Remote Attestation Assurance Arguments for Trusted Execution Environments

Ahmad B. Usman
ahmad.usman@liu.se
Linköping University
Sweden

Nigel Cole
nigelcole.94@gmail.com
Linköping University
Sweden

Mikael Asplund
mikael.asplund@liu.se
Linköping University
Sweden

Felipe Boeira
felipe.boeira@liu.se
Linköping University
Sweden

Christian Vestlund
christian.vestlund@sectra.com
Sectra Communications AB
Sweden

ABSTRACT

Remote attestation (RA) is emerging as an important security mechanism for cyber-physical systems with strict security requirements. Trusted computing at large and Trusted Execution Environments (TEEs) in particular have been identified as key technologies to enable RA since they ideally allow retaining some element of control over remote devices despite them being compromised at the OS level. Unfortunately, sometimes it is claimed that TEEs provide RA support without really substantiating how this support is provided. In this paper we build the assurance arguments for RA to carefully map how secure RA depends on underlying security properties and how these in turn can be provided by TEE capabilities. We base our security analysis of RA on existing literature on security requirements for RA and use Goal Structuring Notation (GSN) as the method to build the security arguments. Our analysis identifies the set of TEE properties (as described in the GlobalPlatform standard) that are needed to support RA, and which goals that cannot be mapped to TEE implementations, and therefore, require other forms of evidence for RA to be trusted at the top level.

CCS CONCEPTS

• **Security and privacy** → *Trust frameworks*; *Security protocols*; • **Computer systems organization** → *Embedded systems.*

KEYWORDS

Remote Attestation, Trusted Execution Environments, Goal Structuring Notation, Assurance, GlobalPlatform, CPS.

ACM Reference Format:
Ahmad B. Usman, Nigel Cole, Mikael Asplund, Felipe Boeira, and Christian Vestlund. 2023. Remote Attestation Assurance Arguments for Trusted Execution Environments. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '23), April 26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3579988.3585056

SaT-CPS '23, April 26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0100-9/23/04.
https://doi.org/10.1145/3579988.3585056

1 INTRODUCTION

As cyber-physical systems are becoming increasingly more complex and intertwined with other systems, the security focus is expanding from protecting a single device to protecting a large (often heterogeneous) set of devices. Devices at the network edge (sensors and actuators) might have limited capabilities for their protection but are still vulnerable to malware through supply chain attacks [31] among others. The motivation behind our work is the challenge of arguing security assurance in such complex systems where RA is used as a basic security mechanism.

RA [26] has been proposed as a basic support mechanism that allows a Verifier to attest the state of a Prover. One can imagine that the Verifier is a cloud-based service that interacts with its users and the world around it using connected IoT devices. Each such IoT device would then be a Prover that would need to show the Verifier that it is in a "good" state in order to interact with the online service. In recent years, there has been a large interest in various RA protocols, including swarm attestation [27], anonymous attestation [7], and mutual attestation [8]. We focus on the most basic direct RA mechanism between two parties and the capabilities required by the Prover device. A major challenge of RA is that it requires some trusted capabilities in the Prover device to work, which increases the trusted computing base and therefore needs careful analysis.

Many works have proposed that the Root of Trust (RoT) in the Prover can be provided in the form of a TEE. A TEE is a feature of the CPU which essentially provides some part of memory which is isolated from all other (untrusted) software. Intuitively, the semantics of the TEE ensures that applications running within a TEE can be trusted even in an adverse environment (e.g., a compromised OS). The major general-purpose CPU architectures today provide such features (e.g., Intel-SGX, ARM TrustZone, AMD SEV), and there is a strong commercial interest in being able to provide trusted computing capabilities as this is seen by many as a necessity in the future cloud solutions. There are also more research-oriented solutions, such as the Keystone framework [29], which is based on the open RISC-V platform. The documentation for several of these TEE platforms claims to provide RA as a supported feature. However, the exact relationship between the capabilities of a TEE and what is required to support RA is often not clearly stated.

This paper aims to analyze the relationship between RA and TEEs carefully. Put another way, what are the TEE features that

enable RA? To answer this question, we first analyze the existing literature on RA that explicitly states what device capabilities are required and synthesize these in a unified hierarchy of properties. We then map these requirements to standard TEE capabilities that are supported by most modern TEE implementations. We use the Goal Structuring Notation (GSN) [1] method to document the requirements found in related work and also to unify and break down the overall goal to achieve secure RA to sub-goals in an iterative fashion until each goal can be mapped to a documented TEE capability (or requiring some other security mechanism to support it). The GSN method was originally developed to support assurance cases in safety-critical systems (thus supporting safety goals rather than security goals). However, as has been pointed out many times, there are strong similarities between safety and security, and methods from one domain can often (but not always) be transferred to the other. Our aim is that by using a structured approach to analyze security mechanisms and their relationships in a specific context, it is possible to validate (or invalidate) arguments about whether TEE supports acceptably secure RA.

We use the GlobalPlatform TEE Protection Profile standard [44] as the basis for determining the capabilities provided by modern TEEs. It has wide industry support and provides a comprehensive framework for specification and certification across a range of stakeholders and application areas. The security properties needed to support RA have been investigated by analyzing five different RA schemes [10, 15, 19, 38, 40] that explicitly state such requirements. Our analysis is focused on the aspects of RA that can be traced to security properties provided at the device level (e.g., by a TEE). We consider one-way, direct, two-party attestation, also excluding privacy requirements. So swarm attestation, and anonymous attestation is out of the scope of this paper. Moreover, we do not consider the problem of how to assess the attestation response and whether that corresponds to the Prover being in a good state or not. To summarize, the contributions of this paper are as follows.

- A systematic analysis based on five selected papers of required security properties for a Prover in an RA scheme.
- Application of the Goal Structuring Notation (GSN) method to the identified security properties, synthesizing them into a unified terminology and determining how they depend on each other.
- A mapping of the relevant security goals to features provided by the GlobalPlatform TEE Protection Profile standard, thereby identifying which parts of RA that can be provided by a TEE solution, and which aspects that need to be provided through other means.

The rest of this paper is organized as follows. In Section 2 we provide a high-level overview of RA, TEE, and GSN. Section 4 briefly discusses the security properties from the five selected papers, and Section 5 contains the main contributions of the paper with the GSN of RA and its mapping to TEE properties. Section 3 contains related work and finally we conclude with Section 6.

2 BACKGROUND

This paper is concerned with the analysis of RA and how it maps to TEE. This analysis is performed with the help of GSN. These three concepts are now further introduced in this section.

