sensors

MDPI

Article

RESEKRA: Remote Enrollment Using SEaled Keys for Remote Attestation

Ernesto Gómez-Marín [1,2,*], Luis Parrilla [2], Gianfranco Mauro [1,2], Antonio Escobar-Molero [1], Diego P. Morales [2] and Encarnación Castillo [2]

[1] Infineon Technologies AG, 85579 Neubiberg, Germany; gianfranco.mauro@infineon.com (G.M.); antonio.escobar@infineon.com (A.E.-M.)
[2] Departamento Electrónica y Tecnología de Computadores, Universidad de Granada, 18071 Granada, Spain; luis@ugr.es (L.P.); diegopm@ugr.es (D.P.M.); encas@ugr.es (E.C.)
* Correspondence: ernesto.gomezmarin@infineon.com

Abstract: This paper presents and implements a novel remote attestation method to ensure the integrity of a device applicable to decentralized infrastructures, such as those found in common edge computing scenarios. Edge computing can be considered as a framework where multiple unsupervised devices communicate with each other with lack of hierarchy, requesting and offering services without a central server to orchestrate them. Because of these characteristics, there are many security threats, and detecting attacks is essential. Many remote attestation systems have been developed to alleviate this problem, but none of them can satisfy the requirements of edge computing: accepting dynamic enrollment and removal of devices to the system, respecting the interrupted activity of devices, and last but not least, providing a decentralized architecture for not trusting in just one Verifier. This security flaw has a negative impact on the development and implementation of edge computing-based technologies because of the impossibility of secure implementation. In this work, we propose a remote attestation system that, through using a Trusted Platform Module (TPM), enables the dynamic enrollment and an efficient and decentralized attestation. We demonstrate and evaluate our work in two use cases, attaining acceptance of intermittent activity by IoT devices, deletion of the dependency of centralized verifiers, and the probation of continuous integrity between unknown devices just by one signature verification.

Keywords: remote attestation; edge computing; Internet of Things; embedded systems; Trusted Platform Module

Citation: Gomez-Marin, E.; Parrilla, L.; Mauro, G.; Escobar-Molero, A.; Morales, D.P.; Castillo, E. RESEKRA: Remote Enrollment Using SEaled Keys for Remote Attestation. *Sensors* **2022**, *22*, 5060. https://doi.org/10.3390/s22135060

Academic Editor: Juan M. Corchado

Received: 31 May 2022
Accepted: 2 July 2022
Published: 5 July 2022

1. Introduction

The number of things connected to the internet (Internet of Things, IoT) is growing exponentially, from 6.1 billion in 2018 to 14.7 billion in 2023 [1]. This growth is continuous, and there is no evidence that it will stop.

The current IoT structure relies on cloud computing. An IoT device collects information from the environment (sensor), sends it to a distant centralized server, which processes it along with currently stored information and information from other sensors, and generates, whether required, an action response. The response is then sent to a new IoT device, namely an actuator, which performs the appropriate action often in the nearby of the sensor. This is a simple and secure system from the service provider's point of view, but due to the remoteness of the server from the nodes, it generates a high volume of data traffic and a high latency not suitable for real-time IoT applications. Moreover, it is not scalable enough with the current growth of IoT devices because of the large network use and the burden on cloud servers [2,3].

In this context, it is convenient for data to be processed at the edge for shorter response times, more efficient processing, and less pressure on the network. The technologies that enable computation at the edge of the network, on the data flow between the cloud services

and the IoT services are known as edge computing [2]. In this scenario, the devices at the edge of the network that provide services (processing and/or data storage) are called edge nodes, while user devices are those that require real-time interaction or high storage capabilities. In this paradigm, user devices will act as edge nodes whenever possible, offloading their computing task to adjacent nodes if too burdensome at a given time [3].

On the other hand, the inherent distributed structure of edge computation implies additional security challenges over cloud computing. Egde computing has to protect different layer of technologies (from cloud to IoT devices) as in cloud computing, but it also needs to provide a distributed global connectivity of heterogeneous devices between the different layers. Rodrigo Roman et al. [4] define it as a combination of "the worst-of-all-worlds". The scenario becomes more complicated when we consider that user devices continuously enter and leave the local network, that means, close mobile users (mobile subscribers), e.g., mobile phones or cars, which is called Mobile Edge Computing (MEC) [5].

In addition, it should be considered that the possible impact of an attack can be very high. Given the wide variety of situations in which edge computing can be used, the consequences can range from affecting our private information, the daily lives of users, and, more indirectly, the industrial ecosystem or critical infrastructure. It is therefore easy to override the possible benefits of edge computing by the losses that can result from relying too much on such technology [4]. Most edge devices do not have a user interface, which causes attacks to go unnoticed by most users [6]. Moreover, one of the main currently open problems in MEC is the security, in particular, the robustness of MEC servers [7]. Hence, verifying the status of IoT devices, edge nodes or not, becomes a complex and critical task in edge computing. This leads to the need of implementing a remote attestation system for edge devices and users [8].

In remote attestation, a Verifier checks the correct status of a device (Attestor) remotely. This solution has been widely investigated in the literature, but when applied to IoT systems, the current RA protocols are hard to scale. To overcome this challenge, several works recently proposed Collective RA (CRA) protocols. However, these solutions brought up new open issues [8]: the need of scalables and decentralized key management allowing mobility, the acceptance of intermittent activity of IoT nodes, which is essential for edge computing, and the resistance against the attack Time Of Check To Time Of Use (TOCTTOU) [9], which is barely covered in the State of the Art (SoA). In this paper, we present a solution for these problems through our remote attestation method, RESEKRA.

In our proposal, we attest remotely the state of an untrusted device through the support of a standard Root of Trust (RoT) consisting of a Trusted Platform Module [10]. This is a work already achieved and applied to edge computing in [11–13], but we go further.

Additionally to the results obtained in [11–13], we provide the following features:

- Proving authenticity in any communication also proves its correctness.
- No pre-shared secret is needed between the Verifier and the Attestor, which makes the system able to easily include new devices which is essential for systems with a variable number of nodes such as edge computing.
- Allowing secure software updates on the Attestor.
- Enabling offline remote attestation. The Attestor can be verified by the end user themselves.

For acheiving these features, we assume that:

- The Attestor shall include the needed hardware root of trust (RoT).
- A trusted Attestor manufacturer is available.
- The attacker cannot modify the RoT.

Our system, called RESEKRA, has been implemented on a Rapberry Pi by connecting a TPM and a pressure sensor to achieve a secure sensor acting as an Attestor that connects remotely to an external computer acting as a Verifier. The system attains an online and dynamic enrollment phase that works even on untrusted state devices, which allows it to be used in decentralized key management, where multiples entities are responsible for

the key generation, distribution and regeneration with the advantage of fault tolerance, scalability [14] and flexibility. This makes RESEKRA the first remote attestation system for large networks of IoT devices with decentralized key management [8]. Moreover, Attestor can prove correct software status directly to the user devices, which make it perfect for users that just entered in the edge network, that means, mobile subscribers in MEC. Additionally, our scheme allows asynchronous booting, where the programs of the IoT devices are initialized in a random order, which is required for Linux-based systems, and finally, RESEKRA does not interfere with updates; therefore, a trusted online software updated system could be easily included in our system. To the best of our knowledge, there is no other system able to provide these services.

The rest of the manuscript is structured as follows: Section 2 is devoted to the related work, while Section 3 presents the technologies needed to understand the system. Section 4 details the RESEKRA design, whose implementation is presented in Section 5 as a solution for real scenarios. In Section 6, we will analyze the effects of the common attacks to the remote attestation systems in RESEKRA, and in Section 7, the conclusions of the work are carried out.

2. Related Work

Remote attestation is a process that has been studied extensively for many years and has recently focused on IoT nodes, creating their own field, Collaborative Remote Attestation (CRA) [8]. Due to the heterogeneity of IoT devices, there are several research studies of software-based solutions to increase scalability such as [15–17], but all require setting up one-hop networks between the Verifier and Attestor. This limitations make their implementation challenging. In addition, software-based solutions make strong assumptions about the limits of attacker capabilities, thus reducing their reliability [8]. To solve the latter problems, the use of a Root of Trust (RoT) residing in the device has been proposed in the literature [18]. This RoT is usually a mix of hardware and software. In [8], the authors divide RoT used in CRA into two main categories: those using hardware with the minimal security capabilities and those using a TPM.

