# ETSI GR NFV-SEC 007 V1.1.1 (2017-10)

**Network Functions Virtualisation (NFV);**
**Trust;**
**Report on Attestation Technologies and Practices for Secure Deployments**

Reference

DGR/NFV-SEC007

Keywords

NFV, trust services

# Contents

# Intellectual Property Rights

# Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

# Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# 1 Scope

The present document discusses existing attestation technologies and practices, as applicable to NFV systems, addressing:

- The identification and definition of levels of assurance
- The assumed capabilities from the NFVI (e.g. TPM, HSM, etc.)
- Operational procedures
- A gap analysis of current (established or newly proposed) attestation technologies
- Recommendations for follow-on PoCs to demonstrate feasibility of the attestation procedures

Given the current status of attestation technologies and standards, the present document is applicable to hypervisor-based NFV deployments only, with the exception of some of the perspectives analysed in clause 8.

# 2 References

## 2.1 Normative references

Normative references are not applicable in the present document.

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS NFV 003 (V1.2.1) (2014-12): "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] ETSI GS NFV 002 (V1.2.1) (2014-12): "Network Functions Virtualisation (NFV); Architectural Framework".

[i.3] ETSI GS NFV-SEC 012 (V3.1.1) (2017-01): "Network Functions Virtualisation (NFV) Release 3; Security; System architecture specification for execution of sensitive NFV components".

[i.4] TCG, PC Client WG: "PC Client Specific Implementation Specification for Conventional BIOS", V1.21 Errata, rev 1.0, 2012-02.

[i.5] TCG, Infrastructure WG: "TCG Attestation, PTS Protocol: Binding to TNC IF-M", V1.0, rev 28, 2011-08.

[i.6] ETSI GR NFV-SEC 009 (V1.2.1) (2017-01): "Network Functions Virtualisation (NFV); NFV Security; Report on use cases and technical approaches for multi-layer host administration".

[i.7] ETSI GR NFV-SEC 003 (V1.2.1) (2016-08): "Network Functions Virtualisation (NFV); NFV Security; Security and Trust Guidance".

[i.8] Unified Extensible Firmware Interface Forum: "Unified Extensible Firmware Interface Specification", V2.7, 2017-05.

[i.9] NIST SP800-155,draft, 2011-12: "BIOS Integrity Measurement Guidelines".

[i.10] TCG PC Client WG: "TCG EFI Platform Specification", V1.22, rev 15, 2014-01.

[i.11] TCG Virtual Platform WG: "Virtualised Trusted Platform Architecture Specification", V1.0, rev 0.26, 2011-09.

[i.12] TCG TPM WG: "Trusted Platform Module Library Specification", V1.38, 2016-09.

[i.13] Andre Rein, 2017: "DRIVE: Dynamic Runtime Integrity Verification and Evaluation", Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.

NOTE: Available at http://dl.acm.org/citation.cfm?id=3052975.

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI GS NFV 003 [i.1] apply.

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.1] and the following apply:

| | |
|---|---|
| CoT | Chain of Trust |
| CRTM | Core Root of Trust for Measurement |
| HBRT | Hardware Based Root of Trust |
| LoA | Level of Assurance |
| RoT | Root of Trust |
| SML | Stored Measurement Log |
| TPM | Trusted Platform Module |

# 4 Attestation Procedures

## 4.0 Introduction

Both authentication (a process of ensuring that the computing platform can prove that it is what it claims to be) and attestation (a process of proving that a computing platform is trustworthy and has not been breached) are necessary steps to ensure secure computing in NFV environment. Attestation procedures create assurances of computing platform's integrity state and ability to protect data in accordance with policy.

## 4.1 Basic Concepts

### 4.1.1 Roots of Trust

#### 4.1.1.1 Overview

The trust status of a computing platform can be determined by a remote party only by using inherently trusted primitives embedded into that platform. These primitives are called Roots of Trust (RoTs). The RoTs are expected to behave always according to their predefined purpose, as no other mechanism is available to fully check their behaviour.

The RoTs are ideally implemented in hardware or protected by hardware mechanisms and provide very specific services to the computing platform they are serving. For attestation of a computing platform, three main services types are required to be supported by its RoTs:

- Protection of cryptographic material (e.g. keys)
- Isolated execution of cryptographic operations
- Bootstrapping code measurement

#### 4.1.1.2 Hardware Based Root of Trust

In NFV deployments it is expected that the virtualisation layer (i.e. hypervisor) will make use of a Hardware Based Root of Trust (HBRT). The HBRT should be implemented in a hardware component that fulfils the requirements defined in ETSI GS NFV-SEC 012 [i.3]. The HBRT provides a subset of the services required for enabling a remote party to compute the trust status of the virtualisation host.

