A Flexible Trust Manager for Remote Attestation in Heterogeneous Critical Infrastructures

A Flexible Trust Manager for Remote Attestation in Heterogeneous Critical Infrastructures

Enrico Bravi
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
enrico.bravi@polito.it

Diana Gratiela Berbecaru
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
diana.berbecaru@polito.it

Antonio Lioy
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
antonio.lioy@polito.it

***Abstract*—Nowadays, critical infrastructures are managed through paradigms such as cloud/fog/edge computing and Network Function Virtualization (NFV), providing advantages as flexibility, availability, and reduced management costs. These paradigms introduce several advantages but – given their nature of physically distributed systems – leave room for various security threats, such as software integrity attacks. To counter these threats, Trusted Computing and Remote Attestation (RA) techniques can be used, to allow a third party (Verifier) to verify the software and configuration integrity of a platform (Attester). In environments composed of different objects, several RA frameworks (hardware-based, software-based, or hybrid) might need to be deployed, depending on the capabilities of the attested elements. To ease this process, we propose a new design and implementation of our *Trust Monitor* (TM) architecture, which implements the Trust Manager specified by ETSI for NFV environments, making it more flexible and usable in different contexts. In addition, we define a generic model for performing RA in heterogeneous environments by employing various RA technologies. More specifically, the extended TM allows flexible RA in hybrid infrastructures composed of different objects, i.e., physical nodes, virtual machines, containers, pods, and enclaves. Through tests performed in an experimental testbed, we show that the proposed implementation is scalable and usable in heterogeneous contexts.**

***Index Terms*—trusted computing, cloud computing, Trust Monitor**

I. Introduction

The security of IT infrastructures, in particular cloud or NFV environments, is increasingly significant in business since more and more resources, like sensitive data and applications, are managed by cloud-enabled or NFV-supported services. The world of networks and infrastructures has been revolutionized by the concept of *virtualization*, allowing for optimized resource utilization and service provision. On the other hand, this innovation has also introduced inherent security issues [1], because if the employed resources are tampered with, they could pose serious threats to the infrastructures themselves. An example of how virtualization can be used to add flexibility in a network context is NFV [2]. This paradigm allows moving all the services provided by a network provider to a higher level, by making them available via Virtualized Network Functions (VNFs) instead of implementing them with dedicated hardware. Also cloud computing is heavily based on virtualization, as it exploits several aspects of these technology but it inherits also some security vulnerabilities, such as the ones related to platform tampering.

One possible approach to detect code manipulation or configuration tampering is through *Remote Attestation* (RA) [3]. The RA process typically involves an entity (called Attester, or Prover) sending its current state, usually in the form of a set of software component measurements, to an external trusted entity (called Verifier) which checks this data against an expected behavior, called golden values. Hardware-based RA techniques exploit a specific device, the Trusted Platform Module (TPM) [4], which is a cryptographic co-processor providing the security capabilities needed by the RA process. However, for IoT devices with limited resources, the TPM may not be practical because it would consume most of the capabilities of the device [5], so alternative solutions like DICE [6] or Keystone [7], have been proposed. A critical problem in heterogeneous network environments is that different entities might be attested through different RA technologies. Thus, the European Telecommunications Standards Institute (ETSI) has created a specific RA standard [8], which proposes a *Trust Manager* entity in charge of the VNFs integrity verification in NFV infrastructure by centralizing these operations.

Contribution. We define first a generic model for object representation, in order to integrate different RA technologies specific to different kinds of objects. We propose a new design of our Trust Monitor (TM) in [9] by adding the Adapters' connector, which allows to manage different RA frameworks in a transparent way. This new TM has been fully implemented and evaluated in a test-bed to measure its performance.

Paper structure. The paper is organized as follows: Section II introduces the main theoretical concepts in the trusted computing, RA, and network virtualization areas. Section III describes the extended design of the new TM architecture. Sections IV and V detail the implementation and testing of the extended TM. Section VI discusses the related work. Finally, Section VII provides conclusions and future work.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the project FISHY (Grant Agreement no. 952644). This work was also partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the European Union - NextGenerationEU.

II. Background

A. Trusted Computing

The concept of *trust* applied to a platform means determining if the platform behaves as expected or not. In case the platform has the expected behavior, it can be considered *trusted*, otherwise, it has to be considered *untrusted*. From this concept, it can be deduced that trustworthiness can be determined if the behavior is predictable.

The Trusted Computing Group (TCG) [10] is a worldwide organization, that defines the specifications and promotes tools related to trusted computing. It has defined the concept of *Trusted Platform* (TP), which is based on the ability to perform integrity measurements on all its hardware and software components. All these measurements are computed over the code or the configuration data of a component using a cryptographic hash algorithm (e.g. SHA-256). The TCG has proposed a possible implementation of the TP, which relies on an additional chip called TPM [11], a hardware Root of Trust (RoT), which is a cryptoprocessor fixed on the motherboard of a platform that permits to store securely the integrity metrics in some special register called Platform Configuration Registers (PCRs) and then report these integrity metrics to third parties in an authenticated way.

