adv_{SPI}). This attacker [12] is capable of capturing a device, and may attempt to exfiltrate information from it.
- *Physical Intrusive Adversary* (Adv_{PI}). This attacker [12] is not only capable of capturing a device, but also to introduce external hardware on it.

III. Background: Device Remote Attestation

Device Remote Attestation (RA) is a well established technique and has been studied extensively in the context of

[2]A device is healthy if it is running latest legitimate software version.

[3]The reader may refer to the survey in [11] for an extensive treatment of attacks on Wireless Sensor Network systems.

IoT [6], [12]. RA allows $\mathbf{V}$ to obtain a proof of the integrity of the configuration (e.g., software, or data) of $\mathbf{P}$. This is carried out over an interactive protocol between $\mathbf{V}$ and $\mathbf{P}$.

RA for IoT devices can be performed in several ways, with different requirements in terms of device capabilities and equipment, and security guarantees. At a high-level, we can distinguish between *software-based attestation*, and *attestation based on root of trust*. The remaining of this section briefly describes each of them. Note that the following does not intend to be a thorough treatment of the various attestation techniques, for which we point the reader to the works in [6], [14]. Rather, in this section we provide the necessary background information to the reader on remote device attestation, in order to make the exposition self contained.

A. Software-Based Attestation

Software-based RA protocols typically rely on timing information to allow $\mathbf{V}$ to assess the correctness of the firmware running on the prover, and have little or no special hard requirements on provers. These approaches usually imply strict timing requirements on the network, which may not be feasible in any generic IoT environment. An example of software-based RA protocol is SoftWare-based ATTestation (SWATT) [15], which leverages the fact that a malicious firmware running on a (compromised) node must redirect the memory access to the memory location where the original code resides, in order to get a valid response to an attestation request. The overhead introduced by this memory redirection has a direct impact on the overall runtime, and thus, on the necessary amount of time for the prover to respond. This would make a compromise detectable by a remote verifier.

Furthermore, software-based RA schemes such as [15]–[17] usually depend on strong assumptions regarding adversarial capabilities. Moreover, these schemes work in scenarios where $\mathbf{V}$ communicates with $\mathbf{P}$ in a one-hop network setting. This makes them hard to deploy over large networks with multi-hop distance between the verifier and provers.

B. Attestation Based on Root of Trust

To overcome some of the limitations of software-based RA, several existing schemes rely on a *Root of Trust* ($R_{\mathbf{P}}$) residing inside $\mathbf{P}$. $R_{\mathbf{P}}$ is assumed to be trusted, and is the endpoint of the attestation protocol. In practice, $R_{\mathbf{P}}$ typically consists of some combination of hardware and software capabilities [14].

An example of attestation based on $R_{\mathbf{P}}$ is Measured Boot, which enables the verification of the integrity of a system at boot time; Measured Boot relies on a $R_{\mathbf{P}}$ that comprises a trusted bootloader and a Trusted Platform Module (TPM) [18]. During boot, integrity measurements of the memory are "recorded" into TPM's Platform Configuration Registers (PCRs), and sent to a remote verifier for verification.

When it comes to IoT security, $R_{\mathbf{P}}$ is realized by leveraging hardware providing minimal security capabilities, in particular code and memory isolation [19]. Examples of research platforms for embedded systems with such capabilities are SMART [20], SPM [21], SANCUS [22] and TyTAN [23]; commercial solutions, such as ARM TrustZone[4], are already available on popular IoT platforms. We will refer to the above as Trusted Execution Environment (TEE) technologies.

Remote attestation schemes based on Root of Trust in IoT assume either a shared secret between $\mathbf{V}$ and $\mathbf{P}$ (i.e., a symmetric key k), or public key cryptography techniques (e.g., traditional public key cryptography, or more complex aggregate multi-signatures [8]). Symmetric cryptography introduces a considerably lower overhead than public key cryptography in resource-constrained devices [7]; as such, it is generally considered to be the preferable security solution to adopt in the IoT space.

From the literature, independently from the cryptographic scheme in use, and based on different assumptions and/or prover's capabilities, we can distinguish, at a high-level, four strategies for a remote attestation protocol based on Root of Trust: *(1) Interactive RA*, *(2) Interactive Self-RA*, *(3) Non-interactive RA*, and *(4) Non-interactive Self-RA*. In what follows, for simplicity we are going to show each of them using a symmetric key scheme.

1) Type 1: Interactive Remote Attestation: This is the most widely used method. It consists of an interactive protocol between $\mathbf{V}$ and $\mathbf{P}$: $\mathbf{V}$ sends a challenge N to $\mathbf{P}$'s $R_{\mathbf{P}}$, which responds with a proof of the device's configuration. This proof usually consists of a hash of $\mathbf{P}$'s configuration $c \leftarrow \texttt{hash}(\texttt{conf}(\mathbf{P}))$, concatenated with N. This proof is signed, in case of public key cryptography, or tagged, in case of symmetric cryptography, using a Message Authentication Code (MAC). In the latter case, the result is a value h. $\mathbf{V}$ verifies the integrity of $\mathbf{P}$'s configuration by verifying the authenticity of h, that is, recomputing h as a MAC using the shared key k over $c'||N$ (or verifying the signature using $\mathbf{P}$'s public key), where c' is the expected configuration for $\mathbf{P}$. Note that, in the simplest scenario where the expected configuration is one and known in advance, only the MAC (or signarure, if using a public key scheme) is sent to $\mathbf{V}$. The protocol is shown in Figure 2. The scheme can be trivially extended to pass along also the computed measurement c to $\mathbf{V}$.

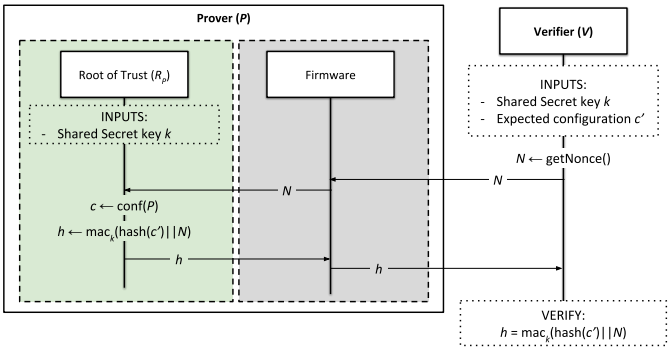


Fig. 2. Type 1: Interactive RA.

2) Type 2: Interactive Self-RA: Leveraging the capabilities of the TEE, it is possible to perform the verification of c directly at $\mathbf{P}$'s side, provided that the list $C = \{c_1, c_2, \ldots, c_n\}$ of potential allowed configurations for $\mathbf{P}$ is securely installed

[4]https://www.arm.com/products/security-on-arm/trustzone

in a write-protected area of the device, and accessible by $R_{\mathbf{P}}$. This type of attestation is depicted in Figure 3. In this case, after receiving N from $\mathbf{V}$, and computing the measurement $c \leftarrow \texttt{hash}(\texttt{conf}(\mathbf{P}))$ of $\mathbf{P}$'s configuration, $R_{\mathbf{P}}$ produces a signed/MACed token h authenticating the binary result $r \in \{\texttt{TRUE},\texttt{FALSE}\}$ of the attestation process, which is then delivered to $\mathbf{V}$. Self-attestation allows $R_{\mathbf{P}}$ to produce a customized token to communicate the result to $\mathbf{V}$, which may be needed, e.g., to scale attestation collection [8]. The main disadvantage of this approach is the need for C to be pre-installed in $\mathbf{P}$'s memory, and/or securely updated.

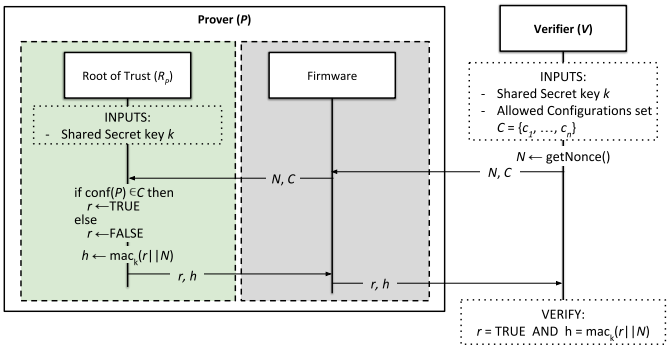


Fig. 3. Type 2: Interactive Self-RA.

3) Type 3: Non-interactive RA: Recently, protocols such as SeED [24] and ERASMUS [13] modify the attestation protocol flow by allowing $\mathbf{P}$ to autonomously determine the time at which attestation happens, and to locally generate a pseudo-random nonce N (ERASMUS uses time t to ensure freshness). This removes the need for $\mathbf{V}$ to initiate the process, giving more freedom to the specific device (see Figure 4). This technique, however, requires additional hardware requirements compared to Type 1 and Type 2 techniques. In particular, non-interactive attestation protocols require some secure source of time. SeED requires [24] a reliable Real Time Clock (RTC), to correctly report the time at which the attestation result refers to; and an Attestation Trigger (AT) circuit, which triggers the attestation process at unpredictable points in time (using a pseudo-random function). ERASMUS requires a Reliable Read-Only Clock (RROC), which is not modifiable by software.

