Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects

Boyu Kuang[a,b], Anmin Fu[a,b,*], Willy Susilo[c], Shui Yu[d], Yansong Gao[a]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, PR China
[b] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, PR China
[c] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
[d] School of Computer Science, University of Technology Sydney, NSW, Australia

ARTICLE INFO

Article history:
Received 25 May 2021
Revised 5 September 2021
Accepted 3 October 2021
Available online 7 October 2021

Keywords:
Internet of Things
Remote attestation
Security
5G
Artificial intelligence

ABSTRACT

The explosive growth of the Internet of Things (IoT) devices is an inevitable trend, especially considering the fact that 5G technology facilitates numerous services building on IoT devices. IoT devices deliver great convenience to our daily lives; nevertheless, they are becoming attractive attacking targets. Compromised IoT devices can result in the exposure of user privacy, damage to network security, or even threats to personal safety. In a rush for convenience and marketability, the security of these devices is usually less considered during production and even ignored. Under these circumstances, Remote Attestation (RA) becomes a valuable security service. It outsources the computation and verification burden to a resource-rich party, e.g., server, to ease its on-device implementation, making it suitable for protocol extensions.

In this paper, we investigate the state-of-the-art RA schemes from different perspectives, aiming to offer a comprehensive understanding of this security service. Specifically, we summarize the basis of RA. We set up an elaborate adversarial model by systematizing existing RA schemes. Then we put forward the evaluation criteria from protection capability, performance, network adaptability, and attestation quality. According to the adversarial model, we classify existing RA schemes into five categories to show the various characteristics. A comparison of representative proposals enables readers to adopt and design suitable protocols in different application scenarios. Finally, we discuss some open challenges and provision prospects for future research.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

The popularity of 5G dramatically improves the communication ability and flexibility of the Internet of things (IoT) (Stellios et al., 2021), and provides an excellent infrastructure for IoT's future development. The era of the "Internet of Everything" is very near. Meanwhile, the rapid growth of advanced technologies, such as artificial intelligence (AI), enriches the IoT. IIoT (Industrial IoT) (Berger et al., 2020) and AIoT (AI & IoT) (Al-Garadi et al., 2020) have become new research

* Corresponding author at: School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, PR China.
E-mail addresses: kuang@njust.edu.cn (B. Kuang), fuam@njust.edu.cn, fam_0522@163.com (A. Fu), wsusilo@uow.edu.au (W. Susilo), shui.yu@uts.edu.au (S. Yu), yansong.gao@njust.edu.cn (Y. Gao).
https://doi.org/10.1016/j.cose.2021.102498
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

hotspots and topics of the times. As a result, the number of IoT devices, as the foundation of IoT, is surging. Nowadays, they have been deployed for intelligent transportation, environmental monitoring, government work, public safety, smart home, industrial monitoring, lighting control, elderly care, personal health, food traceability, etc. Millions of IoT devices have spread throughout our lives (Heartfield et al., 2021; Li et al., 2021; Su et al., 2020). As people are getting used to these devices and relying on them, their security becomes an utmost important concern.

Unfortunately, these IoT devices are facing a number of security vulnerabilities and security risks (Qu et al., 2018). Intelligence is one of the main trends in the current evolution of IoT devices (so-called smart IoT devices). These smart IoT devices are closer to our lives and thus store more private information, which is an interesting asset attractive for attackers (Steiner and Lupu, 2009). Most IoT devices provide remote control interfaces to facilitate unattended settings. However, these interfaces are also usually accessible by attackers, rendering these devices extremely vulnerable. Moreover, many low-end or embedded IoT devices are limited with energy, computing capability, memory, and other necessary resources to defend against various attacks due to the production costs (Ambrosin et al., 2020). Furthermore, some vulnerabilities already exist when the device is produced, and the manufacturers do not provide product updates, upgrades, or patches.

A compromised device would bring severe threats to not only users but also the network (Kolias et al., 2017; Makhdoom et al., 2018; Murray, 2019; Shekari and Beyah, 2020). Malware in IoT devices might reveal users' privacy to third parties secretly or alter the behaviors of these devices (Fu et al., 2017; 2016; 2012; Yu et al., 2018). Particularly, in some safety-critical IoT-enabled infrastructures, such as national defense, healthcare, and industrial process control, the security of those IoT devices is extremely important. On the other hand, with the advancement of information and communications technology (e.g., 5G), IoT's scale grows dramatically. In many large, dynamic, and self-organizing networks, IoT devices connect, interact, and cooperate, forming a *swarm*. A compromised device can be maliciously exploited for launching Distributed Denial of Service (DDoS) attacks (Chew et al., 2021; Praseed and Thilagam, 2021) and then threatening the swarm as well.

Providing a security mechanism for these IoT devices becomes an enormous challenge. In this context, Remote Attestation (RA) is one of the most valuable basic security services, which establishes a static or dynamic trust root in the device. It allows a decision-making party (*verifier*) to assess the state of the other party (*prover*). The *verifier* is usually a trusted party, e.g., cloud server, fog node, or base station, with rich computational and storage capabilities. RA mechanism reduces the computing and energy consumption on the *prover* side. It requires no significant device modification, which is suitable for protocol extensions. Moreover, it can serve as the foundation for other security services such as firmware updates and patches.

1.1. *Related survey and motivation*

Some studies have conducted surveys from the perspectives of RA in wireless sensor networks (WSNs) (Steiner and Lupu, 2009), swarm attestation (Ambrosin et al., 2020), hardware-assisted attestation (har, 2020), the development of RA (Banks et al., 2021), and software-based RA (Ankergård et al., 2021). Specifically, Steiner and Lupu (2009) introduce the RA model, traditional attacks method, common assumptions of RA schemes in WSNs and present a taxonomy in terms of evidence acquisition, integrity measurement, timing, memory traversal, attestation routine, program memory, data memory, and interaction pattern. Ambrosin et al. (2020) analyze a number of attestation schemes in swarms and categorize them into software-based attestation and attestation based on Root of Trust (RoT). The attestation based on RoT includes four types of interaction patterns: interactive RA, interactive self-RA, non-interactive RA, and non-interactive self-RA. har (2020) focus on multiple common hardware used in RA and RA schemes with the assistance of hardware, including the hardware-based RA schemes and hybrid RA schemes. Banks et al. (2021) review RA in chronological order and mainly describe the software-based RA, hardware-based RA, hybrid RA, and swarm attestation. Ankergård et al. (2021) survey the state-of-the-art software-based RA schemes and discuss the strength and weaknesses of each scheme respectively.

Overall, none of the existing surveys has an in-depth investigation of the attack, and there is no comprehensive and fine-grained adversarial model as a guide for taxonomy. As numerous new attacks are proposed (e.g., the runtime attacks), the types of attacks become more diverse, precise, and covert. The RA schemes are also more diversified and complicated. Furthermore, the application scopes of RA are expanding (e.g., the verifiable updates, resets, and memory erasure). Some new technologies are applied and combined in RA (e.g., machine learning), bringing more possibilities to RA. Moreover, a number of new RA architectures (e.g., the formally verified architectures) are put forward to implement richer security features.

In this context, we aim to propose a universal adversarial model to support the RA taxonomy. The adversarial model is a general requirement when devising the RA schemes. Unfortunately, almost every scheme presents a specific adversarial model, which makes their relationship and combination extremely complicated. In our survey, we analyze all adversarial models to the best of our knowledge and then design an elaborate model. We find that this is an important basis for summarizing and classifying the existing methods. Using the model, we can clearly understand the purpose, principle, and strategy of a given RA scheme. We can also quickly grasp the relationship among differing RA schemes and identify their strengths and weaknesses.

1.2. *Contribution*

In this paper, we refer to the latest related literature, covering a more comprehensive range of application scenarios and methods. We are not trying to exhaustively list all RA proposals. We instead aim to form a well-organized taxonomy by comparing and evaluating representative schemes and then summarize unsolved challenges, possible methods, and future research directions. Specifically, we survey the literature on RA over the period 2001–2021. We use Google Scholar, IEEE

Xplore, ACM Digital Library, ScienceDirect, SpringerLink, and Web of Science to search for existing papers with the keywords: "attestation", "remote attestation", "control-flow attestation", and "device attestation". Eventually, we identify over 200 papers, after duplicated entries are removed via a careful manual process. Next, we summarize all the searched literature to build a comprehensive and fine-grained adversary model. Then, we classify the existing literature according to this model and select the representative works in each category to present in this survey, finally including more than 60 papers. More specifically, the majority of representative works are those published in high-profile security venues such as S&P, USENIX Security, CCS, NDSS, ESORICS, IEEE TIFS, IEEE TDSC, Computers & Security, etc., just to name a few. The quality of these venues is determined using the ranking of the security conferences rankings (Gu, Guofei, 2020; Zhou, Jianying, 2020) and the impact factor/citescore of the journals. In addition, there are certain impactful papers that do not belong to those venues but with high citations, which we have further included in the representative papers to discuss. Moreover, we clarify the reasons, methods, advantages, and disadvantages of each strategy to be able to detect a certain type of attack. Based on the evaluation criteria, we compare a number of RA schemes and further present prospects for future research. Our main contributions are as follows:

- We propose a basic RA model and further establish an elaborate adversarial model for IoT devices by abstracting the characteristics of existing attacks.
- We put forward evaluation criteria from protection capability, performance, network adaptability, and attestation quality.
- Based on the adversarial model, we categorize the existing RA schemes into five classes, analyze each category, and compare their merits and limitations.
- According to the analyses, we summarize open problems and challenges of RA, and prospect future research directions.

Roadmap. The remainder of this paper is organized as follows. Section 2 provides a broad overview of remote attestation in IoT. It also explains different types of attacks. Section 3 sets up the evaluation criteria. Sections 4–8 classify the state-of-the-art remote attestation approaches into five categories from the perspective of attacks and analyze the merits and limitations of several typical schemes in each category in each section, respectively. Section 9 summarizes and discusses some challenges for future research, followed by a conclusion in Section 10 concludes for this work.

2. Remote attestation

In this section, the RA system model is presented. Then the objectives of RA and an adversarial model are introduced.

2.1. System overview

The typical system model contains two main entities: a *prover* and a *verifier* (Nunes et al., 2019b; Seshadri et al., 2004; Steiner and Lupu, 2009), as shown in Fig. 1. The general goal of remote attestation is to enable the *verifier* to determine whether the *prover* is in an acceptable state via a challenge-response mechanism. In other words, the process of remote attestation can be defined as an interaction between the *prover* and the *verifier*. When a *prover* is challenged by a *verifier*, it will generate a response corresponding to its current state and then send it back. After verifying the attestation response, the *verifier* can determine the trustworthiness of the *prover* (Ambrosin et al., 2020; Ankergård et al., 2021; Nunes et al., 2019c). A typical RA protocol commonly consists of the following steps:

Challenge. The *verifier* firstly sends an attestation request, so-called a challenge, to the *prover*, triggering the attestation process. The challenges of different RA protocols would contain various elements according to their design goals. Among them, some common elements are as follows.

- *Nonce.* A random nonce is used to ensure the freshness and unpredictability of the challenge.
- *Input.* When the attested program is affected by distinct inputs, the *verifier* will give a fixed one to make the result and process predictable.
- *Attested region.* The region to be attested, which can be the memory region, the hardware configuration, the program control flow, and any other measurable information.
- *Output region.* The memory region used to store the output.

Attestation. Once receiving the request, the attestation procedure in the *prover* measures its internal state and generates an attestation response based on the received challenge. For various attestation purposes, the *prover* would generate different kinds of attestation responses. For instance, it can be the *prover*'s software configuration when checking the software integrity or the control-flow path of a device to detect the control-flow attacks. At last, the *prover* sends back an attestation report to the *verifier*.

Verification. After receiving the attestation report, the *verifier* checks the validity and the correctness of the report. The validity is always guaranteed by Message Authentication Code (MAC) or signatures. The correctness is assessed via comparing the attestation response with the expected state of the *prover*. Note that the *prover*'s expected state is usually prestored or pre-computed by the *verifier* when initializing the device.

The typical RA model is usually applied to the single-prover attestation, where the *verifier* only attests one *prover* in each interaction. Nevertheless, as the scale of the network increases, one *verifier* is connected to multiple *provers*, and single-prover attestation becomes inefficient. Therefore, swarm attestation is proposed, which enables the *verifier* to attest multiple *provers* in parallel. We will discuss the details and differences between single-prover attestation and swarm attestation specifically in Section 4.

2.2. Objectives

As a security service for IoT devices, RA must guarantee that only the honest (not compromised) *prover* can convince the *verifier* that it is in an acceptable (or expected) state. Since the

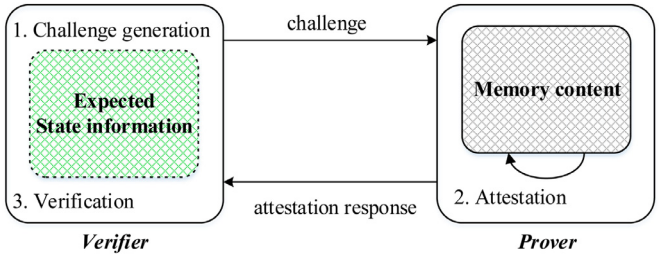


Fig. 1 – System overview of remote attestation.

prover and the *verifier* communicate over an open network, attackers would try their best to evade detection via eavesdropping, replay, forgery, and other means (Steiner and Lupu, 2009). Therefore, a secure and robust RA service needs to satisfy the following properties.