In the branch of RoT based on hardware with the minimal security capabilities—also known as hybrid attestation techniques [19,20]—the authors specify their non-standard security hardware designs. SMART [21], ERASMUS [22], TrustLite [23] and SEED [20] provide high-level details of their hardware solutions. SEED and ERASMUS are the only solutions also requiring a Real-Time Clock, which is used to make the attestation to start from the Attestor itself. LISA [24] requires the same hardware architecture as SMART, but additional hardware is added to authenticate the Verifier and avoid DoS attacks. All the solutions mentioned above require the hardware design and manufacturing of novel security hardware, making them hard to implement. Another solution of the SoA is Hatt [19] where very limited additional hardware is required: "Physical Unclonable Functions" (PUF) and a ROM for the attention code. Still, they do not explain how to protect the PUF from being accessed by a malicious code. Finally, the research SARA [25] defines the requirements of their root of trust without providing any particular details. From all the presented works, only ERASMUS considers TOCTTOU attacks in their respective adversarial model.

On the other hand, other solutions choose security hardware with the highest security standards such as Trusted Plataform Module (TPM). In "Remote Enrollment using SEaled Keys for Remote Attestation" (RESEKRA), we opted for the use of TPM. As Tan et al. argue in "TPM-enabled Remote Attestation Protocol" (TRAP) [13], the cost of current TPMs is relatively low compared to the price of sensors, both in price and area. And as final product that can be found in the market, they can be easly implemented in real applications. As in the work proposed by Miguel Calvo and Marta Beltran [11] and MTRA (TRAP-based system) [12], we propose the use of a TPM where the Platform Configuration Registers (PCR) values, further discussed in Section 3.1.8, are used as proof of the current state of the IoT device. MTRA also considers TOCTTOU in their adversary analysis.

RESEKRA differs from all of the above works, whether based on hardware with the minimal security capabilities or TPM-based, in that none of them have any of the following features: enabling dynamic and online enrollment, which is essential for edge computing and even more for Mobile Edge Computing; decentralized key management and decentralized Verifier structure, removing single points of failure and trusted third parties; and correct analysis of devices with intermittent activity, which is critical in IoT computing because IoT devices are not always online. Additionally, we also consider TOCTTOU attacks, which are only covered in Erasmus and MTRA. All of them are open issues not addressed in the current Collaborative Remote Attestation solutions of the SoA [8].

3. Background

This section will outline the tools and technologies necessary for understanding RESEKRA: TPM2.0, Integrity Measurement Architecture (IMA) [26] and Core Root of Trust of Measurement (CRTM) [27].

These three elements are part of the Attestor. The CRTM ensures the correctness of firmware configuration and IMA. The IMA guarantees the correctness of the software configuration at runtime. Finally, the TPM2.0 ensures the reliability of all the information when shared externally (results generated by IMA and CRTM, among others). In Nomenclature part, we will summarize all the notations used in the document.

3.1. TPM 2.0

The Trust Platform Module 2.0 [10] provides standardized specifications for security coprocessors. Hereafter, security coprocessors that follow these specifications will be called TPM, and the device that has the TPM will be called the Host-TPM. These specifications have many details and functionalities, and only the features needed for our system will be described below.

3.1.1. Virtual Memory

The TPM has a limited protected "real" memory, but credentials and sensitive data can be stored in the non-protected memory of the Host-TPM (computer that is using the TPM). This material is protected through signatures and encryption to ensure integrity and confidentiality. With this system, the TPM can use the Host-TPM device's memory as a virtual protected memory of the TPM.

3.1.2. Key Attribute: "Restricted"

The keys generated by the TPM have attributes that cannot be modified and which limit the use of these keys. Many of them must be verified to guarantee the security of the system. One of these essential attributes is Restricted. Keys with this attribute sign only TPM-generated digests in the signing process and will never sign a document starting with the value "$0xff544347$", which is also called TPM_GENERATED.

3.1.3. TPM_GENERATED

Data that begin with the code TPM_GENERATED can be signed by a Restricted key if and only if it was generated by the TPM. If the authenticity of the Restricted key is guaranteed, so is the veracity of the information related to the TPM. When used correctly, this allows much information to be remotely and reliably derived from the TPM in addition to the PCR values, such as the characteristics of the private keys stored in the TPM. This is a functionality that is rarely used in the state of the art and is the first reason why we stand out. With it, we can remotely complete the entire enrollment process of the IoT device to be attested.

3.1.4. Policies

TPM2.0 offers a large set of policies that are not commonly used in typical applications. To the best of our knowledge, only [13] uses these available policies when sealing secret

keys with PCR values. The use of these policies is one of the contributions carried out in RESEKRA. Only if the policies under which the TPM objects were created are satisfied, the object can be unsealed and then used. The three policies used in this work are the following:

- tpm2_policypcr: Seals the object to the value of one or more PCRs of the TPM.
- tpm2_policycountertimer: Seals the object to the number of times the TPM has been restarted.
- tpm2_policyauthorize: Seals the object to the policies signed by a Trusted Third Party (TTP). This policy allows changing the policies of a remotely signed object. With this policy, in RESEKRA, the Verifier is able to remotely update the other two policies, thus enabling flexible updating of the correct PCR value, and consequently, asynchronous booting and remote updates performing.

The name of an object (Object_name) is computed from the public information of the object (Object_Pubdata) as the public key (Object_PuB), the attributes, and policies. As a consequence, the Object_name can be used to assert the integrity of the object's policies.

3.1.5. Attestation Key

Attestation key (AK) is used to verify the internal information of the TPM. As mentioned above, one of its most important attributes is "Restricted". The key name (AK_name) is computed from the public information of the attestation key (AK_Pubdata), as the public key (AK_PuB), attributes, and policies.

3.1.6. Endorsement Key

The Endorsement Key (EK) is an assymetric key stored in the real memory of the TPM, which comes with a certificate signed from the TPM Manufacturer. The EK together with the Endorsement Key certificate (EK certificate) are used for two purposes: to verify the existence of the TPM and its manufacturer and to verify that an object is stored in the TPM. If an object, such as a private key, is shown to be stored in the TPM, its characteristics and attributes can be trusted as long as the TPM is trusted, such as the "Restricted" attribute.

It is a very restricted key and can only be used to decrypt data with a specific structure. Therefore, the TPM can prove ownership of the EK just through decryption operations [28]. For this task, we use the operations "makecredential/activatecredential".

3.1.7. Makecredential/Activatecredential

An external entity performs the *makecredential* operation using as inputs the Public Endorsement Key (*EK_PuB*), the name of a TPM object, usually the name of an attestation key (*AK_name)*, and a secret value:

$$Credential = makecredential(EK\_PuB, AK\_name, secret) \quad (1)$$

This command creates the so-called *Credential*. The TPM receives it and computes *activecredential* and, if and only if the AK_name object belongs to the TPM, it will decrypt the Credential by retrieving the secret. By showing knowledge of the secret, the Host-TPM proves that it has a TPM with EK and that the object AK_name is real and accurate.

3.1.8. Platform Configuration Registers

The Platform Configuration Registers (*PCR*) are the basis for TPM-based remote attestation, which are also called Trusted Attestation Protocol (TAP) [29]. These registers can be updated through the "Extension" operation only [30]:

$$PCR\ new\ value\ =\ hash(PCR\ old\ value\ ||\ data\ to\ extend) \quad (2)$$

"Because of the one-way nature of a secure digest, there is no way to undo a measurement" [30]. Therefore, once a measurement has been stored in the PCR, no one, even

with the highest privileges can override the effect of this measurement in the PCR, making PCRs perfect for verifying the integrity of the measurement list generated by the Integrity Measurement Architecture (IMA).

3.2. Integrity Measurement Architecture (IMA)

The Integrity Measurement Architecture (IMA) is a tool available in Linux responsible for measuring the files before they are accessed, storing the measurements in a list and extending them to the PCR. The IMA keeps a runtime measurement list; therefore, if any new or edited file is accessed, the path, name and content are measured, as shown in Equations (3) and (4), and they are included in the measurement list with the corresponding extension to the PCR.

$$filedata\ hash\ =\ hash(filedata) \tag{3}$$

$$measurement\ =\ hash(filedata\ hash, file\ path|name) \tag{4}$$

Any program or file that would modify the system architecture such as the IMA itself would be measured before being executed, and it would leave an indelible mark on the PCR. Providing the measurement list together with the PCR will ensure the integrity of the measurement list. Therefore, for trusting in this attestation, it is needed to trust in the first program that started the measuring process and thus was never measured: the Core Root of Trust of Measurement.