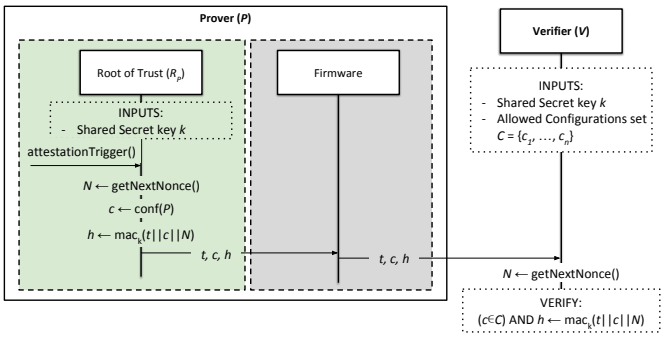


Fig. 4. Type 3: Non-Interactive RA.

4) Type 4: Non-interactive Self-RA: This is a variation of Type 3 used in [25] where, similar to Type 2, $\mathbf{P}$'s TEE knows the set of known configurations $C = \{c_1, c_2, \ldots, c_n\}$, and can therefore perform self-attestation, when triggered by the secure hardware (e.g., by the AT in [24]). The protocol is shown in Figure 5.

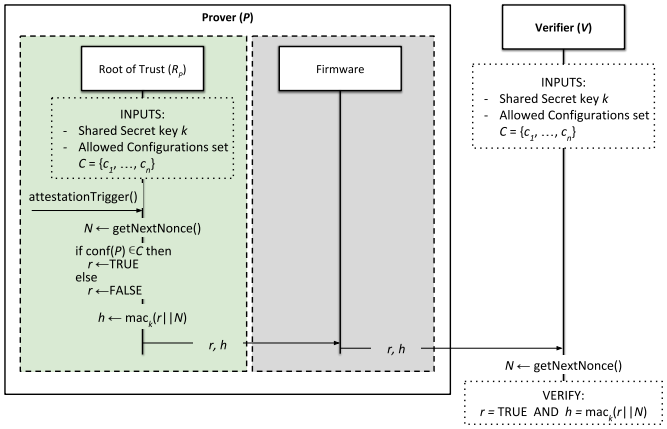


Fig. 5. Type 4: Non-Interactive Self-RA.

IV. State-of-the-art for Collective Remote Attestation

In this section, we overview the CRA literature. All of the works proposed in the literature adopt a similar approach: reduce the communication and computation of the attestation process by securely "offloading" the execution of certain operations to the nodes themselves. Furthermore, all the CRA schemes presented in the literature assume provers are equipped with a hardware-enabled TEE (e.g., TyTAN [23]).

A. CRA Schemes Description

SEDA. The work by Asokan et al. [7] first highlighted the scalability challenges of remote attestation for large swarms of low-end devices; the authors proposed SEDA (Scalable Embedded Device Attestation), a scalable protocol for collective attestation. SEDA allows the verifier to efficiently perform attestation over an (overlay) spanning tree, rooted at the verifier. The protocol comprises two phases: (1) an *offline* phase; and (2) an *online* phase. The offline phase comprises: (i) Device Initialization, performed by the owner (or operator) $\mathbf{O}$ in a trusted setting, which provides provers with their expected correct configuration $\texttt{conf}(\mathbf{P}_i) = c_i$, signed by $\mathbf{O}$ (it may be different per each prover $\mathbf{P}_i$), and credentials; and (ii) Device Registration, where nodes securely establish pairwise keys with their neighbors (e.g., wirelessly reachable devices) using their credentials. The online phase of the protocol is initialized by $\mathbf{V}$, which broadcasts an attestation request to the swarm creating a spanning tree. Each prover, after verifying the authenticity of the request, propagates it to their children in the spanning tree. Every prover $\mathbf{P}_i$ then attests its children using the shared keys established during the offline phase, and sends an "accumulated" result to the parent node indicating whether the subtree routed at $\mathbf{P}_i$ has correct configuration (1) or not (0), as shown in Figure 6.

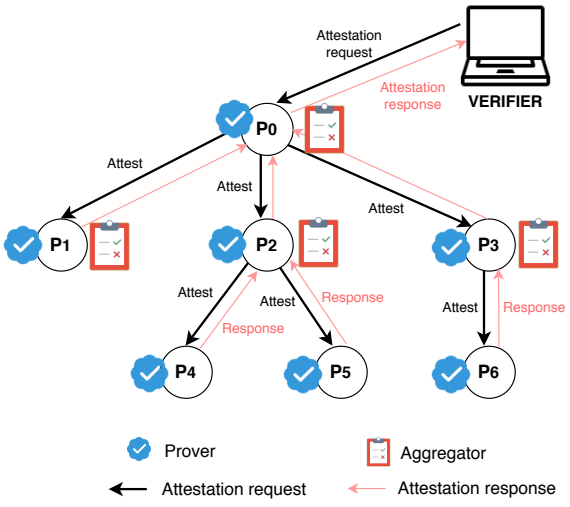


Fig. 6. Spanning Tree based CRA.

This process continues along the spanning tree until reaching $\mathbf{V}$, which assesses the status of the swarm by verifying only the last attestation report. SEDA offers some degree of protection against DoS attacks by limiting the "join-request" frequency or by making them low-priority tasks in the network. The authors of [7] also proposed a variant of SEDA where every device piggybacks the IDs of the compromised devices in the subtree alongside the aggregated attestation result.

While a noteworthy first step towards a more scalable attestation protocol, SEDA has some important limitations. First, it requires the network to be "stable", that is, every node will maintain a fixed set of neighbors throughout the operation time. Second, configurations are statically deployed during the offline phase, and it is unclear how they are updated, e.g., in case of Over-The-Air (OTA) firmware update. Third, the attacker model considered by this protocol is limited to a software-only attacker (Adv_{SW}).

DARPA. Ibrahim et al. [9] proposed DARPA as an attempt to complement SEDA [7] to detect device captured by stronger attackers, i.e., Adv_{SPI} and Adv_{PI}, which may evade detection through attestation. The main essence of this protocol is to detect whether an attack has been occurred rather than detecting the individual malicious devices, i.e., detecting whether a device has been captured. The assumption is that any physical attack requires a non-negligible time t_{cap} to be carried out. Nodes periodically run an *absence detection* protocol with their neighbors, recording present devices in a secure log. Devices generates "heartbeat" messages, along with a timestamp, signed with their secret key and share them to the immediate neighbours. During every absence detection protocol run, every device initiates its own heartbeat based on either an internal secure timer, or an heartbeat message received from one of its neighbours. The neighbouring devices exchange their respective heartbeats. In this way this timestamp-based heartbeats of the devices are stored into a log file, which is then transferred to $\mathbf{V}$ upon the protocol time out.

SANA. Ambrosin et al. [8] proposed SANA that addresses some of the limitations of SEDA [7]. SANA relies on a novel cryptographic method, called as Optimistic Aggregate Signatures (OAS) to aggregate the attestation result of a network through untrusted aggregators. OAS is a multi-party signature scheme which enables SANA to efficiently aggregate attestation report irrespective of the number of the signers due to its short signature size and short verification time. SANA also employs token for verifier assigned by the network owner. This step helps to counter DoS attack, as only legitimate verifiers can initiate attestation requests for the devices in the network. In SANA, the attestation report is publicly verifiable and provides details of compromised devices in the network. Although SANA resolves many shortcomings of SEDA, it still requires full connectivity among devices; furthermore, the aggregate signature scheme introduces severe overhead on low-end devices with respect to computation.

LISA. In [26], Carpent et al. proposed two Lightweight Swarm Attestation (LISA) protocols, LISAα and LISA$\mathtt{s}$. These protocols improve SEDA [7] with respect to scalability and resiliency to physical adversaries. Both versions of the protocol operate over a spanning tree topology. LISAα is the asynchronous version of the protocol. Authors describe the Finite State Machines (FSMs) for provers and the verifier to carry out the attestation procedure in the swarm. Differently from SEDA, the nodes in LISAα collaborate only for propagating the attestation requests, and responses. Furthermore, provers in LISAα verify the session of an attestation response before forwarding it to the parent node in the tree (to prevent reply attacks and/or old messages to reach the verifier). LISA$\mathtt{s}$ is the synchronous version of the protocol, and is similar to SEDA in nature: nodes help "aggregating" attestation response messages by attesting each children in the spanning tree, and sending upstream in the tree only its attestation, together with a list of IDs of the (attested) children.

An additional contribution of the paper in [26] consists in the introduction of a metric named "Quality of Swarm Attestation" (QoSA); QoSA compares the quality of different swarm-attestation protocols based on the report generated by swarm-attestation protocols and the information they yield. Both protocols in LISA use a master key shared among all the devices in the swarm in order to make the swarm attestation less complex. As a consequence, in this scenario even a single compromised node may result in the compromise of the whole network. In cases where this problem is a concern (e.g., in case of a physical attacker), the authors suggest to use a Public Key Infrastructure (PKI) to establish trust among entities, at the price, however, of additional complexity in the system. Despite the effort for bringing collective remote attestation closer to reality, LISAα and LISA$\mathtt{s}$ require full connectivity among nodes during the attestation process, which may limit their applicability in dynamic networks or where there is intermittent connectivity among nodes; it is noteworthy, however, the effort by the authors in trying to formally define the connectivity requirements, in terms of time the nodes shall remain connected for the protocols to succeed.