(1) *Authenticity.* The *provers* must guarantee that the challenges are generated by the authorized *verifier*, which defends against the verifier-impersonation Denial-of-Service (DoS) attacks. Besides, the *verifier* needs to ensure that the *provers*' responses are faithful and trustworthy. This means that the attacker cannot forge a response that the *verifier* will accept.

(2) *Atomicity.* The attestation schemes must prevent attackers from obstructing the execution of the attestation process. Therefore, the attestation process cannot be interrupted to keep attackers from modifying the current memory, moving the malware around during the execution.

(3) *Freshness.* The attestation responses must reflect the current state of the *prover* to prevent replaying attacks. In other words, when receiving the challenge from the *verifier*, the *prover* must perform a fresh process to generate a new attestation response. Besides, the challenges must be refreshed.

(4) *Determinacy.* For each challenge, the corresponding response should be deterministic. This enables the *verifier* to predict the response content of the *prover* accurately.

Furthermore, with emerging applications, more attestation schemes are proposed, and more objectives are required. We have listed the most common ones as follows. Note that these requirements are specific to certain attestation protocols customized for different applications, not any attestation protocol.

(1) *Scalability.* Some attestation schemes are presented for large-scale networks. Therefore, the attestation protocols are required to be scalable for a large number of devices.

(2) *Heterogeneity.* Some complex networks are comprised of a number of heterogeneous devices. Thus, the attestation schemes need to be suitable for different software and hardware.

(3) *Dynamic.* The device movements and the changes in network topology are notable issues. Consequently, the attestation schemes should be compatible with the dynamicity of the network.

2.3. *Adversarial model*

The main goal of an adversary is to compromise the IoT devices (*provers*) to perform some malicious operations or hinder their normal execution. An outline of different types of attacks on RA is illustrated in Fig. 2. In all attestation schemes, we consider two types of attackers:

- *Normal attacker*
- *Knowledgeable attacker*

Each type of attacker has different attack means, as shown in the blue rectangular nodes. The normal attackers may compromise devices at arbitrary times but do not have sufficient knowledge of the RA mechanism. Common methods include software attacks, runtime attacks, and physical attacks.

Software attacks. Traditionally, adversaries may replace or add malicious software snippets and execute them as desired, namely code injection (Francillon and Castelluccia, 2008). Then these malicious codes can misdirect the device behavior or leak users' private information stealthily. This kind of attacks compromise devices via tampering with their internal memory by software invading.

Runtime attacks. The adversaries may alter devices' behavior by manipulating data variables, loop counters, or code pointers, shown as ①, ②, ③ in Fig. 2. All three attack methods exploit existing code snippets but in malicious sequences so that there is no need to inject external codes. Code Reuse Attack (CRA) is a typical runtime attack, which mainly includes return-oriented programming (ROP) attacks (Checkoway et al., 2010) and jump-oriented programming (JOP) attacks (Snow et al., 2013). This kind of attack may cause malicious execution flows or perform privilege paths illegally.

Physical attacks. Physical attacks compromise a device via physical access. Physical attacks can be categorized into non-intrusive, semi-intrusive, and invasive physical attacks (Skorobogatov, 2012). Semi-intrusive and invasive physical attacks both need to capture the device physically so that they can extract sensitive information even in the protected memory or modify the hardware components. Non-intrusive attacks, e.g., side-channels attacks (Carlet et al., 2021), can stealthily deduce devices' private data during normal operations. Non-intrusive attacks are usually resisted by some physical means, such as shielding circuits, where RA is not suitable. Thus, the attestation for physical attacks mentioned in the following paper all refers to semi-intrusive and invasive attacks.

In comparison, with the adoption of the RA mechanism, knowledgeable attackers may utilize some weaknesses of the schemes to achieve the purpose. The most common method

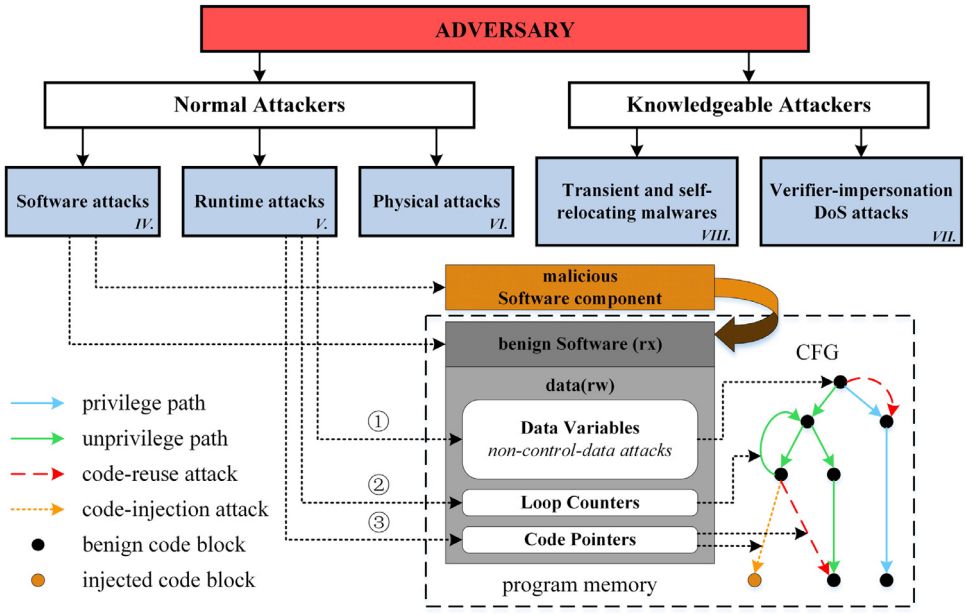


Fig. 2 – Adversarial model.

is verifier-impersonation DOS attacks and transient & self-relocating malware.

Verifier-impersonation DoS attacks. Researchers found that an attacker may disguise itself as a *verifier*, forcing the *provers* to perform attestation procedures by sending forged requests frequently. Unrestricted attestation procedures might obstruct performing normal tasks, leading to DoS attacks.

Transient and self-relocating malware. An adversary moves the location of malware during the attestation process and puts them into the area that has been attested or will not be attested. Note that this attack conflicts with the atomicity property of remote attestation, and we will elaborate on the reasons and mitigation methods in Section 8.

Additionally, an adversary also attempts to bypass the RA mechanism by some traditional means, such as pre-computation, replay, collusion, and so on Steiner and Lupu (2009). These attacks are generally resisted by cryptographic means, which have been adopted by most RA schemes.

3. Evaluation criteria

There are diverse focuses when evaluating a given attestation protocol. Next, we discuss how to evaluate a RA scheme, whether it is practical, whether it can be widely adopted and deployed from *protection*, *accuracy*, *cost*, *network topology*, and *Quality of Swarm Attestation* (*QoSA*). In particular, the *protection* is summarized according to the taxonomy in this survey. The *accuracy* and *network topology* are summarized during the survey of the literature. The *cost* is an important evaluation criterion commonly used in most literature. The concept of *QoSA* is originally proposed by Carpent et al. (2017), and we have further extended it after summarizing the existing literature.

3.1. *Protection*

The protection capability depends on the strategy it adopts and the design goal. In general, we evaluate a RA scheme from the attacks that it can detect or prevent, as we described in the adversarial model. Thus, the taxonomy in this survey is the concretization of this evaluation criterion. Moreover, it should be noted that the restriction degree of assumptions is also a critical factor. Some assumptions are too strict and unrealistic, which appears to be unreasonable. For example, some software-based attestation schemes (discussed in Section 4.1) limit the attackers' ability or require extremely high transmission quality and cannot tolerate delay and packet loss.

3.2. *Accuracy*

Verification accuracy is not considered by most RA protocols. Most RA schemes aim at one single type of attack, even with only one *prover*, so it is unexpected to make mistakes in the rigorous attestation process. Therefore, the accuracy of these schemes is almost 1 without considering the instability of the network. Accuracy is critical for some special RA schemes that share a common feature, i.e., probability or uncertainty. They mainly include two types: the randomization of the attested region and the probability of the attested device in swarm attestation scenarios. For these schemes, the accuracy refers to false negatives to a great extent because the RA will output a conclusive result, and almost no false positives will occur.

3.3. *Cost*

Memory cost. The memory cost of RA schemes mainly comes from the storage of keys, attestation procedure, and some protocol parameters. Although memory cost is still an evaluation

factor, with the reduction of storage cost, it is no longer a key factor.

Time cost. The time cost is an essential factor for evaluating a RA scheme. The atomicity property of the RA mechanism would affect the normal task of the devices during the attestation process, which is more prominent in the swarm attestation schemes. In most swarm attestation schemes, only when all nodes have completed the attestation can the entire swarm operate normally. However, the superposition of communication and computing time of each node leads to a notable total time cost.

Energy cost. One of the original purposes of the RA mechanism is to save the cost of resource-constrained devices, so energy cost is an essential evaluation factor. Energy costs mainly contain the computation overhead and communication overhead. Among them, the communication overhead is concerned due to the proposal of swarm attestation schemes, where a single device needs to interact with neighbor nodes. Besides, the results of the existing schemes show that the computing overhead accounts for a larger proportion of the energy cost. This is because the computing overhead includes the cost of some heavy cryptographic operations, such as MAC or signature, especially for IoT devices.

Hardware cost. When the researchers found that the pure-software RA schemes could no longer provide sufficient security capabilities, they began to seek the support of hardware. Under the same security level, researchers always hope to minimize hardware costs. Hardware cost mainly includes two aspects. One is the utilization of existing hardware units, such as Trusted Platform Module (TPM) (Arbaugh et al., 1997), Physical Unclonable Functions (PUFs) (Gao et al., 2020), Software Guard Extensions (SGX) (Costan and Devadas, 2016), and any trusted hardware components. The other is the redesign of the architecture using Field Programmable Gate Array (FPGA). This kind of RA scheme usually evaluates its hardware cost by the number of additionally required Look-Up Table (LUT), register, and logic gate.

Note that the time and energy cost of the RA schemes are generally highly dependent on the hardware architecture of the device (e.g., devices with TPM usually compute faster). Therefore, only the comparison of time and energy cost in the same hardware architecture setting is meaningful.

The following two evaluation criteria are specific to the swarm attestation.

3.4. *Network topology*

The mobility of devices is one inherent swarm characteristic. However, many swarm RA schemes require that the swarm network topology remains static, where devices cannot move during the attestation process. Since RA is accomplished by the challenge-response mode, it needs a fixed communication channel. Some RA schemes relax this restriction, where the device only needs to remain static during its own attestation procedure without waiting for the end of the entire swarm attestation process.

On the contrary, swarm RA schemes for highly dynamic networks allow devices to move during the attestation process arbitrarily. These schemes usually use the broadcasting method, and detailed descriptions are deferred to Section 4.2.

3.5. *QoSA*

Carpent et al. (2017) propose the concept of Quality of Swarm Attestation (QoSA). Based on their work, we make some further summaries and classify the swarm attestation into five classes to evaluate the granularity of the swarm information received by the *verifier*.

B-QoSA (Binary QoSA). The *verifier* can get a Boolean bit to indicate the trustworthiness of the entire swarm. For this kind of attestation result, the *verifier* is not required to attest each device in the swarm exhaustively. Once a compromised node is detected, it can immediately determine the final result of the swarm and terminate attestation on rest devices.

C-QoSA (Count QoSA). The *verifier* can gain knowledge of the counted number of normal/malicious devices in the swarm but does not know the specific devices that have been compromised.

L-QoSA (List QoSA). The *verifier* can get an identifier list of normal devices. This is the most common QoSA adopted by current swarm attestation schemes.

F-QoSA (Full QoSA). The *verifier* can know the exact attestation result of each device in the swarm and their connectivity, i.e., swarm topology.

N-QoSA (None QoSA). The swarm has the ability to rule out the compromised devices autonomously. That is to say, the *verifier* does not require knowing the identifiers of all compromised devices in the swarm.

For B-QoSA and C-QoSA, the *verifier* can only treat the swarm as a whole to determine its security and is incapable of inspecting and taking actions on particular compromised devices to maintain the security of the swarm. While for F-QoSA and L-QoSA, the *verifier* can remove or repair the compromised devices to ensure security. However, for F-QoSA and L-QoSA, more detailed aggregated reports are often required, which implies a larger message size to be transmitted through the network. N-QoSA combines self-repair with swarm attestation. Therefore, the swarm supporting the N-QoSA attestation is robust against attacks. As a trade-off, extra overheads are induced to the individual device in the swarm for the sake of collaboratively monitoring its neighbor devices' security.

4. Attestation for software attack

Software attacks (e.g., malicious invasion) that compromise software integrity are common attacks performed on IoT devices. An attacker injects malware into a device's memory or replaces benign codes with malicious snippets to compromise the device. In this context, the *verifier* can examine its legality by comparing the memory content with the expected counterpart via evidence. The evidence is usually a checksum over the content in the target memory range or the *prover*'s software configurations. In this section, we will elaborate on several representative RA schemes that detect software attacks according to the number of devices involved: single-prover attestation and swarm attestation.

Table 1 – The comparison among single-prover attestation schemes in chronological order.

