Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2020, 7(12):242-249**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Survey of attack resilient embedded systems design

Krunal Dave, Avani Dave

Abstract The advancement of industrial 4.0 has increased the utilization of resource constrained embedded and IOT devices in application ranging from Internet of Things (*IoT*) and CyberPhysical Systems (*CPS*). These systems are often involve in security-critical user data and/or information transfers. This makes them increasingly popular target for attacks to gain access of security-critical user data and/or information. The resiliency of a system is defined by its ability to detect, prevent and recover from the attacks. Crypto primitives such as secure boot, attestation, TPM, control flow and/or data flow integrity verification are wildly used to enhance device security. However, neither of these provides complete solutions for attack resiliency by supporting detection, prevention and recovery all three features. oftentimes these devices contains security critical user data and/or information which adversary can leverage and misuse. These devices are by design resource constraints and do not have onboard complex attack prevention cryptographic security primitives support. The resiliency of the system is defined by its ability to detect, prevent and recover from the attack.
To this end, this work provides the survey of twenty-plus state-of-the-art security systems solutions classified based on hardware-based, software-based or hybrid techniques. The paper further provides state-of-the-art comparison of different security techniques, identifies the gaps and provides the foundation for future research direction.

Keywords runtime resilient soc, memory modification attacks resilient system

1. Introduction

The utilization of resource constrained embedded and IOT devices in application ranging from Internet of Things (*IoT*) and Cyber-Physical Systems (*CPS*) has increased significantly in last two decades with the advancement of industrial 4.0. These devices are being used in various application ranging from home security systems, water irrigation, automotive controllers, portable devices, cameras, space applications, home appliances, online IOT devices with various sensor systems, geo location based data monitors etc.
This survey layout an overview of the state-of-the-art research landscape in secure resilient system design and summarizes their important contributions. The secure system design should have anomaly detection, prevention, and recovery mechanism in order to provide resilience to recent security attacks. This work clusters twenty-plus new state-of-the-art security system solutions into three groups based on design architecture, namely 1) hardware-based, 2) software-based, and 3) hybrid techniques. Using the proposed taxonomy, section §III presents the comparison of the current techniques, and from that, it derives the conclusion.

2. Secure Systems Research

The requirement of designing the security system for small embedded and IoT system is not new. Many researchers and industrial leaders have proposed solutions that enhance one or multiple security components such as secure boot, integrity check, authenticity check, runtime and boot time attestation, memory isolation, etc. These state-of-the-art solutions can be summarized and classified as shown in Fig. 1.

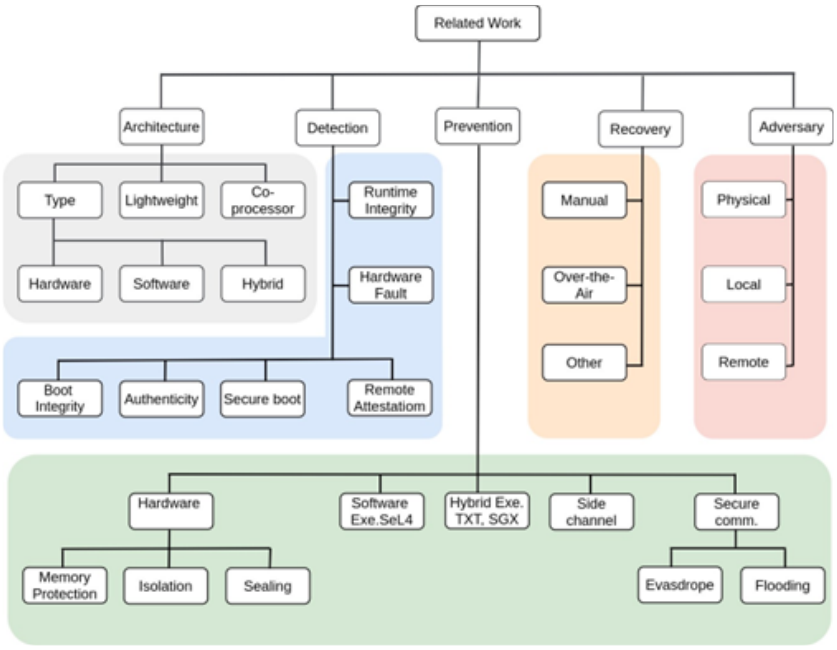

Figure 1: Classification of state-of-the-art attack resilient systems

This work has identified and compared twenty-plus state-ofthe-art implementations. It has classified them in hardwarebased, software-based and hybrid techniques. Suppose the design solution has a detection technique in hardware but provides protection or recovery using software implementation. In that case, the solution is classified as the hybrid implementation.