SeED. The protocol proposed in [24], SeED, enables attestation to be initiated by provers, rather than the verifier. This translates in a reduced energy cost, communication overhead and run-time compared to other verifier-initiated protocols, such as SEDA [7], as well as additional protection against DoS attacks on end devices (e.g., from a malicious verifier). SeED per se is not a CRA protocol, but relies on SEDA [7] to efficiently deliver attestation reports to the verifier. The security of SeED relies on a secure random seed shared between every device and $\mathbf{O}$ during the initial device bootstrapping phase; end devices (provers) store their seed in a Memory Protection Unit (MPU). This shared seed is used to generate a pseudo-random sequence of times at which execute attestation (based on a pseudo-random number generator), preventing a mobile attacker Adv_{MSW} to anticipate the next scheduled attestation round. Every device in the network generates its respective attestation report based on the randomly generated time, which is delivered to $\mathbf{V}$ using SEDA [7]. As a consequence, SeED inherits the same limitations of SEDA.

SCAPI. Kohnhauser et al. [27] presented SCAPI, which improves DARPA [9] by reporting the exact captured devices. In SCAPI the authors propose to periodically distribute a session key among all the devices that are not physically compromised. A "leader" device generates a secret session key for the subsequent time period. When a new session key is generated, devices need to authenticate with the old session key in order to receive the updated session key. The security of SCAPI relies on the assumption that a Adv_{PI} cannot capture a device without turning a device off for a noticeable amount of time. In this way, the captured devices will not be able to obtain the updated session key, and consequently, the protocol allows the detection of physical attacks. SCAPI improves DARPA w.r.t. network communication, energy consumption and run time. However, SCAPI relies on other CRA protocols like [7], [8] for scaling and consequently it has limited application in highly dynamic networks or where there is intermittent connectivity among nodes.

ERASMUS. In [13], Carpent et al. proposed ERASMUS, a RA protocol that, differently from existing RA schemes, aims at detecting mobile adversaries Adv_{MSW} and reducing computation requirements for provers. In ERASMUS, provers self-attest at pre-defined time intervals (Type 3 attestation in Section III-B), and locally store up to n consecutive measurements in its untrusted storage. $\mathbf{V}$ occasionally collects and verifies the latest k measurements from the swarm of (potentially unattended) provers. This technique enables $\mathbf{V}$ to identify whether there is any presence of a mobile adversary between two successive attestation occurrences. ERASMUS achieves roughly 2X improvement w.r.t. simple interactive RA: ≈ 0.3 seconds runtime vs ≈ 0.6 seconds on a single device with memory size of 10 MBytes. Authors claim that this unique feature makes ERASMUS as an ideal candidate for device attestation of unattended or safety-critical resource constraint devices. As for CRA on swarms, ERASMUS can leverage techniques such as SEDA or LISA. However, different from the original schemes that use Type 1 or Type 2 attestation (Section III-B), ERASMUS simplifies the collection phase, which does not involve any online computation (provers simply hand over k store measurements to $\mathbf{V}$); this makes ERASMUS a better fit to highly mobile use cases.

SALAD. Kohnhauser et al. proposed Secure and Lightweight Attestation for Highly dynamic or disruptive networks (SALAD) [28]. SALAD per se does not provide any novel attestation mechanisms. Rather, its main focus is to provide a lightweight message aggregation scheme for highly dynamic networks which has intermittent connectivity. SALAD works in a distributed fashion. $\mathbf{V}$ starts the CRA protocol by sending a (signed) request to a device. The device exchanges the request with its neighbors (i.e., devices in its communication range), and computes its measurement. Neighboring devices also mutually exchange their attestation reports, and locally aggregate them. Eventually, this leads to a state where every node has an (aggregated) attestation result for the whole network. $\mathbf{V}$ can connect to one of the devices to obtain the attestation result for the whole network. In order to optimize storage and communication overhead, SALAD relies on aggregation of Message Authentication Codes (MACs), proposing two MAC schemes: `MACGreedy`, whose goal is to minimize the storage consumption of all the attestation reports, by aggregating MACs in a greedy manner via XOR, and `MACSmart`, which aims at minimizing the size of the transmitted reports among devices. Both MAC schemes try to eliminate duplicate information contained in aggregated attestation reports. As this is equivalent to solving the set coverage problem and the set packing problem, the authors utilize some heuristics that are suitable to resource-constrained devices.

PADS. Ambrosin et al. [25] presented PADS, a protocol that, similar to SALAD, focuses on highly dynamic swarm topologies. PADS turns the problem of attesting a swarm of devices into a consensus problem. Each prover in PADS attests itself using a non-interactive attestation technique similar to SeED [24] (Type 3 attestation), and then broadcasts an attestation report to its neighbors (i.e., devices within its communication range) for a certain number of times. The attestation report contains a representation of the health status of the network from the point of view of the prover, in the form of a bitmap, and it is signed by the TEE of the device. Devices efficiently merge their knowledge of the network with the one received from their neighbors using the `AND` operator. Over time, the view of the network of every prover will converge to the true state of the network. $\mathbf{V}$ can then query an arbitrary device (or a subset of them) for the CRA result. The main advantage of PADS w.r.t. other CRA schemes is its resiliency w.r.t. network topology changes; as devices continue to broadcast for a certain amount of time their "view of the network", and lost transmissions do not affect the result collection. PADS does not rely on a spannig tree to perform attestation. Furthermore, PADS allows an attestation report to provide "partial" knowledge of the network, and introduces the notion of *coverage*, to quantify the completeness of the

information.

Similar to LISA [26], PADS uses symmetric cryptography for efficiency reasons, and a single key is shared by all the devices. This assumption reduces the overall security of the system, making it vulnerable to a hardware adversary.

WISE. Ammar et al. [29] recently proposed a novel approach to perform remote attestation of large-scale IoT network by employing resource efficient machine learning techniques based on Hidden Markov model. The main goal of WISE is to perform attestation over a carefully selected subset of the provers, to reduce the complexity of the attestation process, and reduce memory and energy consumption on devices. WISE consists of two phases. The first phase is an offline phase, where the owner of the deployment provisions the provers with some key material, and groups them into clusters (e.g., based on their geographic location). The second phase is triggered by the verifier ($\mathbf{V}$), which runs a collective attestation protocol over a *subset of devices* (instead of the entire swarm) that are selected depending on: (i) the attestation history of every device, and (ii) some individual characteristics of these devices and constraints (e.g., number of compromised devices in the device's neighborhood, maximum amount of time between attestation rounds for the device, etc.). The first round of attestation is performed for all the network. Similar to other schemes, it uses a spanning tree to propagate the request, and aggregates the response. The "wiser" selection of targets is performed in subsequent rounds of the protocol. The use of spanning tree for communicating and aggregating the results makes the protocol more prone to communication errors, in case of high mobility.

Apart from providing smart, efficient and clustering based RA approach, WISE also provides security against the roving malware[5] due to its unpredictable and variable attestation time and frequency which differs among types of devices in a mesh network.

slimIoT. Ammar et al. proposed slimIoT [30], a protocol that improves SCAPI [27] in the following aspects: (1) unlike SCAPI, it does not assume that majority of the total devices in the network should be healthy, slimIoT can accurately identify the compromised devices with the assumption that at least one device in the network remains healthy; (2) it allows device mobility during attestation. slimIoT organizes the devices into clusters that are periodically attested at pre-define time intervals called *epochs*; these epochs are chosen s.t. the length of the interval is smaller than the time needed by a physical adversary Adv_{PI} to mount an attack and remain undetected. The absence of any device from the cluster indicates the possibility of a physical adversary. slimIoT uses symmetric key cryptography to rely on authenticated parameterized broadcast messages. slimIoT relies on a one-way keychain composed on n symmetric keys, built s.t. $k_{i-1} = \texttt{hash}(k_i)$. In a initialization phase, devices are provided with key k_0, so that they can verify the authenticity of every key used, as $\texttt{hash}(\texttt{hash}(\texttt{hash}(...\texttt{hash}(k_i)...))) = k_0$, but cannot construct any k_i from k_0. Each key is mapped to a series of non-overlapping time intervals, where each epoch is divided into four time sub-intervals. Provers use delayed key disclosure (similar to other broadcast authentication protocols such as TESLA [31]) to authenticate messages from $\mathbf{V}$. During the online phase, at the beginning of the first time sub-interval of each epoch, $\mathbf{V}$ issues four messages: (1) a message with a fresh nonce authenticated by a Hash Message Authentication Code (HMAC), (2) an encrypted and HMAC-authenticated message with the attestation request (using an encryption key derived from the key corresponding to the 2nd time sub-interval in the epoch k_2), (3) a key disclosure message for the key used to compute the HMAC on the first message, and (4) a key disclosure message for the key used to compute the HMAC on the second message. Each prover performs a variation of the Type 2 attestation in Section III-B. Every attestation report carries the ID of the prover, a bit array, s.t., value `1` at position i indicates that prover $\mathbf{P}_i$ is in a correct status, and an authentication value computed by hashing a nonce value received in the (encrypted) attestation request and the correct prover configuration; message authenticity is achieved by HMAC using k_0. Broadcast messages from $\mathbf{V}$ are propagated by devices along the spanning tree maintained by the network; certain fields of the attestation response are aggregated on a hop-by-hop manner using XOR. slimIoT tolerates temporary disconnections of devices by mapping keys to time sub-intervals: if a device misses an attestation round, it will still be able to participate once connection is available in the next epoch, as it will still be able to verify messages from $\mathbf{V}$. Still, slimIoT requires devices to maintain a static formation for the duration of the attestation protocol instance.

SAP. Nunes et al. [32] defined Timely Collective Attestation (TCA) (i.e., an use-case formal model for CRA) model definition and systematically designed a Synchronous Attestation Protocol (SAP) based on TCA. The TCA model serves as an analyzing base for the other proposed CRA schemes and helps in identifying the main design requirements for a CRA approach. The design specifications include parameters like network topology, devices specifications (e.g., hardware and software), adversarial model, device computation power, network communication and attestation results. Attestation results are again further subdivided based on the Quality of Attestation, as proposed in [26]. The proposed TCA model systematically treats any CRA based on the TCA-Efficiency, TCA-Soundness and TCA-Security.