#### 4.1.1.3 RoT for virtualised platforms

Unlike the virtualisation layer, which is expected to run directly on the hardware of the compute node, the VNFCIs will run on virtualised platforms. They may run in an execution environment created by the hypervisor based on dedicated hardware support.

The same principle applies to the RoTs available to the virtualised platform: the virtual platform can make use of dedicated hardware features provided by the HBRT (as defined in ETSI GS NFV-SEC 012 [i.3]), but the hypervisor is involved in configuring this access and sometimes transferring messages between VNFCIs and their associated hardware rooted vRoTs. Therefore, a vRoTs represents the combination of hardware functionality provided by the HBRT and the relevant components of the hypervisor that configure and mediate access to those functionalities.

In NFV deployments, it is highly desirable to restrict as much as possible the influence of the hypervisor on the vRoTs. Coupled with the host hardening requirements of ETSI GS NFV-SEC 012 [i.3], this would ensure the best available protection for the vRoTs.

If HMEE technology is used to host and protect virtual RoTs, one possibility to integrate them in the CoT is to tether them to their corresponding HBRT implementation (e.g. TPM, HSM).

#### 4.1.1.4 Security services of RoTs

A RoT provides one or more security services to the platform, e.g. software measurement service for the Root of Trust for Measurement (RTM), software measurement and measurement validation service for the Root of Trust for Verification (RTV), access controlled and tamper evident or tamper resistant protected storage service for the Root of Trust for Storage (RTS), certification service (providing cryptographic proof that a set of data originates from the RTS) for the Root of Trust for Reporting (RTR).

The term RTM is used in the present document to represent the origin of the Chain of Trust (CoTs) (see clause 4.1.2), as the present document is primarily focused on exploring Remote Attestation technologies. Wherever Secure Boot/Local Attestation is instead referenced within the present document, it should be assumed that the origin of the CoT is, in that case, the RTV.

### 4.1.2 Chain of Trust

A CoT, also known as a Transitive CoT, is used to infer trust in the measurement data of the software component that represents the last link of the chain. Trust is initially only bestowed on the first link in the chain, the Root of Trust for Measurement (RTM).

Starting from the RTM the principle of "first measure and record, then execute" is applied by all software components that are executed on the given platform. It ensures that all software components are measured at load time and cannot tamper with their own measurement procedure. The RTM performs the first measurement, which will be implicitly trusted, because of the defining property of the RTM - immutability (as defined by TCG PC Client Specific Implementation Specification for Conventional BIOS [i.4]). Thus trust can be transferred from the first measurement to the measurement of the last software component in the chain. This process of building a measurement log is also referenced as a Measured Boot process.

When regarded from bottom to top, from the RTM to each of the measurement endpoint software components, this process resembles a tree (of transitive trust), while, when regarded from top to bottom, from the last measured software component to the RTM, it constitutes a CoT.

The CoT is a purely logical construct. It can be explicitly constructed by the Remote Verifier during a Remote Attestation (RA) process as long as the target platform implements either only Measured Boot or both Measured Boot and Secure Boot. For this purpose the data in the measurement log is checked against golden measurements. In case of a match, a new leaf is added to the CoT. Starting from the first mismatch, the data in the measurement log can no longer be trusted and the CoT cannot be expanded further.

When only Secure Boot is implemented by a platform (without Measured Boot), an independent observer (similar to the Remote Verifier used in RA) cannot reconstruct and verify the CoT. It can only trust that the running code was loaded as the last leaf of a properly rooted and locally (on the target platform) constructed CoT.

An example of such a tree/CoT is depicted in Figure 1 (based on TCG Attestation, PTS Protocol: Binding to TNC IF-M [i.5]).

