HolA: Holistic and Autonomous Attestation for IoT Networks

Alessandro Visintin[1(✉)], Flavio Toffalini[2,3], Eleonora Losiouk[1], Mauro Conti[1], and Jianying Zhou[3]

[1] University of Padua, Padua, Italy
alessandro.visintin@studenti.unipd.it , elosiouk@math.unipd.it, mauro.conti@unipd.it
[2] EPFL, Lausanne, Switzerland
flavio.toffalini@epfl.ch
[3] SUTD, Singapore, Singapore
jianying_zhou@sutd.edu.sg

Abstract. Collective Remote Attestation (CRA) is a well-established approach where a single *Verifier* attests the integrity of multiple devices in a single execution of the challenge-response protocol. Current CRA solutions are well-suited for Internet of Things (IoT) networks, where the devices are distributed in a mesh topology and communicate only with their physical neighbours. Recent advancements on low-energy protocols, though, enabled the IoT devices to connected to the Internet, thus disrupting the concept of physical neighbour. In this paper, we propose HolA (Holistic and Autonomous Attestation), the first CRA scheme designed for Internet-like IoT networks. HolA provides defence against attacks targeting both the nodes and the network infrastructure. We deployed HolA on both a network of real devices (*i.e.,* 5 Raspberry Pis) and a simulated environment (*i.e.,* 1M devices in an Omnet++ network). Our results demonstrate that HolA can resist against a disruptive attacker that compromises up to half of the network devices and that tampers with network traffic. HolA can verify the integrity of 1M devices in around 12 s while the state-of-the-art requires 71 s. Finally, HolA requires 7 times less memory per device compared with the state-of-the-art.

Keywords: IoT network · Remote attestation · Distributed IoT services

1 Introduction

Internet of Things (IoT) refers to a category of small independent devices that, when connected to a network, can autonomously collaborate to accomplish complex tasks [4]. The widespread use of IoT technologies attracted the attention of adversaries, leading to the development of a broad class of attacks. These attacks are often partitioned into two groups: (i) *software attacks* that install malicious

J. Zhou et al. (Eds.): ACNS 2022 Workshops, LNCS 13285, pp. 277–296, 2022.
https://doi.org/10.1007/978-3-031-16815-4_16

software inside the device [26,27,45]; (ii) *physical attacks* that tamper directly with the hardware [13,14,24,25]. While *software attacks* are remotely executed by leveraging classic hacking techniques, *physical* ones require an adversary to remove a device from a network for a non negligible amount of time (*e.g.,* 10 min [23,30]).

In this scenario, Remote Attestation (RA) is a major solution for validating the integrity (*software* or *physical*) of remote devices [21]. The classic RA scheme [20] involves a trusted entity (*i.e., Verifier*) that challenges a remote device (*i.e., Prover*) to provide a measurement of its current status. Over the past few years, researchers proposed Collective Remote Attestation (CRA) schemes that better fit the *mesh-like* environment of IoT (*i.e.,* networks with devices communicating only with physically-close neighbours). However, the IoT world is increasingly moving from *mesh-like* to *Internet-like* networks [12,34]. Here, the concept of physical neighbour vanishes and current CRA schemes show limitations in terms of scalability and security. We consider the adoption of *Internet-like* networks prominent in light of the research on new energy-save Wireless protocols (*e.g.,* 6LoWPAN [40], Thread [22]) that promise to connect many more IoT devices to the Internet itself.

In light of these considerations we propose HolA, the first Holistic and Autonomous Attestation protocol for *Internet-like* IoT networks that: (i) guarantees an effective, efficient, and scalable periodic attestation of the whole IoT network; (ii) makes the IoT network resilient to the well-known attacks targeting *mesh-like* networks and the new ones addressing the *Internet-like* networks. We implemented HolA on real devices equipped with a trusted anchor [31] for storing keys and performing cryptographic operations (*i.e.,* 5 Raspberry Pi 3 and a Raspberry Pi 0 for performance reference). We also evaluated HolA performance in a large scale simulated network (*i.e.,* 1M devices) through Omnet++ [43]. To validate our approach, we conducted several attacks in both real and virtual scenarios, encompassing software tampering, lost packets, and corrupted devices.

2 Background

2.1 Remote Attestation

RA schemes consist in protocols that permits the verification of a remote entity. Usually, RA schemes involve two distinct roles: *Verifier* and *Prover*. The *Verifier* is considered trusted and is usually physically protected from attacks (*e.g.,* a remote server). The *Verifier* duty is to verify the integrity of a *Prover* that may be corrupted (*e.g.,* due to a malware). RA schemes require a *Verifier* to start the protocol by sending a challenge to the *Prover*, which measures some properties of its state (*e.g.,* compute a hash of a piece of software) and returns a report. Then, the *Verifier* can validate the *Prover* status by matching the returned report with a database of correct measurements.

In IoT scenarios, it is a common practice to perform *single-device* RA and *collective* RA. In the former, any network device can play the role of the *Verifier* and issue a challenge to another network device, *i.e.,* a *Prover*, to attest its status.

In the latter, only a device from a set of predefined ones can verify the current status of all the other network nodes, *i.e., Provers*, and generate a cumulative report.

2.2 Trusted Anchor

Modern RA schemes require nodes mounting specific hardware, called *trusted anchor*, that correctly implement minimal hardware features for attestation [20]. The nodes use Read-Only Memory (ROM) and Memory Protection-Unity (MPU) to partition the device memory into two zones: (i) *untrusted*, containing general purpose software; (ii) *trusted*, a protected memory region shielding sensitive information, such as cryptographic algorithms, keys, and secure random number generators. In short, the *trusted anchor* guarantees that only the protocol code accesses the cryptographic keys, and the node is booted correctly. Recent works show that Off-The-Shelf IoT devices already provide *trusted anchors* with a minimal hardware features set [39].