Schemes	Architecture	Hardware cost	Fml_ver
SWATT Seshadri et al. (2004)	software-based	-	✗
Pioneer Seshadri et al. (2005)	software-based	-	✗
Y.G. Choi Choi et al. (2007)	software-based	-	✗
PoSE Perito and Tsudik (2010)	hybrid	(MTMEGA 128) MCU & ROM	✗
DR@FT Xu et al. (2012)	hardware-based	TPM	✗
SMART Eldefrawy et al. (2012)	hybrid	(AVR/MSP430) MCU & ROM	✗
TrustLite Koeberl et al. (2014)	hybrid	MPU & ROM & secure clock	✗
PUFatt Kong et al. (2014)	hardware-based	PUF	✗
TyTAN Brasser et al. (2015)	hybrid	MPU & ROM & secure clock	✗
HYDRA Eldefrawy et al. (2017)	hybrid	seL4 & ROM & secure clock	P
Boot Schulz et al. (2017)	hybrid	MCU & ROM	✗
J. Wang Wang et al. (2018)	hardware-based	SGX	✗
AAoT Feng et al. (2018)	hardware-based	PUF	✗
ATT-Auth Aman and Sikdar (2018)	hardware-based	PUF	✗
VRASED Nunes et al. (2019b)	hybrid	(MSP430) MCU & ROM & FPGA	✓
R.V. Steiner Steiner and Lupu (2019)	software-based	-	✗
PURE Nunes et al. (2019c)	hybrid	(MSP430) MCU & ROM & FPGA	✓
APEX Nunes et al. (2020)	hybrid	(MSP430) MCU & ROM & FPGA	✓
M. Kucab Kucab et al. (2021)	hardware-based	SGX	✗
RATA Nunes et al. (2021b)	hybrid	(MSP430) MCU & ROM & FPGA & secure clock	✓

P denotes that the security of HYDRA (Eldefrawy et al., 2017) is based on a formally verified seL4 microkernel, but is not fully formally verified. Fml_ver denotes whether the scheme is based on a formally verified RA architecture.

4.1. *Single-prover attestation*

Single-prover attestation schemes focus on the rigorous interaction process between the *verifier* and the *prover*, as well as the attestation process on the device. According to the RA scheme architectures, there are three main approaches: *software-based, hardware-based*, and *hybrid*. Table 1 presents the architecture of different single-prover attestation schemes, the evaluation criterion of hardware cost, and whether they are based on a formally verified RA architecture, according to the following taxonomy. Note that the non-specific hardware components (such as Random Access Memory (RAM) and flash memory) are not listed in the table.

(1) Software-based

Software-based RA schemes (Choi et al., 2007; Seshadri et al., 2005; 2004; Steiner and Lupu, 2019) usually do not require delicate hardware security components, which has a wider range of applications, especially for resource-constrained IoT devices. *Pseudorandom Memory Traversal (PMT), strict response-time*, and *empty memory space-filling* are commonly used for realizing software-based attestation.

Pseudorandom memory traversal. If the RA traverses the entire memory space in order, an attacker may trivially forge a valid attestation response due to the lack of hardware protection capability. PMT generates the response over random memory blocks, where the order is a secret determined by a pseudorandom seed (e.g., SWATT Seshadri et al., 2004). PMT is a probability method that is unable to cover the whole memory space in one attestation process, which means that the malware still has a chance not to be attested by the attestation procedure. Thus, for the evaluation criterion of accuracy, a memory of size m requires performing $O(m \ln m)$ memory reads to cover the entire memory (Steiner and Lupu, 2009). Besides, the order of pseudorandom traversal may still be guessed by the attacker, especially when the order space is limited.

Strict response-time. Strict response-time can resist attackers from changing the attestation code to forge a valid attestation response via interrupting the attestation process (Seshadri et al., 2005; Steiner and Lupu, 2019). It reduces the possibility of forgery attestation response since adversaries may need extra time to forge a valid response. However, it is inapplicable to unstable network or multi-hop network transmission, where the network latency is hard to be estimated.

Empty memory space-filling. Filling the empty memory space with random noise is frequently used in software-based attestation schemes (e.g., Proactive Code Verification Protocol Choi et al., 2007). Since the *prover* generates a hash over the memory as the input of attestation response, once the empty memory space is filled with uncompressed random noise, there will be no left space available for malware injection by the attacker. Unlike the strict response-time strategy, empty memory space-filling only requires a loose threshold time to tolerate longer processing time and network delay.

(2) Hardware-based

Although software-based solutions can reduce cost and be widely adopted as it relies on non-specific hardware support, most of them build upon strong assumptions. There is a gap between the assumptions and the adversary capabilities. For example, SWATT assumes that the adversary cannot forge a valid attestation response faster than the attestation protocol. However, it is hard to be held in practice. In this context, there is a demand for hardware-based attestation to enhance security with more reasonable assumptions. Hardware-based attestation schemes require some tamper-resistant hardware modules, such as TPM, PUFs, SGX.

TPM provides a root of trust for computing systems. With TPM, the attestation schemes can ensure that the attesta-

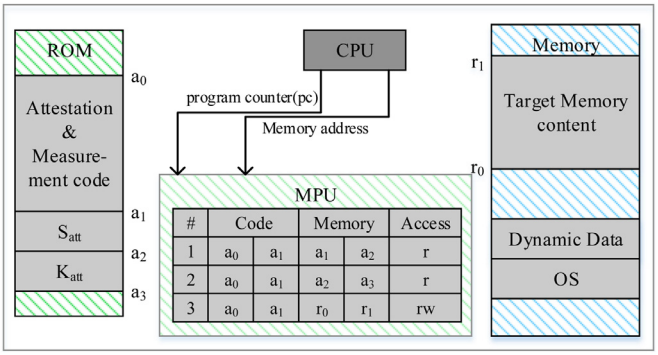


Fig. 3 – Simplified implementation of attestation protocol based on SMART (Eldefrawy et al., 2012).

tion response is trustworthy (Xu et al., 2012). PUFs can provide hardware instance-specific Challenge and Response Pairs (CPRs) by exploiting unavoidable and intrinsic variations induced during chip manufacturing. Thus, the CPRs can play the role of keys to prevent impersonation attacks (Aman and Sikdar, 2018; Feng et al., 2018; Kong et al., 2014). SGX proposed by Intel provides a hardware-enforced isolated execution environment (i.e., enclave) for software execution. The enclave can guarantee the trusted execution of the attestation protocol (Kucab et al., 2021; Wang et al., 2018).

(3) Hybrid

Hardware-based attestation can provide higher security than software-based proposals since the tamper-resistant hardware ensures the correctness of the attestation responses. However, hardware-based solutions are inapplicable to IoT devices that are not provided with the customized hardware components in the production phase. Hybrid attestation schemes leverage the advantages of both hardware-based and software-based schemes. They attempt to overcome the disadvantages of each (Francillon et al., 2014). Hybrid attestation proposals (Brasser et al., 2015; Carpent et al., 2018b; 2018d; Eldefrawy et al., 2017; 2012; Koeberl et al., 2014; Perito and Tsudik, 2010) only require commonly available hardware components, like Micro-Controller Unit(MCU), Memory Protect Unit (MPU), Read-Only Memory (ROM), or FPGA.

Up to now, various lightweight hardware architectures for hybrid-based attestation schemes have been proposed. Traditional architectures (POSE Perito and Tsudik, 2010, SMART Eldefrawy et al., 2012, TrustLite Koeberl et al., 2014, and TyTan Brasser et al., 2015) focus on the protection of secret keys and access control. We choose SMART (Eldefrawy et al., 2012) as an example, which is widely used in various schemes. In SMART, a *prover* computes a MAC of the targeted attestation region to assert authenticity and integrity. Fig. 3 shows a simplified implementation of SMART. The key components contain ROM and MPU. ROM ensures that the stored data cannot be tampered with, such as the attestation protocols and the keys. The MPU controls access to the critical data. For example, the secret key can only be read by the attestation code (as shown as the # 1 of MPU in Fig. 3). Besides, Perito and Tsudik (2010) make use of the ROM and MCU to implement the secure erasure. TrustLite (Koeberl et al., 2014) extends SMART to support interrupt handling. TyTAN (Brasser et al., 2015) provides real-time guarantees and dynamic configuration for some important programs. Schulz et al. (2017) design a boot attestation architecture base on the root of trust integrity using some commercial off-the-shelf MCUs.

At present, hybrid enables the formally verified RA architectures (HYDRA Eldefrawy et al., 2017, VRASED Nunes et al., 2019b, PURE Nunes et al., 2019c, APEX Nunes et al., 2020, and RATA Nunes et al., 2021b). They aim to formally prove the security of each component (i.e., the hardware, software, and the composition of both) and then prove that the entire RA architecture and its implementation are secure. HYDRA (Eldefrawy et al., 2017) makes the first attempt on the verified hybrid RA architecture based on a formally verified seL4 microkernel. Multiple hybrid-based attestation works have been proposed using the HYDRA architecture (Carpent et al., 2018b; 2018d). VRASED (Nunes et al., 2019b) proposes the first formally specified and verified hybrid RA architecture, as shown in Fig. 4. In addition to some common components (e.g., RAM and Flash), similar to previous hybrid architectures (Brasser et al., 2015; Eldefrawy et al., 2012; Koeberl et al., 2014; Schulz et al., 2017), ROM is used to store the secure data. Besides, it exploits FPGA to implement 7 input signals (i.e., *PC*, *irq*, R_{en}, W_{en}, D_{addr}, DMA_{en}, and DMA_{addr} in Fig. 4) and an output signal (*reset*) to guarantee 7 high-level properties of secure attestation: access control, no leakage, secure reset, functional correctness, immutability, atomicity, and controlled invocation. Based on this architecture, it formally verified the architecture via Linear Temporal Logic (LTL) to formalize each sub-module, describing the sub-modules with a Finite State Machine (FSM), and verifying each sub-module and the whole architecture. After VRASED was proposed, many architectures (Nunes et al., 2019c; 2020; 2021b) have made further expansion and research on it. The PURE (Nunes et al., 2019c) provides proofs of remote system-wide reset, secure update, and erasure in RA system. APEX (Nunes et al., 2020) makes use of a 1-bit execution flag EXEC to indicate whether a successful execution is compiled. RATA (Nunes et al., 2021b) introduces a secure logging mechanism to detect the *Time-Of-Check-Time-Of-Use* (TOCTOU) attacks.

4.2. *Swarm attestation*

Since the single-prover attestation schemes are inefficient for large-scale IoT devices that are the trend driven by 5G technology, swarm attestation schemes have attracted great at-

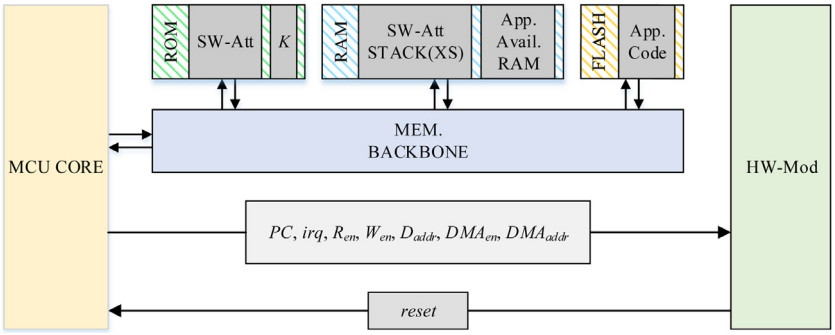


Fig. 4 – VRASED (Nunes et al., 2019b) architecture.

Table 2 – The comparison among swarm attestation schemes in chronological order.

Schemes	Agg_pat	Att_mod	QoSA	Topology	Hardware cost
SEDA Asokan et al. (2015)	Spanning tree	neighbors	C-QoSA	static	SMART Eldefrawy et al. (2012), TrustLite Koeberl et al. (2014)
SANA Ambrosin et al. (2016)	Spanning tree	verifier	L-QoSA	static	TyTAN Brasser et al. (2015)
LISA-s Carpent et al. (2017)	Spanning tree	neighbors	L-QoSA	static	MPU/ROM/secure lock
SeED Ibrahim et al. (2017)	Spanning tree	neighbors	C-QoSA	static	SMART Eldefrawy et al. (2012), TrustLite Koeberl et al. (2014)
Poster Ambrosin et al. (2017)	Broadcast	itself	L-QoSA	dynamic	TEE
B. Gong Gong et al. (2018)	Hierarchy	neighbors	L-QoSA	static	unspecified
SALAD Kohnhäuser et al. (2018)	Broadcast	itself	L-QoSA	dynamic	Boot Schulz et al. (2017)
PADS Ambrosin et al. (2018)	Broadcast	itself	L-QoSA	dynamic	TrustLite Koeberl et al. (2014)
US-AID Ibrahim et al. (2018)	-	neighbors	N-QoSA	dynamic	MPU/ROM/secure clock
WISE Ammar et al. (2018a)	Broadcast	verifier	L-QoSA	static	MPU/ROM/transceiver
HEALED Ibrahim et al. (2019)	-	neighbors	N-QoSA	dynamic	TrustLite Koeberl et al. (2014)
MTRA Tan et al. (2019)	Hierarchy	neighbors	L-QoSA	static	TPM
SAP Nunes et al. (2019a)	Spanning tree	itself	B-QoSA	static	TrustLite Koeberl et al. (2014)
ESDRA Kuang et al. (2019)	Hierarchy	neighbors	L-QoSA	static	MPU/ROM
SHeLA Rabbani et al. (2019)	Hierarchy	verifier	L-QoSA	dynamic	TrustLite Koeberl et al. (2014)
SARA Dushku et al. (2020)	-	verifier	F-QoSA	static	MCU/ROM
SWARNA Kumar et al. (2021)	Spanning tree	verifier	L-QoSA	static	-
FADIA Mansouri et al. (2021)	Spanning tree	itself	L-QoSA	dynamic	MCU/ROM/secure clock
DA Ammar et al. (2021)	Hierarchy	verifier	L-QoSA	static	PoX (APEX Nunes et al. (2020))

Agg_pat and Att_mod denote the aggregate pattern and the attestation mode of the scheme, respectively.

tention recently. A swarm refers to a set of IoT devices with potentially heterogeneous hardware and software configurations. The swarm might be dynamic in terms of both topology and membership. In swarm attestation protocols, the *verifier* hopes to attest the whole swarm with just one interaction. Unlike single-prover attestation, swarm attestation constructions focus on aggregating attestation responses of all devices in the swarm. In swarm attestation, because of the complexity of the network (in terms of the networks topology, message delivery method, attestation mode, aggregation pattern, etc.), one scheme needs to be analyzed from multiple perspectives and thus has different taxonomies. Following, we analyze the peculiarity of classical swarm attestation schemes from two aspects – the *aggregate pattern* and the *attestation mode*, and other features will be discussed in these two categories. Finally, we compare different swarm attestation schemes in chronological order, as shown in Table 2, using the taxonomy discussed below and the evaluation criteria of QoSA, network topology, and hardware cost mentioned in Section 3.