A. Hardware-based Techniques

This work considers secure CPS system design technique to be hardware-based if (1) All the necessary detection, protection, and recovery functionalities are implemented in hardware or (2) The system requires modification or addition to the underlying instruction set architecture. The following subsection provides an overview of a few important types of hardware-based techniques.

AEGIS [1] "AEGIS" proposed the first secure bootstrap with integrity assurance. "AEGIS" defines the boot process as a chain of many small layers of boot codes executed in a specific sequence. According to AEGIS, the integrity of a layer during the boot process can only be guaranteed if the following two rules are satisfied. (1) The integrity is checked for all the lower layers. (2) Transitions to higher layers occur after integrity checks on all lower layers are completed. Trusted Platform Module (TPM) [2] works as a secure co-processor, designed for protecting cryptographic keys and utilizing those keys for protecting data by signing or encrypting. TPM has some" special purpose registers" called Platform Configuration Registers (PCRs), which cannot be overwritten but can only be extended by hashing of software measurements together with previous PCR values. The TPM can sign the PCRs with a private attestation key to generate attestation evidence. TPM2.0 standard recommended by Trusted Computing Group (TCG) has 24 PCRs that can be used for authenticity and integrity check of the device's software by local and remote verifiers. Bit locker and remote attestation are examples of a few use cases of TPM. The TPM-based system's security relies on hardware-based components, and the software must be part of the TCB.

Intel developed Trusted Execution Technology (TXT) [3] to overcome the limitation of all the software that must be part of TCB. TXT uses the TPM chip and dynamically establishes a new RoT for software running in a virtualized environment separate from the normal application. When switching to trusted TXT software, the CPU essentially performs a warm reset and initializes a certain subset of PCRs with a new value. TXT suspends all other applications on the device to run the TXT software, impacting the performance or losing interrupts depending on code size.

SGX Software Guard Extensions (SGX) [4] Intel introduced Software Guard Extensions (Intel SGX) that enables an isolated execution environment called enclave. It ensures that the enclaves' code and data are secure

and accessible by the only enclave. Code execution starts at a specific entry point, and the enclave executes code securely without leaking any private information.

Trusted Execution Environment Trusted Execution Environment (TEE) was designed by Global Platform. TEE provides a secure and isolated execution area for security-critical applications. The TEE is isolated from the Rich Execution Environment (REE), where the untrusted OS runs. The REE resources are accessible from the TEE, while it blocks access from the REE unless explicitly allowed. Since the standard does not specify how manufacturers should implement it, TrustZone [5] is Arm's first implementation of TEE, which is widely used in smartphones. It provides protection for trusted hardware and software resources and switching capability between states at runtime.

Although TPM, TXT, and SGX provide hardware root-of-trust, they are not suitable for small embedded or IoT devices due to space, size, proprietary licensing, and cost constraints. Sancus [6] proposes a security architecture without a trusted code base. Similar to SGX, Sancus provides isolation by implementing additional CPU instructions that enforce the following: (1) all protected module's code is immutable, (2) data of a given software module is limited till the code of the same module is being executed, and (3) execution should start with a well-defined entry point. Sancus remotely attests that a specific software module runs un-compromised and uses a secure communication channel for RA.

CFLAT [7] leverages TEE to verify the execution paths remotely. It computes the hash of the exact sequence of executed instructions on the device using TEE. CFLAT can be used in RA for control-flow integrity. However, It imposes significant runtime overhead due to multiple context switching between the secure and non-secure environments. LoFAT tries to overcome this by implementing the hardware RA engine. Another work LiteHAX includes data-flow integrity checks to detect and protect the system from data-oriented attacks. Secureboot [8] is the first secure boot architecture for RISCV based small embedded devices. It uses a hardware-based state machine for a secure boot with code authentication and key management units. Upon detection of integrity failure, it resets the system to protect it from malicious code execution.