The mechanism used to measure a TP follows the concept of transitive trust for which the trust in a component is used to evaluate the trustworthiness of the subsequent component which will get control of the platform. Thus, it is possible to establish a *chain of trust* which allows to verify whether the system has booted into a trusted environment. After checking if a system booted correctly, the measurement process may continue at the operating system (OS) level in order to determine the trustworthiness of the applications. A remote third party may verify the trustworthiness of a platform by using RA.

B. Remote Attestation

Remote attestation is a security mechanism that enables a trusted platform (Verifier) to verify the integrity of another platform (Attester or Prover) by exploiting golden values saved in a *whitelist*. The protocol used by remote attestation follows a challenge-response model, with two main entities involved, namely the *Prover* (P) and the *Verifier* (V), as shown in Fig. 1.

The process of RA typically starts with the Verifier, which already has a golden value containing the state of the Prover (*P_known_state*), sending a challenge (c) to the Prover. The Prover must provide a response (called Integrity Report) corresponding to its state at the time the attestation was requested by the Verifier, calculated as a signature on the attestation data (*att_data*) and the challenge c with a specific asymmetric private key, i.e. $Sign(att\_data, c, KP.priv)$. Once the Verifier receives the response, it verifies first the signature on the Integrity Report with the public asymmetric key ($KP.pub$) corresponding to the Prover P. Then, it compares the attestation data with the golden value (P_known_state) stored in the local whitelist. If the two values match, the Verifier can assert that the Prover's verified code has not been modified.

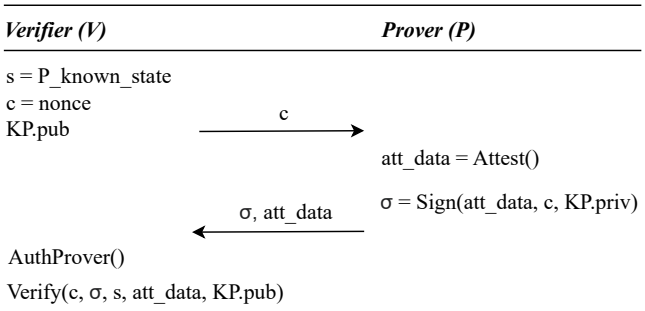

Fig. 1. Simplified workflow of the Remote Attestation process.

Several RA approaches have been proposed, based on different hardware and software technologies used for protecting the sensitive data (like keys) exploited in the RA process.

a) Hardware-based Attestation: These kinds of techniques are based on a dedicated piece of hardware, typically a cryptographic chip, which has to be present on the Prover. One of the most used and common hardware devices used to perform RA is the TPM, described in Section II-A. The TPM specification has been proposed by the TCG, with the first version which was 1.2 [12]. Subsequently, an improved version was proposed, the TPM 2.0 [13], which is the current standard. Other RA approaches exploit a Trusted Execution Environment (TEE) [14], such as Intel SGX [15], or ARM TrustZone [16]. These techniques are typically based on some hardware extension that permits to obtain specific security features.

b) Software-based Attestation: While hardware-based attestation is a highly effective solution for RA, it may not always be practical due to hardware and software constraints, especially in embedded devices. To address this problem, some software-only RA approaches have been proposed to minimize hardware overhead. Pioneer [17] is an example of a software-based primitive that does not rely on CPU architecture extension or any secure co-processor. The core idea of this method is that the *dispatcher* (Verifier) uses Pioneer to create a *dynamic root of trust* [18] on the *untrusted platform* (Prover), thereby ensuring that all contained code is unmodified.

c) Hybrid Attestation: While software-based approaches for RA may not be sufficient in certain networked settings due to potential adversarial capabilities [19], a hybrid approach that incorporates both software and hardware has been developed to address this issue. One example is SMART [20], which is based on a minimal hardware modification of embedded Micro Controller Units and represents the first minimal hardware solution for establishing a dynamic root of trust in such devices.

C. Critical Infrastructures and virtualization

IT infrastructures are nowadays a crucial asset of big companies, public entities, and even small businesses since they support the management of both internal and external

workflows inside companies. In the case of internal flows, it offers organizational-level services to the employees, while for the external ones, they permit to expose offered services and facilities to clients. Paradigms like cloud computing [21] [22] and NFV allow to manage companies' IT infrastructures more easily and in a cost-preserving manner, by exploiting the concept of *virtualization*, which permits the optimization of resource consumption. Traditionally, virtualization introduces new entities, the Virtual Machines (VMs), that allow for a high level of application-level isolation because each VM has its own virtual hardware and separate environment. The *lightweight virtualization* is another option, which employs various OS features and concepts to isolate processes at the kernel level, as an alternative to the hardware virtualization. More recently, *containers* [23] have been increasingly adopted in IT infrastructures due to their flexibility. In practice, containers are entities enabling different levels of isolation based on specific needs, such as at the network, process, or user level, or combinations of them.

Cloud providers offer nowadays different levels of infrastructure management services to the service providers or clients (i.e., companies or organizations in general), such as Infrastructure as a Service (IaaS) where the provider supplies the virtual hardware, and the client builds its infrastructure as it would be on-premise without managing physical nodes. Alternatively, the cloud provider may offer Software as a Service (SaaS), where an application or service is provided to the client, and all infrastructure details are hidden. Lastly, in the Security-as-a-Service (SECaaS) model, the service provider integrates its security services into a cloud infrastructure on a subscription basis, because this approach may be less expensive than most clients can afford on their own when the total (security) cost of ownership is considered. The security services offered often include intrusion detection, anti-malware/spyware, authentication, anti-virus, and even penetration testing or security event management.