(1) Aggregate pattern

In swarm attestation, each device produces a response, which needs to be aggregated effectively by the *verifier*. There are three main aggregate patterns: *spanning tree*, *broadcasting*, and *hierarchy*.

Spanning tree. The most common aggregate pattern is based on the spanning tree structure. We take SEDA (Asokan et al., 2015), the first swarm attestation scheme, as a study case, as shown in Fig. 5. The *verifier* randomly chooses a device as the initiator device (i.e., D_1 in Fig. 5) and sends an attestation challenge to start the swarm attestation. The initiator transfers the challenge to its neighbors, who relay it to other nodes, eventually generating a tree structure network (as shown by the green line in Fig. 5). Then parent devices wait for the responses of children devices. The lower-layer nodes continuously send back the current count number and their own attestation result to the upper-layer nodes. Finally, as the root of the spanning tree, the initiator device can get the state of the whole swarm and send it to the *verifier*. The spanning-tree structure is widely used in a number of swarm RA schemes

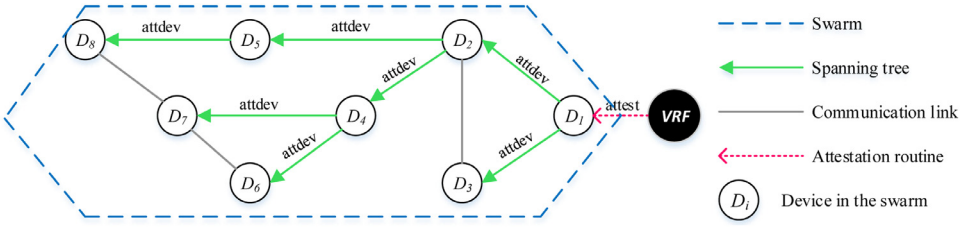


Fig. 5 – Spanning tree structure to aggregate attestation response (adopted from SEDA Asokan et al., 2015).

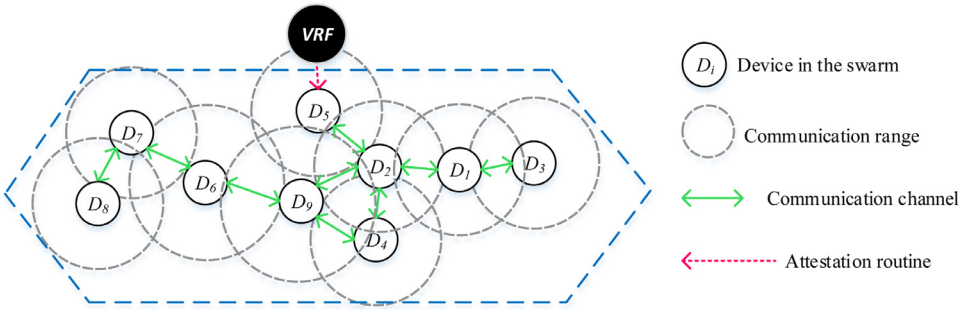


Fig. 6 – Broadcast manner to aggregate attestation response.

(Ambrosin et al., 2016; Carpent et al., 2017; Ibrahim et al., 2017; Mansouri et al., 2021; Nunes et al., 2019a). However, this aggregate pattern usually requires the topology to be static during the attestation. Besides, the unpredictable network delay induced by multi-hop communication makes it challenging to estimate the attestation time. Thus, it is inapplicable to the strict response-time strategy. In this context, the deterministic communication paths (Kumar et al., 2021) utilize IEEE 802.15.4 Time Slotted Channel Hopping (TSCH) link-layer protocol to solve the problem.

Broadcasting. Aggregating attestation report via broadcasting is suitable for highly dynamic networks where devices can move during the attestation process. When a device receives attestation reports from other devices, it will aggregate them with its attestation report and then broadcast reports to all devices within the communication range. Finally, an arbitrary device can aggregate attestation reports of all devices in the swarm. We choose SALAD (Kohnhäuser et al., 2018) as a representative, which is the first swarm attestation scheme for highly dynamic networks, as shown in Fig. 6. The *verifier* initiates the attestation via sending an initiation message to a device (i.e., D_5 in Fig. 6) within its communication range. The receiving devices then propagate the initiation message and check their software integrity to determine whether they are in a healthy state or not (i.e., the mode of *attested by neighbors* in the next subsection). Then the received devices will broadcast attestation reports towards neighboring devices (e.g., the communication between D_5 and D_2). After a period of time, the *verifier* can connect to an arbitrary device to get the attestation result which contains the list of all healthy devices. The broadcasting pattern is adopted by several RA schemes for highly dynamic networks (Ambrosin et al., 2017; 2018) or choosing a subset of devices for attestation (Ammar et al., 2018a) (instead of all devices in the swarm). However, it incurs great communication overhead.

Hierarchy. The hierarchy structure is suitable for the network where devices are with varying resource capabilities, as shown in Fig. 7. In such a network, devices with more robust computing power or a higher level of security features can be selected as cluster head nodes (i.e., C_i in Fig. 7). Then the cluster head nodes could communicate with the devices in their clusters (i.e., D_i in Fig. 7). Different schemes will choose various device characteristics as the classification criteria, such as the security or performance of devices. We choose three representative schemes (Ammar et al., 2021; Gong et al., 2018; Tan et al., 2019) to demonstrate the hierarchy structure. According to their responsibilities, in Gong et al. (2018), devices are divided into three categories: ordinary devices, cluster head devices, and data aggregate devices. Ordinary devices report the attestation responses to the cluster head devices, and cluster head devices report to the data aggregate devices. According to the hardware resources, MART (Tan et al., 2019) divides devices into three types: fully trusted devices (e.g., base station), devices equipped with TPM, and resource-limited devices. The former two types of devices are used to verify the state of the last type of devices. DA (Delegated Attestation) (Ammar et al., 2021) enables the gateways to connect to the attestation proxy (based on PoX architecture Nunes et al., 2020), which then acts as the cluster headers in Fig. 6.

Notably, hierarchy structure is also applicable to highly dynamic networks (e.g., SHeLA Rabbani et al., 2019). The attestation responses of mobile *provers* can be transferred to the originally registered edge *verifiers* via some static edge devices. However, the hierarchy structure may bring some redundant devices (e.g., SHeLA needs to introduce some edge device into the swarm) or result in higher hardware requirements only

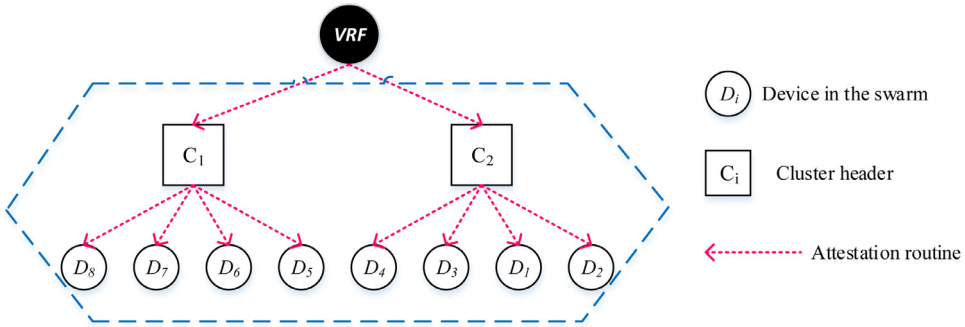


Fig. 7 – Hierarchy structure to aggregate attestation response.

available in high-level devices. Additionally, the hierarchy pattern usually assumes that the nodes of adjacent layers can communicate in one hop, which may not be the case in practice.

(2) Attestation modes

When extending the attestation from an individual device to a swarm, different attestation modes have been proposed. According to which party assesses the state of the target *prover*, modes can be classified into three categories: *attested by neighbors*, *attested by the verifier*, and *attested by the device itself*.

Attested by neighbors. The neighbors are the direct observers of the device. Thus, the state of one device can be attested by its neighboring devices. This attestation mode is widely used in spanning tree structure (Asokan et al., 2015; Carpent et al., 2017). The direct attestation result of one device is given by its parent device in the spanning tree. Since the neighbor devices only provide the attestation results and do not need to transfer the detailed information (e.g., the hash value of the binary code), the attestation reports are small, greatly reducing the communication overhead.

Besides, this attestation mode is applied in certain interesting schemes with specific purposes. US-AID (Ibrahim et al., 2018) utilizes this mode for physical attack detection, as detailed in Section 6.1. HEALED (Ibrahim et al., 2019) is a healing protocol that recovers the malicious software blocks of the compromised devices, ensuring the security of the whole network. ESDRA (Kuang et al., 2019) proposes a distributed attestation scheme, which uses multiple neighbors to attest one node simultaneously and then calculates the final attestation result by the weight-based calculation. The rationale is that determination of the *prover*'s state based on the attestation result of a single neighbor is too rough, especially given such a neighbor device is not a fully trusted device.

Attested by the verifier. Most swarm attestations adopt the mode of *attested by neighbors* in order to reduce the size of attestation reports. However, this mode may suffer from security vulnerabilities as the neighbor may be compromised. Therefore, only the assessment given by the *verifier* can be considered credible. The difficulty of this attestation mode is that the required detailed attestation reports of each device make the size of the attestation report extremely large. A dedicated aggregation algorithm is needed to overcome the above concern.

In this context, the Optimistic Aggregate Signature (OAS) (Ambrosin et al., 2016) protocol guarantees the security of aggregating attestation reports via a spanning tree structure. The parent devices aggregate the attestation responses of children devices, which minimizes the report size while increasing the credibility of the attestation results. In comparison, the publish/subscribe protocol (Dushku et al., 2020) keeps accumulating the attestation responses in the order of publishers and subscribers, which can detect both the software integrity of IoT devices and the non-intended interactions with the malicious devices. But this mechanism cannot reduce the report size.

Attested by the device itself. In some swarm attestation schemes (Ambrosin et al., 2017; 2018; Kohnhäuser et al., 2018; Nunes et al., 2019a), the assessment is done by the *prover* itself. In these schemes, *provers* require extra hardware (e.g., Trusted Execution Environment (TEE)) to ensure the trustworthiness of the assessment. Thus, they can judge their own reliability by comparing the measurement with the expected state. The expected state can be the valid software configuration sent from the *verifier* (Kohnhäuser et al., 2018), or the one pre-stored inside the device (Ambrosin et al., 2017; 2018; Nunes et al., 2019a). Then the *provers* forward the assessment results to the *verifier*. This attestation mode reduces the impact of device mobility on the RA process so that it is preferable for highly dynamic networks.

On this basis, self-measurement is further proposed (Carpent et al., 2018d). Instead of forwarding the results directly to the *verifier*, *provers* attest themselves and temporarily store the results locally. After several attestation rounds, the *verifier* will send a request to the *provers* and collect all the results. Self-measurement can restrict the capabilities of verifier-impersonation DOS attackers (discussed in Section 7.2), as well as reduce the probability of transient and self-relocating malware evading attestation (discussed in Section 8.3).

4.3. *Discussion*

Single-prover attestation schemes often assume a mobile *verifier* or multi-verifiers to ensure that the *prover* is within a *verifier*'s communication range. Unfortunately, single-prover attestation schemes usually exhibit low efficiency when multiple devices are required to be attested. With the explo-

sive growth in the number of IoT devices, swarm attestation schemes are demanded. Compared with single-prover attestation, swarm attestation has the following advantages. (1) The *verifier* can verify the integrity of the swarm as a whole. (2) Swarm attestation is more efficient in the swarm. However, considerations should be taken in swarm scenarios.

- *How to aggregate attestation reports from different devices securely and efficiently?* Different aggregation patterns and attestation modes should be applied according to varying characteristics of the network topology.
- *How to identify the particular compromised devices efficiently?* Some swarm attestation schemes can only determine the swarm's integrity rather than point out the specific compromised devices. However, other L-QoSA swarm attestation schemes that can identify compromised devices make the attestation reports extremely large.
- *How to measure the integrity of devices with different software configurations at one attestation iteration?* It is impractical for the *verifier* to know the expected software configuration of each corresponding device so that the solution should be agnostic to the software configurations.