3.3. Core Root of Trust of Measurement

As pointed out in [27], the Core Root of Trust of Measurement (CRTM) is the "first piece of BIOS code that executes on the main processor during the boot process. On a system with a Trusted Platform Module the CRTM (Core Root of Trust of Measurement) is implicitly trusted to bootstrap the process of building a measurement chain for subsequent attestation of other firmware and software that is executed on the computer system". The CRTM is essential to being able to trust in IMA. The only way to trust in a CTRM is by trusting in the device's manufacturer.

4. RESEKRA Description

As will be detailed in this section, the main contributions of RESEKRA can be summarized as follows:

- The use of Sealed Keys that can be used by the proprietary (the Attestor) temporarily and only when the current device status is approved by the Verifier, thus proving correct Attestor status only by proving ownership of the Sealed Key.
- The same Sealed Key can be used with different device status (always approved beforehand by the Verifier), allowing software updates without continuous revocations and avoiding a complex PKI.
- The creation of the Sealed Key can be realized completely remotely in a untrusted Attestor, allowing great flexibility and a plug-and-play business model perfect for Edge computing.

Figure 1 shows the general scheme of the remote enrollment process in RESEKRA. In the next subsections, the roles and operations performed in each of the steps will be detailed.

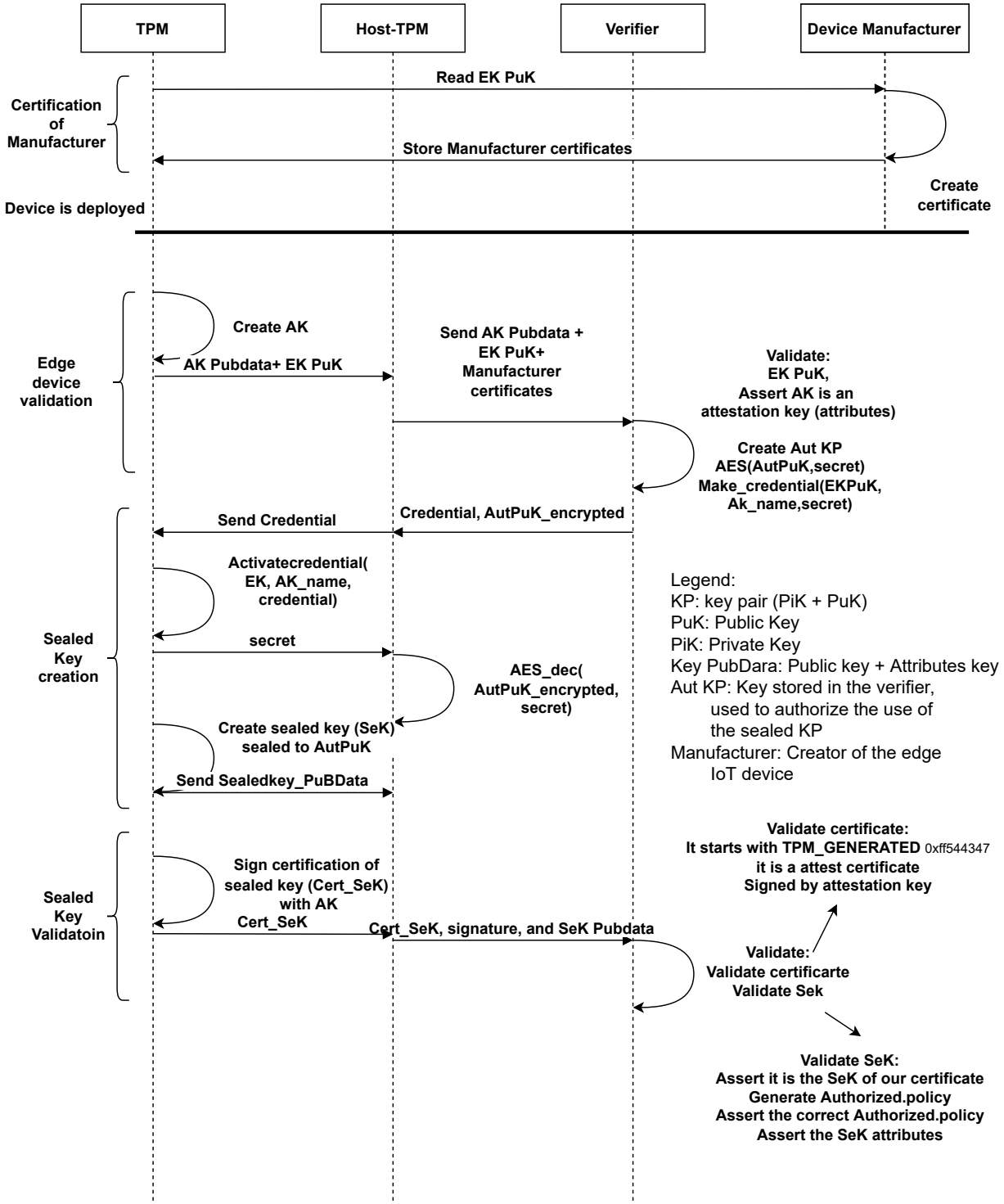

Figure 1. Remote enrollment process. The scheme represents all the enrollment process from the device manufacturer to the supervised creation of the sealed key.

4.1. Roles

The main elements included in RESEKRA are the following:

- Attestor: edge devices with the hardware RoT, i.e., TPM and CRTM.
- Hardware Provider: the entity in charge of manufacturing the Attestor and providing the firmware and RoT. It is considered a trusted entity.
- Software Provider: the entity in charge of designing the Attestor software and creating the RML. It is a trusted entity.
- Programmer: the entity in charge of programming the Attestor. It is a not trusted entity.
- Deployer: the entity responsible for physically deploying the device and providing an authentication method. This entity is semitrusted, because it does not need a high level of knowledge to securely deploy the device, since there is no offline enrollment process and it has no interest in hacking the system because it is one of the stakeholders interested in providing the service. On the other hand, it would require a high level of expertise to manipulate physically the root of trust and go unnoticed by the rest of stakeholders.
- Verifier: the entity in charge of verifying the root of trusts of the Attestor, creating the SeK, verifying the Attestor status and authorizing the use of the SeK in the TPM.
- TPM: Hardware security module following the TPM 2.0 specifications.
- Edge computing user: the entity that requires communication with the Attestor to receive data (sensor) or to analyze/store data (Edge node).

In this paper, we do not focus on authentication. We consider that the Deployer is responsible for providing an authentication method for the Attestor, edge computing user and Verifier.

4.2. Certification of Manufacturer

In this first phase, the Attestor manufacturer and programmer shall create the needed certificates.

4.2.1. Hardware Provider

When the Attestor is fabricated, the Device Manufacturer stores the necessary certificates (Manufacturer Certificates) in the device to identify the existence of the necessary root of trust (TPM and CRTM) on the Attestor. This certificate is signed by the Hardware Provider and linked to the EK. The Verifier can trust several Device Manufacturers.

4.2.2. Software Provider

The software provider installs all the needed software in the Attestor and generates a list of measurements that will be used as reference, the Reference Measurements List (RML). This list is then signed by the software provider. The RML has to be provided to the Verifier and updated in case of software update. We will provide two possible solutions in Section 5. The software provider can be any stakeholder, e.g., Hardware provider, deployer or end user.

4.3. Remote Enrollment

At this moment in the process, the device can be deployed by the owner in any considered location. Because the rest of RESEKRA will work remotely, the deployment can be realized by inexperienced staff. Once the device is deployed, the Verifier has to assert the device's hardware Root of Trust and to manage the creation and the validation of some special keys in the Attestor. It is a completely remote process and can be realized even in a untrusted Attestor. Therefore, the Verifier's role can be dynamically changed, and even multiple Verifiers can work at the same time, sharing trust and responsibility.

4.3.1. Edge Device Validation

Once the IoT device has been deployed, the Host_TPM requests the TPM for the public Endorsement Key (EK_PuK) and orders the creation of the Attestation Key (AK). It sends the public information of Attestation Key (AK_Pubdata) and the public Endorsement Key (EK_PuK) to the Verifier. The Verifier checks that the Endorsement Key belongs to a

genuine TPM from a trusted manufactured edge device. The check is done through the Attestor's Manufacturer Certificates. Then, it verifies the attributes of AK (the Restricted attribute, among others).