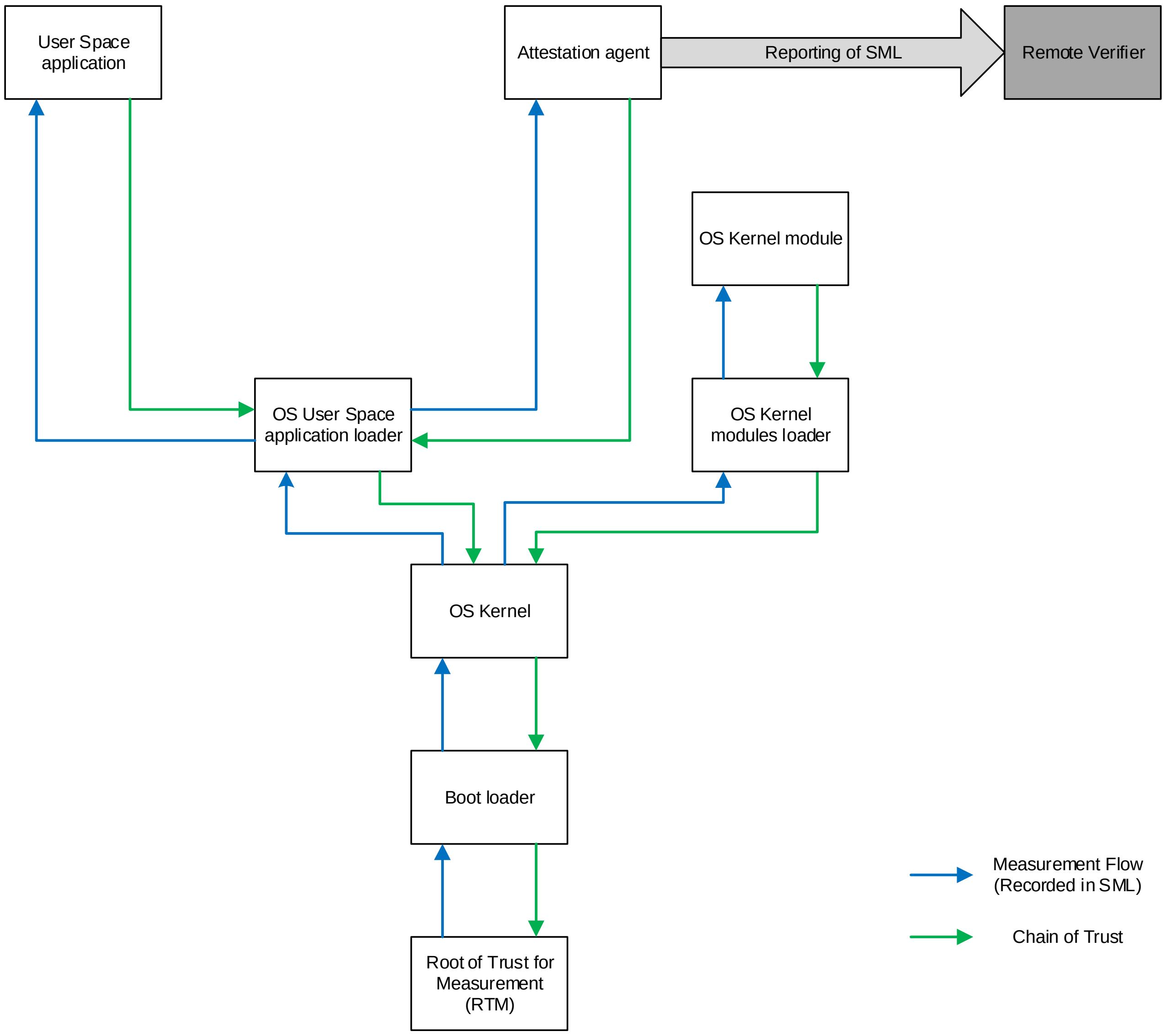


**Figure 1: System Services Chain of Trust and Attestation**

## 4.1.3 Attestation

Attestation is the process through which a remote challenger can retrieve verifiable information regarding a platform's integrity state (as described in TCG PC Client Specific Implementation Specification for Conventional BIOS [i.4]). It is also commonly referenced as Remote Attestation, to highlight that the verification of integrity information is performed by an independent party in a different trust domain.

The platform's integrity information is delivered to the remote challenger in the form of a measurements log. However, as specified in TCG PC Client Specific Implementation Specification for Conventional BIOS [i.4], the information in the measurements log alone is not sufficient to enable a trustworthy assessment of the platform's integrity state. The measurement log is generated by the very software running on the platform being assessed. Therefore, trust in data contained in the measurements log is ensured only if:

- A Chain of Trust (CoT) is established from the platform boot up to any given running application being attested (as defined in TCG Attestation, PTS Protocol: Binding to TNC IF-M [i.5]).