Regarding network management, the NFV paradigm was proposed to address the high hardware resource consumption in traditional networks. This innovative approach replaces traditional network components with a virtualized infrastructure, allowing for the deployment of VNFs like firewalls, intrusion detection systems, and network monitors. These objects are software implementations of physical components running within containers or VMs. This approach offers several advantages, such as the separation of the NFVs from the hardware, obtaining thus reduced hardware dependencies and a faster deployment, configuration, and management time.

The NFV paradigm offers several advantages but also raises security concerns [24]. To address these issues, the NFV-SEC Working Group was established by ETSI. In particular, monitoring activity in an NFV scenario becomes crucial as all VNFs are software implementations, which makes them vulnerable to manipulation from malicious actors. To tackle the security challenges in NFV environments, RA can be employed to verify the trustworthiness of resources and the VNFs. ETSI provides a document [8] that outlines the RA architecture(s) suitable for NFV systems. In this context, ETSI proposes the *Trust Manager*, which has the purpose to manage the trust relationships in NFV deployment environments.

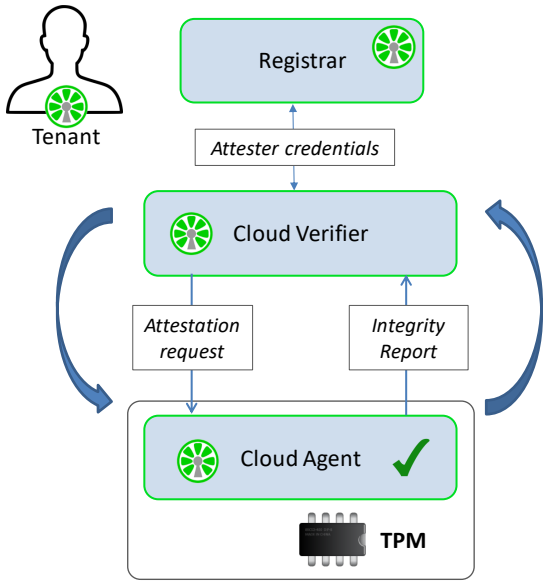

Fig. 2. Keylime Remote Attestation workflow.

D. Keylime Framework

We briefly describe this famous RA framework since we have exploited it in the prototype implementation. Developed by the "Lincoln Laboratory" (a security research group at Massachusetts Institute of Technology - MIT) for cloud computing context [25] and, more specifically, in the scenarios supporting IaaS, the Keylime framework is aimed to provide high scalability to the RA process. In Keylime, the final user (called Tenant) is provisioned with resources (cloud nodes) that could be physical or virtual machines. The resources are given to the user who will deploy and control his software to these nodes. The tenants have no control over the underlying infrastructure, so they cannot ensure, with their own implementation, that the platform given by the IaaS provider remains in a good and safe state during the computation.

The Keylime architecture exploits four main components: the Tenant, the (Cloud) Agent, the Registrar, and the (Cloud) Verifier. The *Tenant*, is the module that registers the *Could Agent* with the *Cloud Verifier* (CV), sending it all the information (that is the whitelist, exclude list, and TPM policy) necessary to start the periodic RA on the node running the Cloud Agent. The *Cloud Agent* (CA), is a service running on the Attester node. It is in charge of sending the Integrity Reports (IRs) to the *Cloud Verifier*. The *Registrar* is the component to which every network node has to register via its own UUID, an alphanumerical identifier. The Keylime RA workflow is shown in Fig. 2

The CV is the core element of the Keylime framework. Once a node is registered, the Tenant can start monitoring it by asking the CV to verify the integrity of the registered

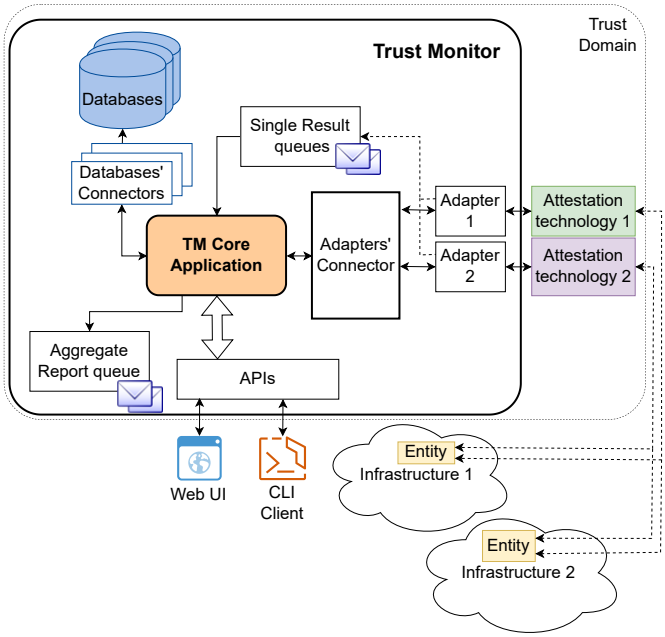

Fig. 3. Proposed extended design for Trust Monitor architecture.

node (Fig. 2). By default, the Verifier sends every 2 seconds an attestation request to the Cloud Agent, even though the attestation period can be changed to verify the Attester more frequently. To evaluate the integrity level of the Attester, the Verifier processes the IRs based on the information received from the Tenant (e.g., the whitelist, the TPM policy), as well as the data received from the *Registrar*.