4.3.2. Sealed Key Creation

The Verifier knows that the EK_PuK belongs to a trusted manufactured edge device but has not yet asserted that the Attestor is the owner of this EK. Then, the Verifier creates the asymmetric key pair, Authorizer, and encrypts the public key (Aut_PuK) with symmetric encryption using "Secret" as the symmetric key.

Next, it encrypts "Secret" using Makecredential and using EK_PuK and AK_Pubdata as input to create Credential. If the Host-TPM shows knowledge of Aut_PuK, it proves to own the TPM, EK, and AK with the asserted attributes. Finally, the Verifier sends Credential and Aut_PuK encrypted to Host-TPM. If Host-TPM shows knowledge of Aut_PuK, it proves the ownership of EK.

When Credential reaches Host-TPM, it decrypts the Credential with Activatecredential obtaining "Secret" and uses it to decrypt Aut_PuK. Now, because it has Aut_PuK, it can instruct TPM to create a sealed key (SeK) with the "tpm2_policyauthorize" using Aut_PuK as the authorizing key. The output of running the sealed key creation script is shown in Figure 2.

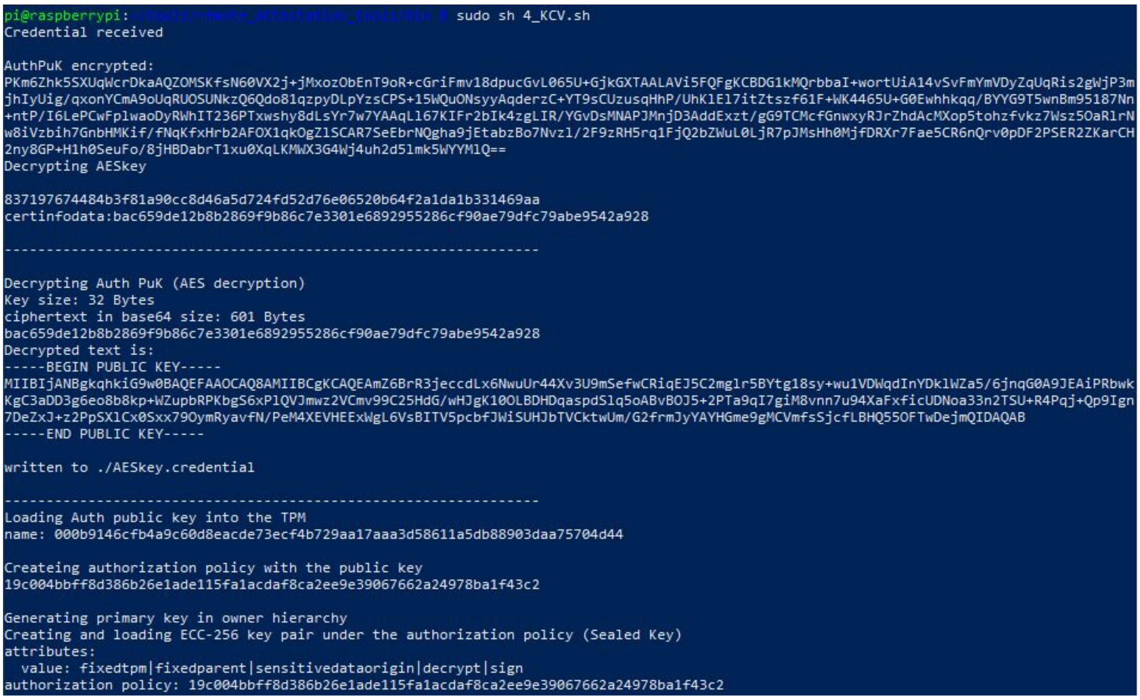

Figure 2. Output of the script for sealed key creation.

4.3.3. Sealed Key Validation

The TPM generates an attestation certificate for SeK (Cert_SeK). This certificate is asserting that the object with the name SeK_name is stored in the TPM; hence, the TPM is bearing it. Then, the TPM signs it with the AK and sends it to the Verifier together with SeK_Pubdata.

The Verifier has to validate this information once received. First, the Verifier checks that the certificate starts with TPM_GENERATED and is signed with the suspected AK. This means that the TPM has generated the certificate if AK is a real AK, which will be verified in the next step. Finally, the Verifier needs to assert that this is the correct type of certificate.

- It starts with TPM_GENERATED;

- It is signed by AK;
- It is an attest certificate.

This validation means that Cert_SeK is genuine. Now, the Verifier will recompute the SeK_name locally from SeK_Pubdata. It shall match with the name backed by the Cert_SeKcheck, and finally, the Verifier will verify all the features of SeK:

- The sealed key is the same being attested by the certificate;
- The policy used to create the sealed key is correct (tpm2_policyauthorize using Aut_Pub). This point is essential, because it proves knowledge of the Aut_Pub;
- Verifying the attributes of the SeK.

If all verifications are successful, the Verifier has been able to create completely remotely, without the need for pre-shared secret, a sealed key in the Host-TPM's TPM. The Verifier would have complete control over this sealed key. The Attestor will be able to use this key only when satisfying the signed requirements imposed by the Verifier.

4.4. Attestation

At this point, the CRTM measures the BIOS and the IMA measures the integrity of all files executed. This is a runtime measurement, so when any new or modified file is executed, it is measured, and the result is stored in the measurement list and extended to the PCR.

The Attestor initiates the remote attestation by sending a request to the Verifier, and the last one replies by sending back a random nonce to the Attestor. The TPM generates a Quote signed by the AK, which includes the nonce to avoid replay attacks. The Quote is a certificate including several important parameters; those most relevant for RESEKRA are:

- TPM_GENERATED, to confirm the veracity when signed by the AK;
- The random nonce, to avoid replay attacks;
- Reset value of TPM. Value that changes when Host-TPM is rebooted;
- Value of PCR.

The Attestor sends the Quote together with the measurement list to the Verifier. It asserts the veracity of the certificate (TPM_GENERATED, signature, and random nonce) and uses the measurement list to rebuild the PCR value found in Quote. If the reconstruction is correct, it means that the measurement list was not adulterated.

Finally, the Verifier compares the measurement list with the Reference Measurement List (RML). The Verifier may have obtained this RML through the Host-TPM signed by the trusted Device Manufacturer or from the cloud. Now, the Verifier can check program by program if there is a difference and where.

When the list is approved by the Verifier, the Verifier can create an authorization to use the SeK for the particular values of RESET and PCR specified in Quote, and it sends the authorization signed by AuTK to the Host-TPM. Figure 3 shows the particular code to generate and sign the combined policy. The complete codes are accessible at: [31].

```
TPM_policies.Policypcr_creation(Hex.decode(computedPcrSha256));

TPM_policies.Policyreset_creation(resetCount);

String authorization_signature = RSAk.sign_byte(TPM_policies.Last_policy);
```

Figure 3. Section of the Verifier code where it computes the policies for using the SeK and signs it if and only if the Attestor passed through the attestation process.

Once the authorization is received, the Host-TPM sends it to the TPM to unlock the SeK. The TPM verifies the signature and checks that it meets the requirements of the authorization (RESET and PCR values), being able to use the SeK until these requirements change, i.e., until it reboots or until a new or modified program is used. If the PCR value

changes, the Attestor will request a new authorization from the Verifier; however, if a program that was not supposed to be executed is run, the Attestor will not obtain a new authorization until it is restarted and ask for a new one. A device will not be able to reuse an old authorization because those are sealed to the RESET value of the TPM, which prevents the use of authorizations for old decommissioned software.