- Evidence of measurements log data protection from local tampering is provided.

As explained in clause 4.1.2, any given software component participating to the chain of trust cannot influence its own measurement procedure, as its execution begins only after it has been measured. However, a CoT does not provide assurances that already recorded measurements have not been tampered at a later time. For this purpose a Root of Trust for Reporting (RTR) is required.

A RTR needs to be able to create cryptographic evidence that the data in the measurements log originates from a RTS and has not been tampered.

A TPM (see clause 6.1) is an example of implementation that could provide RTR and RTS by leveraging the specific tampering detection properties of its Platform Configuration Registers (PCR) and issuing signed quotes of their content (as described in TCG Attestation, PTS Protocol: Binding to TNC IF-M [i.5]).

An HSM (see clause 6.2), is another example of implementation that could provide RTR and RTS, using its capability to provide integrity and confidentiality and cryptographic processing.

Upon receiving a measurements log and the appropriate evidence that its contents has not been tampered with, the remote challenger can determine, in a trustworthy manner, the platform's integrity status. For this purpose the remote party uses reference measurements.

### 4.1.4 Supporting Technologies

#### 4.1.4.1 Measured Boot

As described above, trust in the attestation data is dependent on the establishment of a CoT starting from a RTM. The initialization of this CoT is performed by a measured boot process (according to ETSI GR NFV-SEC 009 [i.6]). The process ensures that all software components starting from the RTM are measured before execution and their measurements recorded in the measurement log, which is integrity protected by the RTR.

Typically, a measured boot covers the measurement of all software components that are executed sequentially, on a linear execution flow, as part of the boot process. This implies constructing the CoT up to, at minimum, the Operating System (OS) kernel. The rest of the CoT from the OS kernel up to any given user space application executed in the system is constructed through load-time measurement.

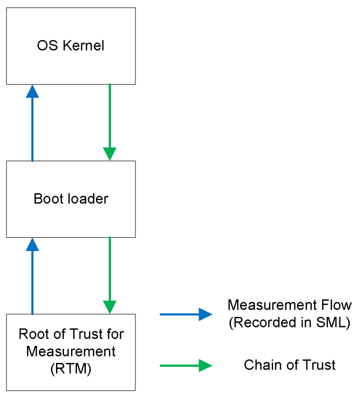

OS Kernel
Boot loader
Root of Trust for Measurement (RTM)
Measurement Flow (Recorded in SML)
Chain of Trust

**Figure 2: Measured boot CoT**

#### 4.1.4.2 Load-Time Measurement

Upon the OS kernel being launched, the execution flow of software components (i.e. applications and kernel modules) is no longer linear. Multiple application and kernel modules can be loaded and executed in parallel and no guarantees as to the order of execution can be provided.

Extending the CoT from the linearly executed boot software components to any given parallel launched user space application can also be accomplished by enforcing the measurement of all applications and modules at load-time, before they are executed, as illustrated in Figure 3. This requires dedicated support, usually within the OS kernel, to ensure that all measurements are properly collected and recorded in the integrity protected SML. It is also sometimes referred to as a "Measurement Architecture".

