Remote Attestation as a Service for IoT

Mauro Conti
Department of Mathematics
University of Padova, Italy
conti@math.unipd.it

Edlira Dushku
Dipartimento di Informatica
Sapienza University of Rome, Italy
dushku@di.uniroma1.it

Luigi V. Mancini
Dipartimento di Informatica
Sapienza University of Rome, Italy
mancini@di.uniroma1.it

Md Masoom Rabbani
Department of Mathematics
University of Padova, Italy
rabbani@math.unipd.it

Silvio Ranise
S&T Unit
Fondazione Bruno Kessler, Italy
ranise@fbk.eu

***Abstract*— Remote attestation is a two-party security protocol that aims to detect the presence of malware in a remote untrusted IoT device. In order to perform the attestation, an IoT device typically has to stop the regular operation and perform expensive computations that will consume the battery life of the device. In this paper, we use cloud/fog computing to attest an IoT device in an efficient way. We propose Remote Attestation as a Service (RAaS) which allows even a low-end IoT device to securely offload the attestation process to the cloud. We argue that RAaS allows the clone of the device, securely created in the cloud, to perform the most expensive attestation computations. Our proposed approach could reduce the number of attestation operations running on the real IoT device, saving energy consumption, and reducing the downtime of the usual operation of an IoT device during the execution of remote attestation.**

***Index Terms*—Remote Attestation, Internet of Things, Security and Privacy.**

I. Introduction

The deployment models of the Internet of Things (IoT) systems consist of a large number of low-end devices supported by cloud platforms. The interactions between an IoT device and cloud are usually facilitated by some devices in the network that have more computational power, e.g., a base station, a smartphone, a fog node. The distribution of the computational power closer to the low-end devices seems a promising deployment model to enable the rapid deployment of large-scale IoT applications that require real-time decision making and low latency [16]. At the same time, the rapid deployment and tremendous growth of the IoT devices have increasingly exposed IoT systems to numerous types of attacks. Incidents like Distributed Denial of Services (DDoS) [28], Stuxnet [3], Jeep-hack [4] have shown that cyber attacks can cause serious consequences in IoT systems.

While IoT devices are usually deployed in distant environments, remote attestation has emerged as a convenient malware detection technique that aims to check remotely the integrity of the software running on these remote untrusted IoT devices. In a typical remote attestation protocol, one trusted party called Verifier (`Vrf`) initiates the remote attestation by sending a challenge to a potentially untrusted device called Prover (`Prv`). Upon the attestation request, the `Prv` stops the usual operation to perform the attestation immediately. The complexity of the state-of-the-art remote attestation protocols consumes energy and may cause a long suspension of the usual work of the device while performing attestation. Thus, from the Prover's perspective, remote attestation is an overhead operation that consumes computational power and battery life. These drawbacks can cause intolerable disruptions especially in time-critical infrastructures such as medical facilities, nuclear plants, etc.

Motivation. Cloud computing has played a key role in broadening IoT applications through data offload, offline data analysis, service management, framework integration, just to name a few. Data collected from IoT devices are currently stored and processed in cloud systems, either directly or through an intermediary device. However, the application of remote attestation in cloud is currently overlooked. To optimize the attestation protocol for IoT devices, instead of improving the attestation protocols deployed on an IoT device, we focus on utilizing the resources offered by the cloud systems. We propose a cloud service (RAaS) that is able to securely perform the remote attestation on behalf of an IoT device. RAaS will be able to get the content of the memory from a low-end IoT device and then run independently the attestation protocol on the cloud.

The basic idea is to enhance the existing ability of low-end devices to upload data on cloud by enabling the device to upload also the content of its memory blocks. This is not a trivial task. While sending the content in cloud, an adversary may destroy or relocate itself on the other memory blocks to avoid uploading on the cloud service, and consequently remain undetected by the attestation protocol.