2.3 Chord

Chord [41,46] is a Distribushed Hash Table (DHT) protocol for managing distributed hash tables. In Chord, each node is identified by an *m-bit* number computed by a hash function. Using these identifiers, nodes are linked to their predecessors and successors, thus creating a ring. To improve resilience, nodes maintain a list of successors called *successors list*. The routing of messages around the ring is made efficient by the introduction of the *fingers table*, which results in an average routing complexity of $O(\log_2(n))$ and renders the operation scalable *w.r.t.* the number of nodes in the network. Chord permits dynamicity in the network by introducing three maintenance tasks [46]: (i) the *join* task, where an outside node contacts a member of the ring to join the topology; (ii) the *stabilize* task, where a node contacts its direct successor to check its presence and possibly adjust disruptions using the *successors list*; (iii) the *rectify* task, where a node receives notification of presence from its predecessor. The *join* task is performed only when a node is entering the network, while the *stabilize* and the *rectify* tasks are periodically executed by all the nodes to maintain the ring topology.

3 Assumptions

3.1 System Model

HolA focuses on *Internet-like* networks where devices are equipped with a trusted anchor. Devices communicate with each other over a secure and reliable channel. The security of the communication is guaranteed by the adoption of known protocols (*e.g.,* Diffie-Hellman [17]) on top of the Internet ones (*e.g.,* WiFi [8], 6loWPAN [40] and TCP/IP [37]), while the reliability comes from the TCP

properties Each device is uniquely identified by a certificate signed by a Certification Authority (CA) controlled by the network owner. The device private key is stored within the trusted anchor, while the public key is shared with other nodes to issue a secure channel. Finally, we assume nodes are already equipped with countermeasures against side channel attacks and having clocks loosely synchronized, as already assumed by previous works [23,28,30].

3.2 Threat Model

The goal of an attacker is to gain control of a network device and compromise it, meanwhile preventing its detection from the rest of the network. To achieve this goal, the attacker can use two different strategies:

- *Software attack* (*i.e.,* A_{sw}): working from a remote location, the attacker can gain control of the untrusted zone of one or more network nodes through classic exploitation techniques, but not of the trusted one. Moreover, she can gain control of one or more network infrastructure nodes (*i.e.,* Dolev-Yao model [18]).
- *Hardware attack* (*i.e.,* A_{hw}): the attacker can gain control of both the trusted and untrusted zone of one or more network nodes by manually tampering with the hardware node. Thus, the attacker needs to be in a physical range with the device, to remove it from the network for a time T_a (*e.g.,* 10 min [13,14,23–25,30]) and compromise it.

Adopting the above-mentioned strategies, the attacker can complete two attacks:

- *Injection attack*: through a A_{hw}, the attacker can inject a compromised device into the network.
- *Compromising attack*: through either a A_{hw} or a A_{sw}, the attacker can compromise a node already belonging to the network.

To inject a compromised device into the network, an adversary needs a valid certificate. Thus, she can either obtain a valid certificate from the CA or steal the certificate from another network node. While the first option is unfeasible, the second one is doable provided that the original owner is excluded from the network. In addition, the adversary can rely on A_{hw} to manipulate the network infrastructure and tamper with the protocol.

To compromise a network node, the attacker can rely on A_{hw} or on A_{sw}. Through A_{hw}, the attacker physically removes a node from the network for a time T_a, installs malicious code inside it, compromising the trusted zone, and makes the node rejoining the network. Moreover, the compromised node might manipulate the network traffic to prevent its detection. Through A_{sw}, the attacker reaches a network node from a remote location, installs malicious code inside it, compromising the untrusted zone, and manipulates the network traffic, either from the compromised node or from a network infrastructure node, to prevent the node detection.

In this work, we do not consider destructive Denial of Service (DoS) attacks that utterly interrupt any communication.

4 Motivation

4.1 CRA Limitations in Internet-Like Networks

To motivate our claim, we discuss here the impact of deploying the following three different CRA schemes in an *Internet-like* network: SANA [6], one of the most scalable protocols and currently used as a baseline; SCAPI [28], one of the CRA schemes that are the most resistant to physical attacks; PASTA [30], a scalable and physical attack resistant CRA scheme, that does not require an external *Verifier*.

Unlike *mesh-like* networks, the *Internet-like* ones do not assume physical connections among nodes and each device is logically connected to any other one. To represent the connections among devices in an *Internet-like* network, SANA and SCAPI schemes can either save the status of the whole network in each single node (S1) or define only a subset of nodes as logical neighbors (S2). S1 requires a high amount of memory allocated for each node, while S2 implies an adaptation of the current protocols. A similar approach may be applied by PASTA, which should require a node to store a key for each other network device or to include a novel mechanism to trace logical neighbor status. *Thus, current CRA schemes lack a mechanism to define logical connections in a Internet-like network.*

From a security perspective, *Internet-like* networks introduce more attack surfaces. The attacker can gain control of the devices from a remote location. Moreover, she can even target network infrastructure components (*e.g.,* switches) to tamper with the packets. In general, the remote access enables the attacker to launch wider attacks that may simultaneously affect a large number of devices. PASTA is the only work that considers such scenario, but its evaluation is limited (up to 10 devices).

4.2 Security Properties

To effectively defend against *injection* and *compromising* attacks, a CRA scheme designed for *Internet-like* networks should have three properties: (i) *neighbourhood attestation*; (ii) *absence detection*; (iii) *network obfuscation.*

Neighbourhood Attestation – It refers to the capability of each network node to verify the integrity of its neighbours. Neighbourhood attestation permits to detect any compromised node that managed to join the network through a *compromising* attack. Neighbourhood attestation is the consequence of removing a central *Verifier* in CRA schemes and distributing its attestation responsibility to all the nodes.

Absence Detection – It is the capability of a CRA scheme to detect whenever a device goes offline for a certain amount of time or even forever. In particular, the absence detection recognizes if a node becomes offline for a time T_a due to an *injection attack* performed through A_{hw}.

Network Obfuscation – This property refers to any strategy adopted by a CRA scheme to harden the network packet inspection and the consequent selective drop. This property prevents any attempt to manipulate the network traffic, which can be either performed during an *injection* or a *compromising* attack through A_{hw} or A_{sw}. Current CRA schemes do not provide any defence against attacks to the network infrastructure, since they assume the network is self-contained.

5 HolA Overview

HolA is a CRA scheme specifically designed for *Internet-like* networks, which guarantees the security properties illustrated in Sect. 4.2. HolA organizes the network devices in a ring by relying on the Chord protocol. To achieve this aim, each device needs to be equipped with specific data structures (Sect. 5.1) and to manage a specific life cycle (Sect. 5.2).