In SAP, the authors propose to construct the swarm network as a balanced binary tree in which the root is the verifier. The attestation challenge propagates along the tree and a secure clock on every device guarantees the verification of the attestation challenge at time t_{att}. Upon receiving the challenge, each device performs attestation and sends the result to its parent. In [32], a parent node performs XOR operation over received result along with own attestation result and forwards the result to its own parent. Upon successful completion of the network-wide attestation, $\mathbf{V}$ receives the

[5]An advanced type of malware that is knowledgeable about the attestation schedule and therefore only active between any two successive attestation routines. It also has the ability to delete itself at the beginning of the attestation to avoid detection.

XORed result of the swarm S and validates the result. SAP depends on symmetric key based cryptography and employs secure, read-only clock to synchronize the attestation time with other nodes in S. In this work, the clock is instrumental to provide security against DoS or man in the middle attacks by validating the attestation time. While SAP provides the base for comprehensive and categorical analysis of any CRA based on the TCA-model, it does not support device mobility. In addition, SAP is not resilient against a stronger attacker such as a physical adversary or a run-time attack. Moreover, SAP does not allow the $\mathbf{V}$ to identify specific malicious nodes in the network because the attestation result of SAP is only a binary output which shows whether the entire S is trustworthy or not.

MTRA. Tan et al. [33] proposed a hierarchical attestation framework for IoT networks named Multiple-Tier Remote Attestation protocol (MTRA). In MTRA there are three types of entities: the trusted verifier $\mathbf{V}$ (e.g., a base station), high-end devices equipped with a Trusted Platform Module (TPM), and low end devices without any secure hardware capability. $\mathbf{V}$ is directly connected with TPM-equipped devices; low-end devices are assumed to be 1-hop connected to a TPM-equipped device, which acts as cluster head(s) in the wireless network. The attestation process is divided into two phases, an offline phase, and an online phase. During the offline phase, $\mathbf{V}$ provides TPM-enabled devices with unique key pairs that are bound to the state of TPM's PCR registers (this, in turn, depends on the device configuration, and can be populated, e.g., through measured boot). $\mathbf{V}$ generates a key chain, i.e., a sequence of $n+1$ keys, $k_0, \ldots, k_n$ such that $k_i = \texttt{hash}(k_{i+1})$, and distributes k_n to all the TPM-equipped devices, sealed (encrypted) against the correct status of the PCRs using the `TPMSeal` function; the device will later be able to unseal (decrypt) this value if the status of its PCRs is valid. End devices are initialized with their application code, and the remaining of their memory is filled with unique random incompressible bits.

In the online phase, $\mathbf{V}$ initiates the i-th round of attestation sending a challenge to one of the TPM-equipped cluster-heads. $\mathbf{V}$ attests the TPM-equipped device first (Type 1 attestation in Section III-B), which in turn will attest the end devices. The challenge contains a nonce and k_{n-i}, and is encrypted with k_{n-i+1}. The TPM-equipped devices have k_{n-i+1} locally stored and sealed (initially k_n). The TPM-equipped device unseals (`TPMUnseal`) k_{n-i+1} (note that, if the boot sequence populated the PCRs with an invalid configuration, this would not be possible) and decrypts the challenge; then, it verifies that $k_{n-i+1} = hash(k_{n-i})$. If this matches (this is used to authenticate the attestation request), the TPM-equipped device sends its configuration to $\mathbf{V}$ (as explained in the paper, the fact that the keys are sealed against PCRs in the device ensures the devices is in a correct state), which verifies it. If the TPM-equipped device does not respond within a certain amount of time, $\mathbf{V}$ assumes it is compromised. The TPM-equipped device then stores and seals k_{n-1}. The provers connected to the TPM-equipped device are attested via software-based attestation, performing similar operations as the TPM-equipped devices, but sending to $\mathbf{V}$ (through the TPM-equipped device) the hash of their entire program flash. The authors argue that this process of measuring the whole flash will prevent adversaries to perform Time Of Check To Time Of Use [34] (TOCTTOU) attacks[6]; furthermore, they claim that the use of key chains prevents other type of network attacks, such as rainbow attacks and wormhole attacks. Local key chains can be employed between TPM-equipped devices and end IoT devices to thwart wormhole attacks. Finally, as acknowledged by the authors, MTRA works only for static networks.

RADIS. Conti et al. proposed RADIS [35], a CRA scheme that turns the CRA problem into a distributed service attestation problem. RADIS aims to detect compromised devices in a distributed IoT service and the legitimate IoT devices that are performing a malicious operation due to corrupted communication data exchanged with the compromised devices. RADIS relies on the so called *control flow attestation*, which aims at measuring the integrity of the execution of a software, rather than only its static properties; this is to detect runtime attacks, such as code-reuse attacks [36]–[38]. Figure 7 shows two services Service S_i and Service S_j (running on Prover $\mathbf{P}_i$ and $\mathbf{P}_j$ respectively) which compose a distributed IoT service. Below each service, it is depicted the corresponding control-flow graph, where the nodes of the graph are the code instructions. The main idea of RADIS is to represent the entire execution flow of a distributed IoT service as a single hash value. RADIS computes this single hash with an accumulative hash algorithm, where for each node N the hash is calculated as $H_i = \texttt{hash}(H_{i-1}, N)$.

The protocol starts with $\mathbf{V}$ sending an attestation request for a service S_i to a prover $\mathbf{P}_i$. When the execution of a distributed service starts at node A (as shown in Figure 7), $\mathbf{P}_i$ will compute the local control flow starting from 0, and will maintain an accumulative hash value for entire service flow execution. When the service S_i in $\mathbf{P}_i$ needs to use another service S_j on another prover $\mathbf{P}_j$, the trust anchor in such device will validate the request from $\mathbf{P}_i$ and this time the computation of the control-flow attestation measurement will not start again from 0 but from the hash of the calling service e.g., computation of H3 in Figure 7. Clearly, if S_j in turn requires another service, this process will be repeated and the accumulated result will be returned to $\mathbf{P}_i$. The final result will then be the accumulation of the entire execution flow of the distributed IoT service.

RADIS improves the state of the art by tackling runtime attacks. However, the need for precomputing all the possible control flows, not only within one device, but for the interactions among devices in the network, along with the need for additional specialized hardware, makes RADIS hard to scale to large systems, and mostly usable for rather small and controlled IoT environments (such as a Smart Home IoT system).

ESDRA. Kuang et al. proposed Efficient and Secure Dis-

[6]Remote attestation checks the state of a device at a given time, but does not provide information about the device's state between consecutive RA executions or between attestation and following use of the device. Therefore, transient malware may be undetected.

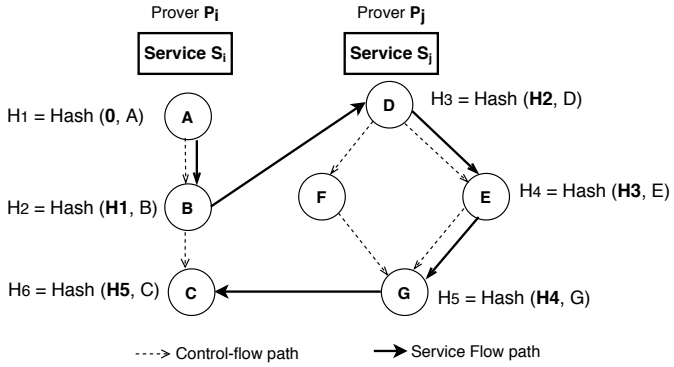


Fig. 7. Attestation of distributed IoT services

tributed Remote Attestation (ESDRA) [39], which aims to reduce the possibility of single node failure. In ESDRA, the network is divided into clusters based on the communication distance among nodes; one of the nodes within a cluster acts as a cluster-head. In ESDRA, every node is attested by three of its neighbours; the cluster head re-checks and sends the attestation report to the verifier. The ESDRA protocol uses elliptic curve cryptography to generate asymmetric key pairs *sk, pk*. These key pairs will be assigned to devices at the bootstrapping phase. To attest a node in the network, ESDRA employs a reputation mechanism, which defines the reputations of nodes based on their previous behaviors. Every node in ESDRA maintains a reputation score (called by the authors *credit score*) for all the neighbours it attests. The score for a $\mathbf{P}$ increases if the $\mathbf{P}$ can be successfully attested by another $\mathbf{P}$ which acts as a $\mathbf{V}$, and decreases if it fails. During attestation, the neighbors of a given $\mathbf{P}$ challenge $\mathbf{P}$ and immediately record the challenge time. The credit score also indicates the current status of any $\mathbf{P}$ in the network. However, it introduces assumption like software only attacker. Moreover, the use of elliptic curve cryptography for attestation and link establishment among devices and public private cryptography for the secure communication incur cost in terms of computation and memory usages for tiny IoT devices.