In this paper, we outline a possible approach for designing RAaS. By offloading the attestation to cloud, the remote attestation procedure will reduce the computations of the low-end devices, will not impede a device for a long time to perform usual work, and will consequently save their battery lifetime.

Contribution. The contribution of this paper is three-fold.

- We introduce a novel idea of the Remote Attestation as Service (RAaS) for low-end IoT devices.

- We describe a possible approach for designing RAaS. RAaS could be a promising solution for IoT networks which work in intermittent connectivity by performing attestation on the cloud and help low-end IoT devices to save precious energy.
- We present the adversary model of RAaS and provide security analysis w.r.t. adversarial assumptions.

Outline. The remainder of this paper is organized as follows. In Section II we present the related works and provide a brief background in Section III. We describe the system model in Section IV and the adversary model in Section V. We provide the protocol description in Section VI and security analysis in Section VII. In Section VIII, we identify the limitations of the approach, and our paper concludes in Section IX with conclusions and future works directions.

II. Related Works

Since past decade, researchers have proposed different working methodologies for remote attestation schemes in terms of their code verification, adversary assumptions, hardware design etc. Remote attestation is broadly classified into three main categories, (1) software-based attestation schemes [32], [33], which are based on “hardware-less” approach; (2) hardware-based attestation schemes [20], which use tamper-resistant special hardware module as the root of trust for attestation, and (3) hybrid attestation schemes [23], [27], [29], [34] which depend on software-hardware co-design and employ “minimal-hardware” for dynamic root of trust.

Traditional software-only attestation techniques proposed in the literature like [31]–[33] rely mainly on time-bounded attestation computation. These schemes assume that a legitimate `Prv` is able to compute the attestation challenge within a restricted period of time, while the presence of malware on the `Prv` will delay the attestation response. That is because when an adversary will force the `Prv` to re-direct the computation to another proxy node, the `Prv` requires more time than usual to perform the attestation. More recently, researchers proposed scalable remote attestation techniques for large IoT networks. These techniques are also known as collective remote attestation techniques. In general, these techniques employ an overlay of spanning tree rooted at the `Vrf` [19], [21], [30] to efficiently propagate attestation requests along the tree. The works in [25], [26] consider the communication data exchanged among a group of IoT devices in the attestation and propose protocols for remote attestation of distributed IoT services. All group attestation schemes force the low-end devices to suspend their usual work until the attestation for the whole network finishes.

Despite the advancement in remote attestation schemes, performing attestation rather than usual work might prove fatal for the scenarios where sensors are also deployed in critical infrastructures (e.g., medical facilities, nuclear plants), that require continuous monitoring. To address this issue, the work in [24] introduces the idea of interruptable remote attestation. To enable an interruptable attestation, this work implements different memory-lock approaches, through which the `Prv` locks the memory to prevent other tasks from accessing the memory blocks during the attestation. Once the attestation for a specific memory block is accomplished, that specific memory block will be unlocked. In this way, the `Prv` resumes quickly the usual work.

Different from other works in the literature, in this paper, we aim to reduce the overhead of remote attestation by offloading the attestation computation to the cloud.

III. Background.

In this section, we present a brief background on the deployment models of IoT in cloud and fog computing.

A. Cloud architecture for IoT

Cloud computing technology offers the potential to use on-demand and scalable resources both in terms of storage and processing services. Cloud services are particularly beneficial for IoT systems due to the requirements for scalable management of a large number of interconnected IoT devices and the requirements for real-time analysis of the vast quantities of data associated with IoT systems.