5.1 HolA Device Architecture

As specified in the system model, network devices are equipped with a trusted anchor (Sect. 5.2). Chord algorithms, together with HolA data structures and logic, are saved in the trusted anchor of each device. Table 1 shows the main components of each device and details are provided below.

`cert` – Every node is equipped with a certificate signed with `privCAKey` and defined as a tuple containing `pubKey`, `role` and `nodeId`. The `nodeId` is an incremental integer used as the index of the node inside the Status List (SL). The certificate represents the device identity, is signed offline by a CA, and used to authenticate messages exchanged among devices. We accept only trusted CAs that are controlled by the network administrators (Sect. 3.1).

`SL` – The SL keeps track of the network devices status. Each entry is a triplet defined as follows:

- `deviceStatus` (`trusted`/`offline`/`compromised`): a device is considered `trusted` as long as it succeeds the attestation from its neighbors. It eventually moves to `offline` status when it becomes inactive. Finally, a device is set as `compromised` when it stays `offline` for more than T_a or it fails the attestations.
- `exitTimestamp`: it is the exact time when a device is found to be `offline`.
- `sessionId`: it is a monotonic counter increasing every time a device enters the network. We use the `sessionId` to handle devices that temporarily go `offline` for less than T_a and that are willing to re-join the network.

We maintain a full copy of the SL in each device for two reasons: (R1) it makes the HolA scheme more robust in case of a simultaneous failure of multiple devices; (R2) it permits any device to be aware of the whole network status, thus efficiently implementing both *single-device* and *collective* attestations (Sect. 6.2).

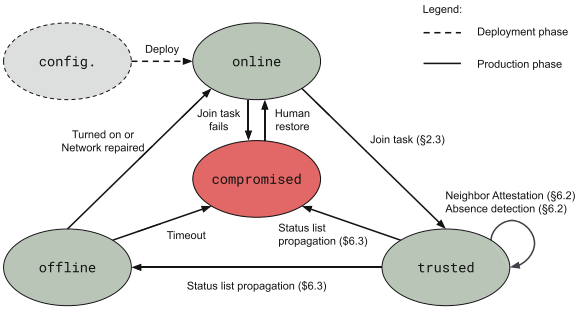

Fig. 1. Lifecycle of a network node deployed in the HolA CRA scheme.

Table 1. Main components of the HolA devices.

Data Structure	Short Description
successors list	List of the direct successors of a device [a]
finger table	List of intermediate devices in the network [a]
`nodeId`	a progressive unique number that identifies a device in the network
`pubKey/privKey`	keys used for issuing secure communication channels.
`cert`	A certificate representing the device identity [b]
`pubCAKey`	the CA `pubKey` used for certificate validation
`Status List (SL)`	a structure containing the status of each device in the network [b]
`verifySF()`	Function to ascertain the healthy status of a device [2,3,11,16,30,42,47]
`role`	the privilege of a device [2]

[a] See Chord protocol in (Sect. 2.3) for more info.
[b] See Device architecture in (Sect. 5.1)

A memory-efficient way to implement the SL is to create a list indexed by the `nodeId`s.

`role` – it represents the device privilege, which could be:

- *User*: generic IoT device, that can add devices to the SL, but not remove them.
- *Admin*: dedicated devices, that can perform network maintenance and remove a device from the SL.

5.2 HolA Device Lifecycle

The HolA device lifecycle is depicted in Fig. 1 and it involves the following states: `configuration`, `online`, `trusted`, `offline` and `compromised`.

`configuration` – Before a device is turned on, the network administrator installs the cryptographic material in the trusted anchor. In particular, the administrator provides the `cert`, the `pubKey`/`privKey`, and sets the `role` and the `nodeId`.

`online` – once the device is configured, it can perform the join phase following the Chord specifications (Sect. 2.3). The join procedure may fail for two reasons: (1) the device has an invalid `cert` (*e.g.,* signed by an unauthorized CA), (2) the node has been already saved in the SL as `compromised` because of a failed neighbourhood attestation or a timeout. In both case, the device is considered `compromised`.

`trusted` – When a device is `trusted`, it performs the neighborhood attestation and the absence detection.

`offline` – Once a member device becomes unreachable, it goes offline and has to re-join the network starting from the `online` status.

`compromised` – A member device can be set as `compromised` by other member devices, thus becoming isolated. A `compromised` device can be restored only due to manual intervention from the network administrator. In this case, the device passes to `online` and starts the join again.

6 HolA: Design

6.1 Status List Propagation

The SL is a data structure containing information about the status of all network devices (*i.e.,* `trusted`, `offline`, `compromised`) and each network device has a local and up-to-date copy of it. Whenever a device intercepts an event that requires an update to its local SL, the device will then propagate the information to the network devices to make them update their local copies of the SL. The events that can cause an SL update are:

- *New device joining the network*: when a new device enters the network, it performs the Chord *join* operation. The *successor* of the new device receives information about the new entrance, updates its own SL and propagates the information to the network.
- *Neighbourhood attestation*: thanks to the properties of Chord, each device periodically attests its *successor*. If the device finds the *successor* either `compromised` or `offline`, it updates its own SL and propagates the information to the network.
- *Absence detection*: if a device has some items in its SL marked as `offline`, it periodically verifies if any of those devices go `online`. If a device is found `offline` for more than T_a, it is marked as `compromised` in the local SL and the information is propagated to the network.

The update of the local SL occurs through a set of priority policies. Given two SL entries (*i.e.,* E_1 and E_2), we assume E_1 has priority over E_2 if (i) the `deviceStatus` of E_1 is `compromised` and the `deviceStatus` of E_2 is not `compromised`; **OR** (ii) the `sessionId` of E_1 is greater than `sessionId` of E_2. Condition (i) makes the network more conservative towards `compromised`

devices, which might become again `trusted` only through the intervention of an *Admin* device. While a tampered node may be able to change its state from `compromised` to `trusted`, it would not be able to propagate it to other nodes as this particular state transition is locked by construction. Condition (ii) handles the scenario where a device turns `online` again, after being `offline`, without the network detecting its exit. The propagation of the SL update to the whole network refers only to the entries that need to be updated, thus reducing the amount of transferred information. A device receives the whole SL only when it joins the network.