US-AID. Ibrahim et al. [40] proposed a CRA scheme called Unattended Scalable Attestation of IoT Devices (US-AID) to attest large autonomous dynamic IoT networks. The authors claim that US-AID is different from the collective attestation state-of-the-art as it performs in-network attestation among neighbours in the network without the presence of a trusted centralized verifier; US-AID also performs absence detection to identify physical adversary in the network. US-AID does not require the network of IoT devices to be static during the attestation phase. The US-AID protocol has an offline and an online phase. During the offline phase the network owner sets up the network and bootstraps devices with necessary attestation details. At this stage all devices are benign. During the online phase neighbour devices perform mutual attestation: two neighbour devices exchange their respective software state upon mutual authentication (i.e., achieved by secure key sharing). Upon successful attestation, devices store the respective results into two separate lists (one for benign state and another one for compromised state devices). US-AID employs DARPA's heartbeat protocol [9] for absence detection. The absence of a centralised verifier makes US-AID robust as removes the single-point failure of a centralised attestation. While in US-AID the devices can perform attestation periodically and maintain a snapshot of the neighbour's health, the complex key establishing mechanism (i.e., Diffie-Hellman) makes this protocol computationally costly for low-end devices.

EAPA. Yan et al. proposed a CRA scheme named Efficient Attestation scheme resilient to Physical Attacks (EAPA) [41]. EAPA utilizes the mechanisms proposed in DARPA [9] and SCAPI [27] by employing heartbeats. In this approach, every device in the network establishes secure communication with its neighbour based on secure key communication. Based on the internal secure clock, every device generates a heartbeat and shares it with its neighbour. Upon receiving the heartbeats from the neighbours, every device authenticates the message. Then, upon successful attestation, the device stores the neighbour's device ID in the present device list and update the corresponding session keys. To detect physical adversary, EAPA relies on the same assumptions and mechanisms as DARPA. In EAPA every device manages two distinct lists for its neighbour: the list of devices that are successfully attested and the list of absent devices. During verification, the network owner will take the report from the prover device which acts as a verifier for its neighbours. The authors claim that EAPA improves SCAPI in terms of runtime and network overhead. Further, authors in EAPA also claim that they have improved SCAPI in terms of security as EAPA eliminates the single point of failure and all the provers mutually attest each other.

SHeLA. Rabbani et al. proposed SHeLA [42], a scalable heterogeneous (i.e. a swarm that consists of nodes that communicate over different wireless communication protocols) layered attestation scheme for large IoT networks. The paper provides an architecture and a protocol for remote attestation of a swarm of IoT devices that considers the mobility of the provers. To this aim, the paper introduces an edge computing layer between the verifier and the swarm network. Multiple edge verifiers are deployed to cover different areas allowing swarm nodes to leave and join as needed. Each edge verifier ($\mathbf{V}$) also maintains information of every swarm registered $\mathbf{P}$, $\mathbf{P}$ moved as guests, and all other $\mathbf{V}$. The root $\mathbf{V}$ has the information of all the other edge verifiers and swarm $\mathbf{P}$.

In SHeLA the attestation process is divided into an offline phase and an online phase. In the offline phase, $\mathbf{P}$ in network are bootstrapped with attestation related details and registered with an edge $\mathbf{V}$. During the online phase, edge verifiers perform attestation of the underlying IoT devices over a spanning tree. Unlike most of the RA schemes, SHeLA supports device

mobility. Through built-in redundancy[7] of the edge verifiers, it allows the swarm nodes to be temporarily unavailable or invisible to one or more edge devices. Therefore, even during attestation, $\mathbf{P}$ does not have to be static. Furthermore, SHeLA helps the root $\mathbf{V}$ to collect detailed information on the sanity of the individual network device. It varies from most current schemes where the granularity of the attestation is restricted to a binary outcome for the whole swarm. Authors claim that SHeLA satisfies all the properties of quality of swarm attestation (QoSA), as proposed by Carpent et al. [26].

HEALED. Ibrahim et al. in [43] proposed Healing and attestation for low-end embedded devices (HEALED). Authors claim that unlike other attestation mechanisms, HEALED not only detects malicious $\mathbf{P}$ in a network, but also provides efficient mechanisms to disinfect the affected $\mathbf{P}$. The authors assume a network of low-end embedded devices where a set of devices act as verifiers and others set of devices as provers. Based on predefined time, the verifier $\mathbf{V}$ attests the prover $\mathbf{P}$. The devices can be heterogeneous in terms of software and hardware configurations, however, $\mathbf{V}$ will share pairwise symmetric keys with all provers in the network. The attestation mechanism is divided into two main parts. In the first phase, one of the network devices acts as a $\mathbf{V}$ and sends the "Nonce" to the potential untrusted $\mathbf{P}$. Upon receiving the Nonce, each $\mathbf{P}$ performs a merkle hash tree based measurement over its software state and computes a MAC using the shared key and sends the result to $\mathbf{V}$. Upon receiving the MAC, $\mathbf{V}$ verifies the result. If the attestation is successful $\mathbf{V}$ stores the result as 1 (benign state), otherwise, if the attestation result does not match with the predefined state, $\mathbf{V}$ initiates the heal function. In the second phase of the attestation mechanism, called Healed, $\mathbf{V}$ broadcasts for a similar class of devices for the affected $\mathbf{P}$. Once $\mathbf{V}$ finds a similar device, it performs the mutual attestation to verify the device's internal state. Upon successful attestation, the similar device will initiate the healing process. HEALED constructs the segments of the Prover's software as a Markle Hash Tree (MHT), where the root of the tree is the measurement of the software state of the Prover. Thus, a compromised code segment generates a non-valid hash value along the path to the MHT root. Upon locating the corrupt memory region, the healer device sends authenticated patch to restore the original software and $\mathbf{P}$, then updates its internal state with the patch. However, a $\mathbf{P}$ can decline the healing process, but in this case it will not be able to prove its trustworthiness to other network devices. HEALED is built based on the design principle of distributed attestation mechanism where devices perform attestation without the presence of centralized entity like $\mathbf{V}$. However, the protocol may not be scalable over a large network due to its mechanism. Furthermore, the healing process does not guarantee whether the affected device will update its affected memory region with the benign code or not. In addition, use of distributed $\mathbf{V}$ raises concern over the safety of $\mathbf{V}$ itself, since there is no mechanism to guarantee the authenticity of $\mathbf{V}$ device.

SARA. Dushku et al. in [44] proposed a Secure Asynchronous Remote Attestation (SARA) protocol which attests asynchronously a group of interacting IoT devices. SARA assumes that each device provides one or many IoT services and IoT services follow a publish/subscribe communication paradigm to interact among themselves composing the so-called distributed IoT services. SARA aims to attest distributed IoT services running on IoT devices by guaranteeing that the software of IoT services have not been compromised and that the exchanged communication data have not maliciously influenced directly or indirectly the operation of the communicating services. In addition, SARA allows the attestation of a group of IoT devices without suspending the regular operation of all the devices at the same time. Given the challenges of physical clock synchronization of IoT devices and unpredictable triggering of events in the publish/subscribe model, SARA traces the execution order of IoT devices by adopting the usage of vector logical clock mechanisms. Specifically, in SARA each service S_i maintains a vector clock VC_i, where initially each value $VC_i[i]$ is 0. Before a service S_i sends a message, it computes $VC_i[i] = VC_i[i] + 1$, and then sends VC_i along with the message. When S_i receives a message associated to another vector clock OV, S_i sets: (1) $VC_i[j] = max\{VC_i[j], OV[j]\}, \forall j \in [1..N]$, (2) $VC_i[i] = VC_i[i] + 1$. In SARA, attestation starts at time T_0 when the $\mathbf{V}$, i.e., an external trusted party, sends an attestation challenge to a Publisher $\mathbf{Pub}$. Then, $\mathbf{Pub}$ registers the input received by environment, registers the output data of its own operation, computes the checksum of $\mathbf{Pub}$'s program binary, and then increments by one its own logical clock. Next, $\mathbf{Pub}$ signs this evidence and sends it along with the published data. When a Subscriber $\mathbf{Sub}$ gets a signed message from $\mathbf{Pub}$, $\mathbf{Sub}$ verifies the signature of the received message, records the input received from $\mathbf{Pub}$, and based on the logical clock sent by $\mathbf{Pub}$, $\mathbf{Sub}$ will update its own logical clock. $\mathbf{V}$ collects the attestation result at time T_1, by sending an attestation request to one (or more) subscriber $\mathbf{Sub}$ (acting as a $\mathbf{P}$) which along with the timestamped attestation result of $\mathbf{Sub}$ will send also the timestamped attestation result of previous interacting services. To verify the attestation result, $\mathbf{V}$ relies on vector logical clock properties to: (1) accurately construct a graph of the order of interactions among services, (2) verify the checksum of each service, and (3) verify the communication data exchanged among them. The vector clock properties allow $\mathbf{V}$ to identify those services that directly or indirectly have influenced the current state of $\mathbf{P}$. The size of the attestation evidence in SARA increases linearly with the number of the services that are included in the evidence.

B. Summary

Table I summarizes the main characteristics of the CRA schemes presented in this section. In particular, Table I shows

[7]In SHeLA, when a $\mathbf{P}$ become mobile and moves between the coverage of edge verifiers then redundancy is introduced through one to one RA between edge verifier and the $\mathbf{P}$. The edge verifiers cove rthe entire swarm network, in the unlikely event of a missing $\mathbf{P}$, that is out of coverage for all the edge $\mathbf{V}$, the edge $\mathbf{V}$ that performs the last attestation of the missing $\mathbf{P}$ keep tracks of the timestamp based attestation result.

