State of the Art on: Interuptible Remote Attestation

DAVIDE LI CALSI, DAVIDE.LI@MAIL.POLIMI.IT

1. INTRODUCTION TO THE RESEARCH TOPIC

Embedded Systems have gained much popularity in the last decade, and are currently employed in a large range of applicative scenarios. You can find them in smart devices installed in people's homes, healthcare devices, industrial machines and many more. These devices generally use simple microcontrollers characterized by an extreme scarcity of resources. We are talking about devices with few kiloBytes of RAM and Flash memory, running relatively simple applications on a CPU with limited resources. Such devices also run on battery, which highly constrains their power consumption. The numerous constraints result in extremely cheap devices, but at the same time represent serious challenges for engineers and researchers. In particular, they pose a serious threat to a fundamental property: security. Security is inherently demanding, in the sense that it requires the allocation of some resources, therefore it is reasonable to expect great challenges when you work with constrained devices. Our research focuses on Remote Attestation[16] for low-power embedded systems. The high-level goal of Attestation is to ensure that the device has not been compromised, and is running the code that we expect it to run. Remote Attestation is useful in several real-world scenarios. Following, you can find some examples:

- **Safety-critical system in a core infrastructure site:** assume to have a MCU employed in an energy-production facility, say to control some physical actuators. Past experience has shown us that these sites are often targets for effective and destructive attacks, that can literally paralyze entire cities or regions. With attestation, you can have an indirect look at the code being run by the MCU, and check that an external attacker has not successfully infected it.
- **Metrology in energy-supply billing:** companies that provide energy to their users need to precisely measure the amount of energy that they supply to a specific user. This is crucial so that they can charge them appropriately. In order to ensure fairness in the billing process, some external entity can attest the device that runs the measurements, and ensure that the code running on it is legitimate.

This topic has been assessed in several prestigious conferences, such as:

- IEEE Symposium on Security and Privacy
- Usenix Security Symposium
- European Conference on Computer Systems
- IEEE International Symposium on Hardware Oriented Security and Trust (HOST)
- European Conference on Computer Systems (EuroSys)
- Asia Conference on Computer and Communications Security (ASIACCS)

- IEEE/ACM International Conference on Computer-Aided Design (ICCAD)
- International Conference on Security, Privacy and Applied Cryptographic Engineering (SPACE)

And some renown journals such as:

- IEEE Access
- IEEE Communications Surveys & Tutorials
- IEEE Internet of Things Journal
- IEEE Transactions on Information Forensics and Security
- Sensors

1.1. Preliminaries

Remote Attestation is a process through which a remote Verifier can attest that a device, the Prover, is in a legitimate state (i.e., uncompromised). In spite of the several existing techniques, RA was also standardized by the Trusted Computing Group[12] and the Remote ATtestation ProcedureS (RATS) working group[3]. Considering RA as a black box, the Verifier indiretly inspects the Prover's memory and outputs *Yes* if Prover is in an uncompromised state, according to an appropriate definition of compromise, *No* otherwise. The process is challenge-and-response-based, to ensure the freshness of the response and thus avoid replay attacks. Although several techniques were proposed, the general basic scheme consists of:

1. Verifier generates a challenge (usually a random nonce n)
2. Verifier sends the challenge to the Prover
3. Prover computes a proof of its inner state (usualy a HMAC over its memory content and n, using a shared key) thus obtaining the attestation reply A
4. Prover sends A to Verifier. Verifier compares A with some some pre-calculated values stored in a database. In case of no match, Verifier outputs No, otherwise it outputs *Yes*

The basic scheme can be seen as a starting point to build more complex and effective RA protocols. The process works well as long as you only attest Program Memory (**Static Remote Attestation**), since its content is static and known a priori. Other approaches attest also Data Memory and the runtime integrity (**Dynamic Remote Attestation**), but require a more sophisticated analysis that often involves the construction of a CFG[24, 18].
RA techniques can be informally divided into three major categories:

1. **Software attestation[2]:** software-based remote attestation does not rely on any hardware performing remote attestation. The core idea is that any malware trying to interfere with attestation to escape detection must inevitably slow down the attestation process. Therefore, a non-negligible delay in the attestation response can be interpreted as some malware trying to escape detection. In spite of its simplicity and flexibility, software attestation suffers from

several shortcomings. Most software-based techniques rely on strong assumptions about the environment, such as the Prover not communicating with other devices during the attestation protocol and that communication is limited to one-hop.