6.2 Neighborhood Attestation and Absence Detection

Neighborhood attestation enables the detection of any compromised node that managed to join the network through a *compromising* attack. Absence detection permits to monitor any device going `offline` and the amount of time it stays unreachable. In HolA, each network device periodically performs a neighborhood attestation against its *successor*. During neighborhood attestation the *successor* receives a challenge asking to check its status through the `verifySF()` function (*i.e.,* healthy or compromised), save it into a `report`, and return it to the sender. The attestation report can depict three outcomes: *correct*, in which case nothing is done; *fail*, in which case the *successor* is marked as `compromised` in the local SL; *timeout*, in which case the *successor* is marked `offline`.

Any update update to the SL is eventually propagated to the neighbors. This local and up-to-date copy of the SL allows each network devices to perform:

- *Single-device attestation* - A device A can verify the integrity of a device B, even if B is not one of A's neighbours by inquiring its SL. Previous works [3, 29] focused only on A_{sw}, while the *single-device* attestation in HolA detects both A_{sw} and A_{hw}.
- *Collective attestation* - an operator connects to a node with *Admin* `role` and looks for `compromised` devices in the SL. The operator can also manually remove the `compromised` nodes from the SL. We remark that the node `role` is part of the `cert`. Thus, an adversary cannot impersonate an *Admin* node, unless she breaks the CA signature or steals a node with *Admin* `role`.

6.3 Network Obfuscation

In our threat model, we assume that the attacker may perform statistical analysis over the exchanged packets (Sect. 3.2). For instance, she may detect and stop those packets belonging to the SL propagation, thus stopping the updates on `compromised` nodes. Since we assume the devices employ secure cryptography primitives, we exclude man-in-the-middle attacks. However, the adversary can still observe the sender IP, the receiver IP, and the packet size. Previous works showed that this information is enough to denanonymize the packets [15,32]. To mitigate this issue, we harden the packet analysis by employing network obfuscation strategies. In particular, we took inspiration from two techniques

used in the mix-networks [1]: (i) all the exchanged messages have the same size by design, and (ii) any device sends extra random packets to a fixed set of devices. These techniques avoid any intermediate device to distinguish between SL propagation, neighborhood attestation, or Chord routines. In addiction, this disrupts a frequency analysis to discover the *successors* of a device [19]. We analyze the packet size and quantify the network overhead in (Sect. 7.7) and (Sect. 7.5), respectively.

7 HolA: Evaluation

7.1 Experimental Setup

We used three setups: Raspberry Pi 0 [35], Raspberry Pi 3 [36], and a simulated network on Omnet++ [43].

We used a Raspberry Pi 0 [35] as a constrained environment to estimate the cost of cryptographic operations used by HolA. A Raspberry Pi 0 mounts a 1 GHz single-core CPU with 512 MB RAM. Each received message requires three cryptographic operations to be performed: (i) authentication, (ii) key negotiation, and (iii) decryption. The authentication mechanism uses an RSA schema with a 2048 bits long key. The certificate verification required on average 0.589 ms (0.02 ms std). The key generation phase produces an AES 256B long key using a Diffie-Hellman exponentiation. This operation required on average 5.92 ms (0.0522 ms std). The decryption mechanism uses an AES-GCM [38] with a 256B long key. It required on average 0.0833 ms (0.126 ms std).

We used a network of 5 Raspberry Pi 3 [36] to prove the feasibility of HolA on real devices mounting ARM TrustZone [44]. We developed the prototype on top of OP-TEE [33] by using the C language. We implemented network communication using two TCP sockets opened in the *untrusted zone*.

We used Omnet++ to simulate an IoT TCP/IP network with up to 1M devices. We set a delay of 10 ms to simulate message processing based on the measurements on the Raspberry Pi 0. We set the communication rate to 250 Kbps based on the defined data-rate of the 6loWPAN specifications [40].

We compared the results of our experiments with state-of-the-art solutions [6,28,30]. Each work has proposed its own experiments and units of measure. We therefore tried to proposed a thorough comparison with the information available.

7.2 HolA Resiliency

HolA requires that the underlying ring is preserved to automatically repair itself in case of disrupted nodes. Therefore, the overall security of our protocol is strictly related to this property. We can adjust the resilience of our network by tuning the *successors list* length ($SLEN$). In particular, $SLEN$ must be longer than the longest sequence of consecutive disrupted nodes.[1] Fig. 2a shows

[1] With consecutive nodes, we mean nodes with a consecutive position in the Chord ring.