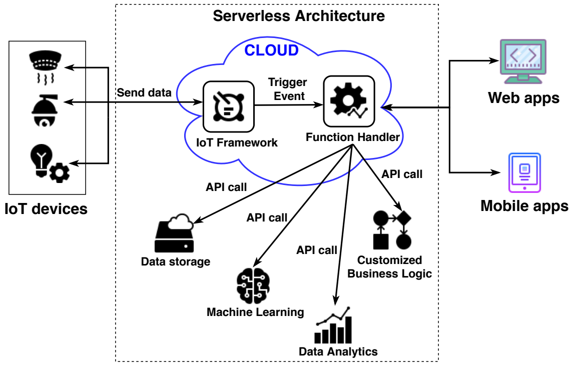

Fig. 1. Serverless architecture in Cloud

Currently, various cloud platforms (such as Cloud IoT Core [13], AWS IoT [6], IBM Watson IoT [15]) facilitate the connectivity of IoT devices to the cloud and other devices. In addition, the major cloud providers have recently launched their serverless computing platforms such as Google Cloud Functions [12], Microsoft Azure Functions [9], Amazon Lambda [8], IBM Cloud Functions [14]. Serverless computing in a new software development paradigm which decomposes monolithic cloud-native applications into a set of functions. Overall, the goal of Serverless computing is to handle the execution management of functions, completely separating it from data management. Each function can be launched through an API call and instantiated in an isolated virtual environment, e.g., Docker container [10]. In this way, the presence of serverless computing platforms allows the cloud providers to support Function-as-a-Service (FaaS). The integration of serverless computing platforms with cloud IoT platforms, simplifies the management of IoT devices, enabling the IoT devices to trigger event-driven execution of functions (as shown in Figure 1).

B. Distributing Cloud in Fog

The requirements of fast response time and low bandwidth usage bring a fundamental necessity to move the data processing from cloud to edge devices. This shift of the computations closer to the data producers is the foundation of the so-called Fog Computing paradigm [16]. In fog computing, facilities or infrastructures that can provide resources for services at the edge of the network are called fog nodes [35]. Many major cloud providers have adopted the fog computing paradigm in various frameworks such as Azure IoT Edge [9], AWS Greengrass [7], EdgeX [11] etc. These edge-oriented frameworks implement the function-based programming model, triggering edge functions based on a topic-based publish/subscribe system (as shown in Figure 2). The IoT edge computing frameworks can be programmed to filter device data and send back to the cloud only necessary information.

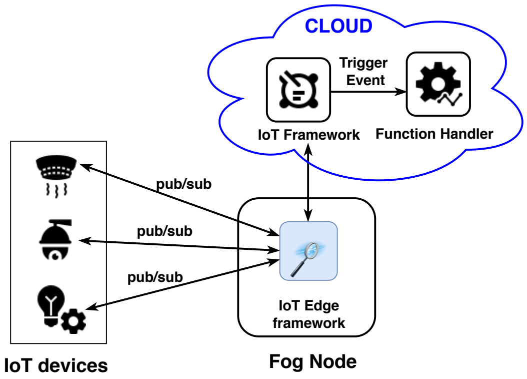

Fig. 2. Overview of Fog Computing paradigm

IV. System Model.

The aim of RAaS is to check the integrity of an untrusted device by performing the attestation computation on the cloud. In our system, we consider three main entities as shown in Figure 3.

- Prover (`Prv`): it is a low-end IoT device (i.e., sensors and actuators). This is a potentially untrusted device that will perform attestation.
- Remote attestation as a service (RAaS): it is a cloud-service, clone of the `Prv`, that will perform the attestation on behalf of the `Prv` and sends attestation result to the `Vrf` upon completion of the attestation.
- Verifier (`Vrf`): it is an external trusted entity that knows in advance the legitimate state of the `Prv`. The purpose of the `Vrf` is to initiate the attestation process and verify the trustworthiness of the `Prv`.

At the attestation time, `Vrf` will send a challenge to RAaS to initiate the attestation (① in Figure 3). When RAaS gets the challenge, RAaS will requests the memory data of the `Prv` (②). Next, the `Prv` will read the memory (③) and will send the content of the memory to RAaS (④). Once the memory is delivered to RAaS, `Prv` will continue the usual work, while RAaS will perform the necessary computation to run the attestation (⑤). At the end, RAaS will send the attestation response to the `Vrf` (⑥), which can afterwards check whether `Prv` is compromised or trustworthy.