5. Attestation for runtime attack

Runtime attacks usually exploit the vulnerabilities in the programming language to alter the behaviors of the device or steal private information. Runtime vulnerabilities could be classified into two categories: control-flow vulnerabilities (Checkoway et al., 2010; Snow et al., 2013), and non-control data vulnerabilities (Chen et al., 2005). Runtime attacks modify the dynamic part in the device's memory, such as dynamic data variables or the values in the stack, which brings a great challenge when verifying the trustiness of these dynamic data. Some mitigation of runtime exploitation technologies have been proposed, e.g., Data Execution Prevention (DEP) (2007), static cookies (Cowan et al., 1998), Address Space Layout Randomization (ASLR) (PaX-Team, 2001), control-flow integrity (CFI) (Abadi et al., 2005), code-pointer integrity (CPI) Kuznetsov et al. (2014). However, their implementations require the change of language, compiler, or binary code, because the protection mechanisms are executed locally. This is often beyond the capabilities of resource-restricted IoT devices. Therefore, dynamic RA schemes are proposed. In this section, we analyze this kind of scheme where the attestation response can be the dynamic integrity evidence or control-flow paths of the programs.

5.1. Dynamic integrity evidence

Although the dynamic data in memory is unpredictable, they often satisfy some dynamic properties. A *verifier* can detect the runtime integrity via these dynamic properties (e.g., ReDAS Kil et al., 2009). ReDAS identifies two types of dynamic properties: *structural integrity* and *global data integrity*. An application having structural integrity means that it satisfies all *structural constraints* at runtime. The structural constraints are the properties that application binary code must be satisfied, such as stack constraints and return address constraints. Similarly, the global data integrity refers to all the *data invariants* being satisfied. The data invariants refer to the values of data variables or relations among them. The integrity measurement component examines these dynamic properties in various applications at runtime and provides integrity evidence. However, it is not easy to identify and collect all the dynamic properties. Additionally, finding the correlation between the dynamic properties and various runtime attacks, as well as detecting all types of attacks, is an arduous task.

5.2. Control-flow path

At present, the most common RA methods against runtime attacks are based on Control Flow Graph (CFG), also known as control-flow attestation. CFG is an abstract of program flow. In CFG, nodes represent a code block, and edges represent control-flow transitions. Runtime attacks mainly aim to abuse the existing code to alter the behavior of the device, which essentially changes the control flow.

Let us take C-FLAT (Abera et al., 2016) as a study case to introduce the basic principles of control-flow attestation, which is the first control-flow attestation scheme against runtime attacks. As shown in Fig. 8, a runtime attacker can alter the control flow at line 3, (i) changing the control flow to the malicious code or (ii) changing the unprivileged path to the privileged path (i.e., control-flow attacks). He can also (iii) modify some critical data to influence the control flow (i.e., non-control data attacks). The key insight of C-FLAT is to verify the control-flow path of the *prover*. The *verifier* firstly performs an offline pre-processing and generates the CFG of the application module via static analysis. Then the *verifier* measures each possible control-flow path and stores them via a measurement function – cumulative hash function as shown in Fig. 9. The cumulative hash function H takes the node id N_i and the previous hash result H_{prev} as input to output a new hash value, i.e., $H_i = H(H_{prev}; N_i)$. Finally, it outputs an *Auth*, which can be regarded as identification of the execution path. During the attestation process, the *prover* also calculates the current control-flow path value using the same method. Thus, the *verifier* can determine whether the *prover* is executed as expected.

Based on the same concept, some other control-flow attestation schemes (Dessouky et al., 2017; Nunes et al., 2021a; Zeitouni et al., 2017) have been proposed to improve security and efficiency, which require the devices to be equipped with some special hardware, such as Branch Filter, Loop monitor, Hash Controller, and Hash Lookup. Moreover, the detection capabilities of control-flow attestation are another focus of researchers. Besides the control-flow integrity, the interactions of devices (Conti et al., 2019), the full uniqueness of control flow (Kuang et al., 2020), and the data-only attacks (e.g., Data-Oriented Programming (DOP)) (Dessouky et al., 2018; Nunes et al., 2021c; Sun et al., 2020) can be detected and verified.

5.3. Discussion

It is a challenge to collect a complete set of dynamic properties that can be used to precisely measure the running devices. The existing dynamic integrity evidence can only detect part of runtime attacks. Therefore, the control-flow attestation

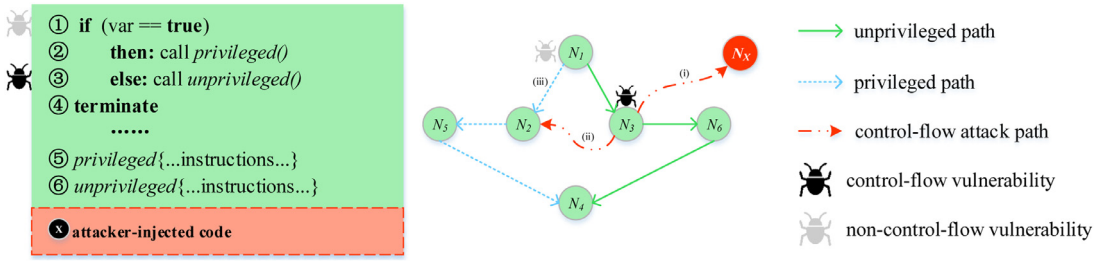


Fig. 8 – Overview of runtime attacks (adopted from C-FLAT Abera et al., 2016).

Table 3 – The comparison among ra schemes for runtime attacks in chronological order.

Schemes	Hardware cost	Control-flow graph
ReDAS Kil et al. (2009)	TPM	No
C-FLAT Abera et al. (2016)	trust anchor (TEE)	Yes
LO-FAT Dessouky et al. (2017)	Special Hardware	Yes
ATRIUM Zeitouni et al. (2017)	Special Hardware	Yes
LiteHAX Dessouky et al. (2018)	Special Hardware	Yes
RADIS Conti et al. (2019)	trust anchor	Yes
Tiny-CFA Nunes et al. (2021a)	PoX (APEX Nunes et al., 2020)	Yes
OAT Sun et al. (2020)	trust anchor (TEE)	Yes
DO-RA Kuang et al. (2020)	trust anchor (TEE)	Yes
DIALED Nunes et al. (2021c)	PoX (APEX Nunes et al., 2020)	Yes

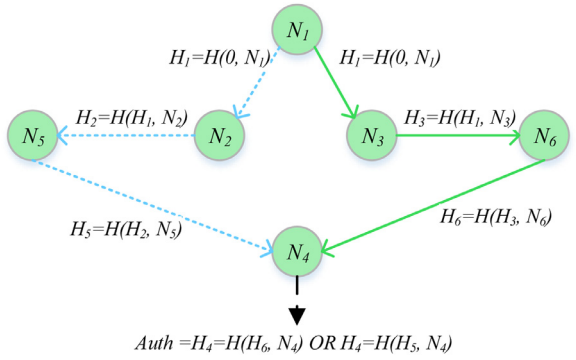


Fig. 9 – C-FLAT control-flow attestation (adopted from C-FLAT Abera et al., 2016).

appears to be a better option. Nonetheless, (1) most current control-flow attestation methods can only detect control-flow attacks but still suffer from non-control data vulnerabilities. In this context, LiteHAX (Dessouky et al., 2018), OAT (Sun et al., 2020), and DIALED (Nunes et al., 2021c) propose the Data-Flow Attestation (DFA) to detect the data-only attacks, whereas they cannot guarantee data confidentiality. (2) Current control-flow attestation methods are mainly based on program segmentation that is a highly complex operation, especially for some intricate structures. Without a proper program segmentation, the *verifier* cannot obtain a well-designed CFG for the attestation. Table 3 summarizes and compares the hardware cost and the attestation method of different attestation schemes against runtime attacks in chronological order.

6. Attestation for physical attack

Both semi-intrusive and invasive physical attacks need to capture the devices for a specific time, from hours to weeks. Therefore, the *verifier* can determine whether physical attacks potentially occur by checking the presence of the target devices (so-called capture detection). Supposing that the minimum time needed for physical attacks is T_{cap}, the *verifier* checks each device's presence every ΔT period ($\Delta T < T_{cap}$). Then, it can identify the (offline) devices that suffered from physical attacks. Note that physical attacks are always considered under the swarm scenarios because only in this case can the device be physically attacked without being noticed. The capture detection methods used in swarm attestation schemes mainly have two types, i.e., heartbeat technology and session key update technology.

6.1. *Heartbeat technology*

The core idea of heartbeat technology is that devices emit their heartbeat messages periodically to prove themselves online. Each device will emit a heartbeat message when receiving a new heartbeat message from neighbors, or ΔT time has elapsed. The heartbeat message includes a timestamp, a counter number, and a device identifier. Then the heartbeat message could be verified by other swarm devices (Ibrahim et al., 2016) or the neighbor devices (Ibrahim et al., 2018).

Compared to other attacks, the attestation of physical attacks may require a higher attestation frequency to detect the continuous presence of devices. Thus, the time overhead is an essential factor when evaluating an attestation scheme.

Table 4 – Comparisons of different attestation schemes for physical attacks in chronological order.

Schemes	Energy cost per device	Time	Topology	QoSA
DARPA Ibrahim et al. (2016)	broadcasting	$O(n^2)$	dynamic	L-QoSA
SCAPI Kohnhäuser et al. (2017)	broadcasting	$O(n)$	dynamic	B/L-QoSA
US-AID Ibrahim et al. (2018)	$E_i \leq 4gE_{mac} + 56gE_{send} + 56gE_{recv}$	$O(1)$	dynamic	N-QoSA
silmIoT Ammar et al. (2018b)	broadcasting	$O(n)$	dynamic	L-QoSA
EAPA Yan et al. (2019)	$E_i \leq gE_{enc} + gE_{dec} + 24gE_{send} + 24gE_{recv}$	$O(1)$	static	L-QoSA

n denotes the number of devices in the swarm and g denotes the number of neighbors. E_i, E_{mac}, E_{send}, E_{recv}, E_{enc}, and E_{dec} denote the energy cost of one device, computing or verifying one MAC, sending one byte, receiving one byte, encryption, and decryption, respectively.

For example, DARPA (Ibrahim et al., 2016), the first attestation scheme for physical attacks using heartbeat technology, utilizes the broadcasting to exchange the heartbeat messages of any two nodes. Thus, it suffers from high time cost, and the complexity of attestation time is $O(n^2)$, where n is the number of devices in the swarm. In comparison, US-AID (Ibrahim et al., 2018) makes the heartbeat messages exchange only with neighboring nodes, so its complexity of attestation time is reduced to be $O(1)$.

6.2. Session key update technology

Session key update technology can naturally rule out the absent devices (regarded as physically attacked). We take SCAPI (Kohnhäuser et al., 2017), the first swarm attestation scheme using the session key update technology, as a study case. It sets a symmetric encryption key between two devices via an XOR operation between the session key sk and channel key ck. The channel key ck is static after the channel key establishment. The session key sk is dynamic and updated every ΔT time generated by a so-called leader device in the swarm. The current sk is denoted as sk_{cur}, and the newly generated sk is denoted as sk_{next}. The leader device then encrypts sk_{next} using sk_{cur} and transmits the ciphertext to other devices. Then, only the device that has the correct sk_{cur} can correctly retrieve the newest sk_{next}. Since the physically attacked device will be offline for T_{cap} time, it can no longer communicate with other devices in the network using the wrong session key. Since all the devices in the swarm need to update their session keys, SCAPI's complexity of attestation time is $O(n)$, where n is the number of devices in the swarm. Based on a similar idea, slimIoT (Ammar et al., 2018b) proposes another attestation scheme against physical attacks based on an efficient broadcast authentication method, whose complexity of attestation time is also $O(n)$. EAPA (Yan et al., 2019) combines the heartbeat technology with session key update technology, which makes the interaction limited to neighbor nodes, and thus complexity of attestation time is $O(1)$.

6.3. Discussion

Compared with the heartbeat technology, the periodically updated session key can naturally rule out the absent devices. However, session key update technology results in higher computing overhead since it relies on more cryptographic operations. Essentially, the ideas of periodic heartbeat-message and session key update strategies share the same concept for detecting physical attacks. They both utilize periodic information to determine the target devices' existence to reflect the occurrence of physical attacks. Therefore, the verification and transmission of periodic messages require significant computation and time consumption, which may affect the regular operation of devices. In addition, session key update technology is vulnerable to desynchronization attacks. If an attacker blocks (through, e.g., channel fuzzing or fake relay) the message that contains sk_{next}, devices will fail to communicate with other devices in the following rounds due to the different session keys. Table 4 compares the evaluation criteria of energy cost, time cost, network topology, and QoSA of different attestation schemes for physical attacks in chronological order, where DARPA (Ibrahim et al., 2016), SCAPI (Kohnhäuser et al., 2017), and silmIoT (Ammar et al., 2018b) utilize broadcasting to deliver the messages, and their energy cost is incalculable. Besides, the evaluation criterion of hardware cost is the same for all of these schemes, i.e., MPU, ROM, and a secure clock.