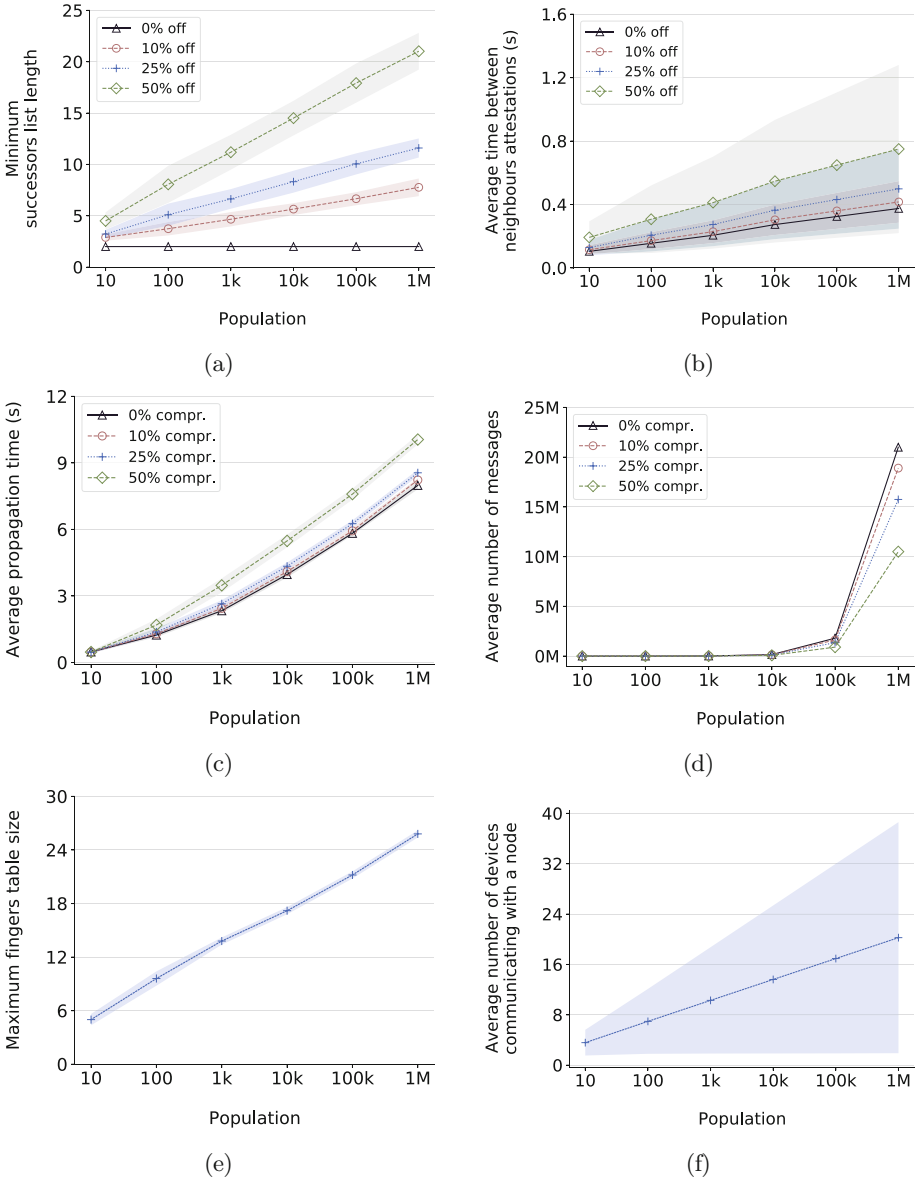

Fig. 2. Minimum successors list length required for preserving the ring structure (a), average time between two consecutive challenges received by a node (b), average time required for an information to be broadcasted (c), average number of messages required to broadcast an information (d), maximum fingers list size (e), average number of devices communicating with a specific node (f). The graphs are the results of 100 experiments run with 6 different populations. The lines represent different percentages of offline nodes (a, b) and compromised nodes (c, d). The points represent the average value and the shaded area the standard deviation. The x-axis has logarithmic base.

the minimum *successors list* length required for preserving the ring structure in case of random nodes disappearing. For the sake of this evaluation, we consider the nodes disappearing either for failure or attack, while we distinguish these cases in (Sect. 7.3). Figure 2a shows a logarithmic trend, therefore, the cost for a single node to maintain the structure scales *w.r.t.* the size of the network. In the remainder of this section, we will consider the 50% average values as a reference (*i.e.,* 4 for 10 nodes, 7 for 100 nodes, 10 for 1K nodes, 14 for 10K nodes, 17 for 100K nodes and 20 for 1M nodes).

Conclusion – Our results show that HolA is in line with [28] by tolerating up to $n/2$ outages in the network. In addition, HolA can adjust its resiliency by increasing the *successors list* length, thus introducing the unique possibility to trade-off between resiliency and memory overhead.

7.3 HolA Security Properties

The objective of HolA is to maintain a healthy status of the network (Sect. 3.2). This is achieved by periodically challenging the nodes and broadcasting the updates. We show the correctness of this approach by formal demonstration.

Definition 1. *A node is compromised if it does not pass the neighbourhood attestation or if it is found offline by the absence detection and it stays offline for more than a time T_a (Sect. 6.2).*

Definition 1 derives directly from the threat model considerations (Sect. 3.2) and HolA design (Sect. 6).

Definition 2. *A network attestation protocol is secure if it prevents compromised nodes to operate in the network.*

Definition 2 indicates that each node belonging to the network must receive *neighbourhood attestation* and that each offline node must be detected by *absence detection*. To avoid compromised nodes to rejoin the network, HolA must additionally broadcast the information to all the devices inside the network (Sect. 6.1).

Theorem 1. *Neighbourhood attestation guarantees continuous check of nodes.*

Proof Sketch. In HolA, each node receives a challenge by its predecessor in the ring (Sect. 6.2). *Given the resiliency property* (Sect. 7.2)*, the ring always remains intact during HolA operations. Therefore, a node participating to the protocol will always have a predecessor.*

In HolA, a node continuously receives challenges to check its status and eventually detect a *compromising attack* carried out through A_{sw} or A_{hw} (Sect. 4.2). Therefore, it is crucial that any node is constantly verified by the other nodes; this time may vary due to the presence of offline nodes. For instance, a node immediately after a disruption needs to wait for the protocol to repair the ring before receiving its next challenge. We deepen this aspect in (Sect. 7.4).

Theorem 2. *Absence detection guarantees the detection of offline nodes.*

Proof Sketch. Each node periodically sends a challenge to its successor and expects a response. In case the communication times out, the node flags the successor as offline in the status list, removes it from the successors list and propagates this information. The sequence of operations is repeated until the node finds the first online device in the successors list. The resiliency property (Sect. 7.2) *assures that at last one node will eventually find an online device. Each node participating to the protocol performs this task. Hence, using again the resiliency property, there will always be a node detecting and propagating the information about its offline successors.*

Since each node of the network is continuously proved, HolA avoids *injection attacks* (Sect. 4.2).

Theorem 3. *Status list propagation reaches all online devices.*

Proof sketch. Each node propagates the information to its successors list. Given the resiliency property (Sect. 7.2)*, the ring always remains intact during HolA operations. Hence, the information will be routed around the circle and will eventually reach all the nodes.*

Without losing in generality, we considered a limit case in which a node has an empty *finger list*, thus only relying on its *successor list*. In reality, the *finger table* usually has some entries that can be used to improve the broadcast speed. We measure the impact of the *finger table* in (Sect. 7.5). Moreover, we distinguish between a random attacker and a selective attacker. Section 7.5 treats the first case demonstrating the robustness of the process with different percentages of compromised nodes. For the second case, a selective attacker needs to physically compromise $SLEN$ consecutive nodes in the ring to block the propagation around the ring. Since we include network obfuscation techniques (Sect. 6.3), an adversary cannot exactly locate the *successors*. To successfully remove all the successors, an adversary must control all the devices contacted by a target node and those contacted by its *successor list*. In practice, in a network of 1M of device, considering a $SLEN$ of 20, and a finger list of 24 devices (more detail in Fig. 2e), an attacker must control 24^{20} devices (around 4×10^{27}) at the same time to block the SL propagation. To summarize, the network obfuscation enables a secure SL propagation to resist against a *compromising attack*.