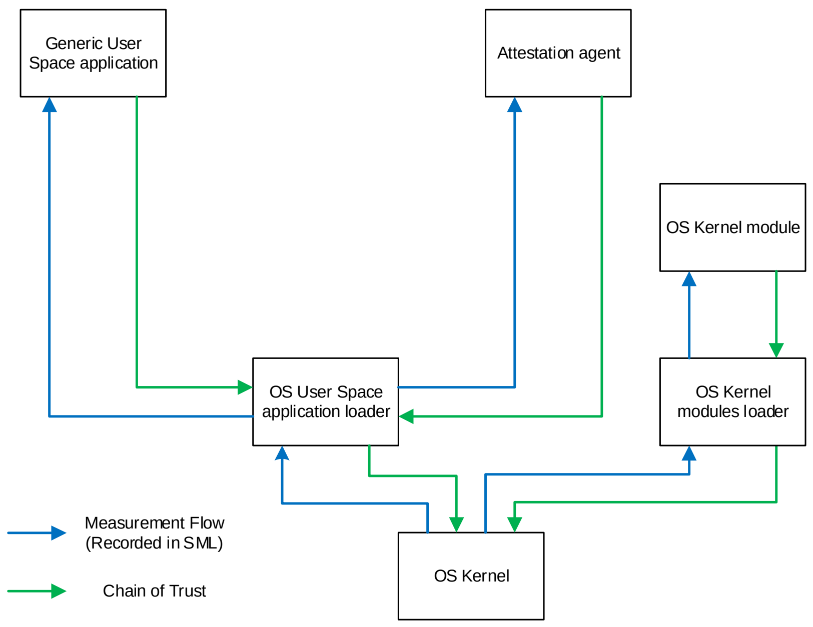

Generic User Space application
Attestation agent
OS Kernel module
OS User Space application loader
OS Kernel modules loader
OS Kernel
Measurement Flow (Recorded in SML)
Chain of Trust

**Figure 3: Load-Time CoT**

## 4.2 Enforcement of System Integrity

Attestation provides only the means to remotely detect platform integrity compromise. For cases where local policy enforcement based on integrity information is desired, the principles of Secure Boot, as opposed to Measured Boot or Trusted Boot (as defined by ETSI GR NFV-SEC 003 [i.7]) are more suitable.

Secure Boot or Verified Boot leverages typically digital signature over software components to detect and block execution of any component that is found to have been compromised. The same principle of "first measure, then execute" is applied. The only difference is that a supplementary step is performed after measurement and before execution: checking that the current software component measurement matches the measurement recorded in the digital signature (usually called the reference measurement). Upon detecting a mismatch between the two measurements, execution of the component is blocked and an appropriate recovery procedure need to be activated. It should be noted that any recovery procedure should be performed only by a component that is already part of the CoT established using Secure Boot, or is booted later using Secure Boot before control of CoT is transferred to it to initiate the recovery process. Otherwise the CoT is broken at the point of failure. An example of recovery procedure might be for the loader to try to revert back to the last known successfully securely booted image to continue the Secure Boot process and report the failure to boot the new process.

Similar to the process used for creating a CoT for attestation, an implicitly trusted component has to exist, which starts the chain of measurements and digital signatures validation (i.e. a CRTM) (as described in the Unified Extensible Firmware Interface Specification [i.8]). Also, the principles behind Secure Boot can be further leveraged to extend the chain of signature based integrity validation up to the last software component in the execution tree, similarly to the CoT for attestation (as described in ETSI GR NFV-SEC 003 [i.7]). This type of process is also known as Local Attestation.

Software components signature validation (extended Secure Boot) cannot provide the necessary data to allow a remote challenger to determine the integrity of the platform. However, the (extended) Secure Boot procedure can be complemented with a procedure for creating a CoT for attestation as described in clause 4.1. The two procedures are not alternatives to each other, but complementary to each other.

## 4.3 Trustworthy Platform Configuration

The trustworthiness of a platform state measured through a process as described in clauses 4.1.4.1 and 4.1.4.2, and verified through attestation (see clause 4.1.2) and/or locally enforced (i.e. local attestation) through a process as described in clause 4.2, can also be influenced by the platform's hardware and software configuration, especially during boot.

Attestation mainly focuses on verifying the integrity of the software (SW) that is executed on that platform. Attestation may lead to integrity corrective actions. The platform hardware (HW) configuration can disrupt the software integrity measuring and/or enforcement process, by effectively breaking the CoT (Chain of Trust) that infers trust in the integrity of the software executed (as described in NIST SP800-155 [i.9]).

In order to attain a high level of trust in the platform's state, the platform's HW configuration should also be verified (during remote attestation and/or local enforcement). To this end, remote attestation and/or local enforcement should be complemented by mechanisms that provide measurements and/or enforce the platform's HW configuration.

Platform HW configuration, which might have an impact on the platform's software integrity measurement and/or enforcement, may include (as defined also by NIST SP800-155 [i.9]), but is not limited to:

- Hardware configuration details:
  - List of attached devices (internal and external)
  - State of hardware interfaces (e.g. USB ports)
  - Option ROMs - typically firmware designed to assist with a specific device initialization/configuration
  - Nonvolatile configuration items (e.g. fuse, state-retention flipflops)
- Boot flow configuration events:
  - Boot flow interruption by user/administrator (e.g. through login)
  - Boot flow changes (e.g. boot from different image)

In the case of using remote attestation to verify the trustworthiness of a platform's boot process, the specifications of the TCG (TCG PC Client Specific Implementation Specification for Conventional BIOS [i.4], TCG EFI Platform Specification [i.10]) explicitly detail the options to extend the measurements also over the platform HW configuration. The platform HW configuration measurements will then be retrieved together with the software integrity measurements by the remote verifier (see clause 4.1.2). Thus, upon performing a validation of the platform's software integrity, the remote verifier can also factor in the additional data on the platform HW configuration integrity.