7. Attestation for verifier-impersonation DoS attack

In most traditional attestation schemes, the *verifier* is always trusted. However, the attacker can indeed impersonate a *verifier* and then send some fake attestation requests that can trigger potentially costly attestation processes in the *prover* to launch a kind of DoS attack. We name such attacks as verifier-impersonation DoS attacks. The existing attestation schemes usually check the validation of attestation requests to prevent verifier-impersonation DoS attacks. Furthermore, the non-interactive attestation approach is recognized to be naturally resilient to verifier-impersonation DoS attacks. Moreover, self-measurement can mitigate the verifier-impersonation DoS attacks by reducing the number of challenges. This section introduces the representative attestation schemes of the above three policies, respectively.

7.1. Validation of attestation request

Verifying the authenticity of attestation requests is a straightforward approach when combating verifier-impersonation DoS attacks. The most common way is to use some cryptographic methods, such as the public key or symmetric cryptography. Among them, PUF has been employed by a number of RA schemes (Aman and Sikdar, 2018; Feng et al., 2018) attributing to its randomness, uniqueness, and ease of use. It exploits the unclonable physical variations in the semiconduc-

tor manufacturing process to generate a "digital fingerprint" as the unique identity of the device.

Another way is to utilize a one-way hash chain (Tan et al., 2019) to verify the attestation requests. A one-way hash chain is a set of hash values, such as $(V_0, V_1, \ldots, V_i, \ldots, V_k)$, where $V_i = hash(V_{i-1})$. Each request contains one of the hash values V_i. The *prover* stores a value V_j locally. Only if $i < j$, the *prover* accepts the request and replaces V_j with V_i. Therefore, it is hard for a malicious *verifier* to find or forge a valid challenge.

However, just verifying the authenticity of requests is not always sufficient. The attacker can pass the authenticity verification by forwarding the eavesdropped requests or replaying the previous requests. Thus, a challenge refresh mechanism can be taken into consideration Brasser et al. (2016) to harden the verifier-impersonation DoS attacks as follows.

- *Nonces*. The *prover* records all the history of nonces. But this method requires a large amount of storage space, can only identify the challenges received before, and cannot identify the challenges forwarded by other devices.
- *Counters*. The *verifier* sets a monotonically increasing counter. Each attestation request must contain a current value of the counter. Only when the new request's value is greater than the last request's value, the *prover* would accept the new request.
- *Timestamps*. The current timestamp is added to the request to serve freshness. But it needs to deploy synchronized clocks on the *prover* and the *verifier*, which would cause additional hardware overhead.

7.2. Non-interactive attestation

Another option to resist verifier-impersonation DoS is the non-interactive attestation mode (Ibrahim et al., 2017), which removes the process of sending challenges in attestation schemes. In other words, the attestation process in a *prover* is no longer triggered by the *verifier* but is automatically triggered by a timing program of the *prover*. In this way, it is inherently resilient to verifier-impersonation DoS attacks. As a premise of non-interactive attestation schemes, both the *verifier* and the *prover* have a loosely synchronized clock.

7.3. Self-measurement

Similar to non-interactive attestation, the ERASMUS (Carpent et al., 2018d) utilizes the self-measurement to reduce the number of challenge processes, which forces each *prover* to attest itself based on a pre-established schedule. The *provers* store the results locally. Then, the *verifier* occasionally contacts the *provers* and collects all stored results to verify the devices' status in this period. Therefore, it can harden the verifier-impersonation DoS attack to a large extent.

7.4. Discussion

Preventing fake attestation requests from unauthorized *verifiers* is the main strategy to avoid verifier-impersonation DoS attacks. Though checking the validation of attestation requests can effectively resist this type of attack, the overhead imposed on the *prover* could be high. Meanwhile, a large number of attestation requests can also force the *prover* to perform verification continuously, which turns out to be another kind of DoS attack. In contrast, non-interactive attestation is inherently resilient to verifier-impersonation DoS attacks. However, the measurement of the *prover*'s software configuration should be tied to its generation time to ensure freshness. Thus, real-time write-protected clocks are required to provide time synchronization between the *verifier* and *provers*. Moreover, the self-measurement mitigates the verifier-impersonation DoS attack, but cannot totally resist it because the *verifier* still needs to collect information using the challenge-response mechanism.

8. Attestation for transient and self-relocating malware

In real-world scenarios, there is a significant conflict between the atomicity of the attestation process and emergency operations. Some safety-critical IoT devices must update data strictly to ensure the correctness and timeliness of some essential functions, such as fire alarm applications. As reported in Carpent et al. (2018a), the attestation process would cost approximately 7 s given a 1 GB attested memory. For a fire alarm, 7 s attestation process might cause disastrous consequences. However, most existing RA mechanisms usually need a time-consuming and atomic attestation process for security requirements. Once the atomicity is relaxed and the attestation process is allowed to be interrupted, the transient and self-relocating malware may evade the attestation. Malware abuses interrupt to relocate themselves tactically in the attestation process, transferring themselves to the undetected region or the region that has been attested. To mitigate the transient and self-relocating malware, *memory sliding locking*, *shuffled measurements*, *self-measurements*, and *distributed attestation* can be used.

8.1. Memory sliding locking

Carpent et al. (2018b) discuss "memory locking" from various aspects. Among them, the sliding lock is a more practical idea. It has two manners: *decreasing lock* and *increasing lock*. *Decreasing lock* locks all memory modules at the beginning of attestation (i.e., temporarily making them read-only). Then, with the execution of the attestation process, it unlocks every module attested. On the contrary, *increasing lock* does not lock any memory at the beginning while locking each module attested during the attestation.

8.2. Shuffled measurements

The shuffled measurements take advantage of randomization. SMARM (Carpent et al., 2018c) is a representative example. During the attestation, SMARM adopts the shuffled measurements where a *prover* computes the MAC on disorderly memory blocks. In this way, the attacker cannot determine which memory block is "safe" and thus cannot move the malware in a targeted manner.

8.3. *Self-measurements*

We have discussed this method in Section 4.2 (*attested by itself*). Because interaction is complicated and time-consuming, self-measurements can significantly reduce the attestation time, thereby minimizing the negative impact resulted from attestation processes on regular tasks of the devices.

8.4. *Distributed attestation*

The distributed attestation mode is proposed for swarm attestation. In swarm environments, the conflict between interruption and atomicity becomes more intense due to the superposition of network delay and node attestation time. Therefore, distributed attestation mode aims to decrease the attestation time for each device in the swarm. Typical distributed attestation schemes are ESDRA (Kuang et al., 2019) and SARA (Dushku et al., 2020). Both can enable the devices to start normal operation immediately after completing the local attestation tasks without waiting for the attestation results of other devices in the swarm.

8.5. *Discussion*

Memory sliding locking is essentially a trade-off between interruption and the atomicity of RA. It is incapable of completely mitigating problems caused by the interruption. If the code of the emergency task is stored in the locking region exactly, the conflict still exists. The shuffled measurement is a probabilistic method, which needs to consider the evaluation criterion of accuracy, especially the false negatives, as explained in Section 3.2. Any attempt to minimize the false negatives will inevitably increase the evaluation criterion of time cost. Besides, the shuffled measurements need additional memory to store the permutation of memory blocks. Both self-measurements and distributed attestation mitigate the conflict caused by atomicity from the perspective of reducing the attestation time of devices.

9. Future research

We have discussed various RA schemes against different attacks. Meanwhile, the advantages and disadvantages of the corresponding schemes are analyzed and compared. According to the evaluation criteria in Section 3.1, we summarize and compare the protection capabilities (for software attacks, control-flow attacks, physical attacks, verifier-impersonation DoS attacks, and transient & self-relocating malware) of attestation schemes from 2015 in chronological order as shown in Table 5. For software attacks, the *prover* must generate the measurement over its current software configuration. The measurement can be a hash over the target memory content. For runtime attacks, measurement must reflect the runtime behavior of the *prover*. Therefore, the measurement can be hash over the control-flow path. The underlying observation of physical attack attestation is that the device under physical attack will be offline for a while. Moreover, Attackers can exploit the flaws of the RA to launch malicious actions, including verifier-impersonation DoS and transient and self-relocating malware. It is worth noting that the attestation proposals can be combined, and one scheme can be effective against more than one type of attack.

We have identified a number of unresolved problems and challenges. Some of them have been recognized by existing research, but efficient solutions have not yet been provided. We discuss these challenges and corresponding potential solutions.

9.1. *How to detect the self-deleting malware?*

Although transient and self-relocating malware already has some solutions, they still cannot solve the problem perfectly. What is more, it reveals that the attacker has the ability to know the time of attestation. In this context, self-deleting malware becomes an attack method that is much more difficult to detect. Remote attestation is a periodic process, and there exists a long "vacuum" period between two attestations. Thus, during this period, attackers can perform malicious operations and then delete their own traces, forming so-called TOCTOU attacks. The existing schemes can hardly detect this self-deleting malware. RATA (Nunes et al., 2021b) makes use of a secure logging mechanism to record the latest memory modification. However, RATA is not suitable for self-modifying code and Just-In-Time (JIT) compilation (jp, 2003). Therefore, a more reliable, automated, and intelligent log authentication mechanism is needed to distinguish abnormal modifications from normal ones.

9.2. *How to design an efficient RA scheme for highly dynamic network topology?*

The existing swarm attestation schemes for highly dynamic network topology usually apply the broadcasting manner to aggregate the attestation reports. However, the broadcasting manner results in greatly increased communication overhead. Furthermore, the more attestation reports forwarded to the network, the higher chance of an attacker successfully eavesdropping and analyzing the attestation reports. It becomes a challenge to distinguish the correct attestation report if differing attestation reports of the same device are received. Therefore, a more efficient and secure swarm attestation scheme for highly dynamic networks should be devised. We believe that this problem has been solved in the mobile communication network. Introducing some redundant and fixed "base stations" may be a good option, but it is necessary to measure whether the overhead is acceptable. SHeLA (Rabbani et al., 2019) seems to have made the first attempt, but there are still some unsolved challenges, and the protocol can be optimized.

9.3. *How to detect the non-control-data attacks?*

Attackers can manipulate some data to alter the device's behavior, gain high privileges, or steal sensitive information. The tampered data variables can be the control data that directly affect the control flow, such as the return of functions or other non-control data that do not directly change the control flow. Most runtime attestation schemes are limited to assess the control flow. In other words, they cannot detect non-control

Table 5 – The protection capability of RA schemes from 2015 in chronological order.

Schemes	Protection				
	Soft_Att	CF_Att	Phy_Att	DoS	Tran-Mal
TyTAN Brasser et al. (2015)	✓	✗	✗	✗	✗
SEDA Asokan et al. (2015)	✓	✗	✗	✗	✗
SANA Ambrosin et al. (2016)	✓	✗	✓	P	✗
C-FLAT Abera et al. (2016)	-	✓	✗	✗	✗
DARPA Ibrahim et al. (2016)	✓	✗	✓	✗	✗
LISA-s Carpent et al. (2017)	✓	✗	✗	✗	✗
LO-FAT Dessouky et al. (2017)	-	✓	✗	✗	✗
SeED Ibrahim et al. (2017)	✓	✗	✗	✓	✗
SCAPI Kohnhäuser et al. (2017)	✓	✗	✓	✓	✗
Poster Ambrosin et al. (2017)	✓	✗	✗	✗	✗
HYDRA Eldefrawy et al. (2017)	✓	✗	✗	✗	✗
Boot Schulz et al. (2017)	✓	✗	✗	✗	✗
ATRIUM Zeitouni et al. (2017)	P	✓	P	✗	✗
B. Gong Gong et al. (2018)	✓	✗	✗	✗	✗
J. Wang Wang et al. (2018)	✓	✗	✗	✗	✗
AAoT Feng et al. (2018)	✓	✗	✗	✗	✗
ERASMUS Carpent et al. (2018d)	✓	✗	✗	P	✗
SMARM Carpent et al. (2018c)	✓	✗	✗	✗	P
SALAD Kohnhäuser et al. (2018)	✓	✗	✗	P	✗
PADS Ambrosin et al. (2018)	✓	✗	✗	✓	✗
US-AID Ibrahim et al. (2018)	✓	✗	✓	✗	✗
WISE Ammar et al. (2018a)	✓	✗	✗	✗	✗
ATT-Auth Aman and Sikdar (2018)	✓	✗	✗	✗	✗
LiteHAX Dessouky et al. (2018)	-	✓	✗	✗	✗
slimIoT Ammar et al. (2018b)	✓	✗	✓	✗	✗
HEALED Ibrahim et al. (2019)	✓	✗	✗	✗	✗
MTRA Tan et al. (2019)	✓	✗	✗	✓	✗
RADIS Conti et al. (2019)	P	✓	✗	✗	✗
SAP Nunes et al. (2019a)	✓	✗	✗	✓	✗
VRASED Nunes et al. (2019b)	✓	✗	✗	✗	✗
ESDRA Kuang et al. (2019)	✓	✗	✗	✗	P
R.V. SteinerSteiner and Lupu (2019)	✓	✗	✗	✗	✗
PURE Nunes et al. (2019c)	✓	✗	✗	✗	✗
SHeLA Rabbani et al. (2019)	✓	✗	✗	✗	✗
EAPA Yan et al. (2019)	✓	✗	✓	✗	✗
SARA Dushku et al. (2020)	✓	✗	✗	✗	P
Tiny-CFA Nunes et al. (2021a)	✓	✓	✗	✗	✗
OAT Sun et al. (2020)	✓	✓	✗	✗	✗
APEX Nunes et al. (2020)	✓	P	✗	✗	✗
DO-RA Kuang et al. (2020)	✓	✓	✗	✗	✗
M. Kucab Kucab et al. (2021)	✓	✗	✗	P	✗
SWARNA Kumar et al. (2021)	✓	✗	✗	P	✗
FADIA Mansouri et al. (2021)	✓	✗	✗	P	✗
DA Ammar et al. (2021)	✓	✗	✗	P	✗
DIALED Nunes et al. (2021c)	✓	✓	✗	✗	✗
RATA Nunes et al. (2021b)	✓	✗	✗	✗	✗

"P" denotes the scheme can only detect this type of attack in some specific cases. "-" denotes the scheme utilizes some existing technologies (e.g., DEP) to prevent this type of attack. Soft_Att, CF_Att, Phy_Att, DoS, and Tran-Mal denote the protection capabilities for software attacks, control-flow attacks, physical attacks, verifier-impersonation DoS attack, and transient & self-relocating malware of the scheme, respectively.

data vulnerabilities. LiteHAX (Dessouky et al., 2018) and DIALED (Nunes et al., 2021c) propose to record all the data modified. But this may cause the attestation response to be particularly large. Therefore, as OAT (Sun et al., 2020) does, identifying the critical non-control data and guaranteeing the critical data integrity locally may be a more realistic option. Only then can costs be controlled and solutions be implemented in practice.