III. Trust Monitor Design

The TM, originally proposed in [9], is a system that permits to handle Remote Attestation of VNFs in a Security-as-a-Service (SECaaS) scenario. The TM was designed to be a stand-alone component in the administrative domain of the Management and Orchestration (MANO) [26] system, managing the attestation process of virtual Network Security Functions (vNSFs) and Network Function Virtualization Infrastructure (NFVI).

The TM architecture is modular and provides several functions, including a) the verification of the integrity of both heterogeneous nodes of the NFVI and vNSFs; b) notification and reporting on integrity status; and c) auditing integrity verification logs of the infrastructure. Attestation of the infrastructure hosts is performed by sub-components known as *Attestation Drivers*, developed as plugins, which allow instantiating different RA workflows based on the type of entity to attest.

The extended TM architecture we propose is based on the original one, maintaining thus coherence with the basic structure. However, the key concept in our extension stands in aiming to make the TM architecture independent from the RA technologies and frameworks used and the type of infrastructure. The updated design of the architecture is illustrated in Fig. 3. The most significant change is the introduction of the *Adapters' Connector*, and the *Attestation Adapters*, which are similar to the Attestation Drivers. This improvement allows the TM to be independent of the attestation technologies used, since it may contact the right attestation adapter for each entity by using data specified during the *registration* phase. To reach this, all the attestation logic was moved on the adapters so that the TM Core Application has no constraints about the RA framework(s) deployed in the attested infrastructure(s).

The attestation results are aggregated into a *report* for each *entity* with an active RA process, providing an instant view of its status. Message queues facilitate sorting attestation results from different technologies in an asynchronous way, allowing for other requests during the wait. Queues receive attestation results from all the adapters and permit the TM core application to read and aggregate them based on the entity. Another queue receives all the aggregated reports created by the TM core application, which can be processed by a custom consumer developed for specific needs. We also defined a generic object model, that permits abstracting objects and it makes possible to register different kinds of entities. This model includes generic fields where specific information can be specified. In this way, the object remains generic to the TM, but the correct adapters can specifically manage it.

The TM provides APIs that enable administrators to control and manage the system, including all of its components and the attestation process. We also developed a graphical web interface that allows all the TM functionalities to be managed more easily.

Databases are also crucial components of the system, as they store all the necessary information for attestation. By maintaining a separate state from the attestation frameworks used, the system can store information independently of the attestation process and the specific frameworks used.

In brief, we extended the original TM design with an additional RA abstraction layer, which allows a simpler integration of different RA technology workflows. By storing information such as data about objects to be attested, verifiers, and attestation technologies to be used, the attestation process becomes independent of the specific technologies utilized. The TM is the only system that needs to be contacted to initiate the integrity verification process, after which it will interact with all necessary attestation frameworks. This approach enables the attestation of an entity using multiple technologies by simply specifying in the registration phase, which frameworks will be used for RA of that particular object. The TM will manage all specified technologies and aggregate all results into a single report.

The components that compose the proposed architecture are:

1) TM Core Application: This is the main component that manages all the RA processes in execution. Moreover, it is responsible for collecting the attestation results, produced by the RA frameworks and aggregating them *for each entity*. It handles the central high-level logic, focusing on managing the attestation process indirectly. When a request is received

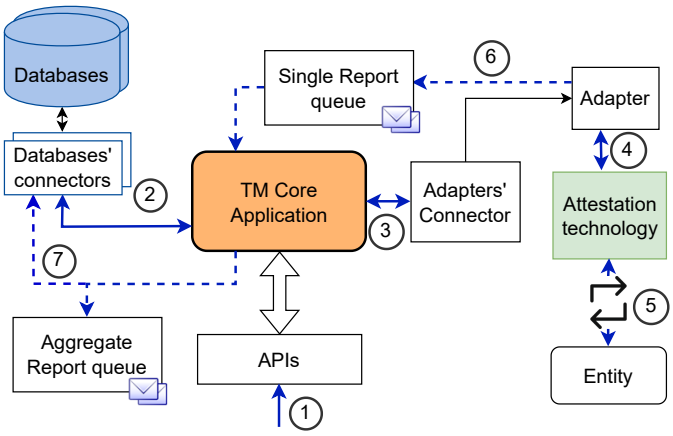

Fig. 4. The defined RA process management.

to initiate the RA process for a specific entity, the TM Core Application retrieves the required data by contacting the Database Connectors. It then passes this data to the adapters responsible for managing the RA process.

2) Adapters' Connector: The Adapters' Connector is a crucial component that enables the TM to communicate with different RA technologies without requiring prior knowledge of the specific framework involved. This allows for the seamless addition and removal of RA technologies without necessitating any changes to the TM core. This component, at load time, read a specific section in a *configuration file*, in order to detect the specified adapters. In this section, there are declared all the developed adapters that will be used. If a new adapter is inserted in the configuration file, there will not be necessary to restart the TM, because each time a new RA process starts, the Adapter's Connector loads all the adapters that have been specified.