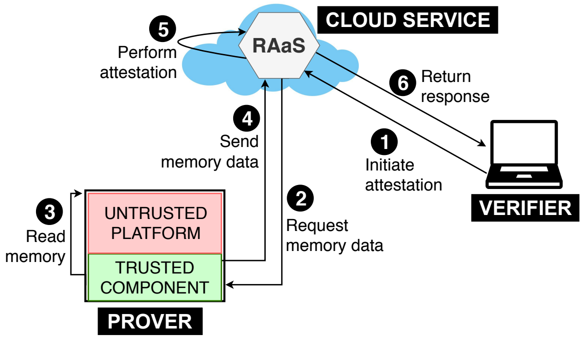

Fig. 3. System model

Although the cloud model can introduce communication delays between a low-end device and the cloud, our model can also be used on a low-end intermediate device. For example, the RAaS can be offered by a fog node in a fog computing paradigm that can provide services at a low latency. In addition, this model can also be extended in edge computing, where the edge device performs some of the calculations and deliver the rest of the attestation protocol in cloud. Certainly, fog computing and edge computing approaches require additional assumptions concerning the trusted execution environments that should be present on each intermediary. To preserve the generality of our approach and to avoid architectural assumptions that do not necessarily affect our vision, we regard the RAaS as a cloud service, and we assume the cloud platform is trusted.

In addition, we assume that `Vrf` is an external entity (e.g., person or device) that periodically monitors the reliability of IoT devices. Alternatively, the cloud service itself can also function as a `Vrf`. This approach could be particularly useful in the Fog computing model, where fog nodes function as `Vrf` of the IoT devices with which they communicate. Since the operations of the `Vrf` do not influence the internal operations of RAaS, to preserve the generality of the system model, we consider the `Vrf` as an external trusted party.

V. Adversary Model.

The goal of the RAaS is to detect a software-only attacker that infects either remotely or being present physically near to the device. The adversary will try to be passive during attestation in order to evade detection. In particular we assume:

- **Software-adversary** ($\texttt{Adv}_{\texttt{sw}}$). The $\texttt{Adv}_{\texttt{sw}}$ exploits a software vulnerability to compromise the device by injecting and executing an unauthorized malicious code. In addition, a $\texttt{Adv}_{\texttt{sw}}$ can also mount replay attack or man-in-the middle attack by sending fake or “old” challenges.
- **Mobile-adversary** ($\texttt{Adv}_{\texttt{mob}}$). The attacker runs malware on the device and during attestation will try to relocate itself in order to evade detection.