TABLE I
FEATURES COMPARISON

Scheme	DoS Mitigation	Attestation Type	Node Mobility
SEDA [7]	✗	Type 1	✗
SANA [8]	✗	Type 2	✗
LISAα, LISAs [26]	✗	Type 1	✗
DARPA [9]	✗	Type 1	✗
SeED [24]	✓	Type 3	✗
SCAPI [27]	✓	Type 1	✗
ERASMUS [13]	✗	Type 3	✓
SALAD [28]	✓	Type 1	✓
PADS [25]	✓	Type 4	✓
WISE [29]	✗	Type 1	✗
slimIoT [30]	✗	Type 1	✗
SAP [32]	✓	Type 1	✗
MTRA [33]	✓	Type 1	✗
RADIS [35]	✗	Type 1	✗
ESDRA [39]	✗	Type 1	✓
US-AID [40]	✗	Type 1	✓
EAPA [41]	✗	Type 1	✓
SHeLA [42]	✗	Type 1	✓
HEALED [43]	✗	Type 1	✓
SARA [44]	✗	Type 1	✗

whether a CRA schemes allows node mobility during attestation or not. This specific characteristic indicates whether the CRA scheme accounts for nodes mobility (and consequently, for potential discontinuity of the connectivity among devices), or only when the devices are static, at least for the duration of the attestation protocol. Furthermore, Table II presents a comparison of the CRA schemes w.r.t. their efficiency in terms of scalability and runtime. Moreover, Table II provides details about implementation setup, mainly the simulation environment and prototype implementation of the respective CRA schemes. Note that, the data reported in Table II is as shown in the original papers (Due to unavailability of the simulation results in terms of scalability and runtime, we exclude ERASMUS [13] from Table II). Due to the heterogeneity of the simulation setting used by the different proposals, it is difficult to provide a fair comparison of their results. What catches the eye is the high complexity of some recent protocols. SALAD [28], for example, trades security for higher overhead, making the proposal potentially impractical for very large node deployments. Similarly, HEALED [43] introduces a healing phase for compromised devices, incurring in additional overhead; it is thus more suitable for small to medium sized deployments: from Table II we can see it takes approximately 1.8 seconds to attest (and heal) 20 devices, an order of magnitude larger than the time taken by RADIS [35] or LISAα, LISAs [26] for a comparable network size.

V. SECURITY ANALYSIS

In this section, we compare the various CRA schemes proposed in the literature according to the attacker model presented in Section II-B.

Table III compares the surveyed schemes based on the attestation adversary types they defend against (introduced in Section II-B). As shown in the table, all the discussed CRA schemes can detect software adversary, due to their core motivation of finding the legitimacy of the underlying software of $\mathbf{P}$. On the other hand, only ERASMUS [13], SeED [24], WISE [29] and SARA [44] provide solutions to tackle Adv_{MSW} (i.e., an adversary which changes its location continuously to evade detection and perform malicious activities during two successive attestation periods). ERASMUS relies on continuous monitoring, irrespective of the attestation period, and stores the attestation results in a tamper-resistant hardware module in order to secure it from unauthorized accesses. With ERASMUS, during attestation period $\mathbf{V}$ gets the chain of previous attestation results thus enabling the detection of a mobile adversary. WISE leverages continuous monitoring and unpredictable and variable attestation times to protect against Adv_{MSW}. In MTRA, authors rely on TPM enabled devices to perform attestation where TPM enabled higher-end devices act as cluster-head and perform attestation of the underlying low-end devices which lacks in tamper-proof hardware protection.

Physical attacks, performed by strong adversaries such as Adv_{SPI} and Adv_{PI}, require adversary to capture the device for a non-negligible amount of time in order to tamper with the hardware. So far, few CRA schemes like DARPA [9], SeED [24], SCAPI [27], slimIoT [30], US-AID [40], and EAPA [41] are capable of noticing the absence of any device in the network, as these specific attestation schemes run an absence detection protocol where each node registers their neighbors' "heartbeats". Since a physical adversary requires a non-negligible time to capture and perform malicious activities on devices, the compromised nodes will remain absent during one or more attestation runs. PADS [25] uses a specific label to identify devices whose state is "unknown", with the goal of disambiguating cases where a device is not reached by the attestation protocol versus cases where the device is simply inactive during the protocol. In PADS, only after several successive attestation rounds where a device is reported an "unknown" this device is considered as "compromised". SALAD [28] considers physical adversaries in its adversarial model. The authors define realistic security requirements in presence of attackers that can violate the TEE, defining the security of the system against a physical attacker as the inability for the attacker to fake the attestation response for a non-physically compromised device. This requirement is not met by PADS [25] and LISAα, LISAs [26]. DARPA [9], SeED [24], SCAPI [27], SALAD [28], and slimIoT [30] rely on specialized hardware and absence detection for detecting device captures.

VI. OPEN ISSUES FOR COLLECTIVE REMOTE ATTESTATION

Collective Remote Attestation (CRA) is a relatively low-cost solution to check the integrity of IoT swarms. In what follows, we highlight some limitations of the existing CRA literature. Moreover, we sum up a few open challenges associated with CRA that the research community is currently working on, but that still need practical and complete solutions.

TABLE II
COMPLEXITY COMPARISON

Scheme	Number of Devices in the Simulation	Simulation Runtime	Communication Protocol	Simulation Environment	Prototype Platform
SANA [8]	1,000,000	$\approx$ 2.5 sec	ZigBee	OMNeT++	Tytan
SEDA [7]	1,000,000	$\approx$ 1.4 sec	ZigBee	OMNeT++	SMART and TrustLite
LISAα, LISAs [26]	40	$\approx$ 0.1 sec	WiFi	CORE	Unspecified
DARPA [9]	1,000,000	$\approx$ 0.4 sec	ZigBee	OMNeT++	SMART
SeED [24]	1,000,000	$\approx$ 0.8 sec	ZigBee	OMNeT++	SMART and TrustLite
SCAPI [27]	500,000	$\approx$ 128 sec	ZigBee	OMNeT++	Stellaris EK-LM4F120XL
SALAD [28]	3,000	$\approx$ 10 to 20 minutes	ZigBee	OMNeT++	Stellaris LM4F120H5QR
PADS [25]	16,000	$\approx$ 2 sec	IEEE 802.15.4	OMNeT++	Unspecified
WISE [29]	10,000	$\approx$ 12.5 sec	IEEE 802.15.4	OMNeT++	Raspberry PI, Arduino (Zero, Uno and Due), Microship STK 600 with AVR Atmega644P MCU, and MicroPnP IoT
slimIoT [30]	1,000,000	$\approx$ 18 sec	IEEE 802.15.4	OMNeT++	Unspecified
SAP [32]	1,000,000	$\approx$ 0.6 sec	Unspecified	OMNeT++	TrustLite
MTRA [33]	8,000	$\approx$ 24 sec	Unspecified	NS3	Odroid XU4
RADIS [35]	10	$\approx$ 0.1 sec	WiFi	Docker based	Unspecified
ESDRA [39]	1,000,000	$\approx$ 0.5 sec	ZigBee	OMNeT++	Unspecified
US-AID [40]	1,000,000	$\approx$ 1.5 sec	ZigBee	OMNeT++	Raspberry Pi 3 Model B
EAPA [41]	25,000	$\approx$ 0.2 sec	Unspecified	OMNeT++	Unspecified
SHeLA [42]	50	$\approx$ 10.1 ms$^{(*)}$	WiFi	Unspecified	Xilinx FPGA
HEALED [43]	20	$\approx$ 1.8 sec	WiFi	OMNeT++	SMART and Trustlite
SARA [44]	250	$\approx$ 19 sec	IEEE 802.15.4	Instant Contiki	Unspecified

(*) Overhead measured on the prototype platform

TABLE III
MITIGATION CAPABILITIES W.R.T. ATTESTATION ADVERSARIES

Scheme	Adv_{SW}	Adv_{MSW}	Adv_{SPI}	Adv_{PI}	Adv_{PNI}
SANA [8]	✓	✗	✗	✗	✗
SEDA [7]	✓	✗	✗	✗	✗
LISA [26]	✓	✗	✗	✗	✗
DARPA [9]	✓	✗	✗	✓	✗
SeED [24]	✓	✓	✗	✓	✗
SCAPI [27]	✓	✗	✗	✓	✗
ERASMUS [13]	✓	✓	✓	✗	✗
SALAD [28]	✓	✗	✓	✗	✓
PADS [25]	✓	✗	✗	✗	✗
WISE [29]	✓	✓	✗	✗	✗
slimIoT [30]	✓	✗	✗	✓	✗
SAP [32]	✓	✗	✗	✗	✗
MTRA [33]	✓	✗	✗	✓	✗
RADIS [35]	✓	✗	✗	✗	✗
ESDRA [39]	✓	✗	✗	✗	✗
US-AID [40]	✓	✗	✗	✓	✗
EAPA [41]	✓	✗	✗	✓	✗
SHeLA [42]	✓	✗	✗	✗	✗
HEALED [43]	✓	✗	✗	✗	✗
SARA [44]	✓	✓	✗	✗	✗

Key Management. Choosing the most appropriate key mechanism is critical. CRA solutions in the literature have used various approaches, trying to balance security and efficiency.

We can broadly classify key management schemes into centralized and decentralized [11]. With a centralized key management scheme, a central authority is responsible for key generation and distribution. In a decentralized scheme, key generation, distribution and regeneration are not under the control of any central authority, but rather managed by more than one entity. CRA schemes predominantly use a centralized key distribution model, usually with the network/system owner in charge of it, being the central trusted authority for CRA schemes (i.e., $\mathbf{O}$ usually bootstraps the devices in the offline phase). However, this method of key generation and distribution may be tedious (i.e., in case of large network) and time-consuming. On the other side, decentralized management has advantages over the centralized approach due to its fault tolerance and scalability [11]. Unfortunately, to the best of our knowledge, so far none of the CRA schemes is employing decentralized key management for their respective operations.