B. Software-based Techniques

The software-based techniques perform malicious code detection using software only. Software-based RA's are some use-cases of it, as explained below.

Pioneer [9] takes a one-time special checksum of the memory in unpredictable fashion. This technique prevents memory shadowing and forging checksum result attacks.

Viper [10] uses a time-sensitive checksum computation to verify integrity of peripherals' firmware. In all softwarebased methods device's integrity assessment relies on digest computation and is received within a time limit to the verifier. It makes time-optimized crypto core implementation vital. An alternative approach exploits memory constraints by not leaving any free space for malicious code to hide. SWATT [11] uses a similar principle that verifies the integrity of the prover by digest matching.

Choi et al. [12] propose to fi ll all unused memory with pseudo random numbers derived from a Vrf-generated random seed. Afterward, Prv can construct an attestation report by computing a hash over its entire memory and submitting it to Vrf. Vrf can validate the received report since it knows the expected entire memory snapshot, consisting of original memory contents and contents of previously empty memory filled with pseudo random numbers.

Yang et al. [13] further extended this approach to perform distributed software-based attestation, where the integrity of Prv's software is determined in a distributed manner by all Prv's neighbors. The security of this approach is dependent on two assumptions. First, it assumes space optimality of original memory contents (including the checksum code). Otherwise, an adversary can compress and gain enough free space not filled by pseudo random numbers to store and run malicious code to evade attestation [14]. The second assumption is that Vrf must be aware of all communication between Prv and other entities. This assumption is necessary to prevent a socalled "proxy" attack where malicious Prv asks for help from a more powerful accomplice device to compute a valid response. In particular, the authors in [14] propose a new primitive,

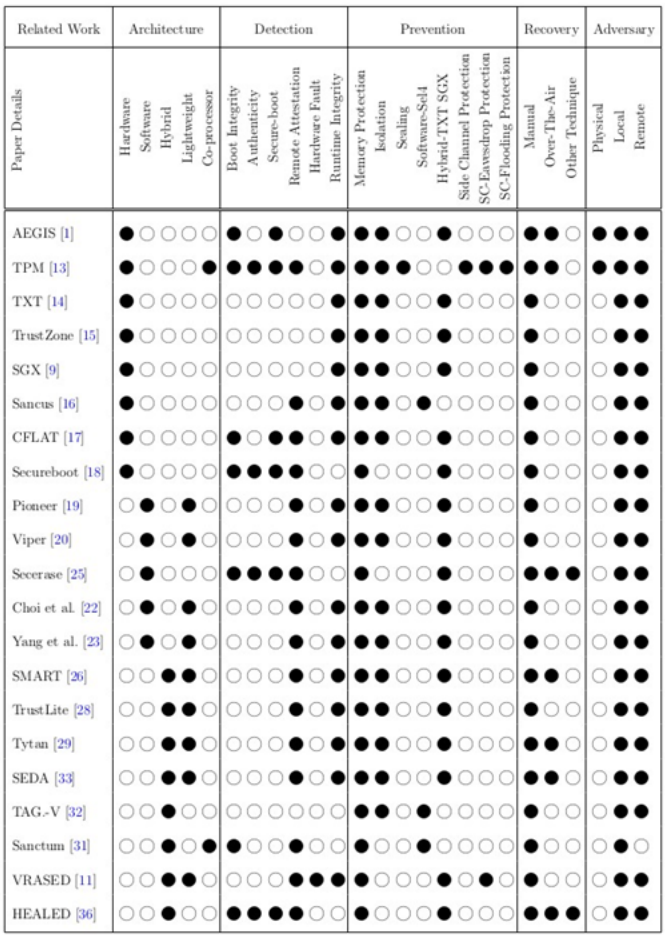

Figure 2: High level summary of research works discussed in chronological order

called randomized polynomial, and mathematically proves its space-time optimality in a formal model of their target. Intuitively, optimality guarantees that malware cannot evaluate a randomized polynomial faster than the theoretical lower bound. Whereas space optimality implies that no malware can hide in code implementing a randomized polynomial.