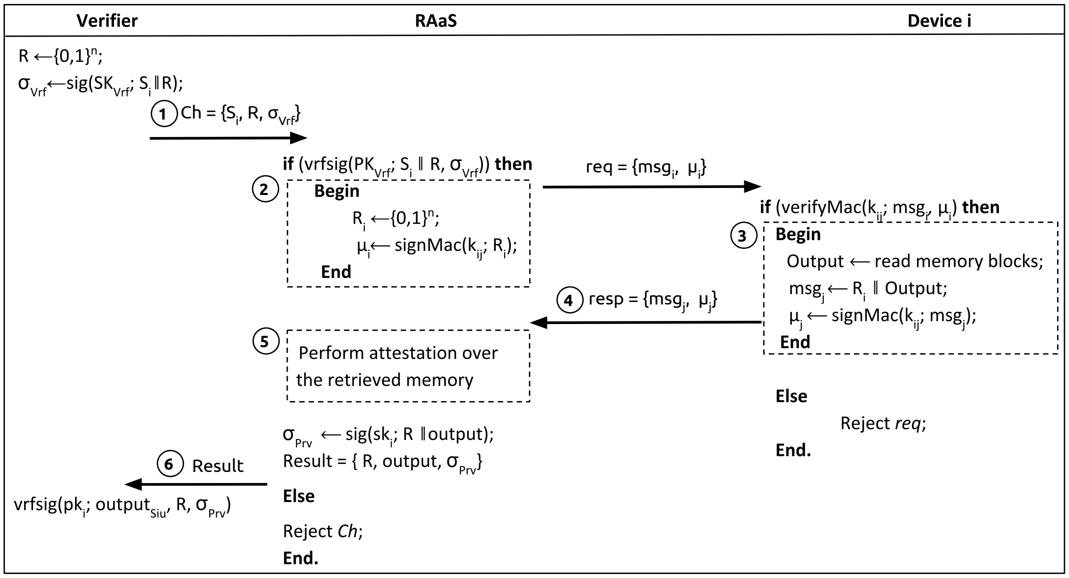
Verifier
RAaS
Device i
$R \leftarrow \{0,1\}^n$;
$\sigma_{Vrf} \leftarrow sig(SK_{Vrf}; S_i \| R)$;
1 $Ch = \{S_i, R, \sigma_{Vrf}\}$
if $(vrfsig(PK_{Vrf}; S_i \| R, \sigma_{Vrf}))$ **then**
2 **Begin**
$R_i \leftarrow \{0,1\}^n$;
$\mu_i \leftarrow signMac(k_{ij}; R_i)$;
End
$req = \{msg_i, \mu_i\}$
if $(verifyMac(k_{ij}; msg_i, \mu_i)$ **then**
3 **Begin**
Output ⟵ read memory blocks;
$msg_j \leftarrow R_i \|$ Output;
$\mu_j \leftarrow signMac(k_{ij}; msg_j)$;
End
4 $resp = \{msg_j, \mu_j\}$
5 Perform attestation over the retrieved memory
Else
Reject *req*;
End.
$\sigma_{Prv} \leftarrow sig(sk_i; R \|output)$;
Result = { R, output, σ_{Prv}}
6 Result
$vrfsig(pk_i; output_{Siu}, R, \sigma_{Prv})$
Else
Reject *Ch*;
End.

Fig. 4. Algorithm overview

In line with other remote attestation techniques in literature, we exclude hardware attacks (i.e., physical adversary) and network-wide Distributed Denial of Service (DDoS) attacks from the current scope of this paper. For simplicity, we assume that the cloud-based service is secure and can not be compromised. However, we describe possible approaches that address software attacks in cloud platforms.

VI. Protocol Description.

The main objective of our protocol is to make remote attestation lightweight for low-end IoT devices. Therefore, we introduce RAaS as a mechanism that is able to perform attestation on behalf of the Prv and consequently reduces the overhead that remote attestation introduces in the device.

A. General Overview.

In Figure 4, we present a general overview of the algorithm performed by RAaS. The Vrf initiates the attestation by sending a challenge Ch to the RAaS (Step ① in Figure 4). Upon receiving the challenge, RAaS will verify the request *vrfsig()* and then will request a copy of the memory blocks from the Prv (Step ②). The Prv will read the memory blocks (Step ③) and send it to RAaS (Step ④). We describe the secure memory offload mechanism in Section VI-D. Once the RAaS gets the memory blocks, it will perform attestation on the memory content (Step ⑤) and send the result along with the challenge to the Vrf (Step ⑥). The challenge serves two purposes: first, it initiates the attestation process; second, it proves the freshness of the attestation computation.

The RAaS will perform authentication[1] of the Prv before copying the memory content. The authentication can be achieved through secure key-mechanism which can be chosen based on network and application requirements.

B. RAaS Working mechanism.

In the following, we describe the main steps of the operations of RAaS (i.e., cloud service) and the Verifier (Vrf).

1) RAaS operations: The RAaS performs the following main operations:

- *Receive Attestation request*: The cloud service (RAaS) receives challenge from the Vrf to initiate the attestation process.
- *Verify request*: RAaS verifies the attestation initiation challenge to prevent replay and man-in-the-middle attacks.
- *Memory block request*: RAaS sends a request to a Prv to retrieve the content of the memory blocks.
- *Perform attestation*: Upon receiving the entire memory blocks, RAaS performs attestation.
- *Report to* Vrf: Upon completion of attestation, RAaS sends the attestation report to the Vrf.