3) Attestation Adapters: The TM's flexibility in managing the attestation process is achieved through the *Attestation Adapters*. These components enable the connection of attestation frameworks with the TM core application while keeping the frameworks' details hidden from the TM. An Adapter follows a specific structure to expose a common interface to the TM, allowing it to use the framework without knowledge of its implementation and workflow. Custom data can be stored in metadata fields in the databases, which will be utilized by developed adapters. Whitelist structures are also customizable, as different attestation technologies support different structures, allowing the adapter to manage and utilize the structure for the attestation process. The TM core application is kept unaware of these details.

4) Attestation Results management: The extended TM architecture includes two types of queues: the single result queue and the aggregate report queue. The first one gathers all attestation results from the attestation technologies through the adapters. It enables the TM Core Application to receive attestation results in an asynchronous manner, enabling more efficient management of the results to create reports that aggregate the results related to a specific entity. A single queue is created for each entity under integrity verification, and it collects all the results produced by every RA framework that is attesting that specific entity. The purpose of the aggregate report queue is to make reports available for live consumption. As a result, a customized consumer can be developed to process the generated reports and make decisions or take actions based on them. There will only be a single queue in this case.

5) High-level RA attestation process: The attestation procedure (Fig. 4) begins with a request to the API manager (1), which contacts the TM Core Application. The TM Core Application retrieves all the necessary data (2) to begin the RA process. Upon gathering the required information, the TM Core Application communicates with the Adapters' Connector (3) to select the appropriate attestation adapter. If multiple attestation technologies are employed, the Adapters' Connector contacts all the necessary adapters. Once contacted, the selected adapter begins communication with the attestation framework (4) to manage the attestation process performed with the specific technology (5). The results of this process are published on the single report queue (6), which makes them accessible to the TM Core Application for reading. Periodically, the TM Core Application generates a report that consolidates this information and publishes it on the aggregate report queue (7). Additionally, the reports are stored in the internal database, enabling historical analysis of all reports created for a specific entity.

6) APIs and Web Interface: The TM exposes a set of APIs that can be used to interact with it. The APIs Manager handles all the operations provided by the TM, receiving requests and subsequently communicating with the TM Core Application to execute the requested procedures. This component has been implemented as a web server, which can run on `http` or `https`.

The APIs can be used if there is the intention to integrate the TM into a more complex architecture. In this case, for example, it is possible to use or develop an orchestrator's integration that will automatically contact the TM. On the other hand, in case the TM is used as a stand-alone component, we developed a simple Web Interface to manage all the

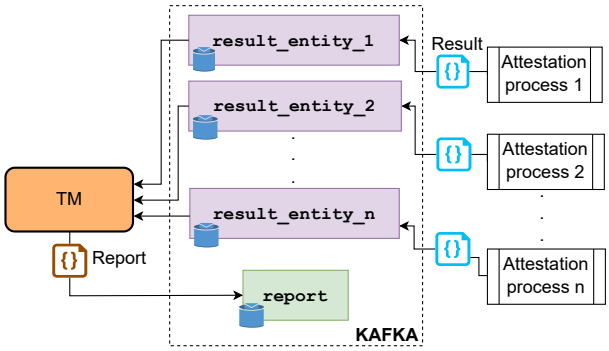

Fig. 5. Queue-based attestation results collection mechanism.

possible operations. This interface is an isolated component that can be deployed autonomously, independent from the TM deployment. Additional details on the full design of the proposed TM architecture can be found in [27].

IV. Trust Monitor Implementation

The TM system has been developed using the Python programming language, version 3.8. Each logical component has been developed as a single Python file which exposes all the necessary methods in order that other components can interface with it.

The APIs Manager component is implemented as a web server, and the Quart Python framework [28] was utilized. Quart is an asyncio [29] reimplementation of the popular Flask microframework API [30], enabling the handling of requests asynchronously without blocking the execution flow until a request is served. Asyncio, a component of the Python standard library, provides an event loop with input/output operations, enabling the implementation of concurrency and improving CPU utilization performance.

The implementation of the necessary queues (Fig. 5) is accomplished using Apache Kafka [31], which allows for the implementation of a publish/subscribe protocol that is utilized to transmit attestation results and aggregate reports. Messages sent in this infrastructure are arranged in topics, which can be considered similar to folders in a filesystem. Each topic can have multiple producers that publish messages on the topic, and multiple consumers that read and process messages sent on the topic. In the specific case of the TM, there are two types of topics:

- `result_entity_<entity_uuid>`: it collects attestation results, produced by all the attestation technologies that are in execution, related to a specific entity. This is a dynamic topic that is created when the RA process begins, for a specific entity, and deleted when it ends;
- `report`: it makes available aggregate reports, for each entity under attestation. There is only one instance of this topic that collects all reports.

As RA technology for the deployed nodes, we used the Keylime [25] framework (version 6.3.2), which requires nodes to be equipped with a TPM 2.0 chip. We used an extended version [32] which has been proposed in order to support Kubernetes' pods RA. Moreover, we have developed a specific attestation adapter for the selected RA framework to enable the TM to interact with the Keylime software. Each pod was composed of one container based on the `nginx` image.