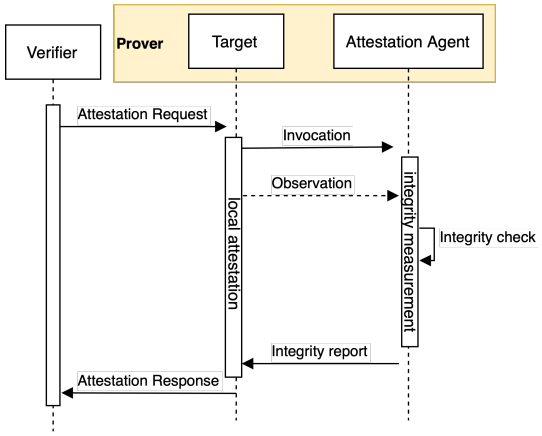

Figure 1: Architecture of remote attestation model

2.1 Remote Attestation

Remote attestation (RA) [43] is a modern technique that (remotely) detects malicious activities and compromised software by validating the integrity of a non-trusted party. This technique has been widely used across several platforms, particularly in low-end devices. Figure 1 shows the basic idea of RA where a Prover attests itself to a Verifier. Local attestation can be established between a trusted application running within an enclave (an isolated portion of memory) on another untrusted application which shares the same hardware. On the other hand, when an entity is interested in verifying the correctness of a remote application, such a process requires a protocol or Remote attestation mechanism. As illustrated in Figure 1, Remote attestation accomplishes this by first creating an attestation request from the Verifier. The Target in the Prover forwards this request by making an invocation to the attestation agent.

The attestation agent performs the integrity measurement based on an observation of the state of the Target and checks its integrity (in some cases, the outcome of the integrity check is not done here, but in the verifier). If the attestation agent performs the integrity check successfully, it then forwards this integrity report to the Target. The Prover will finalize the process and send an attestation response to the Verifier.

2.2 Trusted Execution Environment

A Trusted Execution Environment (TEE) is an isolated environment used for trusted execution. It is isolated from other entities in the sense that it can perform secure executions, meaning that the nature and results of the executions are protected, and also have secure storage for data and cryptography keys [41]. The isolation allows the TEE to interact upon requests from entities that it trusts. The TEE performs a secure boot, ensuring that the system is trusted and reliable by comparing hash values.

TEE promise to resolve issues and security challenges by providing data and code integrity, confidentiality, and more importantly attestability. They achieve this by isolating the OS, Applications and other elements into two environments, commonly known as Normal world and Secure world. Alternatively. Some well-known TEE

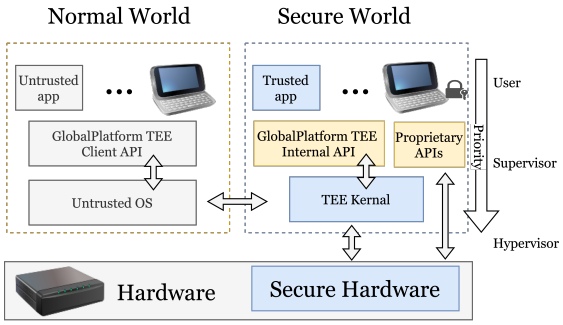

Figure 2: Typical architecture of TEE

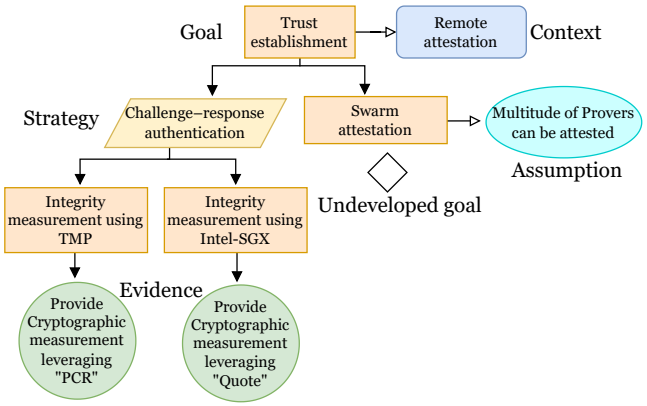

Figure 3: GSN Trust establishment example

implementations include SGX from Intel [12], Security Encrypted Virtualization (SEV) from AMD [16], TrustZone from Arm [39] and RISC-V [4].

Figure 2 shows how a typical TEE is usually configured. The figure has a logical separation of three priority privileged executions, starting from the least priority 1) User mode, which includes applications/enclaves, 2) Supervisory mode and it includes Operating system/kernel, 3) Machine/Hypervisor mode with the most priority for accessing the memory.

The exact implementation of a TEE will differ depending on the context or the use case. Although the goal of isolating the Secure world from the Normal world is shared between TEE implementations, there is a difference on how the goal is fulfilled by either the HW or SW. For instance, TrustZone TEE [36] partition its memory of both SW and HW in either the Normal world for anything, or the Secure world for sensitive sub-systems. Keystone [29] on the other hand uses a Physical Memory Protection (PMP) HW component to isolate memory into partitions. PMP features are provided by RISC-V.

The GlobalPlatform [44] provides and publishes a set of established industry standards that specify essential TEE use cases and capabilities. GlobalPlatform provides a specification for securing digital devices and services through standardized certifications and technologies, allowing interoperability, and a common understanding of what it means to provide TEE capabilities.

2.3 Goal Structuring Notation

The term Assurance in computer security refers to the measure of confidence when determining that a security claim is true. Assurance in itself is subjective, making it hard to determine the degree of assurance for an assurance case and its arguments.

The GSN [22] method can be used to explicitly define an assurance case for a specific environment and provide a convincing argument that a system is acceptably safe (or in our case, secure). It does this by graphically representing core elements and the relationships between them. These core elements are:

- Goal – A claim about the system.
- Strategy – A means to address a goal.
- Solution – Evidence to support claims.
- Context – The scope or made assertions.
- Undeveloped Goal – A goal that is to be developed further.

Structuring a security case with the above elements creates the goal structure. Acquiring evidence for the solutions will determine whether or not the sub-goals are acceptably fulfilled, thereafter determining the outcome of the main goal. The GSN community standard (GSNCS) [1] further develops the GSN method and adds the following core elements:

- Assumption – An intentionally unsubstantiated statement.
- Justification – A rationale statement.
- Undeveloped element decorator – Undeveloped argument that can be applied to goals and strategies.