4.5. Daily Life Functionality

Whenever the egde device proves ownership of the SeK, the TPM will first verify the signature of the policy provided by the server, secondly, it will assert the signed polices (values of PCR and Reset), and finally, it will grant access to sign with the SeK, proving the correctness of the software, and therefore, no additional attestation is needed. Figure 4 shows the script using the SeK to sign the measure of a pressure sensor satisfying the policy that was set when the SeK was generated in Section 4.3.2 (authorization policy in Figure 2).

```
pi@raspberrypi:~/tools/remote_attestation_tools/bin # sudo sh test_SeK.sh
name: 000b9146cfb4a9c60d8eacde73ecf4b729aa17aaa3d58611a5db88903daa75704d44
c83a71dba04be92b2bb3a0b47c437ef3540ffd193d7236ad30a9168146ded143
5291ea4bb3ae260837aa280477b84ecd9410e8bb520195f306a87c790f3ecba7
verifying signature
Satisfying policies
c83a71dba04be92b2bb3a0b47c437ef3540ffd193d7236ad30a9168146ded143
5291ea4bb3ae260837aa280477b84ecd9410e8bb520195f306a87c790f3ecba7
policies satisfied
19c004bbff8d386b26e1ade115fa1acdaf8ca2ee9e39067662a24978ba1f43c2
```

Figure 4. Output of the script using the sealed key to sign a measured value after satisfying the policies.

This key can also be used to prove authentication. In protocols such as the well-known cryptography protocol Transport Layer Security (TLS) [32], or signing transactions for blockchain, asymmetric keys are used for the authentication, thus; the Sealed key pair can be use to prove correctness at the same time that authentication. The end user can trust in the SeK because it is coming with a certificate from a trusted third party, which is commonly the Verifier itself.

4.6. Implementation

For implementing our system, we employ a low-price System on Chip (SoC), Raspberry Pi 4B: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz 8 GB LPDDR4-3200 SDRAM and a pressure sensor DPS310 Pressure Shield2Go [33]. Additionally, we have used a TPM IRIDIUM9670 TPM2.0 LINUX [34], a hardware security module by Infineon Technologies GMBH specifically designed for Raspberry Pi SoCs. In Figure 5 is shown the setup. Our implementation lacks CTRM, but it is a feasible solution for the real-world market [35].

The software has been developed over WenXin's code, which is available in a public repository [36]. This software presents a classic remote attestation with Attestor and Verifier. We have used this repository as a baseline for the implementation of our presented novelties, and the result can be found in [37] for the Attestor and [31] for the Verifier.

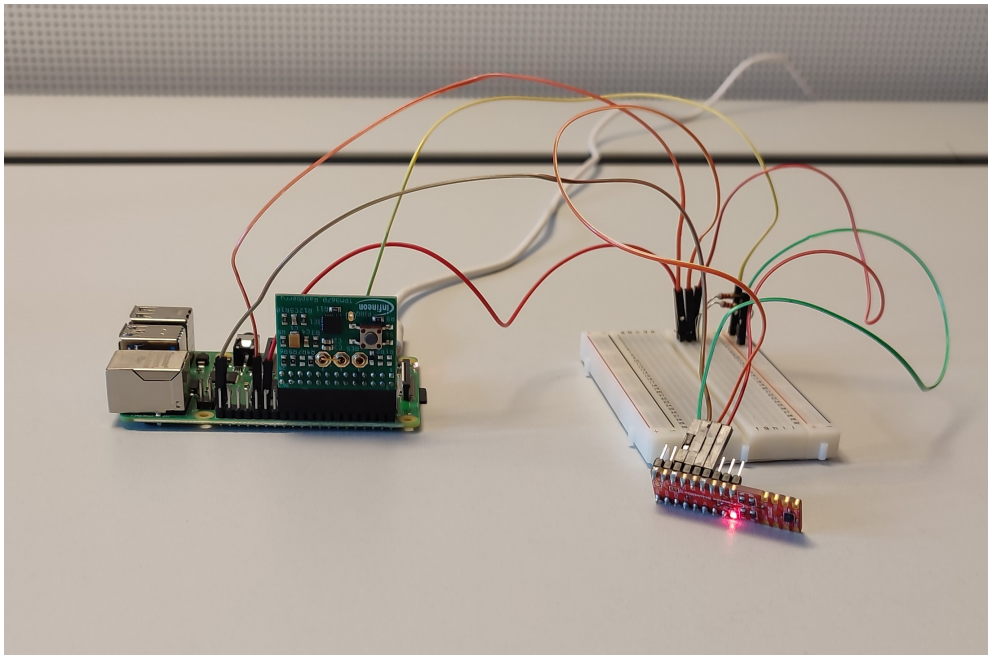

Figure 5. Experimental setup. Raspberry Pi 4 with TPM (green hardware) and pressure sensor (red hardware).

5. RESEKRA Use Cases

There are several ways to use RESEKRA, depending on whether the edge device programmer is the same as the manufacturer, regarding how the Verifier obtains the RML, how it obtains the trust on the sealed key, or who takes the role of the Verifier. In the process, we consider that there is a trusted system to provide the identity of the ED devices to the end user. Two use cases were considered for RESEKRA: edge computing as a trusted service and edge computing as a trusted service—No TTP online.

In the case of edge computing as a trusted service, we have the case of a company that offers edge computing for a fee. This company is responsible for the deployment and maintenance of edge nodes. Additionally, in this use case, we can find the manufacturers of the edge devices, which are trusted by all. The Verifiers, in this case, are a cloud service with high latency and some downtime. Finally, we find the end users who rely on the Verifiers to use the reliable edge device services requiring very low latency.

The second use-case, edge computing as a trusted service—No TTP online, is similar but with a fundamental difference: there are no cloud services offering the Verifier service.

There are other possible use cases where the Reference Measurement List is trusted through a voting method or is provided along with the edge device software from a public repository such as [38]. Edge devices that have already been validated can also be used as a Verifier.

5.1. Edge Computing as Trusted Service

In edge computing as a trusted service, we have (1) the trusted entity in charge of designing, programming, and manufacturing the edge devices for edge computing (Trusted Designers); (2) several semi-trusted investors in charge of deploying and maintaining the edge devices (Semitrusted Maintainers); (3) Trusted Clouds and (4) mobile end users, e.g., a car.

5.1.1. Roles

The Trusted Designers are responsible for designing an edge device with CRTM and TPM. They are also responsible for designing the secure software. Finally, they manufacture the product and store all necessary certificates on the edge device.

Semitrusted Maintainers are entities with little technical knowledge. They are semitrusted because they have no interest in making attacks into the system, but they have no knowledge of how to prevent them. They have the capacity and interest to deploy and maintain an edge device in a specific area, such as a non-profit interested organization, e.g., a government, or an interested profit organization with a business model based on subscriptions or blockchain using utility tokens, e.g., Helium [39].

The Semitrusted Cloud (SC) is responsible for verifying and signing the authorizations for the SeKs of edge devices. Every end user has a set of trusted SC. The SC is semitrusted because the end user only trusts some of them.

The last role is the end user, who needs to trust edge devices to use edge computing services.

5.1.2. Process

First, Semitrusted Maintainers purchase an Edge-Device N (ED_N) $\forall N \in \mathbb{N}$ from Trusted Designers and deploy it wherever they see a fit. Thanks to the remote and versatile key creation system in RESEKRA, there can be several independent SCs verifying the integrity and authenticity of the received documents. These SCs can belong to third parties. The ED_N communicates with a set O_N of SCs, sends them the reference measurement list and the manufacturing certificates signed by the Trusted Designers. A subset P_N of O_N approves the Section 4.3 Remote Enrollment and issues a certificate with the SeK generated and ED_N's identifier. The identifier has to be an universally unique identifier [40] to avoid Wormhole attacks [41], which is explained in more detail in Section 6.3. In our scenario, we propose using the last 128 bits of the hash of the EK_PuB as an identifier.

Because the process Section 4.3.1 Edge device validation has to be realized just once per Verifier, and the processes Section 4.3.2 Sealed Key creation and Section 4.3.3 Sealed Key validation are performed just in unusual moments, normally after each reset, the interactions between ED_N and SCs are infrequent. Therefore, the number of Verifiers in the subset P_N will not affect the service quality. Moreover, the SeKs are stored in the virtual memory of the TPM.

The edge device subsequently requests a Section 4.4 Attestation to every SC in P_N. A subset Q_N of P_N approves the attestation and grants the authorization to use the SeKs, as long as the device is not rebooted or its software is not modified. Additionally, a time limit can be added to this authorization.

At this moment, the ED_N can act as a Verifier for the nearest IoT devices by using a software-based remote attestation such as SoftWare-based ATTestation (SWATT) [15], where the time response is essential, and therefore, a one-hop network is a fundamental requirement. We do not provide further details here, since it is beyond the scope of this paper.