Besides, there is another potential solution from our perspectives. Even though non-control-data attacks delicately deviate the program execution from the CFG, they make the program traverse distinct control-flow paths from the legitimate paths. Thus, an interesting direction would be to see whether it is possible to detect such deviation, where machine learning can be a useful tool.

9.4. *How to design an efficient RA scheme for physical attacks?*

The existing attestation schemes for physical attacks are all based on the assumption that physical attacks need to take the target device offline for a non-negligible time. However,

they require almost zero tolerance for network delay and packet loss since they may mistakenly regard temporarily unreachable devices as physically attacked. Moreover, the existing heartbeat and session key update mechanism would bring notable overhead because the heartbeat cycle needs to be less than the minimum physical attack time, which requires a higher attestation frequency. Therefore, it is worth considering detecting the physical attacks from new perspectives, such as examining and verifying the device's hardware configuration.

9.5. *How to design an intelligent attestation scheme?*

AI technology, such as machine learning, has demonstrated stunning performance on a wide range of applications. It is worth integrating AI into RA services. Suppose the *verifier* can utilize a machine learning model to judge the state of a device through an arbitrary program attestation rather than a program with a fixed start and end. In that case, it can harden an attacker to predict, forge, or replay the attestation response. Moreover, it would be more desirable to implement this attestation method in runtime attack detection. Because this means that the attestation program can directly verify the currently executing program, it will not affect the normal execution of the device.

In addition, the *verifier* can utilize machine learning to predict a *prover*'s state by collecting the behaviors of the *prover*. This intelligent attestation mode can decrease the overhead of the *prover* and reduce the impact on its regular operation. Note that considering the complexity of various attacks, the accuracy of the machine learning method in determining the devices' states needs to be further studied.

9.6. *How to choose the swarm attestation mechanism?*

The majority of swarm attestation schemes follow a collective attestation mechanism that collects the entire swarm's information to the *verifier*. This is because the *verifier* has to determine the security state of each device. Though a detailed report is preferable, it incurs substantial communication overhead when such a detailed report of every device has to be transmitted. Another attestation mechanism is the accusation mechanism (Kuang et al., 2019). It does not aggregate all the state information of each node but exploits neighbor nodes to act as supervisors. The accusation mode only requires reporting the compromised nodes instead of transferring the benign ones' information, which significantly reduces the communication cost. However, the challenges are ensuring that the compromised device will be reported and distinguishing the malicious reports.

10. Conclusion

The 5G communication technology has become a fertile ground for IoT, which makes the number of IoT devices explode. As IoT devices are ubiquitously integrated into every corner of our society, the security service for IoT devices has to be carefully examined and implemented. Remote attestation is regarded as one of the fundamental security services to ensure their security. In this paper, we summarized the state-of-the-art RA schemes to provide a comprehensive review. Analyzing all adversarial models to the best of our knowledge, we designed an elaborate adversarial model that divides attackers into two categories: normal attackers (including software attacks, runtime attacks, and physical attacks) and knowledgeable attackers (including verifier-impersonation DOS attacks and transient & self-relocating malware). Then, building upon the adversarial model and evaluation criteria, we analyzed and compared various RA schemes from the perspective of attack types.

Specifically, the RA for software attacks contains single-prover attestation and swarm attestation. Single-prover attestation focus on generating a trustworthy and verifiable attestation response, while swarm attestation utilizes parallel computing and inter-device interaction to check the security status of all devices in the network. In addition, most RA schemes against runtime attacks are based on the CFG to record the execution path of the program. Furthermore, all RA schemes against physical attacks rely on capture detection to verify the online states of devices. Moreover, the strategies to resist verifier-impersonation DOS attacks and transient & self-relocating malware are enhancements to the existing RA scheme, but they may incur some additional overhead, such as secure clocks.

Finally, we identified and discussed multiple challenges confronted by current RA schemes, which are worth to be considered in future investigations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank Prof. N. Asokan of the University of Waterloo. He presents valuable feedback and suggestions on the background, system model, future research, and paper writing for this work.

This work is supported by National Natural Science Foundation of China (62072239, 62002167, and 61901209), Natural Science Foundation of Jiangsu Province, China (BK20211192 and BK20200461),Open Foundation of the State Key Laboratory of Information Security of China (2021-MS-07), Fundamental Research Funds for the Central Universities (30921013111), China Scholarship Council (No. 202006840154), and Australia ARC (DP200101374).

REFERENCES

A survey on hardware approaches for remote attestation in network infrastructures. arXiv preprint arXiv:2005.12453 (2020).

Abadi M, Budiu M, Erlingsson Ú, Ligatti J. Control-flow integrity. In: Proceedings of the CCS; 2005. p. 340–53.

Abera T, Asokan N, Davi L, Ekberg J-E, Nyman T, Paverd A, Sadeghi A-R, Tsudik G. C-FLAT: control-flow attestation for embedded systems software. In: Proceedings of the CCS; 2016. p. 743–54.
Al-Garadi MA, Mohamed A, Al-Ali AK, Du X, Ali I, Guizani M. A survey of machine and deep learning methods for Internet of Things (IoT) security. IEEE Commun. Surv. Tutor. 2020;22(3):1646–85.
Aman MN, Sikdar B. ATT-auth: a hybrid protocol for industrial IoT attestation with authentication. IEEE Internet Things J. 2018;5(6):5119–31.
Ambrosin M, Conti M, Ibrahim A, Neven G, Schunter M. SANA: secure and scalable aggregate network attestation. In: Proceedings of the CCS; 2016. p. 731–42.
Ambrosin M, Conti M, Lazzeretti R, Rabbani MM, Ranise S. Toward secure and efficient attestation for highly dynamic swarms: poster. In: Proceedings of the WiSec; 2017. p. 281–2.
Ambrosin M, Conti M, Lazzeretti R, Rabbani MM, Ranise S. PADS: practical attestation for highly dynamic swarm topologies. In: Proceedings of the SIoT; 2018. p. 18–27.
Ambrosin M, Conti M, Lazzeretti R, Rabbani MM, Ranise S. Collective remote attestation at the internet of things scale: state-of-the-art and future challenges. IEEE Commun. Surv. Tutor. 2020;22(4):2447–61.
Ammar M, Crispo B, Nunes IDO, Tsudik G. Delegated attestation: Scalable remote attestation of commodity CPS by blending proofs of execution with software attestation. In: Proceedings of the Wisec; 2021. p. 37–47.
Ammar M, Washha M, Crispo B. WISE: lightweight intelligent swarm attestation scheme for IoT (the verifier's perspective). In: Proceedings of the WiMob; 2018a. p. 1–8.
Ammar M, Washha M, Ramabhadran GS, Crispo B. SlimIoT: scalable lightweight attestation protocol for the Internet of Things. In: Proceedings of the DSC; 2018b. p. 1–8.
Ankergård SFJJ, Dushku E, Dragoni N. State-of-the-art software-based remote attestation: opportunities and open issues for Internet of Things. Sensors 2021;21:1598.
Arbaugh W, Farber D, Smith J. A secure and reliable bootstrap architecture. In: Proceedings of the S&P; 1997. p. 65–71.
Asokan N, Brasser F, Ibrahim A, Sadeghi AR, Schunter M, Tsudik G, Wachsmann C. SEDA: scalable embedded device attestation. In: Proceedings of the CCS; 2015. p. 964–75.
Banks, A. S., Kisiel, M., Korsholm, P., 2021. Remote attestation: a literature review. arXiv preprint arXiv:2105.02466.
Berger S, Bürger O, Röglinger M. Attacks on the industrial Internet of Things - development of a multi-layer taxonomy. Comput. Secur. 2020;93:101790.
Brasser F, El Mahjoub B, Sadeghi A-R, Wachsmann C, Koeberl P. TyTAN: tiny trust anchor for tiny devices. In: Proceedings of the DAC; 2015. p. 1–6.
Brasser F, Rasmussen KB, Sadeghi AR, Tsudik G. Remote attestation for low-end embedded devices: the prover's perspective. In: Proceedings of the DAC; 2016. p. 1–6.
Carlet C, de Chérisey É, Guilley S, Kavut S, Tang D. Intrinsic resiliency of s-boxes against side-channel attacks-best and worst scenarios. IEEE Trans. Inf. Forensics Secur. 2021;16:203–18.
Carpent X, Eldefrawy K, Rattanavipanon N, Sadeghi A-R, Tsudik G. Invited: reconciling remote attestation and safety-critical operation on simple IoT devices. In: Proceedings of the DAC; 2018a. p. 1–6.
Carpent X, Eldefrawy K, Rattanavipanon N, Tsudik G. Lightweight swarm attestation: a tale of two LISA-s. In: Proceedings of the ASIACCS; 2017. p. 86–100.
Carpent X, Eldefrawy K, Rattanavipanon N, Tsudik G. Temporal consistency of integrity-ensuring computations and applications to embedded systems security. In: Proceedings of the ASIACCS; 2018b. p. 313–27.
Carpent X, Rattanavipanon N, Tsudik G. Remote attestation of IoT devices via SMARM: shuffled measurements against roving Malware. In: Proceedings of the HOST; 2018c. p. 9–16.
Carpent X, Tsudik G, Rattanavipanon N. ERASMUS: efficient remote attestation via self-measurement for unattended settings. In: Proceedings of the DATE; 2018d. p. 1191–4.
Checkoway S, Davi L, Dmitrienko A, Sadeghi A-R, Shacham H, Winandy M. Return-oriented programming without returns. In: Proceedings of the CCS; 2010. p. 559–72.
Chen S, Xu J, Sezer EC, Gauriar P, Iyer RK. Non-control-data attacks are realistic threats. In: Proceedings of the USENIX Security; 2005. p. 177–91.
Chew C-J, Chen Y-C, Lee J-S, Chen C-L, Tsai K-Y. Preserving indomitable DDoS vitality through resurrection social hybrid botnet. Comput. Secur. 2021;106:102284.
Choi Y-G, Kang J, Nyang D. Proactive code verification protocol in wireless sensor network. In: Proceedings of the ICCSA; 2007. p. 1085–96.
Conti M, Dushku E, Mancini LV. RADIS: remote attestation of distributed IoT services. In: Proceedings of the SDS; 2019. p. 25–32.
Costan V, Devadas S. Intel SGX explained. IACR Cryptol. ePrint Arch. 2016;2016(86):1–118.
Cowan C, Pu C, Maier D, Walpole J, Bakke P, Beattie S, Grier A, Wagle P, Zhang Q, Hinton H. StackGuard : automatic adaptive detection and prevention of buffer-overflow attacks. In: Proceedings of the USENIX Security; 1998. p. 63–78.
Data execution prevention. https://docs.microsoft.com/zh-cn/windows/desktop/Memory/data-execution-prevention (2007).
Dessouky G, Abera T, Ibrahim A, Sadeghi A-R. LiteHAX: lightweight hardware-assisted attestation of program execution. In: Proceedings of the ICCAD; 2018. p. 1–8.
Dessouky G, Zeitouni S, Nyman T, Paverd A, Davi L, Koeberl P, Asokan N, Sadeghi A-R. LO-FAT: low-overhead control flow attestation in hardware. In: Proceedings of the DAC; 2017. p. 1–7.
Dushku E, Rabbani MM, Conti M, Mancini LV, Ranise S. SARA: secure asynchronous remote attestation for IoT systems. IEEE Trans. Inf. Forensics Secur. 2020;15:3123–36.
Eldefrawy K, Rattanavipanon N, Tsudik G. HYDRA: hybrid design for remote attestation (using a formally verified microkernel). In: Proceedings of the tenWiSEC; 2017. p. 99–110.
Eldefrawy K, Tsudik G, Francillon A, Perito D. SMART: secure and minimal architecture for (establishing a dynamic) root of trust. In: Proceedings of the NDSS; 2012. p. 1–15.
Feng W, Qin Y, Zhao S, Feng D. AAoT: lightweight attestation and authentication of low-resource things in IoT and CPS. Comput. Netw. 2018;134:167–82.
Francillon A, Castelluccia C. Code injection attacks on harvard-architecture devices. In: Proceeding of the CCS; 2008. p. 15–26.
Francillon A, Nguyen Q, Rasmussen KB, Tsudik G. A minimalist approach to remote attestation. In: Proceedings of the DATE; 2014. p. 1–6.
Fu A, Qin N, Wang Y, Li Q, Zhang G. Nframe: a privacy-preserving with non-frameability handover authentication protocol based on (t, n) secret sharing for LTE/LTE-A networks. Wirel. Netw. 2017;23(7):2165–76.
Fu A, Song J, Li S, Zhang G, Zhang Y. A privacy-preserving group authentication protocol for machine-type communication in LTE/LTE-A networks. Secur. Commun. Netw. 2016;9(13):2002–14.
Fu A, Zhang Y, Zhu Z, Jing Q, Feng J. An efficient handover authentication scheme with privacy preservation for IEEE 802.16m network. Comput. Secur. 2012;31(6):741–9.
Gao Y, F Al-Sarawi S, Abbott D. Physical unclonable functions. Nat. Electron. 2020;3(2):81–91.