An example of how a GSN top-down method is formulated is presented in Figure 3. Coloring is added by us and is not part of the GSNCS. An arrow with a solid arrowhead represents *SupportedBy* which means that a goal is supported by another goal, strategy or solution. It can also be used for a strategy that is supported by a goal. There is also an arrow with a hollow arrowhead that represents *InContextOf* which is used to declare a contextual relationship between a goal and context, assumption or justification. It can also be used between a strategy and context, assumption or justification. There are two main methods described in the GSNCS. The GSN Six-Step method which was obtained from the method given by Kelly [23], and the Bottom-Up method. The former method (used in this paper), is a top-down method that focuses on creating goals as the starting point. While the latter method, which is not used in this paper, is a bottom-up method that focuses instead on the available evidence to use as solutions.

Figure 4 shows the six-step method from the GSNCS. The first step is to identify a goal that is to be added to the GSN. Secondly, the context of this goal needs to be explicitly stated. Then, for the third step there may be a need for a strategy to support the new goal. In step four the strategy needs to be justified similarly to step two. Step five is taken when the goal needs to be further developed since there does not exist a sufficient and detailed goal that can be fulfilled with a solution. If the claim is at a level where a solution can be connected, step six is taken instead where a solution is identified. This does not mean that every goal or strategy needs an added context if it is already understood from the context of higher level goals or through clearly motivated sub-goals.

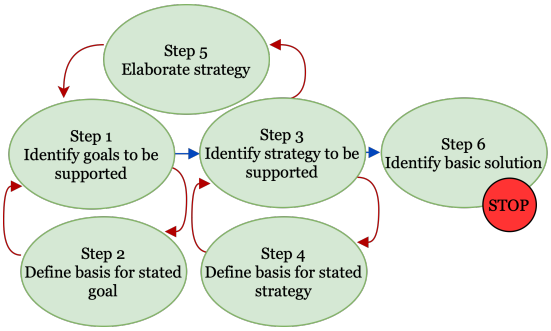

Figure 4: The recursive six-step process for creating GSNs

3 RELATED WORK

In the first part of this section, we focus on TEE studies that provide RA mechanisms. We then relate works that identify security requirements for different attestation architectures and break down their security properties and guarantees. Finally, we describe some other uses of the GSN method to argue security assurance.

TEE-based RA implementations. Intel-SGX is one of the major TEE platforms available in the market, and it has been excessively investigated [28] in numerous studies relating to RA. For example, Dhar et al. [17] propose ProximiTEE, which provide RA

and also provides a solution against relay attacks in order to achieve a successful attestation regardless of the compromised software. OPERA by Chen et al. [9] is another Intel-SGX RA. It is an open platform RA mechanism. Even though OPERA was built with Intel premises, the attestation mechanism does not involve the attestation services provided by Intel.

The hardware element provided by Arm TrustZone [39] allows users to establish one TEE for every system. The two isolated environments in this hardware-based architecture are called the Normal world and Secure world as in Figure 2; however, they are often called Rich Execution Environment (REE) and TEE, respectively by TrustZone. The entire isolation between the two environments is done through the security extension of the TrustZone's system hardware, which includes CPU, memory and peripherals. Arm TrustZone doesn't provide a full RA mechanism [35]. However, it may be possible to check the signatures which are signed by a trusted mechanism on an Arm device, such as the one signed by Intel CPU, for example. In addition to that, there are several proposed protocols for Arm TrustZone that perform both one-way RA such as SecTEE by Zhao et al. [47] (which is a software-based enclave designed to provide RA, integrity measurement and along with other services), AdAttester [30] (a verifiable smartphone framework that attests two primitives operations for attestation based on TrustZone in the Secure world) and mutual RA [2, 40].

RISC-V is an open-source ISA introduced to address the limitation of the TEE vendors. Intel-SGX, Arm TrustZone, and AMD SEV are proprietary and are constrained for a certain HW and implementation. The main purpose of Keystone [29] is to provide isolation; however, it also supports RA. Sanctum [13] simulates the behavior of Intel-SGX in terms of isolation and RA, where the RA derives its trust from Root of Trust (RoT). Other successful solutions on RICS-V are TIMBER-V [45] and LIRA-V [42]. While both provide isolation and RA, the former focuses on isolation implementation and the latter focuses on providing comprehensive mutual RA.

AMD Secure Encrypted Virtualization (SEV) [16] provides secure encryption virtualization, as the name implies. It does so by isolating its environment with a virtual machine, container or similar mechanism, taking advantage of the trusted hypervisors. The SEV-SE version of AMD encrypts the entire contents of the CPU register when the VM shuts down to prevent the CPU from leakage of data. SNP added new functionality for memory integrity to prevent malicious activities, while Vanilla was introduced to support and run on macOS. All three versions provide a full RA mechanism.

RA properties. Conti et al. [11] proposed RADIS, a *RA of Distributed IoT Services* protocol, targeting distributed IoT services for trustworthiness verification. Among the contributions of RADIS is defining the requirements for security properties for RA while taking advantage of distributed IoT services.

Ammar et al [3] introduced a novel scheme called SIMPLE, it is a RA Approach designed for resource-constrained IoT devices. For providing secure RA through reliable software, SIMPLE requires the minimal HW supports. SIMPLE derived its RA security properties from VRASED [38].

Ménétrey et al. [35] present RA mechanisms for TEEs. Aiming at comparing the RA's state-of-the-art schemes by highlighting four different architectural hardware-assisted TEEs. The authors review and explain RA principles in modern TEEs; however, they do not break down the RA principles with regard to the TEE security functionality.

Koeberl et al. [24] propose a TEE-based approach to allow different parties to gain access to computation data and capabilities, assuming that the parties agree upon security assurances beforehand. The proposed model is reliant on the assumption that TEEs provide good assurance. The article states that the assurance depends on the implementation of the TEE and that the analysis of the security of TEEs needs to be further researched. Furthermore, it states that while there exist multiple TEE solutions, their security properties differ, possibly revealing TEE shortcomings when TEE solutions are employed for various usages. This further emphasises the need for evaluating the assurance of TEEs.

Bognar et al. [5] architecturally investigated security issues presented in the VRASED [38] and Sancus 2.0 [37] articles. Their findings identified some limitations of the security properties in both the Sancus assumption and seven assumption by VRASED, which ultimately led them to report several attacks on the implementation and provide possible solutions to mitigate these attacks.

Maene et al. [32] defined security properties offered by trusted computing architectures leveraging attestation and isolation. Furthermore, the studies specified and compared against 12 architectures from both industry and academia, focusing on the isolation and attestation from the hardware-based perspective. The studies concluded that the 12 architectures in the comparison provide strong guarantees, but few of them can support the entire mechanisms for trusted computing.

Other important schemes discusses RA security properties in this context are SMART [18], TyTAN [6], TrustLite [25] and Sanctum [14].

Use of GSN to argue security assurance. Yamamoto [46] provides a method to add attributes in GSN which are used to denote security. The attributes are given a value ranging from very unsatisfied to very satisfied. Yamamoto also provides a use case for a LAN Device Management System.