V. Test and Evaluation

A. Testbed description

The implemented TM system was tested to evaluate the performance and correctness of the functionalities. The experimental testbed (shown in Fig. 6) used was composed of the following elements:

- two PCs, used as *attesters*, that were Intel NUC equipped with an Intel Core i5-5300U Processor, 16 GB of RAM, and a TPM 2.0 chip, running Ubuntu server 20.04 LTS with a patched Linux kernel [32] (to support Pods RA) based on version 5.13 and the RA agent of Keylime.
- one PC, used as a *verifier*, which was an Intel NUC equipped with an Intel Core i5-5300U Processor, 16 GB of RAM, and a TPM 2.0 chip, running the Ubuntu server 20.04 LTS and the verifier software of Keylime, described further below.
- one PC, running the *Trust Monitor*, which was a DELL XPS 15 9500 equipped with an Intel Core i7-10750H Processor, 16 GB of RAM. The OS used was Ubuntu Desktop 20.04 LTS.

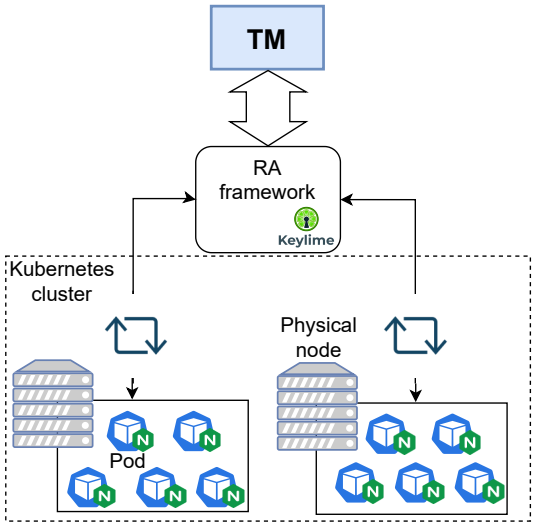

Fig. 6. Experimental physical testbed.

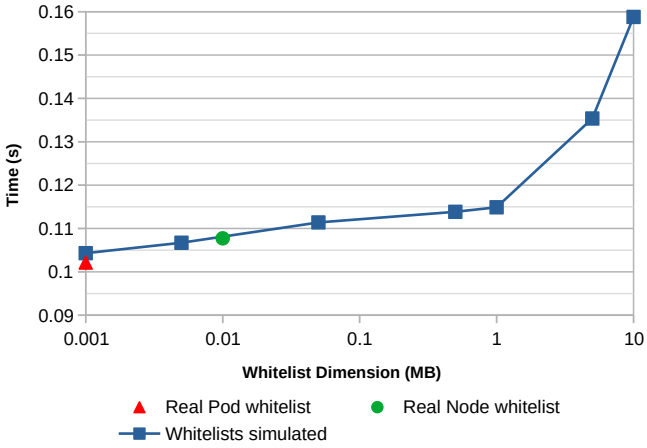

Fig. 7. Overhead time, introduced by the TM during the RA process instantiation, about the whitelist dimension.

B. Results evaluation

The performance evaluation focused on the following metrics:

1) the time (duration) for initiating a new attestation process by the TM, depending on the size of the whitelist;

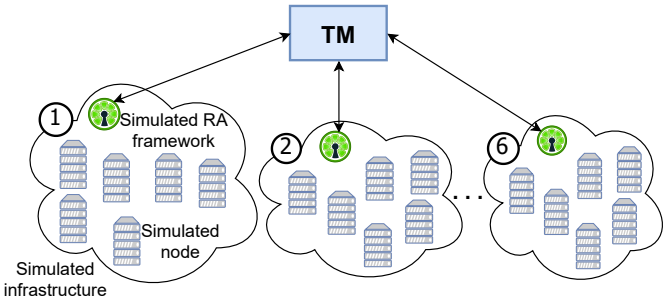

Fig. 8. Virtual testbed composed of 6 simulated infrastructures and a total of 36 simulated nodes.

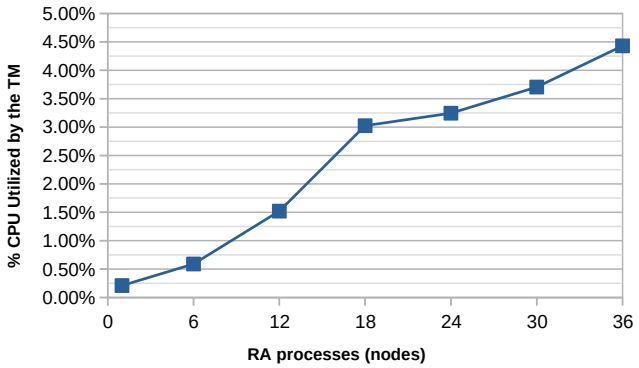

Fig. 9. CPU consumption in relation to the number of active RA processes.

2) the CPU utilization, depending on the number of running RA attestation processes.