7.4 Time Delay for Neighbourhood Attestation

We evaluated the time delay that neighbourhood attestations could suffer due to network disruptions. We experimented with 6 different populations presenting 4 different percentages of random offline nodes (from 0 to 50%). Figure 2b shows a logarithmic trend, therefore, the time period scales *w.r.t.* network size. The largest period is reached with a population of 1M device, where the maximum expected period is 1.3 s.

Conclusion – Despite major disruptions, the neighbourhood attestation procedure demonstrates optimal performance in terms of scalability and availability.

7.5 SL Propagation Performance

Theorem 3 demonstrates the correctness of HolA propagation. However, the overall performance of the naive implementation (*i.e.,* propagating only to *successors list* nodes) are linear *w.r.t.* the network size. To improve it, we experimented with a more efficient propagation where the information is additionally sent to one entry in the *finger table*. From the propagation perspective, this is equivalent to spread the information using a binary tree topology. We measured the time and number of messages required for the propagation to reach every node in the network. We experimented with 6 different populations and 4 different percentages of random offline nodes. For the *successors list* lengths, we considered the suggested values in (Sect. 7.2).

Figure 2c shows the average propagation time required for HolA to propagate an information to the whole network. The propagation time is almost logarithmic, demonstrating the scalability of the process *w.r.t.* the network size and that the process is slightly affected by the rate of offline nodes. The propagation time ranges from 8 s to 10 s with a population of 1M devices. In terms of messages, Fig. 2d shows the average number of messages required increases linearly with the online population and the number of messages sent by each node. In fact, each online node sends out an exact number of messages (all the successors plus a finger). As an example, for a population of 1M devices where 50% of them are offline we measured exactly $10,500,000$ messages $(50\% \times 1M) \times (20+1)$, where 20 is the *successors list* length and 1 the additional finger. This simplifies the estimate of the burden introduced by extra random packets (Sect. 6.3). In fact, the number of random packets sent proportionally affects the global number of packets exchanged. In other words, r extra random packets per each node would bring an increase of r-times in the messages.

Conclusion – HolA guarantees an efficient network propagation that scales quasi-logarithmically *w.r.t.* the network size. A complete execution of the protocol comprising both the attestation and the propagation takes an average time period of around 12 s (Sect. 7.4). This is a major improvement *w.r.t.* to previous works, where the elapsed time could be as long as 71.7 s [30] and 1421 s [6]. The time required by PASTA [30] highly depends on the network's state, ranging from 3 s to 71.7 s while our results are more stable.

7.6 Memory Consumption

Successor List – The *successor list* size ($SLEN$) is a constant value to be set before deploying the network. Each entry in the *successor list* stores an IP address (4B) together with the `pubKey` of the related node (128B), for a total of 132B. Hence, a complete *successor list* requires $(132 \times SLEN)$B.

Finger Table – Every entry of the *finger table* requires 132B as for the *successor list* (4B for the IP plus 128B for the `pubKey`). The total number of entries in the *finger table* depends on the network size. Figure 2e shows the maximum number of fingers of the *finger table* with different network populations (up to

1M devices). The plot shows a logarithmic trend, suggesting that the *fingers table* scales *w.r.t.* the network size. In particular, a *finger table* contains $132 \times log_2(n)$B.

Status List – Each SL entry stores a `deviceStatus` (1B), an `exitTimestamp` (8B) and a `sessionId` (1B), totalling 10B. The SL must contain all the devices in the network, hence the overall memory overhead is $(10 \times n)$B.

Cache – We introduced a cache for storing the `pubKey` and `nodeId` of other devices and speed up the communications. In particular, we tune the cache to contain as many entries as the expected number of nodes to contact (Fig. 2f). The graph indicates a logarithmic growth *w.r.t.* the global population of the network. This suggests that a cache could reduce the burden of certificate verification without imposing too much overhead on the memory. Each entry in cache occupies 130B (128B for the `pubKey` and 2B for the `nodeId`), setting the cache memory to $(130 \times log_2(n))$B.

Conclusion – The overall memory used for data structures is $10 \times n + 262 \times log_2(n) + 132 \times SLEN$. The other decentralized autonomous network (PASTA [30]) claims an overall memory cost of at most $(78,140 + |token| \times 1,280)$B (around 700 Kb). In the same scenario (*i.e.,* 10K), HolA has an overhead of at most $(103,668 + 132 \times SLEN)$B. Considering 14 as $SLEN$ (Sect. 7.2), the size becomes 105,516B (around 100 Kb), that is seven times less overhead *w.r.t.* PASTA.

7.7 Communication Overhead

We measure the communication overhead in terms of message size. In our implementation, the messages contains a header of 274B, where the majority part is dedicated to a *certificate* (256B) and the rest is for the HolA internal working. The payload of a message depends in which phase HolA is operating: (i) *join phase*, and (ii) *operational phase*. During the *join phase*, the largest message sent has size $(10 \times n + 132 \times SLEN + 274)$B and it is sent once. The *operational phase*, instead, comprises all the tasks executed by a node during its permanence inside the ring. Since we adopt network obfuscation techniques, all the *operational* messages has the same length, which is of $(264 \times (SLEN + 1))$B). In case of a network of 10K devices and a resiliency rate of 50% (Sect. 7.2), the HolA *operational* messages have a weight of 3,960B.

Conclusion – SANA [6] has a communication costs in the same order of magnitude of HolA. The same can be said for SCAPI [28] that claims an average cost of 1,314B for a network of 10K devices.

8 Related Works

CRA on Spanning Tree Topology – SEDA [7], SANA [6], and LISA [10] are the first CRA schemes. They initiate a spanning tree topology over the network

and use it for distributing the burden of computation among all the devices. A limitation regarding these approaches is the static topology assumption that forces the devices to not disrupt the tree during the protocol execution. Moreover, they do not consider hardware attackers [6,7,10], they provide coarse-grained results [7] or expensive aggregation methods [6] and they propose inefficient secret keys management [7,10].