2. **Hardware attestation:** Hardware-based remote attestation leverage physical chips and modules to achieve remote attestation. These modules include Trusted Platform Modules (TPMs) which are hardware modules that enables secure storage and computation, and dedicated processor architectures such as Intel SGX[9] and ARM TrustZone[20]. Hardware-based protection is obviously much more solid than that provided by software mechanisms. Nevertheless, adding more hardware comes at a cost, and could highly affect the price of each device.

3. **Hybrid attestation:** Hybrid remote attestation is a hardware/software co-design that is based on a minimal trust anchor. These techniques rely on minimal hardware that is able to support and secure software-based checks. Hybrid attestation is a very promising research field, with new papers and solutions showing up every year. VRASED[21] is surely one of the most promising ones. It is a formally verified RA protocol, therefore it can claim solid mathematical guarantees of security.

1.2. Research topic

Our focus was on the problem of interrupts in RA. Some approaches to RA require disabling interrupts before its execution. In fact, if an attacker has taken over the device, it might use spurious interrupts to interfere with the process and avoid detection, e.g., by self-deleting before the attestation routine can attest its memory region or by self-relocating to a new memory area that was already attested. Some state of art RA schemes require interrupts to be disabled for the entire duration of Prover-side calculations. Solutions were proposed in the last years, but none of them allows for a fully-interruptible RA scheme, or they are just not suitable for low-end devices. For this type of devices many researchers have simply suggested methods to minimize the duration of the attestation routine, thus disabling interrupts for a less significant amount of time.

Unfortunately, for devices operating in real-time contexts, interrupts are a key feature that should not be disabled for too long. Devices operating in time-critical contexts require interrupts to be enabled at all times, as disabling them for few hundreds of milliseconds might result in catastrophic events. A fully interruptible RA scheme would guarantee the responsiveness of our system even during attestation.

2. Main related works

2.1. Classification of the main related works

RA is quite a broad topic, with researchers focusing on its several aspects. We have examined a rich variety of papers, each examining RA from a different standpoint. Some works focused on the formulation of new methodologies for software-based, hardware-based, or hybrid RA. Contrary to that, we have also read about a number of attacks that are meant to break RA. The most remarkable result is definitely by Francillon et al.[8], consisting in compression and ROP attacks to escape RA

detection. Further works try to speed up RA when it is aimed at verifying the state of multiple Provers, also known as Swarm Attestation[17, 5]. Finally, recent researches have targeted interrupts in RA. These are surely the most relevant ones to our purpose.
We also placed our focus on the results regarding malware detection by means of HPC. These works can be classified according to two major dimensions:

1. Type of counters they use
2. Type of classification algorithm

During the literature review process, we dwelled on two workflows in parallel. One was aimed at collecting works related to interrupts in RA, the other one aimed at collecting information about the use of Performance Counters for malware detection.
Although RA has been an active research direction for almost two decades now, interruptible RA has gained significant popularity recently. As briefly mentioned before, two major paths were taken into account by researchers worldwide. The first one simply gives up on introducing new constraints and checks in order to counteract malicious interrupts, and only tries to drastically reduce the duration of the attestation routine. The second consists of slight variations on some existing RA scheme. Each of these variations is added to make malware relocation harder or even impossible.
On the other hand, malware detection sparked the interest of a significant number of researchers worldwide. In that context, behavioral malware detection has gained popularity w.r.t. signature-based detection. The foundation of this method is the assumption that an infected device's behavior deviates from that of legitimate activity. In order to actively detect malware, you need to measure some quantities that encompass this behavioral gap. With that purpose in mind, several papers examined whether it is possible to use Hardware Performance Counters for malware detection. We assessed those articles with a focus on which counters they use, and ultimately how they use them to classify the observed activity.

2.2. Brief description of the main related works

Interruptible RA is still regarded as an open problem, yet some partial solutions were proposed. We will now illustrate them, showing their strengths and pitfalls.
TyTan[4] and TrustLite[15] are two of the first proposals that address the issue of interrupts in RA. TrusLite handles interrupts by using a secure exception engine. TyTan further builds upon TrustLite. It attests one process at a time, and thanks to a EA-MPU enforces the constraint that only processes that are not being attested can interrupt the routine. Both approaches require some extra hardware, that might not be present in low-end MCUs. Furthermore, as Tsudik et al.[7] observe,TrustLite and (in case of broken process isolation) TyTan are unable to counteract roving malware
SMARM[7] is a variation on the SMART[11] hybrid protocol. It allows interrupts by attesting memory in random order, an approach similar to that employed in the software-based RA protocol SWATT[23]. Assuming that memory divided into n blocks $b_1, b_2, ..., b_n$, the Prover attests each block according to a random permutation σ. In addition to that, interrupts are enabled only between the attestations of two consecutive (according to σ) blocks. This leads to some probabilistic guarantees about malware detection. The probability of evasion is dependent on the attacker's knowledge about the system. Assuming malware only knows the amount of memory that still has to be