Finally, end users, by requesting the services of ED_N, have to provide a priority organized set G of SCs and a Random Nonce (RN). If any of the elements of G belong to Q_N, the ED_N will sign the RN using the SeK approved by the most priority SC belonging to G and Q_N, and it will furnish the corresponding certificate. Since the identity in the certificate is based on the EK_PuK, the identity provider just has to provide one identifier for ED_N, and it is valid for all the ED_N's SeKs. Otherwise, the ED_N is considered untrusted by the end user.

When ED_N is trusted, the end user requests the edge computing services (gathering data or processing data) and measures the required time of the service. At the end of the service, the ED_N signs the hash of the RN with the results of the service using the SeK. If the final signature never arrives or takes longer than expected, the end user does not trust the results and reports the irregularity.

However, it needs to check the correctness of the edge devices before using their services. The car only has the name of the edge devices and a set of cloud services that are trusted for this car but are not accessible in real time.

Figure 6 shows an scenario where a car enters in a location with an edge computing network, and it needs edge services from 3 EDs such as gathering data from sensors or using computer processing. However, it needs to check the correctness of the edge devices before using their services. The car only has the identifier of the edge devices and a set of SCs G that are not accessible in real time.

Firstly, the car requests a signed RN from the three EDs, each one with three different subsets Q, but only ED_1 and ED_3 have passed through a remote attestation of at least one of the SCs belonging to G. Then, by simply signing the RN with their SeKs, ED_1 and ED_3 prove their correctness, and the car puts the trust in them and starts using their services.

5.2. Edge Computing as Service—No TTP Online

In edge computing as service—No TTP online, we consider the need of avoiding the use of a Semitrusted Cloud. Perhaps, because the internet connection is not available, there are not SCs of G belonging to Q or simply to avoid the use of a TTP.

In this field, we have the same roles as in the previous case, except for the absence of SC.

Processes

When the end user wants to use the services of the edge device, and there is not any trusted third party available to act as a Verifier, the end user himself can act as a Verifier.

The end user himself realizes the Section 4.3 Remote Enrollment. Then, the ED provides the RML signed by the ED's programmer. If ED passes the Attestation Section 4.4, the end user will authorize the use of the SeK and requests the Edge computing services (gathering data or processing data). Upon completion of the service, the ED will use the SeK to sign the service result.

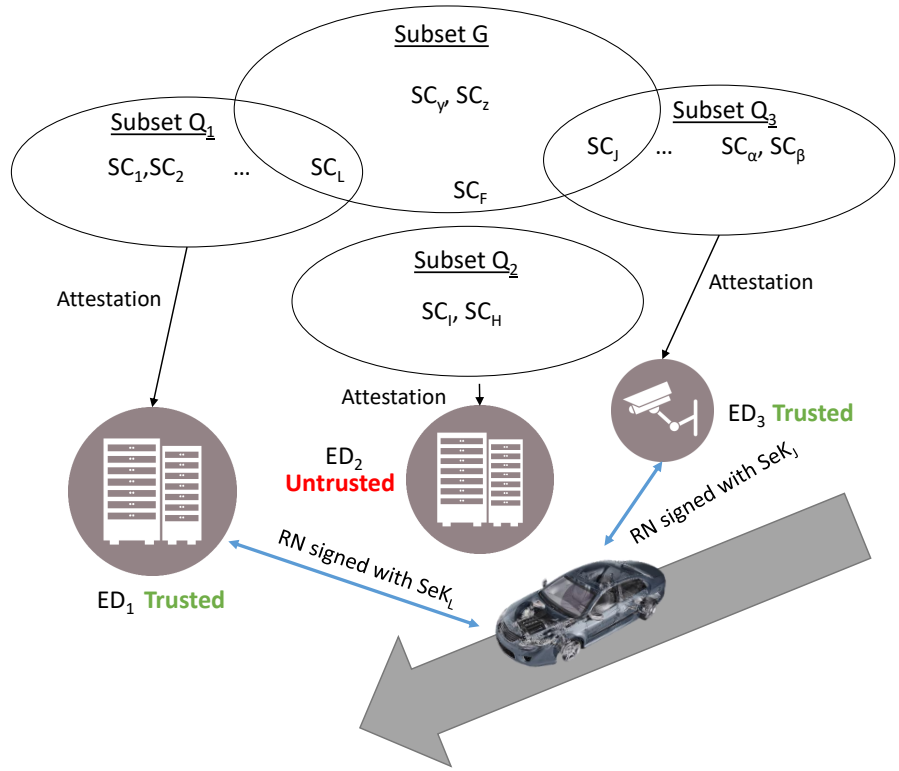

Figure 6. Visualization of edge computing as trusted service. A car reaching a location needs the local services of several edge and IoT devices. All these devices have many SeKs available validated by an SC each. The car just trusts in some of the SCs, those which belong to subset G.

Throughout the communication process, a constant status of the software is ensured by continuously proving the SeK ownership. The process of document verification and SeK creation should be created beforehand to avoid delaying the start of the edge computing service. Figure 7 represents the differences between the two use-cases presented in this section. On the right of the image, all the phases (in purple) are processed between car and edge device, and there is no need of third party acting as a Verifier.

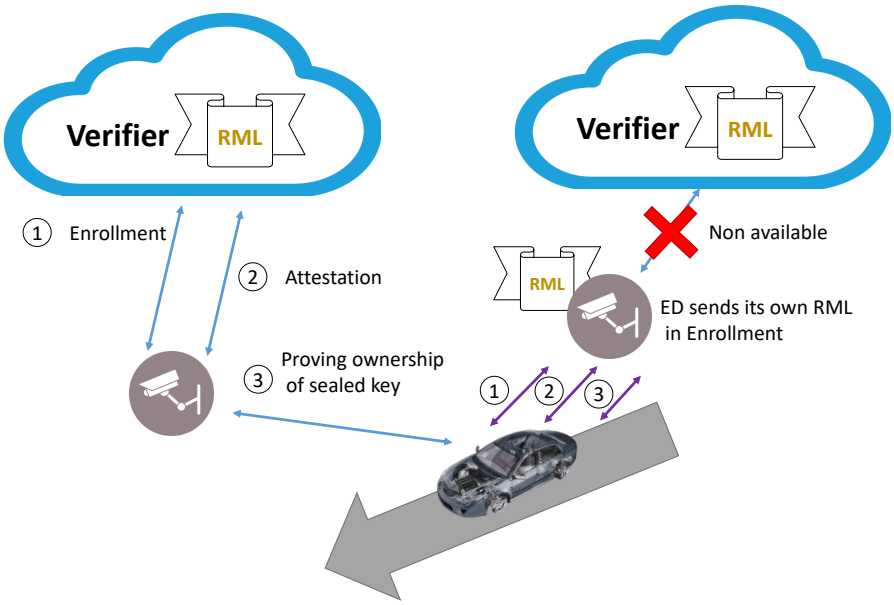

Figure 7. Visualization of the two use cases, edge computing as service (in blue) and edge computing as service—No TTP online (in purple).

6. Security Analysis

In this section, we provide an analysis of the most common attacks on remote attestation present in the state of the art. We do not consider Denial of Service in all the analysis.

6.1. Man-In-the-Middle Attack (MIM Attack)

The entire RESEKRA scheme was developed with MIM attacks in mind. The most sensitive section of RESEKRA is Section 4.3 Remote Enrollment, since we consider the Host-TPM untrusted for the remote communication.

All communications between the ED and Verifier are protected with TLS, but we take into account that even in this communication, an MIM can obtain complete control of Host-TPM. All comunication between the Verifier and TPM goes through the Host-TPM. The attacker would be able to read all certificates and information transferred, but it would not be able to modify it, because it is signed by the TPM. Moreover, even with the complete control of the Host-TPM, the attacker cannot force the TPM to sign fake certificates or quotes because all the TPM generated data are signed by a restricted key (AK).

The AK is proved to be a restricted key by using the Endorsement Key. The attacker could obtain knowledge of Auth_PuK and generate a fake SeK out of the TPM and store it in the TPM. Following this process, the attacker could generate a Cert_SeK signed by the AK, but it would fail while asserting the SeK attributes Section 4.3.3. Therefore, all Section 4.3 Remote Enrollment is protected from MIM attack.

6.2. Impersonation and Replay Attack

An attacker could obtain the certificates stored in Host-TPM, but it would not be able to prove EK ownership and thus would fail to perform the Section 4.3 Remote Enrollment.

Validation of AK and SeK is done through Auth_PuK, which is randomly generated. Therefore, we avoid replay attacks in Section 4.3 Remote Enrollment.