The chart depicted in Fig. 7 illustrates the impact of increasing whitelist size on the time required for the TM to initiate the RA process. This is the most significant performance measurement because the subsequent time of the RA process depends only on the RA framework that is used, so this is the most notable time overhead that could be introduced by the TM regarding the RA process. This measurement reflects the time needed by the TM core to retrieve necessary information and start the whole attestation process. The tested whitelist sizes include 1 kB, 5 kB, 50 kB, 500 kB, 1 MB, 5 MB, and 10 MB. The chart also presents results obtained from two actual use cases: node and pod whitelists. As indicated in the chart, the time required to start the attestation process begins to significantly increase when the whitelist size exceeds 1 MB but remains still acceptable.

To assess the CPU usage caused by the attestation processes handled by the TM, a *simulated environment* (Fig. 8) was established, entirely deployed on the TM PC. This environment comprises six infrastructures completely simulated. Each infrastructure was composed of a simulated Keylime instance and six simulated nodes. There were simulated only physical nodes because the purpose of this test was to measure the performance, independently of the entity attested. The Keylime framework's actions were emulated by web servers, deployed as Docker containers, that expose the same interfaces of the real framework. To simulate a node, some information was registered, in order to create the object into the TM. In this way, it was possible to start a RA process on that object. In this case, the RA process was just emulated because the Keylime instance was a simulated one, but this permitted to create several RA management processes in the TM, to evaluate the CPU consumption. As depicted in Fig. 9, the results indicate a nearly linear rise in the CPU utilization (expressed as a percentage) on the TM node.

VI. Related work

In a cloud context, a RA architecture aimed to verify the trustworthiness of users' VMs, specifically for the IaaS scenario, was proposed in [33]. Unlike our work, this paper aimed to manage the trustworthiness of objects but it was explicitly specialized for managing VMs. Another work proposed that aims to monitor the trustworthiness and integrity of network infrastructures is the Trust Monitor [9], but it has the limitation to be applicable only in an NFV context.

For IoT environments, composed of numerous smart devices that are potentially vulnerable to cyberattacks, new solutions have started to be investigated. In particular, the Collective Remote Attestation (CRA) schemes may remotely perform attestation of large networks of IoT devices [34] that could be even mobile nodes. However, such schemes require the presence of an additional component in the network (in addition to the Attester and Verifier), named Aggregator, whose purpose is to relay messages among entities in a network and, when possible, aggregate inputs from neighbors.

Authors in [35] proposed two scalable remote attestation scheme suitable for private cloud, NFV use cases supporting large amounts of VM attestations by exploiting physical TPM device. In particular, in the Hypervisor-based attestation scheme, a so-called *appraiser* attests the target including n-VMs. However, the appraiser does not contact and attest VMs directly, instead, it contacts the hypervisor and receives a collection of VM attestations attached to a single physical TPM attestation.

VII. Conclusion and future work

In this paper, we presented a solution for flexible RA attestation in heterogeneous environments composed of different objects employing various attestation technologies. The Trust Monitor architecture proposed aims to be independent of the attestation technologies, by introducing a so-called *Adapter's Connector*. This component allows to dynamically load attestation drivers called Adapters, that implement the interface with different RA attestation technologies. This mechanism permits to add more easily new RA attestation technologies by developing the corresponding adapter, without modifying the core TM application. In addition, a new object model was designed to support multiple object types. This model permits to abstract an entity (e.g. physical node, pod, VM, container, enclave) by specifying all the information necessary for RA, with custom metadata fields added for flexibility. We described

the prototype implementation as well as the evaluation of the extended TM for hardware-based RA in environments composed of physical machines, equipped with TPM 2.0, and Kubernetes Pods by using the Keylime framework. We evaluated TM performance both at instantiation time as well as during runtime, in terms of the overhead introduced by the TM with the increase of the whitelist dimension, and the CPU consumption based on the number of attested nodes.

The proposed system could be further improved by introducing an automatic whitelist generation system, which could permit a more automatic registration phase. Future tests could verify and improve the use of TEE technologies such as Intel TXT, AMD SEV, ARM TrustZone, and Keystone. Due to its flexibility, the proposed TM system has good potential for expansion and improvement to support emerging IT infrastructure scenarios.

REFERENCES

[1] T. T. Brooks, C. Caicedo, and J. S. Park, "Security vulnerability analysis in virtualized computing environments," *International Journal of Intelligent Computing Research*, vol. 3, no. 1/2, pp. 277–291, December 2012, doi: 10.20533/ijicr.2042.4655.2012.0034.

[2] ETSI, "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," in *SDN and OpenFlow World Congress*, Darmstadt (Germany), 2012, http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[3] G. Coker et al., "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, June 2011, doi: 10.1007/s10207-011-0124-7.

[4] H. Brandl, "Trusted computing: The TCG Trusted Platform Module specification," 2004, http://osgug.ucaiug.org/utilisec/embedded/Shared%20Documents/Device%20Security/VariousInputs/ShrinathInputs/Basic_Knowledge_EC2004.pdf.

[5] D. G. Berbecaru and S. Sisinni, "Counteracting software integrity attacks on IoT devices with remote attestation: a prototype," in *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, October 2022, pp. 380–385, doi: 10.1109/ICSTCC55426.2022.9931765.

[6] TCG, "Hardware Requirements for a Device Identifier Composition Engine," 2018, https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf.

[7] D. Lee et al., "Keystone: An open framework for architecting trusted execution environments," in *Fifteenth European Conference on Computer Systems*. Heraklion (Greece): ACM, April 2020, pp. 1–6, doi: 10.1145/3342195.3387532.