Enforcement (local attestation) of platform HW configuration is not fully supported by relevant industry specifications (e.g. UEFI Secure Boot [i.8] supports just Option ROMs signature checking). This represents a gap between the current industry standards and NFV security requirements. That is not to say the already defined mechanisms cannot be extended to ensure full enforcement of a platform configuration. Already available hardware features can be leveraged for this purpose (e.g. Dynamic Root of Trust for Measurement (DRTM)).

## 4.4 Remote Attestation of VNFs

### 4.4.1 Introduction

Remote Attestation can be used as a tool for checking the LoA (see clause 5) of VNFs, by a Remote Verifier. The Remote Verifier can determine the trust status of a VNF, within a certain LoA, upon receiving qualifying data through the Remote Attestation process. According to ETSI GR NFV-SEC 003 [i.7] the qualifying data for assigning trust in a VNF derives from three main sources:

- Signed VNF package
- Hypervisor software integrity state
- VNFCs software integrity state

The VNF package integrity and provenance data is a source of data for trust establishment, but it is not meant to be transferred through a Remote Attestation to the Remote Verifier for validating. Instead, as described in ETSI GR NFV-SEC 003 [i.7], this data is meant to be validated before the instantiation of the VNFC. Nevertheless, Remote Attestation can be perhaps used to validate the trust status of the component, which performs the checks on this data.

The hypervisor and VNFC's software integrity state are both suitable for collection through Remote Attestation. Based on the hypervisor software integrity state, the Remote Verifier can assign a trust level to the hypervisor operations, including the setup and execution of the VNFCs virtualisation containers. Together with the VNF package integrity and provenance data this infers a certain trust level in the instantiation of the VNFCs.

Depending on the LoA, which a specific VNF is expected to meet, its constituting VNFC's software integrity validation through Remote Attestation might also be required. As also described in ETSI GR NFV-SEC 003 [i.7], this is a mechanism that enables validation of the trust level of a VNF after instantiation.

Figure 4 highlights the coverage of the two different sets of data - hypervisor and VNFC integrity state data - which can be collected through Remote Attestation to enable validation of the VNF trust status. The execution flow is built according to ETSI GR NFV-SEC 003 [i.7].

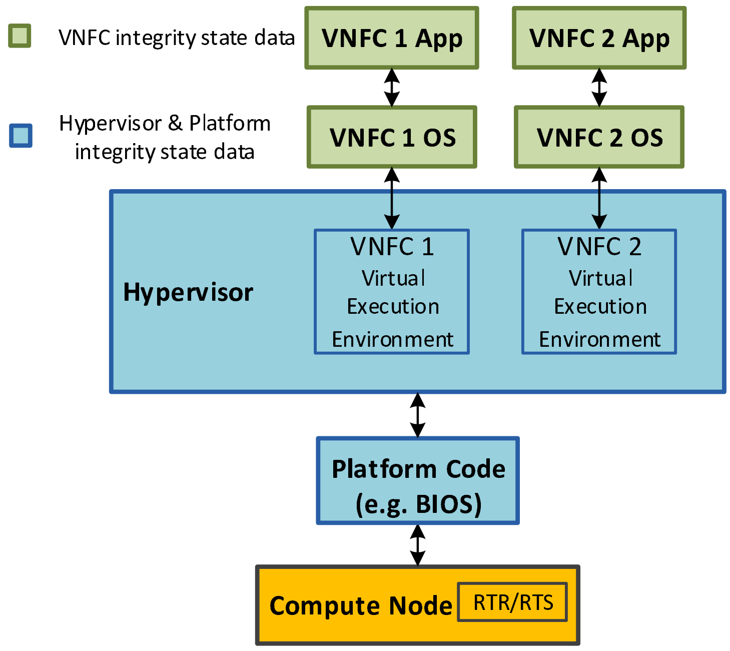

VNFC integrity state data
Hypervisor & Platform integrity state data
VNFC 1 App
VNFC 2 App
VNFC 1 OS
VNFC 2 OS
Hypervisor
VNFC 1 Virtual Execution Environment
VNFC 2 Virtual Execution Environment
Platform Code (e.g. BIOS)
Compute Node
RTR/RTS

**Figure 4: Coverage of integrity data**

### 4.4.2 Known Challenges

Attestation of a VNF implies reporting and verification steps for all of the constituting software components. These include the primary software building blocks of any VNF, the VMs that the VNF is running on top of (according to ETSI GS NFV 002 [i.2]), together with the corresponding infrastructure components (i.e. virtualisation layer). The core of this process is, therefore, the employed VM attestation model.