Another aspect related to key management in CRA schemes that is worth mentioning is the use of only symmetric keys vs a mix of public and symmetric keys. Several of the surveyed schemes, such as LISAα, LISAs [26], and PADS [25], adopt a simplistic approach where a single symmetric key is shared among all the devices in the network. Despite this is technically fitting their security model, as they consider a software-only adversary and the key is protected by the TEE, this is a potentially unsafe practice; when it comes to stronger adversaries, e.g., Adv_{PI}, even in the case of a single device compromise the adversary will be able to simulate a valid attestation response for virtually every device in the swarm. On the other hand, the use of public-key cryptography provides stronger security, but requires additional computation capabilities for resource-constrained devices, and in some cases, its use may be unfeasible (e.g., due to the computational cost of public key encryption in low end platforms).

Intermittent Connectivity. It is a common assumption in the CRA literature that devices in the swarm maintain connectivity during the attestation process. In reality, stable connectivity among IoT devices may not be feasible, as swarms can be deployed in mobile scenarios (e.g., drones). Assuming devices will maintain a static formation (i.e., a spanning tree) for the duration of the attestation protocol may limit the usability of existing schemes in scenarios with mobile devices. Only a few recently proposed protocols like PADS [25], SALAD [28], US-AID [40], and SHeLA [42], facilitate device mobility during attestation. Furthermore, clusterization can be the basis for

more efficient protocols, if a small number of mobile devices in a cluster can be relied on to be in a static formation for the attestation runtime. Then, attestation of larger networks can be performed with existing or new solutions that allow mobility.

The absence of devices during swarm attestation has been so far only considered w.r.t to the presence of physical adversaries. That is, the absence of a device is considered sign of a device compromise. However, IoT devices can have intermittent activity, due to device necessities (e.g., power saving) or other constraints (e.g., precedence to real-time activities), which is true also when considering a static network formation. Thus, CRA schemes should provide ways to distinguish between the case in which a device was absent as a result of a compromise, or for other "legitimate" reasons. "Legitimate" intermittent connectivity has been taken in consideration only in few papers, with different interpretations. PADS [25] and SALAD [28] attest only provers that are reachable at the attestation time, delegating to following attestation rounds the validation of missing provers. Furthermore, some attestation results may not travel the whole network, and hence being unavailable when $\mathbf{V}$ collects the attestation report. For this purpose, PADS introduces the concept of "coverage" of the attestation result w.r.t. the network. Coverage implies the knowledge of one device about other devices in the network (e.g., for instance a prover $\mathbf{P}$ has 80% coverage implies that $\mathbf{P}$ has the knowledge about 80% other devices in the network). Again, works such as SCAPI [27], slimIoT [30] and US-AID [40] interpret device absence in the attestation protocol as the will to hinder the detection of compromised devices, even if slimIoT [30] tolerates short disconnections.

Time of check to Time of use. An important issue in attestation is the so-called Time Of Check To Time Of Use (TOCTTOU) [34], [45] problem. In brief, an attestation report is relative to the time where the remote attestation is carried out. As such, it neither guarantees whether the prover was malicious *before* the attestation time, nor that the state of the prover is correct right after attestation. An intelligent adversary can evade the attestation by using TOCTTOU gap. In [46] authors have demonstrated vulnerabilities of different attestation schemes e.g., C-FLAT [36], LO-FAT [37] and SMART [20] against TOCTTOU attacks. Unfortunately, except for ERASMUS [13], and MTRA [33], none of the CRA schemes consider TOCTTOU attack in their respective adversarial model. Recently, the work in [45] tries to address the TOCTTOU problem by formally defining the problem and proposing a solution that is suitable for low-end devices. New CRA schemes should focus on using solutions such as [45] to overcome this issue.

Dealing with physical adversaries. Physical adversaries are a serious concern for IoT devices, as they may be deployed and left unsupervised for a long time. Unfortunately, these adversaries are mostly overlooked in the existing CRA literature. Only a few schemes, i.e., DARPA [9], and SCAPI [27], SeED [24], slimIoT [30], MTRA [33], EAPA [41], and US-AID [40], are capable of partially mitigating physical attacks. In particular, these schemes identify physical adversarial presence by detecting device absence in the network. These schemes follow the assumption that in order to capture and modify a device in a network, a Adv_{PI} will require to take the device offline for a non-negligible amount of time. The absence of device(s) identified by other neighbour devices or by attestation procedure will indicate possible physical attack on the missing device.

Architectural issues. Most of the swarm attestation schemes employ specialized hardware, and some hardware-software co-design to carry out their attestation schemes. In existing deployments, however, the presence of such components is unlikely. Future research should look into solutions that are practical even for existing deployments, studying necessary tradeoffs between security and usability of such techniques.

VII. Future Directions

Future works are not only limited to current open issues. Several new directions can be explored by researchers. Here, we provide some hints.

Novel IoT architectures. The proliferation of IoT devices has opened the way for innovative architectures such as Edge computing [47], [48] and Fog Computing [49], [50], which are "horizontal, system-level architectures that distribute computing, storage, control, and networking functions closer to the users along a cloud-to-thing continuum". Fog Computing supports a general computing model, where data processing tends to be close to the edge where they are generated. Some works in literature propose some security solutions for fog architectures [51], [52]. Indeed, it is necessary to secure IoT devices composing them. Therefore, the use of attestation should be validated to secure fog and edge computing architectures. Moreover, collective remote attestation protocols should be tailored around fog or edge architectures, by first locally aggregating attestation reports, still leveraging on the clusterized nature of the architectures, and later broadcasting aggregated results to the cloud for attestation result fusion.

Future Internet technologies. Several novel internet architectures, such as Information-Centric Networking [53] and Named-Data Networking [54], have been proposed in recent years. Several works have shown the advantages of using such technologies in IoT [55]–[58]. Future research should look into leveraging some of the benefits of these architectures, such as in-network caching, to, e.g., improve resiliency against intermittent connectivity in CRA protocols.

Blockchain. Typically, most of the swarm attestation schemes work with the presence of a centralized entity, under the assumption of a single "owner" for the deployment of devices. In cases in which this is not true, a centralized architecture may not be ideal. To address the issue, Blockchain [59] based solutions have been explored to provide trusted and distributed mechanism for autonomous IoT environments [60]. However, its application in CRA has not been investigated so far. The blockchain can be used to store correct configuration of

devices, update events, as well as which devices are healthy or compromised at a given timestamp. Nonetheless, in our view, the characteristics of the blockchain are detriment for CRA protocols, especially due to the long latency in transaction validation, while CRA protocols must be fast. Moreover, the blockchain introduces significant computational and storage overhead that could be unsustainable for low power IoT devices. On the other hand, the joint use of blockchain and novel architectures [61] could be instead helpful for swarm attestation, moving blockchain management tasks to more powerful devices.

VIII. Concluding Remarks

Collective Remote Attestation (CRA) protocols aim at scaling remote attestation for large deployments of IoT devices. In this paper we provide a critical review of the state of the art of CRA schemes, comparing them w.r.t. their security features and assumptions. We also highlight open issues and suggest possible future research directions. Existing solutions are able to solve only some of the open issues highlighted through the paper. As an example, some solutions are efficient, but cannot be used in dynamic networks, while others provide high security guarantees, at the cost of an increased performance overhead. We believe this work can serve as a guideline for the research community, which is actively working on new solutions for CRA.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[4] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Anatomy of threats to the internet of things," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1636–1675, 2018.

[5] M. Guri and D. Bykhovsky, "aIR-Jumper: Covert Air-Gap Exfiltration/Infiltration via Security Cameras & Infrared (IR)," *Computers & Security*, vol. 82, pp. 15–29, 2019.

[6] R. V. Steiner and E. Lupu, "Attestation in Wireless Sensor Networks: A Survey," *ACM Computing Surveys*, vol. 49, no. 3, p. 51, 2016.

[7] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable embedded device attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, 2015, pp. 964–975.

[8] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "SANA: Secure and Scalable Aggregate Network Attestation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, 2016, pp. 731–742.

[9] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, and S. Zeitouni, "DARPA: Device attestation resilient to physical attacks," in *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '16, 2016, pp. 171–182.

[10] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014. [Online]. Available: http://science.sciencemag.org/content/345/6198/795

[11] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 8, no. 2, pp. 2–23, Second 2006.

[12] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "Invited – Things, Trouble, Trust: on Building Trust in IoT Systems," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 121.

[13] X. Carpent, G. Tsudik, and N. Rattanavipanon, "ERASMUS: Efficient remote attestation via self-measurement for unattended settings," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ser. DATE '18. IEEE, 2018, pp. 1191–1194.

[14] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, "Systematic treatment of remote attestation," *IACR Cryptology ePrint Archive*, vol. 2012, p. 713, 2012.

[15] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *Proceedings of the 2004 IEEE Symposium on Security & Privacy*, ser. IEEE S&P '04, 2004, pp. 272–282.

[16] D. Spinellis, "Reflection as a mechanism for software integrity verification," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 51–62, 2000.

[17] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla, "Scuba: Secure code update by attestation in sensor networks," in *Proceedings of the 5th ACM workshop on Wireless security*. ACM, 2006, pp. 85–94.

[18] W. Arthur and D. Challener, *A practical guide to TPM 2.0: using the Trusted Platform Module in the new age of security*. Apress, 2015.

[19] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, "A minimalist approach to remote attestation," in *Proceedings of the conference on Design, Automation & Test in Europe*, ser. DATE '14, 2014, p. 244.