Note that Section 4.4 Attestation is realized with an RN and signed by AK; therefore, replay attacks and impersonation are avoided.

Finally, to show the correct status of the software to end users, the SeK is used to sign RNs, avoiding replay attacks.

6.3. Wormhole Attack

This attack was introduced by Yih-Chun Hu [41]. In this attack, the attacker uses legitimate data or information from an edge computing network at a given location and "tunnels" it in some way to another edge computing network without adding appreciable delay. The attack will then be explained, which is followed by a possible defense strategy:

1. The attacker compromises one edge device close to the end user (ED_C).
2. The attacker tunnels the network of ED_C with another far network with an ED from the same deployer (ED_f).
3. The end user starts the authentication method with ED_C and passes through because it is talking with the expected node.
4. When the attacker receives a random nonce to be signed with SeK, it tunnels this random nonce to ED_f, obtains the signature and certificates ED_f's SeK, and sends them back to the end user.
5. Finally, the end user trusts ED_C without access to ED_C's SeK.

Since RESEKRA relies on asymmetric cryptography (EK, AK, and SeK) that is randomly generated and is therefore unique, it is easy to avoid this kind of attack.

To avoid this attack, the certificate issued from SCs shall include the universally unique identifier [40] (ID) of ED_C, and this ID should be provided to the end user as part of the ED_C's identification information. In Section 5.1, we use the last 128 bits of the hash of the EK_PuB as the ID.

When verifying the SeK's certificate, the end user will find out that the SeK of ED_f does not belong to ED_C.

6.4. Interference Attacks

In the interference attack, an MiM provides a false attestation response (Quote) to Verifier. It will not pass the process, and ED will be considered untrustworthy, even though the Verifier was unable to verify the actual Quote.

The Quote is coming with a signature and RN. When a false Quote is provided, the Verifier will recognize it is not coming from the real ED and will refuse it without affecting the attestation process.

6.5. Time-of-Check-to-Time-of-Use Attack

Time-Of-Check-To-Time-Of-Use (TOCTTOU) is a present vulnerability in Remote Attestation scenarios [9]. In TOCTTOU, the ED provides evidence of having the appropriate software at the time of the Attestation, but it uses different software when the ED is about to provide services.

At a general level, this attack should be avoided by proving access to the SeK at the moment ED provides the services.

One possible attack would be to have the correct software when proving ownership of the SeK at the beginning of the service and then calling a manipulated program when providing the real services. This attack should not be possible in an IMA that is bugs-free because any program able to call a manipulated program would not pass the attestation.

However, if in an unlikely situation were to happen, the ED will not be able to provide the second signature of the result, and thus, the end user will notice this problem when receiving the result. If so, the ED owner can initiate a fast debugging process.

6.6. Verifier-Based DoS Attack

In Verifier-Based DoS (VBDoS) attack, the attacker acts as a Verifier and requests many attestations to the ED, which becomes overloaded and cannot provide its normal service.

The remote enrollment and attestation from ED to SC is started by the ED, and they are not routine procedures.

The attack could come from the end user when it is in charge of attesting the ED through the full RESEKRA process or just validating the use of the SeK. The solution for this attack should be provided by the ED owner as part of the normal DoS attacks to their EDs. It is the responsibility of the ED owner to develop a scheme for providing ED services to the end user, allocate resources, and control and avoid misuses or malicious uses of the ED resources by the end user.

6.7. Non-Verification DoS Attack

It is a new kind of attack developed by us. In this attack, the attacker is able to interrupt the Verifier service on an edge computing network. It could happen through attacking one of the hops in the connection between the Verifier and Attestor or with a DoS attack to the Verifier.

In a common remote attestation scheme, throughout the interruption, the end user cannot trust the EDs, and even when the EDs are in a correct state, if trust is essential, the attacker produces a DoS to the local edge computing infrastructure.

Due to the decentralized Verifier infrastructure in RESEKRA, a DoS to multiple Verifiers is very unlikely. If the connection between EDs and Verifier is broken, the ED still has access to the SeK; therefore, it can still prove its correct status. If by an unlikely circumstance, the ED loses access to the SeK, several local Verifiers can be run-time assigned, or the end user itself can perform the remote attestation. To the best of our knowledge, our system is the first remote attestation protocol able to resist this attack.

7. Conclusions

In this paper, a new process for remote attestation focused on edge computing has been presented. The process, called RESEKRA, has been designed and implemented in a real sensor, introducing new features with regard to the SoA. The first characteristic is the remote enrollment, when an IoT device can connect with a new Verifier without having to know a shared secret, allowing Attestors and Verifiers to dynamically join and leave the structure. The second special feature of RESEKRA is the use of asymmetric keys sealed to the correct state of the software, which allows the correct state of the device to be proven directly to the end user while reducing the workload on the Verifier and allowing the IoT sensor to operate with intermittent activity by requesting a remote attestation when it deems it necessary.

These qualities enable the sought-after decentralized verification system in the SoA, integrate remote attestation into Mobile Edge Computing, and finally, allow the end user to trust the edge devices without any trusted authority or entity within the edge. It allows for further research into edge computing by taking for granted the trust of the devices in the environment. Furthermore, RESEKRA can also have a significant academic impact on other lines of research in IoT computation where trust in IoT devices is crucial, e.g., supply chain, IoT in blockchain, or Industry 4.0.

On the other side, RESEKRA—as with all Remote Attestation systems based on RoT—requires a reliable manufacturer to produce the trusted device, but then, RESEKRA can be used in many other scenarios inside and outside Mobile Edge Computing. It can be used from applications as simple as smart homes—where a simple set up is essential—attesting the end users's devices using a mobile phone. Other applications as complex as Industry 4.0 use several Verifiers from the same owner to avoid the highly feared DoS attacks in manufacturing affairs. It can also be used for lower consumption applications, where sensors operate with intermittent activity to save battery life. In general, RESEKRA and therefore its results can be applied in any scenario beyond the use cases presented where the assumptions detailed in the introduction are met.

In other respects and future work, we envision integrating the Direct Anonymous Attestation Scheme. It would allow proving ownership of an SeK without giving any

additional device identity information. This would preserve the privacy of the Attestor and enable the attestation of privacy-sensitive devices such as cars or mobile phones. On the other side, we are currently working to delete the use of private keys in the Verifier and move it to a smart contract in the blockchain, making the blockchain itself being able to attest the IoT devices before accepting a transaction.

Author Contributions: Conceptualization, E.G.-M.; methodology, E.G.-M.; software, E.G.-M.; validation, E.G.-M.; formal analysis, E.G.-M.; investigation, E.G.-M.; resources, E.G.-M.; data curation, E.G.-M.; writing—original draft preparation, E.G.-M.; writing—review and editing, E.G.-M., G.M., E.C. and L.P.; visualization, E.G.-M.; supervision, E.C. and L.P.; project administration, A.E.-M.; funding acquisition, D.P.M. and A.E.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Union's Horizon 2020 Research and Innovation program under grant agreement No. 871518, A COmprehensive cyber-intelligence framework for resilient coLLABorative manufacturing Systems, COLLABS, and by FEDER/Junta de Andalucía-Consejería de Transformación Económica, Industria, Conocimiento y Universidades under Project B-TIC-588-UGR20.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The work has been developed on a open source base with MIT license: https://github.com/Infineon/remote-attestation-optiga-tpm (accessed on 24 June 2022). Moreover, I thank the community of TPM developers, TPM.dev, for its help in the key queries of specific elements of the TPM standard. However, any errors or problems found in the present work or in the code are solely and exclusively the responsibility of the authors of this document.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

The following notations are used in this manuscript:

IMA	Integrity Measurement Architecture
CTRM	Core Root of Trust of Measurement
AK	Attestation Key
EK	Endorsement Key
SeK	Sealed Key
KP	Key Pair
PuB	Public Key
Pubdata	Public information of a key (public key, policies and attributes)
TPM	Hardware security module following TPM standard
Host-TPM	Device owner of TPM
TPM_GENERATED	4 bytes-code "0xff544347". Files starting with this code and signed by a restricted key were generated by a TPM
PCR	Platform Configuration Registers
ML	Measurement List
RML	Reference Measurement List
Cert_SeK	Attestation certificate of Sealed Key
ED_N	Edge device number N
SC	Semitrusted Cloud
O_N	Set of SCs ED_N commnicate with
P_N	Set of SCs that accept the Remote enrollment of ED_N
Q_N	Set of SCs that accept the Attestation of ED_N
G	Set of priority organized SCs trusted by the end user
LD	Linear dichroism