[8] ETSI, GRNFV, "ETSI GR NFV-SEC 003 V1.2.1: Network Functions Virtualisation (NFV); NFV Security; Security and Trust Guidance," August 2016, https://www.etsi.org/deliver/etsi_gr/NFV-SEC/001_099/003/01.02.01_60/gr_NFV-SEC003v010201p.pdf.

[9] M. De Benedictis and A. Lioy, "A proposal for trust monitoring in a network functions virtualisation infrastructure," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. Paris (France): IEEE, June 2019, pp. 1–9, doi: 10.1109/NETSOFT.2019.8806655.

[10] "The Trusted Computing Group," https://trustedcomputinggroup.org/.

[11] TCG, "TPM main part 1 design principles," TCG White Paper, 2011, https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf.

[12] TCG, "Trusted Computing Group Protection Profile PC Client Specific Trusted Platform Module TPM Family 1.2," 2014, https://www.commoncriteriaportal.org/files/ppfiles/pp0030b.pdf.

[13] TCG, "Protection Profile PC Client Specific TPM Library Specification Family 2.0," 2021, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClient_PP_1p3_for_Library_1p59_pub_29sept2021.pdf.

[14] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Helsinki (Finland), August 2015, pp. 57–64, doi: 10.1109/Trustcom.2015.357.

[15] V. Costan and S. Devadas, "Intel SGX explained," 2016, https://eprint.iacr.org/2016/086.pdf.

[16] S. Pinto and N. Santos, "Demystifying ARM TrustZone: A Comprehensive Survey," *ACM Comput. Surv.*, vol. 51, no. 6, January 2019, doi: 10.1145/3291047.

[17] A. Seshadri et al., "Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems," in *20th ACM Symposium on Operating Systems Principles*, ser. SOSP '05, Brighton (UK), October 2005, p. 1–16, doi: 10.1145/1095810.1095812.

[18] C. Nie, "Dynamic root of trust in trusted computing," TKK T1105290 Seminar on Network Security, Citeseer, October 2007, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a446b7b281c55d4e94bb79c034cf3f2d86bd3267.

[19] K. Eldefrawy, N. Rattanavipanon, and G. Tsudik, "Hydra: Hybrid design for remote attestation (using a formally verified microkernel)," in *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '17, Boston (MA, USA), July 2017, p. 99–110, doi: 10.1145/3098243.3098261.

[20] K. Eldefrawy et al., "Smart: secure and minimal architecture for (establishing dynamic) root of trust," in *NDSS 2012, 19th Annual Network and Distributed System Security Symposium*, vol. 12, San Diego (CA, USA), February 2012, pp. 1–15.

[21] L. Qian et al., "Cloud Computing: An Overview," in *CloudCom: IEEE International Conference on Cloud Computing*, Beijing (China), December 2009, pp. 626–631, doi: 10.1007/978-3-642-10665-1_63.

[22] G. Boss et al., "Cloud computing," IBM white paper, October 2007, https://www.academia.edu/download/30844302/Cloud_computing_wp_final_8Oct.pdf.

[23] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 677–692, May 2019, doi: 10.1109/TCC.2017.2702586.

[24] A. M. Alwakeel, A. K. Alnaim, and E. B. Fernandez, "A Survey of Network Function Virtualization Security," in *SoutheastCon 2018*. St. Petersburg (FL, USA): IEEE, October 2018, pp. 1–8, doi: 10.1109/SECON.2018.8479121.

[25] N. Schea et al., "Bootstrapping and maintaining trust in the cloud," in *32nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. Los Angeles (CA, USA): ACM, December 2016, p. 65–77, doi: 10.1145/2991079.2991104.

[26] M. Ersue, "ETSI NFV management and orchestration-An overview," Presentation at the IETF, 2013, https://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf.

[27] E. Bravi, "Use of Trusted Computing Techniques to Counteract Cybersecurity Attacks in Critical Infrastructures," MSc Thesis, Politecnico di Torino, Italy, December 2022, https://webthesis.biblio.polito.it/24553/.

[28] "The Quart project," https://quart.palletsprojects.com/en/latest/index.html.

[29] The asyncio library, https://docs.python.org/3/library/asyncio.html.

[30] The Flask project, https://flask.palletsprojects.com/en/latest/.

[31] The Apache Kafka Project, https://kafka.apache.org/.

[32] C. Piras, "TPM 2.0-based Attestation of a Kubernetes Cluster," MSc Thesis, Politecnico di Torino, Italy, December 2022, https://webthesis.biblio.polito.it/24507/.

[33] C. Li et al., "An Implementation of Trusted Remote Attestation Oriented the IaaSCloud," in *ISCTCS: International Conference on Trustworthy Computing and Services*. Beijing (China): Springer, June 2012, pp. 194–202, doi: 10.1007/978-3-642-35795-4_25.

[34] M. Ambrosin et al., "Collective Remote Attestation at the Internet of Things Scale: State-of-the-Art and Future Challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2447–2461, 2020, doi: 10.1109/COMST.2020.3008879.

[35] H. Lauer and N. Kuntze, "Hypervisor-Based Attestation of Virtual Environments," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. Toulouse (France): IEEE, 2016, pp. 333–340, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0067.