Zhou et al. [48] proposed a quantitative approach to support developing assurance cases and argue for cybersecurity in vehicle compliance on the road by using GSN. Their strategy initially started by analyzing the standard approach, later proposing the cybersecurity assurance model-based approach leveraging GSN.

He et al. [20] showed how GSN can be utilized for constructing generic cases for security purposes, by analyzing cybersecurity incidents reported previously, targeting healthcare organizations. They claimed that they are the first who attempted at extending an approach for generic modeling from a safety perspective. The generic approach was supported by two practical use-cases at different geographical areas, veterans' affairs administration in the United State and Shenzhen hospital in China.

Other works [21, 34] either implement GSN assurance cases directly or develop GSN with extensions and other techniques. Matsuno [33] has pushed to spread assurance cases in Japan.

There is clearly a great interest in building assurance cases with GSN. However, none of the related work modeled the relationship between the requirements of RA and features provided by TEEs.

4 ANALYSIS OF RA GOALS

Our analysis of ra goals are derived from the RA security properties in the literature. To extract these security properties, we performed a targeted literature search focused on papers that explicitly list security properties needed to implement secure RA. The papers were found through a combination of searching for the keywords "remote attestation" and "security properties" as well as other relevant papers that we found through related works. We performed an initial filter to reduce the number of more carefully studied papers by excluding papers published before 2010, survey papers, or papers that do not discuss platform security properties related to RA (this includes papers that focus on the network layer rather than the platform properties, e.g., collective RA and anonymous RA). Moreover, in some cases we include only a subset of the papers published by a group of authors that use similar properties in a series of papers to avoid unnecessary repetition.

We briefly describe the five selected papers, which all explicitly state and discuss at least four RA-related platform properties. Thus, for a paper to be included, we require that the properties are named and given definitions.

- Dave et al. [15] propose the SRACARE, *Secure RA with Code Authentication and Resilience Engine (SRACARE)*, a framework for RA on a RISC-V platform with 8 specific security properties for secure boot and RA system design.
- Nunes et al. [38] present VRASED, *Verifiable RA for Simple Embedded Devices*, claiming that no formally verified RA or HW/SW co-design does exist prior to this work. VRASED enables RA capabilities for embedded devices and stated seven relevant security properties for RA.
- Shepherd et al. [42] present another RA scheme for low-end devices called LIRA-V, *Lightweight RA for Constrained RISC-V*, a RA solution that leverages the RISC-V PMP. Furthermore, LIRA-V uses SCYTHER (symbolic analysis tool for security protocols) to formally verify a mutual attestation protocol that has been proposed in the communication between trusted devices.
- In 2011 Coker et al. [10] identified the five *principles of RA*. The security properties and their semantics allows the developer to uniquely identify common vocabulary among different RA mechanisms. This scheme is widely used across studies for determining the necessary requirements for RA.
- Francillon et al. [19] presented *A Minimalist Approach to RA*, claiming to introduce the first step towards RA study systemically. The work presented in this paper systematically provides a treatment of a RA and identifies the sufficient and necessary properties for secure RA, ultimately mapped them into a combination of HW and SW.

Note that the focus of our analysis is on the security properties and goals that are stated as central for RA in these works. We do not consider any of the solutions provided in the papers. In fact, most of these schemes target low-end devices that try to achieve RA without powerful TEE capabilities as provided by general-purpose CPUs. Furthermore, the properties stated in the papers are not always unambiguous and might be influenced by the proposed solution. In the following section, we have made an attempt to interpret these requirements and explain them in separate subsections in order to later unify them into a joint terminology, even if they are originally stated with different words.

4.1 SRACARE

There are three main domains provided in SRACARE [15], constituting eight security properties **[A1- A8]** as follows; **1) Secure Communication: [A1]** - *Eavesdrop Protection* (to protect against eavesdropping, replay and Man in the middle (MITM) attacks, the devices should be equipped with wiretap detection mechanisms), **[A2]** - *Flooding Protection* (protection against attacks such as both Denial of Service (DoS) and Distributed Denial of Service (DDoS)). **2) Key Protection: [A3]** - *Key Confidentiality* (protection of the key against adversarial attacks by storing the key in a secure ROM), **[A4]** - *Access Control Enforcement* (prevention against unauthorized access to the memory where the key and data are stored by providing access control policy). **3) Safe Execution: [A5]** - *Correct Implementation* (protect against modification of sub-modules implementation), **[A6]** - *Atomicity* (interruption-free execution of integrity measurement), **[A7]** - *Error Free Execution* (for all the supported hardware and software submodules) and **[A8]** - *Controlled Invocation* (integrity measurement runs as first-to-last).

4.2 VRASED

Both SRACARE [15] and VRASED [38] share similar properties (in fact, SRACARE refers to VRASED as a starting point). While the former has three domains associated with eight security properties, the latter has only two domains with seven security properties **[P1-P7]**; **1) Key Protection: [P1]** - *Access Control* (restricting access to the key, reachable only through software attestation), **[P2]** - *No Leakage* (memory clearance after execution), **[P3]** - *Secure Reset* (resetting the system to its default configuration). **2) Safe Execution:**

[P4] - *Functional Correctness* (implementation of the RA protocol expects the correct behavior of the entities), **[P5]** - *Immutability* (of the software attestation execution), **[P6]** - *Atomicity* and **[P7]** - *Controlled Invocation*. As mentioned earlier, those properties overlap with SRACARE. We omit to show a corresponding GSN for VRASED and the other analyzed papers, as we will later synthesize these in a joint GSN diagram.

4.3 LIRA-V

LIRA-V [42] addresses four security goals **[GL1 - GL4]**; **[GL1]** - *Measurement Procedure Integrity* (during measurement process of a device, the proposal is considered to be protected against privilege attacks), **[GL2]** - *Signing Key Secrecy* (the key used by the Prover to sign response quotes shall be protected against adversaries), **[GL3]** - *Secure Channel Creation* (the communication channel between the Verifier and the Prover shall be created securely) and **[GL4]** - *Bi-Directional Attestation* (the secure channel between the Verifier and the Prover shall be attested within a single protocol).