References

1. Cisco, U. *Cisco Annual Internet Report (2018–2023) White Paper*; Cisco: San Jose, CA, USA, 2020.
2. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
3. Liu, D.; Yan, Z.; Ding, W.; Atiquzzaman, M. A survey on secure data analytics in edge computing. *IEEE Internet Things J.* **2019**, *6*, 4946–4967. [CrossRef]
4. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
5. ISG; ETSI; GMI; Mobile Edge Computing; Market Acceleration; MEC Metrics Best Practice and Guidelines. Available online: https://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/006/01.01.01_60/gs_MEC-IEG006v010101p.pdf (accessed on 23 May 2022).
6. Xiao, Y.; Jia, Y.; Liu, C.; Cheng, X.; Yu, J.; Lv, W. Edge Computing Security: State of the Art and Challenges. *Proc. IEEE* **2019**, *107*, 1608–1631. [CrossRef]
7. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. *IEEE Internet Things J.* **2018**, *5*, 450–465. [CrossRef]
8. Ambrosin, M.; Conti, M.; Lazzeretti, R.; Rabbani, M.M.; Ranise, S. Collective Remote Attestation at the Internet of Things Scale: State-of-the-art and Future Challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2447–2461. [CrossRef]
9. Raducu, R.; Rodríguez, R.J.; Álvarez, P. Defense and Attack Techniques Against File-Based TOCTOU Vulnerabilities: A Systematic Review. *IEEE Access* **2022**, *10*, 21742–21758. [CrossRef]
10. Trusted Computing Group TPM 2.0 Library. 2021. Available online: https://trustedcomputinggroup.org/resource/tpm-library-specification/ (accessed on 23 May 2022).
11. Calvo, M.; Beltrán, M. Remote Attestation as a Service for Edge-Enabled IoT. In Proceedings of the 2021 IEEE International Conference on Services Computing (SCC), Chicago, IL, USA, 5–10 September 2021; pp. 329–339. [CrossRef]
12. Tan, H.; Tsudik, G.; Jha, S. MTRA: Multi-Tier randomized remote attestation in IoT networks. *Comput. Secur.* **2019**, *81*, 78–93. [CrossRef]
13. Tan, H.; Hu, W.; Jha, S. A TPM-enabled remote attestation protocol (TRAP) in wireless sensor networks. In Proceedings of the 6th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, Miami, FL, USA, 31 October 2011; pp. 9–16.
14. Wang, Y.; Attebury, G.; Ramamurthy, B. A survey of security issues in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **2006**, *8*, 2–23. [CrossRef]
15. Seshadri, A.; Perrig, A.; Van Doorn, L.; Khosla, P. SWATT: Software-based attestation for embedded devices. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 12 May 2004; pp. 272–282.
16. Spinellis, D. Reflection as a mechanism for software integrity verification. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2000**, *3*, 51–62. [CrossRef]
17. Seshadri, A.; Luk, M.; Perrig, A.; Van Doorn, L.; Khosla, P. SCUBA: Secure code update by attestation in sensor networks. In Proceedings of the 5th ACM workshop on Wireless Security, Los Angeles, CA, USA, 29 September 2006; pp. 85–94.
18. Francillon, A.; Nguyen, Q.; Rasmussen, K.B.; Tsudik, G. Systematic Treatment of Remote Attestation. Cryptology ePrint Archive, Report 2012/713. 2012. Available online: https://ia.cr/2012/713 (accessed on 24 June 2022).
19. Aman, M.N.; Basheer, M.H.; Dash, S.; Wong, J.W.; Xu, J.; Lim, H.W.; Sikdar, B. HAtt: Hybrid remote attestation for the Internet of Things with high availability. *IEEE Internet Things J.* **2020**, *7*, 7220–7233. [CrossRef]
20. Ibrahim, A.; Sadeghi, A.R.; Zeitouni, S. SeED: Secure non-interactive attestation for embedded devices. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston, MA, USA, 18–20 July 2017; pp. 64–74.
21. Eldefrawy, K.; Tsudik, G.; Francillon, A.; Perito, D. Smart: Secure and minimal architecture for (establishing dynamic) root of trust. In Proceedings of the Ndss, San Diego, CA, USA, 5–8 February 2012; Volume 12, pp. 1–15.
22. Carpent, X.; Tsudik, G.; Rattanavipanon, N. ERASMUS: Efficient remote attestation via self-measurement for unattended settings. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1191–1194.
23. Koeberl, P.; Schulz, S.; Sadeghi, A.R.; Varadharajan, V. TrustLite: A security architecture for tiny embedded devices. In Proceedings of the Ninth European Conference on Computer Systems, Amsterdam, The Netherlands, 14–16 April 2014; pp. 1–14.
24. Carpent, X.; ElDefrawy, K.; Rattanavipanon, N.; 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, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 86–100.
25. Dushku, E.; Rabbani, M.M.; Conti, M.; Mancini, L.V.; Ranise, S. SARA: Secure asynchronous remote attestation for IoT systems. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3123–3136. [CrossRef]
26. Safford, D.; Kasatkin, D.; Mzohar. Integrity Measurement Architecture (IMA). 2013. Available online: https://sourceforge.net/projects/linux-ima/ (accessed on 24 June 2022).
27. Cooper, D.; Polk, W.; Regenscheid, A.; Souppaya, M. BIOS Protection Guidelines—NIST, 2011. National Institute of Standards and Technology. Available online: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-147.pdf (accessed on 7 June 2022).

28. Trusted Computing Group™. Endorsement Key (EK) and Platform Certificate Enrollment Specification. 2013. Available online: https://trustedcomputinggroup.org/wp-content/uploads/IWG-EK-CMC-enrollment-for-TPM-v1-2-FAQ-rev-April-3-2013.pdf (accessed on 20 May 2022).
29. Trusted Computing Group. TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0 Revision 0.36. Available online: https://trustedcomputinggroup.org/wp-content/uploads/TNC_TAP_Information_Model_v1.00_r0.36-FINAL.pdf (accessed on 24 June 2022).
30. Arthur, W.; Challener, D.; Goldman, K. *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*; Springer Nature: London, UK, 2015.
31. Ernesto. RESEKRA: Verifier. Available online: https://github.com/ernesto418/Remote_attestation_server/ (accessed on 21 June 2022).
32. Dierks, T.; Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC5246. 2008. Available online: https://www.rfc-editor.org/info/rfc5246 (accessed on 20 May 2022).
33. Technologies, I. DPS310 Pressure Shield2Go. Available online: https://www.infineon.com/cms/en/product/evaluation-boards/s2go-pressure-dps310/ (accessed on 21 June 2022).
34. Technologies, I. IRIDIUM9670 TPM2.0 LINUX. Available online: https://www.infineon.com/cms/en/product/evaluation-boards/iridium9670-tpm2.0-linux/ (accessed on 23 May 2022).
35. Mode, M. Secured Boot for ARM Processor Platforms. Available online: https://www.infineon.com/dgdl/Infineon-ISPN-Use-Case-Secured-boot-for-ARM-processor-platforms-ABR-v01_00-EN.pdf?fileId=5546d4625bd71aa0015c0b1fceee0851 (accessed on 20 May 2022).
36. Leong, W.; Yushev, A. OPTIGA™ TPM Application Note Remote Attestation. Available online: https://github.com/Infineon/remote-attestation-optiga-tpm (accessed on 23 May 2022).
37. Ernesto. RESEKRA: Attestor. Available online: https://github.com/ernesto418/Remote_attestation_TPM/(accessed on 21 June 2022).
38. Linux Kernel. Available online: https://github.com/torvalds/linux (accessed on 20 May 2022).
39. Helium©, People-Powered Network, 2021 Helium Systems Inc. Available online: https://www.helium.com/ (accessed on 28 January 2022).
40. Leach, P.; Mealling, M.; Salz, R. A Universally Unique Identifier (uuid) urn Namespace. RFC 4122. 2005. Available online: https://www.rfc-editor.org/rfc/rfc4122 (accessed on 24 June 2022).
41. Hu, Y.C.; Perrig, A.; Johnson, D.B. Wormhole attacks in wireless networks. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 370–380.