attested, SMARM can detect it with probability $1 - e^{-1}$, which is around 63%. However, if malware also knows which blocks have already been attested, the probability of detection drops to $1/n$. Finally, in the worst-case scenario, any malware that also knows the future order of measured blocks can escape detection with certainty (although this event can only occur in case of insecure/leaky implementations). Even in the best-case scenario, the probability of evasion is non-negligible, so the authors suggest repeating attestation multiple times. HATT[1] also relies on Shuffled Measurements RA, and achieves the same probabilistic guarantees as SMARM. They also use Physical Unclonable Functions[13] to protect the secrets of the device from external physical attacks.
Other solutions exploit the concept of memory locking to prevent malware from escaping detection at attestation time. Tsudik et al.[6] introduce this concept and assess its effectiveness to secure the memory's content even in presence of malicious interrupts. Locking a memory region consists in protecting it to be read-only. Since self-relocating and transient malware strictly require performing some writes, making a memory area read-only is enough to stop them. The authors identify several locking strategies, depending on what you lock and when. In fact, the most obvious goal is to lock only the minimum amount of memory. Regardless of the best locking strategy, the major shortcoming of this method is its inapplicability to low-end MCUs. In fact, "In higher-end devices, memory locking can be used to prevent modifications until the end of attestation, as discussed in [22]. However, in low-end devices, where applications run on bare-metal and there is no architectural support for memory locking, temporal consistency is attained by enforcing that attestation software (SW-Att) runs atomically: once it starts, it can not be interrupted by any software running on the Prover, thus preventing malware from interrupting RA and relocating itself"[10].

When it comes to malware detection through HPCs, we were able to find several other works. ConFirm[25] uses event counters to monitor the device activity and eventually detect changes in the device's firmware. They rely on HPC present both in ARM and PowerPC processors. These counters belong to the PMU, and were natively designed to monitor and possibly reduce the power consumption of the CPU. HPCs measure low-level events, such as executed branch instructions, cache misses... with a variation in the number and nature of counted events that depend on the implementation. These HPC measurements are collected in a vector V, while classification works by querying a signature DB. The latter should store valid signature over some values of V that are associated to a benign activity, and that have been collected in a previous offline phase. If a match with some offline positive reference happens, the device is deemed as not infected. Several other works [14, 22, 19]exploit HPCs and feed them to some Machine Learning classifier that was trained to distinguish between legitimate and anomalous behavior. In the last decades, several models were evaluated, from Decision Trees to Deep Neural Networks and SVM. Most of these papers claim to have achieved great performances, with accuracies usually above 95%.

2.3. Discussion

We believe that Event/Performance counters could be a useful tool to detect the activity of a malware that tries to escape detection by means of malicious interrupts. Therefore, we aim to combine the use of Performance Counters with RA to ensure interruptible RA. Although you could argue that HPC-based anomaly detection techniques already exist, the context of RA for deeply constrained devices provides new challenges and opportunities.
The aforementioned works only use low-level, architectural event counters. While these counters

certainly work, they cannot properly capture the high-level behavior of applications. We believe that an applicative-level event trace is crucial to discriminate between benign and malicious software, therefore we will try to include some high-level counters for malware detection and investigate their effectiveness. In addition to that, to the best of our knowledge nobody has applied HPC to RA yet. So far, they were only applied to detect a wide range of malware attacks. However, transient or relocating malware performs a very specific type of attack (with negligible difference among the two). This reduced "attack-space" might lead to achieve high accuracy even with simpler models. These issues surely deserve further investigation, and we promise to address the issue.

References

[1] Aman, M. N., Basheer, M. H., Dash, S., Wong, J. W., Xu, J., Lim, H. W., and Sikdar, B. Hatt: Hybrid remote attestation for the internet of things with high availability. *IEEE Internet of Things Journal 7*, 8 (2020), 7220–7233. 2.2

[2] Ankergård, S. F. J. J., Dushku, E., and Dragoni, N. State-of-the-art software-based remote attestation: Opportunities and open issues for internet of things. *Sensors 21*, 5 (Feb 2021), 1598. 1

[3] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and Pan, W. Remote Attestation Procedures Architecture. Internet-Draft draft-ietf-rats-architecture-15, Internet Engineering Task Force, Feb. 2022. Work in Progress. 1.1