4.4 Five Principles of Remote Attestation

The paper [10] denoted the five principles **[PR1- PR5]**; **[PR1]** - *Fresh Information* (reflecting the state of the Prover by the Target during attestation), **[PR2]** - *Comprehensive Information* (capability of the attestation mechanisms to allow Verifier to deliver comprehensive information including the full internal state of the Target), **[PR3]** - *Constrained Disclosure* (Target's ability of enforcing policies to Verifier in order to control the data that has been delivered), **[PR4]** - *Semantic Explicitness* (attestation mechanisms should allow the Verifier to collect the attestation data by uniquely identifying the semantic of the Target) and **[PR5]** - *Trustworthy Mechanism* (ability of Verifier to correctly reason Prover's data even in the existence of an attack). In addition to the high-level principles discussed, they also stated five *abilities* that an RA architecture should provide. These abilities are focused more on the device platform, whereas the principles concern the whole attestation mechanism.

4.5 Minimalist Approach

Minimalist approach by Francillon et al. [19] describes five security properties that are necessary for secure RA. These properties are also overlapping with the main studies as in VRASED; but some are worded differently. For instance, *Uninterruptibility* mentioned in Minimalist, corresponds to *Atomicity* found in VRASED in Nunes et al. [38]. In particular, the RA security properties found in Minimalist approach are**[M1-M5]**; **[M1]** - *Exclusive Access* (to the key k), **[M2]** - *No Leaks* (of the key k), **[M3]** - *Immutability* (of the code), **[M4]** - *Uninterruptibility* (to execute the attestation), **[M5]** - *Controlled Invocation* (to restrict the invocation of the attestation).

5 SYNTHESIS AND MAPPING

This synthesis and mapping section presents the result we have obtained by performing the GSN analysis. Firstly, we provide an overview and a general description of the GSN representation of acceptably secure RA. Secondly, we describe in more detail the security goals and explain how they are connected to each other, and finally, we provide a short summary of the outcome.

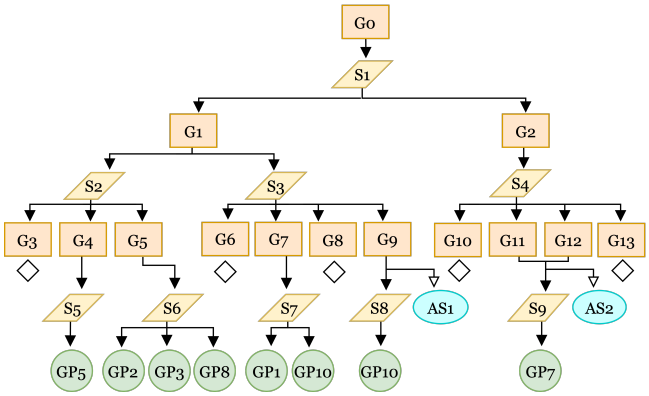

Figure 5: Synthesized GSN for secure RA and TEE. The goals are described in Table 2, strategies in Table 1

.

Table 1: Description of Strategies in Figure 5

ID	Description of Strategy
S1	Protection of RA entities and communication
S2	Memory isolation
S3	Safe execution
S4	Protocol ensures secure communication
S5	Secure storage provided by TEE
S6	Isolation & instantiation enabled by TEE
S7	Proper execution of integrity measurement by TEE
S8	TEE ensures proper execution of the integrity measurement
S9	Using cryptographic functions in TEE

5.1 Overview

The resulting GSN for RA, as shown in Figure 5, is based on the security properties described in Section 4. The GSN's main goal is to provide an acceptably secure RA. To ensure that each goal is upheld, a strategy element as discribed in Table 1 is added under the goal, which can then be achieved by fulfilling a number of sub-goals. For each goal and strategy in the figure. Table 2 contains a short description together with a mapping to the properties gathered from the five papers above. IDs are sorted left-right based on Figure 5. (SRACARE [15], VRASED [38], LIRA-V [42], Five Principles of RA [10], Minimalist [19]). A '-' means that no explicit mention of such property was provided in the paper. However, this does not mean that the topic is not discussed or acknowledged, which is the case more often than not. The bottom part of the GSN is a TEE security functionality that is meant to provide solutions to the connected goals indicated, as evidenced in the diagram.

When comparing the goals in the GSN diagram and the properties and requirements elicited from the analyzed papers, one can see that some goals are similar but worded differently and are therefore merged into a single goal when applicable or split into multiple goals due to the structure of the GSN tree.

Table 2: Comparison of goals among RA schemes based on Figure 5. The table considers explicit security properties provided in the corresponding schemes, other properties might be implicit or argued in the respective solutions

ID	Description	SRACARE	VRASED	LIRA	Five	Minimalist
G0	Acceptably secure remote attestation.	-	-	-	-	-
G1	Protection of Verifier, Target and Attestation Agent.	-	-	-	-	-
G2	Secure communication between Target and Verifier.	-	-	**GL3**	-	-
G3	Attestation Agent cannot modify Target state.	-	-	-	-	-
G4	Key protection.	**A3, A4**	**P1-P3**	**GL2**	-	**M1, M2**
G5	Integrity of the Attestation Agent state.	-	**P5**	-	-	**M3**
G6	Attestation Agent has (read-only) access to the full Target state.	-	-	-	**PR2**	-
G7	Controlled invocation - first-to-last integrity measurement.	**A8**	**P7**	-	-	**M5**
G8	Atomicity -interruption-free execution of integrity measurement.	**A6**	**P6**	-	-	**M4**
G9	Functional correctness of the integrity measurement.	**A5, A7**	**P4**	**GL1**	**PR5**	-
G10	Freshness of Attestation Response.	-	-	-	**PR1**	-
G11	Confidentiality of the channel between Target and Verifier.	**A1**	-	-	-	-
G12	Integrity of the channel between Target and Verifier.	-	-	-	-	-
G13	Availability of the channel between Target and Verifier.	**A2**	-	-	-	-
AS1	Assumption: integrity measurement code is functionally correct.	-	-	-	-	-
AS2	Assumption: protocol uses cryptographic primitives correctly.	-	-	-	-	-

Table 3 relates the TEE security functionalities as stated by GlobalPlatform in the Technology TEE protection profile [44]. We denote these functionalities as GP1-GP10. There are also undeveloped goals in the diagram, which represent the goals that are out of scope, either because they are not directly related or supported by the TEE or the TEE doesn't provide security functionality to the corresponding goal.

Notice that in our methodology we have not attempted to propose new goals based on a clean-slate analysis. Instead, we have studied the security properties described in the literature, synthesizing and matching them against each other when applicable. However, when structuring these goals using the GSN method, we have identified the need to complement the existing goals with additional ones that either are implicitly assumed in those works such as (G1), or are missing for the higher goal to be really satisfied if all subgoals are satisfied (e.g., G13 and G10). Example of implicit property in VRASED, freshness of attestation response using the challenge is achieved through the hash-based key derivation function (HKDF), however explicitly not fulfilled as in Table 2.