For attestation, each VM can leverage the virtual trusted platform instance that is uniquely associated to it, which includes security capabilities such as virtual Root of Trust for Measurement (vRTM), virtual Root of Trust for Reporting (vRTR) or virtual Root of Trust for Storage (vRTS), as defined by TCG's Virtualised Trusted Platform Architecture Specification [i.11]. These capabilities can be provided by the virtualisation layer in the form of a virtual Root of Trust for Measurement (vRTM) and virtual RTS and virtual RTR implementation.

The purpose of the virtual Roots of Trust (vRoTs) is to infer trust in the reporting of VM software measurements through the establishment of a chain of trust originating in these vRoTs. As the vRoTs do not belong to the same execution domain as the VM software on which they report, an implicitly higher level of trust is assigned to their functionality. This is similar to the transitive chain of trust established for attestation in non-virtualised platforms, which is based on hardware protected RoTs (e.g. physical TPM - pTPM, HSM).

The virtualisation layer, typically embodied as a hypervisor (according to ETSI GS NFV 002 [i.2]), maintains the potential to influence the functionality of the vRoTs. An accurate assessment of the VM state can, therefore, only be performed, if the hypervisor's chain of trust is also validated as part of the VM attestation. This process is called deep attestation (in TCG's Virtualised Trusted Platform Architecture Specification [i.11]) and it allows trust to be extended from the hardware protected RoTs of the computing node, through the hypervisor established chain of trust, to the vRoTs implementation and from there forward, through the VM chain of trust, to the VNF.

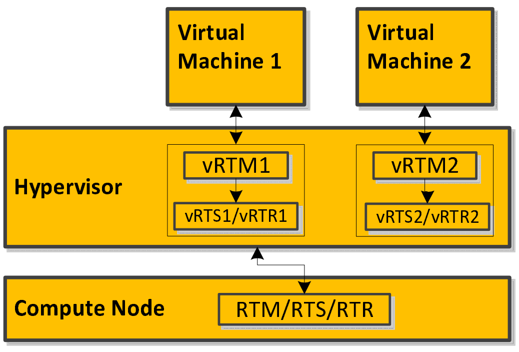


**Figure 5: Roots of Trust in a NFV deployment**

Deep attestation implies retrieval and subsequent validation, by a Remote Verifier, of the attestation data from the VM and the attestation data of the underlying hypervisor. From the Remote Verifier's perspective these two attestation reports can be provided from two independent entities. Therefore, the Remote Verifier requires proof that the hypervisor, which it is attesting as part of a deep attestation, is actually executing the vRoTs instances (i.e. vRTM, vRTS, vRTR instances) uniquely associated with the VM that is the main target of the deep attestation. The problem of creating an explicit and verifiable binding between the VM attestation and the attestation of the hypervisor on which the VM's vRoTs instances are executing is also known as the layer binding problem (in TCG's Virtualised Trusted Platform Architecture Specification [i.11]).

The TCG Virtualised Platform WG developed an in-depth analysis of the requirements for attestation of VMs in Virtualised Trusted Platform Architecture Specification [i.11]. The WG has focused mainly on two deep attestation models, which differ in the way hypervisor attestation data is reported: through the existing VM attestation channel or through a separate, dedicated channel.

Besides the mechanisms that enable trust in the attestation data provided by each of the models under review, the issue of scalability of the models has also to be analysed. It can be foreseen that in a NFV deployment, attestation of multiple VNFs will be performed simultaneously. This implies a possible attestation of all or a large set of VMs running on the same hypervisor. Therefore, one will ensure that the attestation of the hypervisor does not become a bottleneck for the entire VNF attestation process.

### 4.4.3 Single-Channel VM-Based Deep Attestation

In this model the Remote Verifier will establish a communication channel with the VM through which it will request both the VM specific attestation data and also the attestation data of the hypervisor on which the VM is running.