Physical Attacks Detection – Due to the increasing size of the networks, researchers investigated mechanisms to detect *physical* adversaries that may capture the devices. These works assume a *physical* adversary removes a device from the network for a non-negligible amount of time. DARPA [24] and SCAPI [28] are the main works in this direction. They both rely on an heartbeat token exchanged between neighbors that permits an external *Verifier* to detect devices' absence. These works act as overlays placed above existing solutions (SEDA [7], SANA [6]). Thus, they inherit both the complexity and the limitations of the underlying attestation scheme.

CRA for Highly Dynamic Swarms – To tackle the static topology issue of first CRA schemes, more recent ones proposed a scheme that incrementally creates a complete snapshot of the network status. SALAD [29] and PADS [5] achieve this goal through a shared structure that contains the status of all the devices, and an external *Verifier* that needs to retrieve the structure from a random device.

Autonomous Networks – A new branch of CRA works proposes autonomous networks that do not rely on an external *Verifier* to maintain their healthy status. DIAT [3] assumes a *mesh-like* connection and focuses on *software* adversaries and run-time RA (*i.e.,* they validate runtime device status). US-AID [23] can detect both *software* tampering and device disconnections. PASTA [30] handles both *software* and *physical* adversaries by relying on a periodical generation of tokens that attests the integrity of all the devices that participated in its generation. These autonomous CRA schemes focus on *mesh-like* networks where a device can only connect with its physical neighbours. Hence, they do not provide support for those environments in which a device can potentially connect with all the others.

9 Discussion

Certificate Revocation/Expiration – In our design, we ruled out the certification revocation and expiration. This may let adversaries forge fake certificates and allow malicious devices to join the network. We can overcome this issue with the introduction of probabilistic filters, as described in previous works [9].

False Positive – A device is considered `compromised` through A_{hw}, if it is `offline` for more than T_a. However, this does not mean a device has been actually under attack. This is also considered as an open problem in previous

works [23,28,30]. In HolA, we mitigate this issue by relying on *Admin* devices that can manually control the network status and restore outage devices.

Devices Loosely Synchronized – As assumed also in previous works [23,28,30], the network devices require some clock synchronization strategy to detect a A_{hw}. We aim at overcoming this limitation by storing the relative time at which a device becomes `offline`, instead of the absolute timestamp. However, this introduces other synchronization challenges, such as considering random network propagation delays. We plan to investigate new solutions for this issue in future versions of HolA.

10 Conclusion

In this paper, we proposed HolA, the first Holistic and Autonomous Attestation protocol for *Internet-like* networks. HolA guarantees a strong defence against both *compromising* and *injection* attacks.

We demonstrated the feasibility of the HolA protocol over real devices (*i.e.,* Raspberry Pi 0 and 3) and on a network of 1M of simulated devices (*i.e.,* Omnet++). In our evaluation, we stressed the resilience of HolA against a network with 50% of nodes disrupted. HolA showed an attestation time in between 8 s and 12 s, that is similar and more stable than previous works (*i.e.,* from 3 s to 72 s [30]). In addition, HolA can resist to adversaries that perform network analysis and selectively drop packets. In terms of scalability, HolA requires only 100 Kb per device in a network of 10K nodes, which is in contrast with the 700 Kb required by previous works [30].

Acknowledgements. The work is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund - Pre Positioning (IAF-PP) Award A19D6a0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of A*STAR.

References

1. Abe, M.: Mix-networks on permutation networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-48000-6_21
2. Abera, T., et al.: C-FLAT: control-flow attestation for embedded systems software. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 743–754. ACM, New York (2016). https://doi.org/10.1145/2976749.2978358, https://doi.acm.org/10.1145/2976749.2978358
3. Abera, T., Bahmani, R., Brasser, F., Ibrahim, A., Sadeghi, A., Schunter, M.: DIAT: data integrity attestation for resilient collaboration of autonomous systems. In: 26th Annual Network & Distributed System Security Symposium (NDSS). The Internet Society (2019). http://tubiblio.ulb.tu-darmstadt.de/110632/
4. Alaba, F.A., Othman, M., Hashem, I.A.T., Alotaibi, F.: Internet of things security: a survey. J. Netw. Comput. Appl. **88**, 10–28 (2017)

5. Ambrosin, M., Conti, M., Lazzeretti, R., Rabbani, M.M., Ranise, S.: PADS: practical attestation for highly dynamic swarm topologies. In: 2018 International Workshop on Secure Internet of Things (SIoT), pp. 18–27 (2018). https://doi.org/10.1109/SIoT.2018.00009
6. Ambrosin, M., Conti, M., Ibrahim, A., Neven, G., Sadeghi, A.R., Schunter, M.: SANA: secure and scalable aggregate network attestation. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 731–742. ACM, New York (2016). https://doi.org/10.1145/2976749.2978335, https://doi.acm.org/10.1145/2976749.2978335
7. Asokan, N., et al.: SEDA: scalable embedded device attestation. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015, pp. 964–975. ACM, New York (2015). https://doi.org/10.1145/2810103.2813670, http://doi.acm.org/10.1145/2810103.2813670
8. Bhatt, A., Patoliya, J.: Cost effective digitization of home appliances for home automation with low-power WiFi devices. In: 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), pp. 643–648 (2016). https://doi.org/10.1109/AEEICB.2016.7538368
9. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: a survey. Internet Math. **1**(4), 485–509 (2004). https://doi.org/10.1080/15427951.2004.10129096
10. Carpent, X., ElDefrawy, K., Rattanavipanon, N., Tsudik, G.: Lightweight swarm attestation: a tale of two LISA-s. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS 2017, pp. 86–100. ACM, New York (2017). https://doi.org/10.1145/3052973.3053010, http://doi.acm.org/10.1145/3052973.3053010
11. Challener, D.: Trusted platform module. In: Encyclopedia of Cryptography and Security, pp. 1308–1310 (2011)
12. Cisco Systems, I.: Why IP is the right foundation for the smart grid. https://www.cisco.com/c/dam/assets/docs/c11-581079-wp.pdf. Accessed November 2020
13. Conti, M., Di Pietro, R., Gabrielli, A., Mancini, L.V., Mei, A.: The smallville effect: social ties make mobile networks more secure against node capture attack. In: Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access, pp. 99–106 (2010)
14. Conti, M., Di Pietro, R., Mancini, L.V., Mei, A.: Emergent properties: detection of the node-capture attack in mobile wireless sensor networks. In: Proceedings of the First ACM Conference on Wireless Network Security, pp. 214–219 (2008)
15. Conti, M., Rigoni, G., Toffalini, F.: ASAINT: a spy app identification system based on network traffic. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–8 (2020)
16. Dessouky, G., et al.: Lo-fat: Low-overhead control flow attestation in hardware. In: Proceedings of the 54th Annual Design Automation Conference 2017, pp. 1–6 (2017)
17. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
18. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science. In: SFCS 1981, pp. 350–357. IEEE Computer Society, Washington, DC (1981). https://doi.org/10.1109/SFCS.1981.32
19. Feistel, H.: Cryptography and computer privacy. Sci. Am. **228**(5), 15–23 (1973)