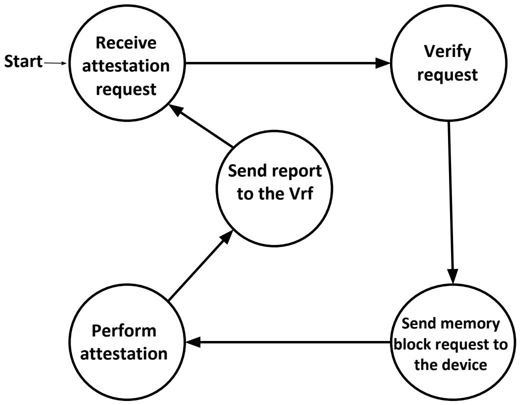
Start
Receive attestation request
Verify request
Send report to the Vrf
Perform attestation
Send memory block request to the device

Fig. 5. FSM-s of RAaS (cloud service)

[1] Authentication can prevent proxy attacks.

Figure 5 presents the aforementioned operations of RAaS in the finite state machine (FSM) model.

2) Verifier operations: The Vrf has four main functions as follows:

- *Attestation initiation*: Vrf initiates the attestation process.
- *Send challenge to RAaS*: To counter replay attacks, Vrf sends challenge to the RAaS.
- *Attestation report gathering*: Upon completion of attestation operation over Prv 's memory blocks, Vrf receives the attestation result from RAaS.
- *Verify*: Vrf compares the received attestation report with the expected legitimate one.

The operations of the Vrf are shown in Figure 6 in the finite state machine (FSM) model.

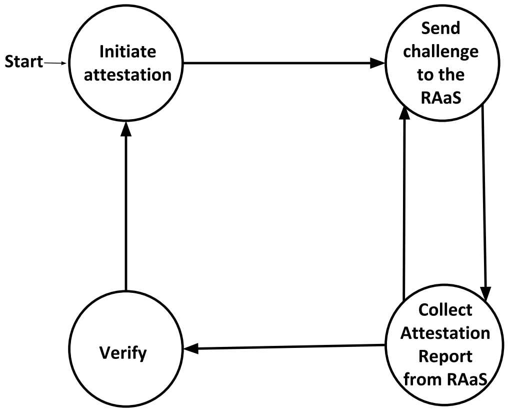
Start
Initiate attestation
Send challenge to the RAaS
Verify
Collect Attestation Report from RAaS

Fig. 6. FSM-s of Verifier

C. Data transmission vs Computational overhead.

In evaluating the efficiency of the RAaS approach, one crucial consideration has to do with the overhead of data transmission that the Prv has to send to the cloud service. RAaS is an efficient approach in the scenarios when the overhead of data transmission to the cloud is lower than the overhead of performing attestation locally in the device. We argue that IoT devices are becoming multi-functional; for instance, ST [18], Arduino [5], SensorTile [17] have simplified the development of multi-sensor solutions. Considering that in some scenarios, not all the IoT services are active all the time, a context-aware remote attestation procedure may need to attest only the relevant services instead of attesting the entire IoT device. In this way, the amount of data transmitted to the cloud service will be reduced to only a portion of the Prv's memory that has recently been changed. Moreover, we expect that remote attestation performed by the cloud service increases the effectiveness of the attestation. That is because, unlike resource-constraint IoT systems that perform lightweight remote attestation, the cloud service will have unlimited resources to conduct profound memory content analysis and identify prospective new threats.