Table 3: TEE Security Functionalities as stated in GlobalPlatform [44] and their mapping to Table 2

ID	Functionality	Goal
GP1	TEE instantiation through secure process using assets bound to System on Chip (SoC) that ensures TEE firmware's authenticity, integrity and downgrade prevention.	G7
GP2	Isolation of the TEE services, the TEE resources and the Trusted Applications from the Regular Execution Environment (REE).	G5
GP3	Isolation between Trusted Applications (TAs) and TEE from TAs.	G5
GP4	Protected communication interface between Client Applications (CAs) in the REE and TAs in the TEE, including communication endpoints in the TEE.	-
GP5	Trusted storage of TA, TEE data and keys, ensuring consistency, confidentiality, atomicity and binding.	G4
GP6	Random Number Generation.	-
GP7	Cryptographic API, e.g. generation and derivation of keys and key pairs, hashing, symmetric and asymmetric operations.	G11, G12
GP8	TA instantiation that ensures the authenticity and consistency of the TA code.	G5
GP9	Monotonic TA instance time.	-
GP10	Correct execution of TAs.	G7, G9

5.2 Detailed Approach Description

Recall that the main goal of our GSN is to argue the assurance case for RA. Paying close attention to Figure 5 and Table 1, the initial RA goal G0 is to deliver acceptably secure RA. To achieve this goal, we started with strategy S1 (protection of RA entities and communication) at a higher level. Under this strategy, goals G1 and G2 were introduced. The former goal is to provide protection for all the entities (Verifier, Target and Attestation Agent), and the latter goal is to provide secure communication between the Verifier and the Target within the Prover. See Figure 1.

5.2.1 Strategy S2: Goal G1 is supported by two strategies. Strategy S2 (memory isolation) to enforce memory isolation, and strategy S3 (safe execution) to provide safe execution of the integrity measurement. The reasoning of these two strategies is that the isolation of the memory shall provide protection against the untrusted influence and that the execution of the entities shall be untampered, which leads to the next three sub-goals (G3, G4 and G5). Goal G3 states that the Target state shall be unmodified by the Attestation

Agent. Notice that this process is accomplished only by the Attestation Agent and the Target, and it is not provided by the TEE; as a result, we denote it as an undeveloped goal. Both goals G4 and G5 promise key protection to enforce access control and to prevent leakage of the key after execution, and ensure the integrity of the Attestation Agent, respectively.

5.2.2 Strategy S3: Goal G6 ensures that the Attestation Agent has read-only access to the full Target state. We denote it as an undeveloped goal since this functionality is not provided by the TEE. Goal G7 provides control invocation. As discussed earlier, the invocation of the integrity measurement should be executed from the very first instruction to the last. The integrity measurement should not be interrupted during the execution. This is not achieved by the TEE and is, therefore, marked as an undeveloped goal but achieved by the atomicity feature in the RA, and we denote it by goal G8. The functional correctness of the integrity measurement is achieved by goal G9. We assumed that G9 required additional support, especially with regards to GlobalPlatform, thus we added an assumption (AS1).

5.2.3 Strategy S4: The remaining four goals (G10, G11, G12 and G13) are sub-goals for providing secure communication through strategy S4 (protocol ensures secure communication). This RA protocol shall provide freshness of attestation response (G10) of the Prover from the request initiated by the Verifier, this is an undeveloped goal from the TEE perspective since it is mainly done by RA protocol. The protocol shall also provide confidentiality (G11), integrity (G12), and availability (G13) of the attestation channel between the Target and the Verifier. We also assumed that G11 and G12 required additional support, thus we added a second assumption (AS2) to clarify that the protocol implements cryptographic mechanisms correctly. Availability, however, is not directly guaranteed by the TEE; thus, we denoted G13 as an undeveloped goal.

5.2.4 GlobalPlatform: The developed goals from Table 2 are further mapped with the corresponding TEE security functionality provided by the GlobalPlatform, that has been described in Table 3. The key protection (G4) corresponds to GP5 through strategy S5 (secure storage provided by TEE), which provides several features, including key protection. The integrity of the Attestation Agent state (G5) is achieved by three TEE features from the GlobalPlatform (GP2, GP3 and GP8) through strategy S6 (isolation & instantiation enabled by TEE). The control invocation (G7) is instantiated with a secure process (GP1) and correct execution of trusted applications (GP10) through strategy S7 (proper execution of integrity measurement by TEE). Functional correctness (G9) corresponds to correct executions (GP10) through strategy S8 (TEE ensures proper execution of the integrity measurement). And finally, confidentiality (G11) and integrity (G12) through strategy S9 (using cryptographic functions in TEE) is guaranteed by several functionalities provided by GP7. Notice that the choice of the identifiers in Section 4 is according to the way they appear in their corresponding articles. Except for subsection 4.3, changed to GL1-GL4 due to their similarities with our synthesized goals G0-G13, subsection 4.4, named PR1-PR5, short for Five Principles of RA, and finally, subsection 4.5, shortened to M1-M5 as a minimalist approach.

5.3 Summary

In this section, we have analyzed the security properties discussed in five selected papers on RA from the literature. Using the GSN method, we relate these properties in a hierarchical diagram where the top goal states the overall security property, and for every layer downward, the properties are further refined. We found that similar properties are worded differently in the analyzed papers (e.g., atomicity vs uninterruptability) and that they tend to focus on only a subset of the goals we found in our analysis. Out of the 13 goals we listed, none was included (again explicitly) in all the papers. Moreover, we mapped the detailed (lowest layer) security goals to the TEE features listed in the GlobalPlatform standard. Out of the 10 TEE features, seven are needed to support RA, namely GP1-GP3, GP5, GP7, GP8 and GP10. These properties guarantee isolation, correct instantiation, trusted storage, cryptographic APIs and correct execution. The three remaining TEE features (protected communication between TEE and REE, RNG, and monotonic time) might also be useful to support some of the undeveloped goals in our analysis, but we have not found any direct dependency on these from the developed RA goals.

There were a number of low-level security goals required to support RA that we could not map to TEE features, thus requiring other mechanisms or solutions. Examples of such security goals are that the Attestation Agent has full (read-only) access to the target state (G6) and freshness of attestation response (G10). While it is not surprising that such features cannot be guaranteed by a TEE, we believe that it is important to highlight also these features as they are sometimes neglected when statements are made that certain TEE platforms provide RA, when in fact they typically only provide part of the solution needed.