Secure erasure and update [15] has shown a method of putting software receiver-transmitter code into trusted ROM for connecting to a recovery server, which also requires additional ROM storage.

C. Hybrid Techniques

Hybrid techniques use hardware and software co-design to provide detection, protection, or recovery function. Examples of common hardware components include simple read-only memory storage (ROM) or dedicated hardware to monitor and enforce memory access control rules. Below, we overview two important RA architectures that have become a foundation for subsequent work in hybrid RA.

SMART [16] provides dynamic root of trust by minimal hardware modification to current micro-controller units (MCUs). It assumes that attestation keys and software are stored in immutable ROM and apply access control to memory. SMART ensures the attestation code's execution starts with specific reset instructions, and it cannot be interpreted to protect the device from replay attacks. Whenever any violation is detected, it erases all data memory and resets the device. Subsequent work [17] extends SMART to protect against denial-of-service (DoS) attacks and requires a Reliable ReadOnly Clock (RROC) for authentication.

TrustLite [18] extends SMART to provide strongly isolated software modules by using an execution-aware memory protection unit (EA-MPU), which can be programmed at compile time. TrustLite ensures no access to attestation key or code by other trusts, and truslets have a well-defined entry point. TrustLite achieves secure interrupt handling by modifying the CPU's exception engine for storing the trustlet's context in a protected memory region. It also clears the CPU registers before switching to an untrusted interrupt handler to protect the

system from key leakage. In addition to the Trustlite system needs a mechanism to disable interrupts or memory locking for preventing malicious code modification.

TyTAN [19] provides runtime programmability feature to Trustlite by using EA-MPU which provides program counter based isolation. It makes trustlets to be dynamically loaded and unloaded at runtime.

VRASED [20] is first hybrid implementation of verified remote attestation based similar mechanism as SMART. It utilizes a formally verified hardware module and crypto core for attestation. Upon failure, the device will reset, and thus VRASED protects the device from malicious code execution. PURE [21] is implemented on top of VRASED to provide secure code update and execution proofs to the remote verifier. Sanctum [22] is a hybrid approach with minimal hardware modifications and a trusted software component that offers isolation similar to SGX and allows enclaves to run at the user level. SGX uses a memory Encryption Engine (MEE) to encrypt and protect code and data memory, but Sanctum does not support it. It modifies the MMU using two Page Table Base Registers (PTBRs), one for untrusted code and one for the currently running enclave. The security monitor can only change the content of those registers.

Timber-V [23] uses tag-based to achieve memory isolation with MPU based access control for runtime. It presents a flexible and fine-grained lightweight isolation mechanism managed by a trusted software component. It can be used for remote attestation and sealing. Scalability is the issue as Timber-V uses 2-bit tags. Apart from this, few remote attestations are based on SEDA, SANA, and DARPA implementations for distributed IoT device attestation.

SEDA [24] was designed upon SMART and TrustLite as distributed attestation protocol. In SEDA, the verifier sends the attestation request to provers. Each prover generates the hash of its current state and sends the result to the central node, which aggregates the report and generates the final report to be sent to the verifier.

SANA [25] extends SEDA by implementing efficient and scalable Optimistic Aggregate Signatures (OAS) based central RA scheme. OAS combines many individual signatures into a single aggregated signature for fast verification.

DARPA [26] detects the time of absence based on the rationale that an adversary needs to spend a non-negligible amount of time to compromise the device physically. DARPA requires each device to monitor other devices by recording their heartbeats periodically. the remote verifier can detect any absent device from collected heartbeats.

Healed [27] has demonstrated the first recovery mechanism using Merkle Hash Tree (MHT), which assumes at least one node in the network is untempered, and firmware of it can be used to reflash corrupted nodes.