[20] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust." in *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, ser. NDSS '12, 2012, pp. 1–15.

[21] R. Strackx, F. Piessens, and B. Preneel, "Efficient isolation of trusted subsystems in embedded systems," in *SecureComm*, 2010.

[22] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens, "Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base," in *22nd USENIX Security Symposium*, ser. USENIX Security '13, 2013, pp. 479–498.

[23] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "Tytan: tiny trust anchor for tiny devices," in *Proceedings of the 52nd Design Automation Conference*, ser. DAC '15, 2015, pp. 1–6.

[24] A. Ibrahim, A.-R. Sadeghi, and 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*, ser. WiSec '17, 2017, pp. 64–74.

[25] M. Ambrosin, M. Conti, R. Lazzeretti, M. Masoom Rabbani, and S. Ranise, "PADS: Practical Attestation for Highly Dynamic Swarm Topologies," in *Proceedings of the 1st International Workshop on Secure Internet of Things*, ser. SIoT '18, 2018.

[26] X. Carpent, K. ElDefrawy, N. Rattanavipanon, and G. Tsudik, "LIghtweight Swarm Attestation: a Tale of Two LISA-s," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIACCS '17. ACM, 2017, pp. 86–100.

[27] F. Kohnhäuser, N. Büscher, S. Gabmeyer, and 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*, ser. WiSec '17, 2017, pp. 75–86.

[28] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser, "Salad: Secure and lightweight attestation of highly dynamic and disruptive networks," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 329–342.

[29] M. Ammar, M. Washha, and B. Crispo, "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, 2018, pp. 1–8.

[30] M. Ammar, M. Washha, G. S. Ramabhadran, and B. Crispo, "slimiot: Scalable lightweight attestation protocol for the internet of things," in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2018, pp. 1–8.

[31] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *Rsa Cryptobytes*, vol. 5, no. 2, pp. 2–13, 2002.

[32] I. D. O. Nunes, G. Dessouky, A. Ibrahim, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik, "Towards systematic design of collective remote attestation protocols," in *2019 IEEE 39th International Conference on Distributed Computing Systems*, ser. ICDCS '19. IEEE, 2019, pp. 1188–1198.

[33] H. Tan, G. Tsudik, and S. Jha, "MTRA: Multi-Tier randomized remote attestation in IoT networks," *Computers & Security*, vol. 81, pp. 78–93, 2019.

[34] S. Bratus, N. D'Cunha, E. Sparks, and S. W. Smith, "Toctou, traps, and trusted computing," in *International Conference on Trusted Computing*. Springer, 2008, pp. 14–32.

[35] M. Conti, E. Dushku, and L. V. Mancini, "RADIS: Remote Attestation of Distributed IoT Services," in *2019 Sixth International Conference on Software Defined Systems*, ser. SDS '19. IEEE, 2019, pp. 25–32.

[36] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-FLAT: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, 2016, pp. 743–754.

[37] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N. Asokan, and A.-R. Sadeghi, "Lo-fat: Low-overhead control flow attestation in hardware," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, p. 24.

[38] G. Dessouky, T. Abera, A. Ibrahim, and A.-R. Sadeghi, "Litehax: Lightweight hardware-assisted attestation of program execution," in *2018 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '18. IEEE, 2018, pp. 1–8.

[39] B. Kuang, A. Fu, S. Yu, G. Yang, M. Su, and Y. Zhang, "ESDRA: An Efficient and Secure Distributed Remote Attestation Scheme for IoT Swarms," *IEEE Internet of Things Journal*, 2019.

[40] A. Ibrahim, A.-R. Sadeghi, and G. Tsudik, "US-AID: Unattended scalable attestation of IoT devices," in *2018 IEEE 37th Symposium on Reliable Distributed Systems*, ser. SRDS '18. IEEE, 2018, pp. 21–30.

[41] W. Yan, A. Fu, Y. Mu, X. Zhe, S. Yu, and 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*. ACM, 2019, pp. 2–7.

[42] M. M. Rabbani, J. Vliegen, J. Winderickx, M. Conti, and N. Mentens, "SHeLA: Scalable Heterogeneous Layered Attestation," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 240–10 250, Dec 2019.

[43] A. Ibrahim, A.-R. Sadeghi, and G. Tsudik, "Healed: Healing & attestation for low-end embedded devices," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 627–645.

[44] E. Dushku, M. M. Rabbani, M. Conti, L. V. Mancini, and S. Ranise, "Sara: Secure asynchronous remote attestation for iot systems," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3123–3136, 2020.

[45] I. D. O. Nunes, S. Jakkamsetti, N. Rattanavipanon, and G. Tsudik, "On the toctou problem in remote attestation," *arXiv preprint arXiv:2005.03873*, 2020.

[46] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A.-R. Sadeghi, "Atrium: Runtime attestation resilient under memory attacks," in *Proceedings of the 36th International Conference on Computer-Aided Design*, ser. ICCAD '17. IEEE Press, 2017, pp. 384–391.

[47] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[48] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, Fourthquarter 2018.

[49] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, Firstquarter 2018.

[50] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: a complete survey," *Journal of Systems Architecture*, 2019.

[51] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 601–628, Firstquarter 2018.

[52] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[53] C. Fang, H. Yao, Z. Wang, W. Wu, X. Jin, and F. R. Yu, "A survey of mobile information-centric networking: Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2353–2371, thirdquarter 2018.

[54] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[55] O. Waltari and J. Kangasharju, "Content-centric networking in the internet of things," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 73–78.

[56] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the iot: Experiments with ndn in the wild," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 77–86.

[57] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, "Information-centric networking for the internet of things: challenges and opportunities," *IEEE Network*, vol. 30, no. 2, pp. 92–100, 2016.

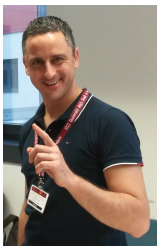
[58] G. Carofiglio, A. Compagno, M. Conti, F. De Gaspari, and L. Muscariello, "Iaas-aided access control for information-centric iot," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 208–216.

[59] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3796–3838, Fourthquarter 2019.

[60] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1676–1717, Secondquarter 2019.

[61] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1508–1532, Secondquarter 2019.

Moreno Ambrosin Moreno Ambrosin received his Ph.D from the University of Padova, Italy, in 2017 under the supervision of Prof. Conti. His research interest is mostly security and privacy. He has been a visiting researcher at the University of Washington (2015), and a Research Scientist at Intel Labs (2017-2019). He is now a Software Engineer at Google.



Mauro Conti is Full Professor at the University of Padua, Italy, and Affiliate Professor at the University of Washington, Seattle, USA. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. Currently, he is a Full Professor at University of Padova. He has been Visiting Researcher at GMU (2008, 2016), UCLA (2010), UCI (2012, 2013, 2014, 2017), TU Darmstadt (2013), UF (2015), and FIU (2015, 2016, 2018). He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His research is also funded by companies, including Cisco, Intel and Huawei. His main research interest is in the area of security and privacy. In this area, he published more than 250 papers in topmost international peer-reviewed journals and conferences. He is Area Editor-in-Chief for IEEE Communications Surveys & Tutorials, and Associate Editor for several journals, including IEEE Communications Surveys & Tutorials, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, and IEEE Transactions on Network and Service Management. He was Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, and General Chair for SecureComm 2012 and ACM SACMAT 2013. He is Senior Member of the IEEE.

Riccardo Lazzeretti received the Computer Science Engineering degree (cum laude) and the PhD degree from the University of Siena, Italy, respectively in 2007 and 2012. From November 2009 to May 2010, he was with Philips Lab, Eindhoven, The Netherlands. From 2012 to 2015, he continued his research at University of Siena, Italy, and from 2016 to March 2017 at University of Padua, Italy. Since 2017 he is assistant professor with the Department of Computer, Control and Management Engineering of Sapienza University of Rome, Italy. His research activity mainly focuses on security and privacy. He is senior member of IEEE, elected member of IEEE Information Forensics and Security Technical Committee, and associate editor of Elsevier Journal of Information Security and Applications.

Md Masoom Rabbani received his PhD degree in 2020 at the University of Padova under the supervision of Prof. Mauro Conti. Currently, he is working as a postdoctoral research fellow at ESAT-COSIC group (KU Leuven). Prior to joining to PhD, he worked in IBM India Pvt Ltd as an Associate System Engineer (2013-2016). His research interest includes security and privacy, Blockchain and embedded systems.

Silvio Ranise received his Ms. Eng. in 1997 at the University of Genova (Italy) and his PhD in Computer Engineering from the University of Genova (Italy) and the University H. Poincaré (Nancy, France) in 2002. He works at FBK in the S&T Research Unit as Senior Researcher since April 2010. His previous appointments are: assistant professor at the U. H. Poincarín 2001-2002, INRIA researcher at the LORIA computer science laboratory of Nancy in 2002-2008, research associate at the University of Verona (in the context of the EU Project AVANTSSAR) in 2008-2010, visiting professor at the Department of Computer Science of the University of Milano. His research focuses on formal methods for the automatic analysis of security-sensitive applications and he has published more than 65 papers in international conferences and journals on automated analysis of security policies, infinite state model checking, and Satisfiability Modulo Theories (SMT) solving. He has been initiator and coordinator of the SMT-Lib initiative, and started the SMT workshop series on SMT techniques. He has also given tutorials on SMT and infinite state model checking techniques at international conferences. In 2010, he received the HVC award for his “pivotal and continuous role in building and promoting the SMT community.”