6 CONCLUSION

This work presents an application of the GSN method to create security assurance cases of RA mechanisms, using TEE capabilities as evidence. We have used five different RA articles to gather information about relevant security goals and synthesize these into a unified security assurance analysis. Moreover, we have mapped how these security goals can be provided by TEE functionalities as specified by the GlobalPlatform protection profile. We identified seven specific TEE features (GP1-GP3, GP5, GP7-GP8, GP10) needed to support RA, thus answering the question we posed in the introduction. We found many differently worded, but similar security properties listed in the analyzed papers. Moreover, there were some discrepancies in which properties that were listed and which properties that were merely implicitly assumed (or explicitly mentioned, but not listed as a property). In addition to the directly excluded aspects such as mutual attestation, there are many aspects of RA that could be investigated in more detail. For future work, it would be relevant to expand the analysis of the undeveloped goals and how they can be supported by other forms of security mechanisms.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] The Assurance Case Working Group (ACWG). 2018. *Goal Structuring Notation Community Standard Version 2.* The Assurance Case Working Group (ACWG), Edinburgh, UK. https://scsc.uk/r141B:1?t=1 last visited: 2022-12-14.

[2] Jaehwan Ahn, Il-Gu Lee, and Myungchul Kim. 2020. Design and Implementation of Hardware-Based Remote Attestation for a Secure Internet of Things. *Wirel. Pers. Commun.* 114, 1 (sep 2020), 295–327. https://doi.org/10.1007/s11277-020-07364-5

[3] Mahmoud Ammar, Bruno Crispo, and Gene Tsudik. 2020. SIMPLE: A Remote Attestation Approach for Resource-constrained IoT devices. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS).* IEEE, Sydney, NSW, Australia, 247–258. https://doi.org/10.1109/ICCPS48487.2020.00036

[4] Krste Asanović and David A. Patterson. 2014. *Instruction Sets Should Be Free: The Case For RISC-V.* Technical Report UCB/EECS-2014-146. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html last visited: 2022-12-14.

[5] Marton Bognar, Jo Van Bulck, and Frank Piessens. 2022. Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures. In *2022 IEEE Symposium on Security and Privacy (SP).* IEEE, San Francisco, CA, USA, 1638–1655. https://doi.org/10.1109/SP46214.2022.9833735

[6] Ferdinand Brasser, Brahim El Mahjoub, Ahmad-Reza Sadeghi, Christian Wachsmann, and Patrick Koeberl. 2015. TyTAN: Tiny trust anchor for tiny devices. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC).* IEEE, San Francisco, CA, USA, 1–6. https://doi.org/10.1145/2744769.2744922

[7] Ernie Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security* (Washington DC, USA) *(CCS '04).* Association for Computing Machinery, New York, NY, USA, 132–145. https://doi.org/10.1145/1030083.1030103

[8] Guoxing Chen and Yinqian Zhang. 2022. MAGE: Mutual Attestation for a Group of Enclaves without Trusted Third Parties. In *31st USENIX Security Symposium (USENIX Security 22).* USENIX Association, Boston, MA, 4095–4110. https://www.usenix.org/conference/usenixsecurity22/presentation/chen-guoxing

[9] Guoxing Chen, Yinqian Zhang, and Ten-Hwang Lai. 2019. OPERA: Open Remote Attestation for Intel's Secure Enclaves. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19).* Association for Computing Machinery, New York, NY, USA, 2317–2331. https://doi.org/10.1145/3319535.3354220

[10] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of remote attestation. *International Journal of Information Security* 10, 2 (2011), 63–81. https://doi.org/10.1007/s10207-011-0124-7

[11] Mauro Conti, Edlira Dushku, and Luigi V. Mancini. 2019. RADIS: Remote Attestation of Distributed IoT Services. In *2019 Sixth International Conference on Software Defined Systems (SDS).* IEEE, Rome, Italy, 25–32. https://doi.org/10.1109/SDS.2019.8768670

[12] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. Cryptology ePrint Archive, Report 2016/086. https://ia.cr/2016/086.

[13] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *25th USENIX Security Symposium (USENIX Security 16).* USENIX Association, Austin, TX, 857–874. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[14] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *25th USENIX Security Symposium (USENIX Security 16).* USENIX Association, Austin, TX, 857–874. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[15] Avani Dave, Nilanjan Banerjee, and Chintan Patel. 2020. SRACARE: Secure Remote Attestation with Code Authentication and Resilience Engine. In *2020 IEEE International Conference on Embedded Software and Systems (ICESS).* IEEE, Shanghai, China, 1–8. https://doi.org/10.1109/ICESS49830.2020.9301516

[16] Advanced Micro Devices. 2019. Secure Encrypted Virtualization API: technical preview. Technical report. https://developer.amd.com/wp-content/resources/55766.PDF last visited: 2022-12-14.

[17] Aritra Dhar, Ivan Puddu, Kari Kostiainen, and Srdjan Capkun. 2020. *ProximiTEE: Hardened SGX Attestation by Proximity Verification.* Association for Computing Machinery, New York, NY, USA, 5–16. https://doi.org/10.1145/3374664.3375726

[18] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. 2012. Smart: secure and minimal architecture for (establishing dynamic) root of trust. In *Ndss*, Vol. 12. ISOC, San Diego, 1–15. https://www.ics.uci.edu/~gts/paps/smart.pdf

[19] Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, and Gene Tsudik. 2014. A minimalist approach to Remote Attestation. In *2014 Design, Automation and Test in Europe Conference and Exhibition (DATE).* IEEE, Dresden, Germany, 1–6. https://doi.org/10.7873/DATE.2014.257

[20] Y. He and C.W. Johnson. 2012. Generic security cases for information system security in healthcare systems. In *7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012.* 1–6. https://doi.org/10.1049/cp.2012.1507

[21] Celso Hirata and Simin Nadjm-Tehrani. 2019. Combining GSN and STPA for Safety Arguments. In *Computer Safety, Reliability, and Security*, Alexander Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 5–15. https://link.springer.com/chapter/10.1007/978-3-030-26250-1_1

[22] Tim Kelly and Rob Weaver. 2004. The goal structuring notation–a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases.* Citeseer, IEEE Computer Society, 1730 Massachusetts Ave., NW Washington DC, United States, 6.