3. Comparison

Fig. 2 table summarizes state-of-the-art research based on the taxonomy proposed earlier in this survey paper. Our observation indicates that the majority of the techniques provide anomaly detection and prevention technique. Healed and Secure erase are only the two latest techniques that provide recovery mechanisms. Some RA techniques provide communication channel security for protecting the system from replay and flooding types of attacks. Hardware and hybrid techniques focus on detecting and providing prevention and protection using memory protection units or isolated execution with little overhead on hardware resources. Software-based solutions are more prone to attacks and require secure storage, along with it increases the latency of device performance. Essentially, the majority of the hardware-based techniques provide runtime attestation and security.

A. Analysis of Related Works

As can be seen from Table III, the state-of-the-art works comparison has a lot of data. Figure 3 represents bar-graph analysis of all twenty-four sample implementations, to visualize current state and potential scope of future opportunities.

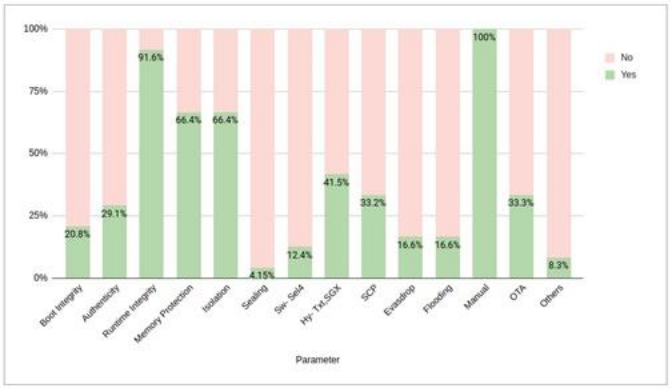

Figure 3: Analysis of Related Works

As can be seen from Table III, nearly each of the technique has one or few methods of malicious code modification attacks detection. Relatively low numbers of them has runtime prevention methods and almost all are supporting manual recovery.

B. Improvement Opportunities

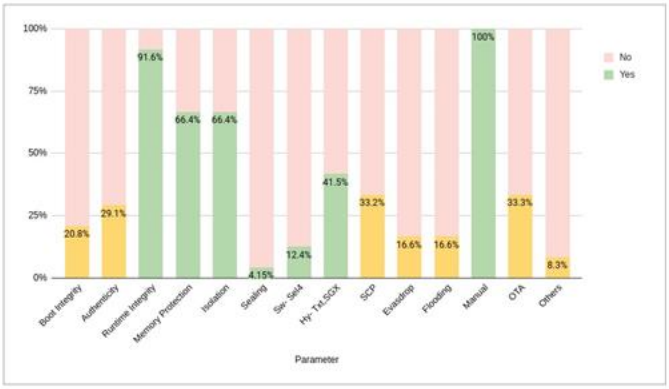

Figure 4: Improvement Opportunity

Figure 4 high-lights (in yellow) the potential opportunities for future work in the resilient systems design domain to enhance overall system security. Apart from this as discussed in comparison III section, only two implementations (Healed and Secure Erase) have demonstrated other recovery techniques. However, they lack in providing proper secure boot or run-time attestation. In addition to this, majority of them lacks in providing SCP, lightweight secure communication and onboard or application specific recovery process.

4. Conclusion

As this survey focuses on secure and attacks resilient system design of low-end embedded and IoT devices, Our analysis considers hardware-based techniques expensive. As such techniques require dedicated hardware features only available in more powerful (high-end) devices, e.g., personal computers, laptops, or smartphones. However, the same features are considered a luxury for low-end embedded and IoT devices. On the other hand, software-based techniques rely on strong security assumptions about their adversarial capabilities, which are unrealistic in networked (multi-hop) settings. Additionally, resource optimization becomes a challenging problem with a software-based solution. Therefore, this work have considered hybrid techniques to be the best fit for resource constrained target devices.

References

[1]. W. A. Arbaugh, D. J. Farber, and J. M. Smith, “A secure and reliable bootstrap architecture,” in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, ser. SP ’97. USA: IEEE Computer Society, 1997, p. 65.