D. Secure memory Offload

To securely copy Prv's data-memory blocks, we rely on the usage of “memory-locking” mechanism by operating system. Based on this mechanism, the entire memory of the Prv will be locked at the beginning of the attestation process and individual memory blocks will be released once the offload for that specific block is completed. Main advantages of this approach are two-fold; firstly, Prv's entire memory-content is not locked for the whole attestation period and secondly, individual memory blocks upon their release from RAaS can perform their usual work. Specifically, locked-memory blocks prevent other tasks from accessing these blocks during the attestation phase. Thus, this mechanism prevents $\mathtt{Adv}_{\mathtt{mob}}$ to evade detection by relocating itself in different positions during the memory copy process from the Prv to the cloud service. Additionally, since this technique does not lock the entire Prv's memory for the whole duration of the attestation process, the usual work of the Prv will not be hindered for a long time.

E. Execution of cloud service.

The cloud platform may pose serious security and privacy concern [1], [2] for end users. Although this paper do not provide details how to cope with these concerns, a possible solution can be “shielding applications” as discussed in [22]. In this approach, an application can securely use hardware protection[2] to execute applications in an untrusted cloud environment.

VII. Security Analysis

In Section V, we introduced the adversary model for our proposal RAaS. According to adversarial capabilities, an Adv can launch software-only attacks. This section provides a security analysis of RAaS w.r.t. adversary model.

- **Software-adversary** ($\mathtt{Adv}_{\mathtt{sw}}$). Firstly, a software adversary can manipulate IoT devices software by executing malicious code. However, in RAaS we use secure memory lock functionalities where the Prv's entire memory will be locked at the initial phase of attestation. A software adversary can not evade detection, due to the secure memory copy mechanism and RAaS will identify the malicious code in a Prv. Further, the use of *challenge* preserves the freshness and prevents an $\mathtt{Adv}_{\mathtt{sw}}$ from launching a replay attacks by using “fake” or “old” *challenge*.
- **Mobile-adversary** ($\mathtt{Adv}_{\mathtt{mob}}$). Secondly, a mobile adversary can relocate itself to prevent detection. In RAaS, however, a $\mathtt{Adv}_{\mathtt{mob}}$ can not avoid detection, as in RAaS, secure lock mechanism will lock the entire Prv's memory at initialization of attestation process and release individual memory blocks upon successful completion of memory copying. Thus, a $\mathtt{Adv}_{\mathtt{mob}}$ can not relocate itself during attestation to thwart detection.

[2] In [22] authors employ Intel SGX as the hardware protection against privileged code and physical attacks.

VIII. Limitations.

In this paper, our main aim is to obtain clear security guarantees and maximize the efficiency of IoT devices by employing cloud-based services for performing remote attestation on behalf of resource-constrained IoT devices. Mainly, we aim to propose an efficient remote attestation scheme of IoT devices that is able to reduce the suspension time of their regular work during the attestation. However, despite its many advantages, RAaS has also limitations.

In particular, in line with other state-of-the-art remote attestation techniques [19], [21] we do not consider physical adversaries. A physical adversary can tamper with the attestation process. A more comprehensive approach is missing in RAaS to counter this threat.

Additionally, the use of the *challenge* helps to counter replay attacks, but the same may not contribute to counter network-wide DDoS attacks. In real network applications, DDoS attacks are a real threat. Thus we need a counter mechanism to thwart any DDoS attack on RAaS.

Further, we have to improve three main areas for RAaS; (1) A secure communication process should be employed between the cloud service and the underlying IoT devices, (2) the implementation of RAaS over a large heterogeneous IoT network (i.e., swarm), to validate its performance, and (3) energy-consumption, average packet delivery ratio (APDR) data needs to be checked while performing attestation in RAaS.

IX. Conclusion and Future directions.

Providing security for low-end devices is a difficult task. Remote attestation protocols intend to improve the security of these devices by detecting the adversarial presence. However, remote attestation is significantly expensive in terms of computational power and battery consumption. To address these challenges, we propose a cloud-hosted remote attestation protocol that aims to offload the computation of the attestation protocol.

This work is a ongoing effort, and we are evaluating the pros and cons of the approach, particularly with regard to performance issues, i.e., computational time, communication overhead, and scalability. We believe that this research direction opens up new perspectives in developing lightweight remote attestation protocols for low-end devices that rely on the cloud.