[23] Timothy Patrick Kelly. 1999. *Arguing safety: a systematic approach to managing safety cases.* Ph. D. Dissertation. University of York York, UK. https://www-users.cs.york.ac.uk/~tpk/tpkthesis.pdf

[24] Patrick Koeberl, Vinay Phegade, Anand Rajan, Thomas Schneider, Steffen Schulz, and Maria Zhdanova. 2015. Time to Rethink: Trust Brokerage Using Trusted Execution Environments. In *Trust and Trustworthy Computing*, Mauro Conti, Matthias Schunter, and Ioannis Askoxylakis (Eds.). Springer International Publishing, Cham, 181–190. https://doi.org/10.1007/978-3-319-22846-4_11

[25] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. 2014. TrustLite: A Security Architecture for Tiny Embedded Devices. In *Proceedings of the Ninth European Conference on Computer Systems* (Amsterdam, The Netherlands) *(EuroSys '14).* Association for Computing Machinery, New York, NY, USA, Article 10, 14 pages. https://doi.org/10.1145/2592798.2592824

[26] Florian Kohnhäuser, Niklas Büscher, and Stefan Katzenbeisser. 2019. A Practical Attestation Protocol for Autonomous Embedded Systems. In *2019 IEEE European Symposium on Security and Privacy (EuroS and P).* IEEE, Stockholm, Sweden, 263–278. 10.1109/EuroSP.2019.00028

[27] Seema Kumar, Patrick Eugster, and Silvia Santini. 2021. Software-based Remote Network Attestation. *IEEE Transactions on Dependable and Secure Computing* 19, 5 (2021), 1–1. https://doi.org/10.1109/TDSC.2021.3077993

[28] David Lantz, Felipe Boeira, and Mikael Asplund. 2022. Towards Self-monitoring Enclaves: Side-Channel Detection Using Performance Counters. In *Secure IT Systems*, Hans P. Reiser and Marcel Kyas (Eds.). Springer International Publishing, Cham, 120–138. https://doi.org/10.1007/978-3-031-22295-5_7

[29] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20).* Association for Computing Machinery, New York, NY, USA, Article 38, 16 pages. https://doi.org/10.1145/3342195.3387532

[30] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. 2015. AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15).* Association for Computing Machinery, New York, NY, USA, 75–88. https://doi.org/10.1145/2742647.2742676

[31] Bao Liu and Brandon Wang. 2014. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *2014 Design, Automation and Test in Europe Conference and Exhibition (DATE).* IEEE, Dresden, Germany, 1–6. https://doi.org/10.7873/DATE.2014.256

[32] Pieter Maene, Johannes Götzfried, Ruan de Clercq, Tilo Müller, Felix Freiling, and Ingrid Verbauwhede. 2018. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Comput.* 67, 3 (2018), 361–374. https://doi.org/10.1109/TC.2017.2647955

[33] Yutaka Matsuno. 2017. D-Case Communicator: A Web Based GSN Editor for Multiple Stakeholders. In *Computer Safety, Reliability, and Security*, Stefano Tonetta, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 64–69. http://dx.doi.org/10.1007/978-3-319-66284-8_6

[34] Yutaka Matsuno and Shuichiro Yamamoto. 2013. An implementation of GSN community standard. In *2013 1st International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE).* IEEE, San Francisco, CA, USA, 24–28. https://doi.org/10.1109/ASSURE.2013.6614267

[35] Jämes Ménétrey, Christian Göttel, Anum Khurshid, Marcelo Pasin, Pascal Felber, Valerio Schiavoni, and Shahid Raza. 2022. Attestation Mechanisms for Trusted Execution Environments Demystified. In *Distributed Applications and Interoperable Systems*, David Eyers and Spyros Voulgaris (Eds.). Springer International Publishing, Cham, 95–113.

[36] Bernard Ngabonziza, Daniel Martin, Anna Bailey, Haehyun Cho, and Sarah Martin. 2016. TrustZone Explained: Architectural Features and Use Cases. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC).* 445–451. https://doi.org/10.1109/CIC.2016.065

[37] Job Noorman, Jo Van Bulck, Jan Tobias Mühlberg, Frank Piessens, Pieter Maene, Bart Preneel, Ingrid Verbauwhede, Johannes Götzfried, Tilo Müller, and Felix Freiling. 2017. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Trans. Priv. Secur.* 20, 3, Article 7 (jul 2017), 33 pages. https://doi.org/10.1145/3079763

[38] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, Michael Steiner, and Gene Tsudik. 2019. VRASED: A Verified Hardware/Software Co-Design for Remote Attestation. In *28th USENIX Security Symposium (USENIX Security 19).* USENIX Association, Santa Clara, CA, 1429–1446. https://www.

usenix.org/conference/usenixsecurity19/presentation/de-oliveira-nunes

[39] Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* 51, 6, Article 130 (jan 2019), 36 pages. https://doi.org/10.1145/3291047

[40] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. 2017. Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments. In *Proceedings of the 12th International Conference on Availability, Reliability and Security* (Reggio Calabria, Italy) *(ARES '17)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/3098954.3098971

[41] Carlton Shepherd, Ghada Arfaoui, Iakovos Gurulian, Robert P. Lee, Konstantinos Markantonakis, Raja Naeem Akram, Damien Sauveron, and Emmanuel Conchon. 2016. Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems. In *2016 IEEE Trustcom/BigDataSE/ISPA*. 168–177. https://doi.org/10.1109/TrustCom.2016.0060

[42] Carlton Shepherd, Konstantinos Markantonakis, and Georges-Axel Jaloyan. 2021. LIRA-V: Lightweight Remote Attestation for Constrained RISC-V Devices. arXiv:2102.08804 [cs.CR]

[43] Rodrigo Vieira Steiner and Emil Lupu. 2016. Attestation in Wireless Sensor Networks: A Survey. *ACM Comput. Surv.* 49, 3, Article 51 (sep 2016), 31 pages. https://doi.org/10.1145/2988546

[44] GlobalPlatform: the Standard for Secure Digital Services and Devices. 2020. *GlobalPlatform Technology TEE Protection Profile Version 1.3*. GlobalPlatform, the Standard for Secure Digital Services and Devices, California, United State. https://globalplatform.org/specs-library/?filter-committee=tee 2022-12-14.

[45] Samuel Weiser, Mario Werner, Ferdinand Brasser, Maja Malenko, Stefan Mangard, and Ahmad-Reza Sadeghi. 2019. TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V. In *19th Network and Distributed System Security Symposium*. NDSS, San Diego, USA. https://doi.org/10.14722/ndss.2019.23068

[46] Shuichiro Yamamoto. 2015. Assuring Security through Attribute GSN. In *2015 5th International Conference on IT Convergence and Security (ICITCS)*. 1–5. https://doi.org/10.1109/ICITCS.2015.7292954

[47] Shijun Zhao, Qianying Zhang, Yu Qin, Wei Feng, and Dengguo Feng. 2019. SecTEE: A Software-Based Approach to Secure Enclave Architecture Using TEE. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, New York, NY, USA, 1723–1740. https://doi.org/10.1145/3319535.3363205

[48] Zhengshu Zhou, Yutaka Matsubara, and Hiroaki Takada. 2021. Quantitative Security Assurance Case for In-vehicle Embedded Systems. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 43–50. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00022