[2]. "TPM wiki," https://en.wikipedia.org/wiki/Trusted Platform Module, 2010.
[3]. D. Grawrock, "Dynamics of a trusted platform: A building block approach," Intel Press, 2009.
[4]. F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2487726.2488368
[5]. "Amd platform security processor," https://ebrary.net/24869/computer science/secure technology.
[6]. J. Noorman, P. Agten, W. Daniels, R. Strackx, A. V. 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.
[7]. T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-flat: Control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p.743–754. [Online]. Available: https://doi.org/10.1145/2976749.2978358
[8]. J. Haj-Yahya, M. M. Wong, V. Pudi, S. Bhasin, and A. Chattopadhyay, "Lightweight secure-boot architecture for risc-v system-on-chip," in *20th International Symposium on Quality Electronic Design (ISQED)*, March 2019, pp. 216–223.
[9]. A.Seshadri,M.Luk,E.Shi,A.Perrig, L.van Doorn,and P. Khosla, "Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems," *SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, p. 1–16, Oct. 2005. [Online]. Available: https://doi.org/10.1145/1095809.1095812
[10]. Y. Li, J. M. McCune, and A. Perrig, "Viper: Verifying the integrity of peripherals' firmware," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 3–16. [Online]. Available: https://doi.org/10.1145/2046707.2046711
[11]. A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "Swatt: softwarebased attestation for embedded devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, 2004, pp. 272–282.
[12]. Y.-G. Choi, J. Kang, and D. Nyang, "Proactive code verification protocol in wireless sensor network," in *Proceedings of the 2007 International Conference on Computational Science and Its Applications - Volume Part II*, ser. ICCSA'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 1085–1096.
[13]. Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, ser. SRDS '07. USA: IEEE Computer Society, 2007, p. 219–230.
[14]. C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 400–409. [Online]. Available: https://doi.org/10.1145/1653662.1653711
[15]. D. Perito and G. Tsudik, "Secure code update for embedded devices via proofs of secure erasure," in *ESORICS*, 2010.
[16]. K. M. E. Defrawy, G. Tsudik, A. Francillon, and D. Perito, "Smart: Secure and minimal architecture for (establishing dynamic) root of trust," in *NDSS*, 2012.
[17]. F. Brasser, K. B. Rasmussen, A.-R. Sadeghi, and G. Tsudik, "Remote attestation for low-end embedded devices: The prover's perspective," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2897937.2898083 [18] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "Trustlite: A security architecture for tiny embedded devices," in *Proceedings of the Ninth European Conference on Computer Systems*, ser. EuroSys '14. New York, NY, USA: Association for Computing Machinery,2014. [Online]. Available: https://doi.org/10.1145/2592798.2592824

[18]. F. Brasser, B. El Mahjoub, A. Sadeghi, C. Wachsmann, and P. Koeberl, "Tytan: Tiny trust anchor for tiny devices," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[19]. I. D. O. Nunes, K. Eldefrawy, N. Rattanavipanon, M. Steiner, and G. Tsudik, "VRASED: A verified hardware/software co-design for remote attestation," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1429–1446. [Online]. Available: https://www.usenix.org/conference/ usenixsecurity19/presentation/de-oliveira-nunes

[20]. I. de Oliveira Nunes, K. Eldefrawy, N. Rattanavipanon, and G. Tsudik, "Pure: Using verified remote attestation to obtain proofs of update, reset and erasure in low-end embedded systems," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2019, pp. 1–8.

[21]. I. Lebedev, K. Hogan, and S. Devadas, "Invited paper: Secure boot and remote attestation in the sanctum processor," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, July 2018, pp. 46– 60.

[22]. A. Bradbury, G. R. Ferris, and R. Mullins, "Tagged memory and minion cores in the lowrisc soc," 2019.

[23]. N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "Seda: Scalable embedded device attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 964–975. [Online]. Available: https://doi.org/10.1145/2810103.2813670

[24]. M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "Sana: Secure and scalable aggregate network attestation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 731–742. [Online]. Available: https://doi.org/10.1145/2976749.2978335

[25]. C. Lab, "Intelligent Design of Electronic Assets (IDEA) and Posh Open Source Hardware (POSH)," 2018. [Online]. Available:https://www.darpa.mil/attachments/eridesignproposersday.pdf

[26]. A. Ibrahim, A.-R. Sadeghi, and G. Tsudik, "Healed: Healing & attestation for low-end embedded devices," in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds. Cham: Springer International Publishing, 2019, pp. 627–645.