[4] Brasser, F., Mahjoub, B., Sadeghi, A.-R., Wachsmann, C., and Koeberl, P. Tytan. pp. 1–6. 2.2

[5] Carpent, X., ElDefrawy, K., Rattanavipanon, N., and Tsudik, G. Lightweight swarm attestation: a tale of two lisa-s. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (2017), pp. 86–100. 2.1

[6] Carpent, X., Eldefrawy, K., Rattanavipanon, N., and Tsudik, G. 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* (New York, NY, USA, 2018), ASIACCS '18, Association for Computing Machinery, p. 313–327. 2.2

[7] Carpent, X., Rattanavipanon, N., and Tsudik, G. Remote attestation of iot devices via smarm: Shuffled measurements against roving malware. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (2018), pp. 9–16. 2.2

[8] Castelluccia, C., Francillon, A., Perito, D., and Soriente, C. On the difficulty of software-based attestation of embedded devices. Association for Computing Machinery. 2.1

[9] Costan, V., and Devadas, S. Intel sgx explained. *IACR Cryptol. ePrint Arch. 2016* (2016), 86. 2

[10] De Oliveira Nunes, I., Jakkamsetti, S., Rattanavipanon, N., and Tsudik, G. On the toctou problem in remote attestation. 2.2

[11] ELDEFRAWY, K., PERITO, D., AND TSUDIK, G. Smart: Secure and minimal architecture for (establishing a dynamic) root of trust. 2.2

[12] GROUP, T. C. Tcg remote integrity verification: Network equipment remote attestation system. 1.1

[13] HERDER, C., YU, M.-D., KOUSHANFAR, F., AND DEVADAS, S. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE 102*, 8 (2014), 1126–1141. 2.2

[14] KADIYALA, S. P., JADHAV, P., LAM, S.-K., AND SRIKANTHAN, T. Hardware performance counter-based fine-grained malware detection. *ACM Trans. Embed. Comput. Syst. 19*, 5 (sep 2020). 2.2

[15] KOEBERL, P., SCHULZ, S., VARADHARAJAN, V., AND SADEGHI, A.-R. Trustlite: A security architecture for tiny embedded devices. 2.2

[16] KUANG, B., FU, A., SUSILO, W., YU, S., AND GAO, Y. A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects. *Computers Security 112* (2022), 102498. 1

[17] KUANG, B., FU, A., YU, S., YANG, G., SU, M., AND ZHANG, Y. Esdra: An efficient and secure distributed remote attestation scheme for iot swarms. *IEEE Internet of Things Journal 6*, 5 (2019), 8372–8383. 2.1

[18] KUANG, B., FU, A., ZHOU, L., SUSILO, W., AND ZHANG, Y. Do-ra: Data-oriented runtime attestation for iot devices. *Computers Security 97* (2020), 101945. 1.1

[19] KURUVILA, A. P., MENG, X., KUNDU, S., PANDEY, G., AND BASU, K. Explainable machine learning for intrusion detection via hardware performance counters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022), 1–1. 2.2

[20] NGABONZIZA, B., MARTIN, D., BAILEY, A., CHO, H., AND MARTIN, S. Trustzone explained: Architectural features and use cases. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)* (2016), pp. 445–451. 2

[21] NUNES, I. D. O., ELDEFRAWY, K., RATTANAVIPANON, N., STEINER, M., AND TSUDIK, G. VRASED: A verified Hardware/Software Co-Design for remote attestation. In *28th USENIX Security Symposium (USENIX Security 19)* (Santa Clara, CA, Aug. 2019), USENIX Association, pp. 1429–1446. 3

[22] ROSSO, M., RENES, J., VESHCHIKOV, N., ALVARENGA, E., AND DEN HARTOG, J. Actionable malware classification in embedded environments using hardware performance counters. SPACE 2021: Eleventh International Conference on<br/>Security, Privacy and Applied Cryptographic Engineering, SPACE 2021 ; Conference date: 10-12-2021 Through 13-12-2021. 2.2

[23] SESHADRI, A., PERRIG, A., VAN DOORN, L., AND KHOSLA, P. Swatt: Software-based attestation for embedded devices. vol. 2004, pp. 272– 282. 2.2

[24] TIGIST ABERA, N. ASOKAN, L. D. J.-E. E. T. N. A. P. A. R. S. G. T. C-flat:control-flow attestation for embedded systems software. *CoRR abs/1605.07763* (2016). 1.1

[25] WANG, X., KONSTANTINOU, C., MANIATAKOS, M., AND KARRI, R. Confirm: Detecting firmware modifications in embedded systems using hardware performance counters. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2015), pp. 544–551. 2.2