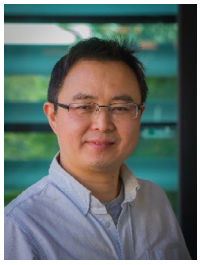
Gong B, Zhang Y, Wang Y. A remote attestation mechanism for the sensing layer nodes of the internet of things. Future Gener. Comput. Syst. 2018;78:867–86.
Gu, Guofei, 2020. Computer security conference ranking and statistic. https://people.engr.tamu.edu/guofei/sec_conf_stat.htm.
Heartfield R, Loukas G, Bezemskij A, Panaousis E. Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. IEEE Trans. Inf. Forensics Secur. 2021;16:1720–35.
Ibrahim A, Sadeghi A-R, Tsudik G. US-AID: unattended scalable attestation of IoT device. In: Proceedings of the SRDS; 2018. p. 21–30.
Ibrahim A, Sadeghi A-R, Tsudik G. HEALED: healing & attestation for low-end embedded devices. In: Proceedings of the FC; 2019. p. 627–45.
Ibrahim A, Sadeghi A-R, Tsudik G, Zeitouni S. DARPA: device attestation resilient to physical attacks. In: Proceedings of the WiSec; 2016. p. 171–82.
Ibrahim A, Sadeghi A-R, Zeitouni S. SeED: secure non-interactive attestation for embedded devices. In: Proceedings of the WiSec; 2017. p. 64–74.
jp. Advanced doug Lea's Malloc exploits. Phrack 2003;11(61).
Kil C, Sezer EC, Azab AM, Ning P, Zhang X. Remote attestation to dynamic system properties: towards providing complete system integrity evidence. In: Proceedings of the DSN; 2009. p. 115–24.
Koeberl P, Schulz S, Sadeghi A-R, Varadharajan V. TrustLite: a security architecture for tiny embedded devices. Proceedings of the EuroSys, 2014.
Kohnhäuser F, Büscher N, Gabmeyer S, Katzenbeisser S. SCAPI: a scalable attestation protocol to detect software and physical attacks. In: Proceedings of the WiSec; 2017. p. 75–86.
Kohnhäuser F, Büscher N, Katzenbeisser S. SALAD: secure and lightweight attestation of highly dynamic and disruptive networks. In: Proceedings of the ASIACCS; 2018. p. 329–42.
Kolias C, Kambourakis G, Stavrou A, Voas JM. DDoS in the IoT: Mirai and other botnets. Computer 2017;50(7):80–4.
Kong J, Koushanfar F, Pendyala PK, Sadeghi A-R, Wachsmann C. PUFatt: embedded platform attestation based on novel processor-based PUFs. In: Proceedings of the DAC; 2014. p. 1–6.
Kuang B, Fu A, Yu S, Yang G, Su M, Zhang Y. ESDRA: an efficient and secure distributed remote attestation scheme for IoT swarms. IEEE Internet Things J. 2019;6(5):8372–83.
Kuang B, Fu A, Zhou L, Susilo W, Zhang Y. DO-RA: data-oriented runtime attestation for IoT devices. Comput. Secur. 2020;97:101945.
Kucab M, Boryo P, Choda P. Remote attestation and integrity measurements with intel SGX for virtual machines. Comput. Secur. 2021;106:102300.
Kumar S, Eugster P, Santini S. Software-based remote network attestation. IEEE Trans. Dependable Secure Comput. 2021:1–12. doi:10.1109/TDSC.2021.3077993.
Kuznetsov V, Szekeres L, Payer M, Candea G, Sekar R, Song D. Code-pointer integrity. In: Proceedings of the OSDI; 2014. p. 147–63.
Li Y, Cheng Y, Meng W, Li Y, Deng RH. Designing leakage-resilient password entry on head-mounted smart wearable glass devices. IEEE Trans. Inf. Forensics Secur. 2021;16:307–21.
Makhdoom I, Abolhasan M, Lipman J, Liu RP, Ni W. Anatomy of threats to the Internet of Things. IEEE Commun. Surv. Tutor. 2018;21(2):16361675.
Mansouri M, Jaballah WB, Önen M, Rabbani MM, Conti M. FADIA: fairness-driven collaborative remote attestation. In: Proceedings of the Wisec; 2021. p. 60–71.
Murray V. Legal GNSS spoofing and its effects on autonomous vehicles. Proceedings of the BlackHat, 2019.
Nunes IDO, Dessouky G, Ibrahim A, Rattanavipanon N, Sadeghi A-R, Tsudik G. Towards systematic design of collective remote attestation protocols. In: Proceedings of the ICDCS; 2019a. p. 1188–98.
Nunes IDO, Eldefrawy K, Rattanavipanon N, Steiner M, Tsudik G. VRASED: a verified hardware/software co-design for remote attestation. In: Proceedings of the USENIX Security; 2019b. p. 1429–46.
Nunes IDO, Eldefrawy K, Rattanavipanon N, Tsudik G. PURE: Using verified remote attestation to obtain proofs of update, reset and erasure in low-end embedded systems. In: Proceedings of the ICCAD; 2019c. p. 1–8.
Nunes IDO, Eldefrawy K, Rattanavipanon N, Tsudik G. APEX: a verified architecture for proofs of execution on remote devices under full software compromise. In: Proceedings of the USENIX Security; 2020. p. 771–88.
Nunes IDO, Jakkamsetti S, Rattanavipanon N, Tsudik G. On the TOCTOU problem in remote attestation. In: Proceedings of the CCS; 2021b. p. 1–16.
Nunes DOI, Jakkamsetti S, Tsudik G. Tiny-CFA: a minimalistic approach for control flow attestation using verified proofs of execution. Proceedings of the DATE, 2021a.
Nunes IDO, Jakkamsetti S, Tsudik G. DIALED: Data integrity attestation for low-end embedded devices. In: Proceedings of the DAC; 2021c. p. 1–6.
PaX-Team, 2001. PaX ASLR (address space layout randomization). https://pax.grsecurity.net/docs/aslr.txt.
Perito D, Tsudik G. Secure code update for embedded devices via proofs of secure erasure. In: Proceedings of the ESORICS; 2010. p. 643–62.
Praseed A, Thilagam PS. Modelling behavioural dynamics for asymmetric application layer DDoS detection. IEEE Trans. Inf. Forensics Secur. 2021;16:617–26.
Qu Y, Yu S, Zhou W, Peng S, Wang G, Xiao K. Privacy of things: emerging challenges and opportunities in wireless Internet of Things. IEEE Wirel. Commun. 2018;25(6):91–7.
Rabbani MM, Vliegen J, Winderickx J, Conti M, Mentens N. SHeLa: scalable heterogeneous layered attestation. IEEE Internet Things J. 2019;6(6):10240–50.
Schulz S, SchallerFlorian A, Katzenbeisser K. Boot attestation: secure remote reporting with off-the-shelf IoT sensors. In: Proceedings of the ESORICS; 2017. p. 437–55.
Seshadri A, Luk M, Shi E, Perrig A, Van Doorn L, Khosla P. Pioneer: verifying integrity and guaranteeing execution of code on legacy platforms. In: Proceedings of the SOSP; 2005. p. 10–1145.
Seshadri A, Perrig A, Doorn LV, Khosla P. SWATT: software-based attestation for embedded devices. In: Proceedings of the S&P; 2004. p. 272–82.
Shekari T, Beyah R. IoT skimmer: energy market manipulation through high-wattage IoT botnets. Proceedings of the BlackHat, 2020.
Skorobogatov S. Physical Attacks and Tamper Resistance. New York, NY: Springer; 2012. p. 143–73.
Snow KZ, Monrose F, Davi L, Dmitrienko A, Liebchen C, Sadeghi A-R. Just-in-time code reuse: on the effectiveness of fine-grained address space layout randomization. In: Proceedings of the S&P; 2013. p. 574–88.
Steiner RV, Lupu E. Attestation in wireless sensor networks: a survey. ACM Comput. Surv. 2009;49(3):51–82.
Steiner RV, Lupu E. Towards more practical software-based attestation. Comput. Netw. 2019;149:43–55.
Stellios I, Kotzanikolaou P, Grigoriadis C. Assessing IoT enabled cyber-physical attack paths against critical systems. Comput. Secur. 2021:102316.
Su M, Zhou B, Fu A, Yu Y, Zhang G. PRTA: a proxy re-encryption based trusted authorization scheme for nodes on CloudIoT. Inf. Sci. 2020;527:533–47.

Sun Z, Feng B, Lu L, Jha S. OAT: attesting operation integrity of embedded devices. In: Proceedings of the S&P; 2020. p. 1433–49.
Tan H, Tsudik G, Jha S. MTRA: multi-tier randomized remote attestation in IoT networks. Comput. Secur. 2019;81:78–93.
Wang J, Hong Z, Zhang Y, Jin Y. Enabling security-enhanced attestation with intel SGX for remote terminal and IoT. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2018;37(1):88–96.
Xu W, Zhang X, Hu H, Ahn G, Seifert J. Remote attestation with domain-based integrity model and policy analysis 2012;9(3):429–42.
Yan W, Fu A, Mu Y, Zhe X, Yu S, Kuang B. EAPA: efficient attestation resilient to physical attacks for IoT devices. In: Proceedings of the IoT S&P; 2019. p. 2–7.
Yu Y, Li Y, Tian J, Liu J. Blockchain-based solutions to security and privacy issues in the Internet of Things. IEEE Wirel. Commun. 2018;25(6):12–18.
Zeitouni S, Dessouky G, Arias O, Sullivan D, Sadeghi AR. ATRIUM: runtime attestation resilient under memory attacks. In: Proceedings of the ICCAD; 2017. p. 384–91.
Zhou, Jianying, 2020. Top cyber security conferences ranking (2020). http://jianying.space/conference-ranking.html.

Boyu Kuang is currently a Ph.D. student in School of Computer Science and Engineering, Nanjing University of Science and Technology, China. He received his Bachelor degree in Computer Science and Technology from Nanjing University of Science and Technology, China, in 2016. His research interest includes IoT Security and Privacy Preserving.

Anmin Fu received the Ph.D. degree in Information Security from Xidian University in 2011. From 2017 to 2018, he was a Visiting Research Fellow with the University of Wollongong, Australia. He is currently a professor of Nanjing University of Science and Technology, China. His research interest includes IoT Security, Cloud Computing Security and Privacy Preserving. He has published more than 80 technical papers, including international journals and conferences, such as IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Services Computing, IEEE Internet of Things Journal, Computers & Security, IEEE ICC, IEEE GLOBECOM and ACISP.

Willy Susilo is a Distinguished Professor in the School of Computing and Information Technology, Faculty of Engineering and Information Sciences at the University of Wollongong (UOW), Australia. He is an IEEE Fellow, the director of Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, UOW and the Head of School of Computing and Information Technology at UOW (2015–now). Prior to this role, he was awarded the prestigious Australian Research Council Future Fellowship in 2009. In 2016, he was awarded the æResearcher of the Yearg at UOW, due to his research excellence and contributions. He is the Editor-in-Chief of the Elseviers Computer Standards and Interfaces and the MDPIs Information journal. He is currently an Associate Editor of IEEE Transactions on Dependable and Secure Computing, ACM Computing Surveys and Elseviers Computers and Security. He has also served as the program committee member of several international conferences.



Shui Yu is a Professor of School of Computer Science, University of Technology Sydney, Australia. Dr Yu's research interest includes Big Data, Security and Privacy, Networking, and Mathematical Modelling. He has published three monographs and edited two books, more than 400 technical papers, including top journals and top conferences, such as IEEE TPDS, TC, TIFS, TMC, TKDE, TETC, ToN, and INFOCOM. His h-index is 58. Dr Yu initiated the research field of networking for big data in 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. He is currently serving a number of prestigious editorial boards, including IEEE Communications Surveys and Tutorials (Area Editor), IEEE Communications Magazine, IEEE Internet of Things Journal, and so on. He is a Senior Member of IEEE, a member of AAAS and ACM, a Distinguished Lecturer of IEEE Communications Society, and an elected member of Board of Governor of IEEE Vehicular Technology Society.

Yansong Gao received his M.Sc degree from University of Electronic Science and Technology of China in 2013 and Ph.D degree from the University of Adelaide, Australia, in 2017. He is with Nanjing University of Science and Technology, China. His current research interests are hardware security, AI security and privacy, and system security.