20. Francillon, A., Nguyen, Q., Rasmussen, K.B., Tsudik, G.: A minimalist approach to remote attestation. In: 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1–6. IEEE (2014)
21. Gong, B., Zhang, Y., Wang, Y.: A remote attestation mechanism for the sensing layer nodes of the internet of things. Futur. Gener. Comput. Syst. **78**, 867–886 (2018)
22. Thread Group: Thread. https://www.threadgroup.org/. Accessed November 2020
23. Ibrahim, A., Sadeghi, A.R., Tsudik, G.: US-AID: unattended scalable attestation of IoT devices. In: 37th IEEE International Symposium on Reliable Distributed Systems (2018). https://doi.org/10.1109/SRDS.2018.00013, https://ieeexplore.ieee.org/document/8613950
24. Ibrahim, A., Sadeghi, A.R., Tsudik, G., Zeitouni, S.: DARPA: device attestation resilient to physical attacks. In: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec 2016, pp. 171–182. ACM, New York (2016). https://doi.org/10.1145/2939918.2939938, http://doi.acm.org/10.1145/2939918.2939938
25. Ibrahim, A., Sadeghi, A.R., Zeitouni, S.: SeED: secure non-interactive attestation for embedded devices. In: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 64–74 (2017)
26. Islam, S.A., Katkoori, S.: SafeController: efficient and transparent control-flow integrity for RTL design. In: 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 270–275. IEEE (2020)
27. Jeong, S., Hwang, J., Kwon, H., Shin, D.: A CFI countermeasure against got overwrite attacks. IEEE Access **8**, 36267–36280 (2020)
28. Kohnhäuser, F., Büscher, N., Gabmeyer, S., Katzenbeisser, S.: SCAPI: a scalable attestation protocol to detect software and physical attacks. In: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2017, pp. 75–86. ACM, New York (2017). https://doi.org/10.1145/3098243.3098255, http://doi.acm.org/10.1145/3098243.3098255
29. Kohnhäuser, F., Büscher, N., Katzenbeisser, S.: SALAD: secure and lightweight attestation of highly dynamic and disruptive networks. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, ASIACCS 2018. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3196494.3196544
30. Kohnhäuser, F., Büscher, N., Katzenbeisser, S.: A practical attestation protocol for autonomous embedded systems. In: 4th IEEE European Symposium on Security and Privacy (EuroS&P 2019) (2019). https://doi.org/10.1109/EuroSP.2019.00028, http://tubiblio.ulb.tu-darmstadt.de/114633/
31. Kylänpää, M., Rantala, A.: Remote attestation for embedded systems. In: Bécue, A., Cuppens-Boulahia, N., Cuppens, F., Katsikas, S., Lambrinoudakis, C. (eds.) CyberICS/WOS-CPS -2015. LNCS, vol. 9588, pp. 79–92. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40385-4_6
32. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 255–263. Association for Computing Machinery, New York (2006). https://doi.org/10.1145/1180405.1180437, https://doi.org/10.1145/1180405.1180437
33. Linaro: Op-tee (2015). https://github.com/OP-TEE/optee_os. Accessed June 2019
34. Mandula, K., Parupalli, R., Murty, C.A., Magesh, E., Lunagariya, R.: Mobile based home automation using internet of things (IoT). In: 2015 International Confer-

ence on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 340–343. IEEE (2015)
35. Pi, R.: Raspberry pi zero. https://www.raspberrypi.org/products/raspberry-pi-zero/
36. Pi, R.: Raspberry pi 3 model b (2015). https://www.raspberrypi.org
37. Rayes, A., Salam, S.: The internet in IoT. In: Internet of Things From Hype to Reality, pp. 37–65. Springer, Heidelberg (2019)
38. Salowey, J., Choudhury, A., McGrew, D.: AES galois counter mode (GCM) cipher suites for TLS. Request for Comments 5288 (2008)
39. Schulz, S., Schaller, A., Kohnhäuser, F., Katzenbeisser, S.: Boot attestation: secure remote reporting with off-the-shelf IoT sensors. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 437–455. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_24
40. Shelby, Z., Bormann, C.: 6LoWPAN: The Wireless Embedded Internet, vol. 43. Wiley, Hoboen (2011)
41. Stoica, I., et al.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw. **11**(1), 17–32 (2003). https://doi.org/10.1109/TNET.2002.808407, http://dx.doi.org/10.1109/TNET.2002.808407
42. Toffalini, F., Losiouk, E., Biondo, A., Zhou, J., Conti, M.: SCARR: scalable runtime remote attestation for complex systems. In: 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), pp. 121–134. USENIX Association, Chaoyang District, Beijing (2019). https://www.usenix.org/conference/raid2019/presentation/toffalini
43. Varga, A.: OMNet++. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation, pp. 35–59. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_3
44. Winter, J.: Trusted computing building blocks for embedded linux-based arm trustzone platforms. In: Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing, STC 2008, pp. 21–30. ACM, New York (2008). https://doi.org/10.1145/1456455.1456460, http://doi.acm.org/10.1145/1456455.1456460
45. Xia, H.: Capability memory protection for embedded systems. Ph.D. thesis, University of Cambridge (2020)
46. Zave, P.: How to make chord correct (using a stable base). CoRR abs/1502.06461 (2015). http://arxiv.org/abs/1502.06461
47. Zeitouni, S., et al.: ATRIUM: runtime attestation resilient under memory attacks. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 384–391. IEEE (2017)