Acknowledgement

Md Masoom Rabbani is supported by Fondazione Bruno Kessler fund for his PhD. This work is also supported in part by EU LOCARD Project under Grant H2020-SU-SEC-2018-832735. The work of Edlira Dushku and Luigi V. Mancini is supported by the MIUR under grant Dipartimenti di eccellenza 2018-2022 of the Dipartimento di Informatica of Sapienza University of Rome.

References

[1] C. C. Miller. Revelations of N.S.A. spying cost U.S. tech companies. https://www.nytimes.com/2014/03/22/business/fallout-from-snowden-hurting-bottom-line-of-tech-companies.html.

[2] Cloud security alliance. government access to information survey. https://cloudsecurityalliance.org/research/surveys/#nsaprism, 2013.

[3] Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon. https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet, 2014.

[4] Jeep hacking 101. https://goo.gl/ulBt4U, 2015.

[5] Arduino. https://www.arduino.cc/, 2019.

[6] AWS IoT Core. https://aws.amazon.com/iot-core/, 2019.

[7] AWS IoT Greengrass. https://aws.amazon.com/greengrass/, 2019.

[8] AWS Lambda. https://aws.amazon.com/lambda/, 2019.

[9] Azure IoT Edge documentation. https://docs.microsoft.com/en-us/azure/iot-edge/, 2019.

[10] Docker: Enterprise Container Platform. œDocker,https://www.docker.com,2019., 2019.

[11] EdgeX. œEdgex,https://www.edgexfoundry.org/,2019., 2019.

[12] Google Cloud Functions. https://cloud.google.com/functions/, 2019.

[13] Google IoT Core. https://cloud.google.com/iot-core/, 2019.

[14] IBM Cloud Functions- Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk. https://cloud.ibm.com/functions/, 2019.

[15] IBM WATSON- The Internet of Things delivers the data. AI powers the insights. https://www.ibm.com/internet-of-things, 2019.

[16] Open fog consortium. https://www.openfogconsortium.org/, 2019.

[17] SensorTile. https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html, 2019.

[18] STMicroelectronics. https://www.st.com/content/st_com/en.html, 2019.

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

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

[21] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. SEDA: Scalable embedded device attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[22] A. Baumann, M. Peinado, and G. Hunt. Shielding applications from an untrusted cloud with haven. *ACM Trans. Comput. Syst.*, 2015.

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

[24] X. Carpent, K. Eldefrawy, N. Rattanavipanon, and G. Tsudik. Temporal consistency of integrity-ensuring computations and applications to embedded systems security. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18.

[25] M. Conti, E. Dushku, and L. V. Mancini. Distributed Services Attestation in IoT. In *From Database to Cyber Security*, pages 261–273. Springer, 2018. ISBN: 978-3-030-04834-1.

[26] M. Conti, E. Dushku, and L. V. Mancini. RADIS: Remote Attestation of Distributed IoT Services. In *6th IEEE International Conference on Software Defined Systems, SDS 2019*, pages 25–32, 2019.

[27] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, 2012.

[28] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84, 2017.

[29] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens. Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In *USENIX Security Symposium*, 2013.

[30] M. M. Rabbani, J. Vliegen, J. Winderickx, M. Conti, and N. Mentens. Shela: Scalable heterogeneous layered attestation. *IEEE Internet of Things Journal*, 2019.

[31] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. Scuba: Secure code update by attestation in sensor networks. In *Proceedings of the 5th ACM Workshop on Wireless Security*, WiSe '06, 2006.

[32] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla. SWATT: Software-based attestation for embedded devices. In *Proceedings of the 2004 IEEE Symposium on Security & Privacy*, IEEE S&P '04.

[33] D. Spinellis. Reflection as a mechanism for software integrity verification. *ACM Trans. Inf. Syst. Secur.*, 2000.

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

[35] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15. ACM, 2015.