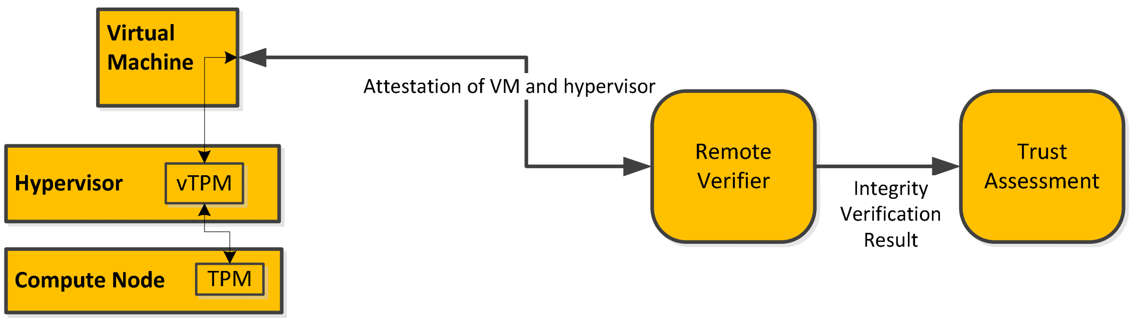


**Figure 6: Single-channel, VM-based attestation**

The issue of the layer binding can be solved by including, in the attestation data of the hypervisor, vTPM specific data, thus providing verifiable cryptographic proof of layer binding.

An attestation of the hypervisor will be retrieved for each attestation of the VM through an additional set of vTPM functionalities (i.e. new API or extensions to existing API).

In terms of scalability, this model is not suitable for deployments in which large numbers of VMs are running on the same hypervisor. As the hypervisor attestation report is based on hardware protected RoTs, it means that the generation of this report might be time consuming. Furthermore, generation of multiple reports cannot be performed in parallel, but only in a sequential manner.

### 4.4.4 Multiple-Channel Independent Deep Attestation

In this model, the Remote Verifier will establish a communication channel with the VM to retrieve its attestation data and, subsequently, it will establish another, independent, communication channel with the hypervisor to retrieve its attestation data.

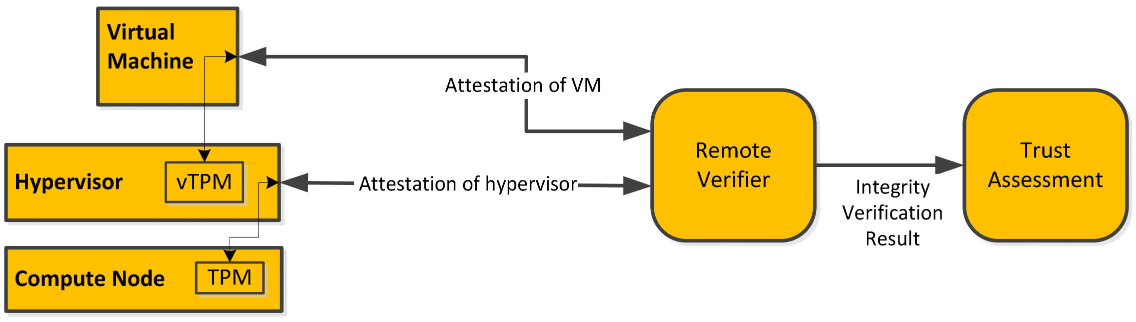


**Figure 7: Multiple-channel, independent attestation**

The binding between the attestation data of the different layers is very loose. The management data regarding the VM execution environment (i.e. to which compute node it is allocated) is the only information that connects the two attestation data reports. This is not sufficient to enable complete trust in the reported attestation data.

However, this model scales very well for deployments that imply large number of VMs that are running on the same hypervisor. The Remote Verifier (which can be tightly integrated with the management system) can, for the sake of scalability, first perform an attestation of all VMs that are of interest and are running on a specific hypervisor. Subsequently, once all VM attestations are completed, it can perform a, one time, attestation of the hypervisor. In this way, the attestation of the hypervisor is no longer a bottleneck.

# 5 Levels of Assurance

## 5.0 Introduction

The NFV environment includes not only all the infrastructural elements, but also the software components running on them, and especially the VNFs actually deployed. Attestation of computing platform's integrity (i.e. measurement and verification) leads to the ability to establish information security assurance. Such security assurance directly translates into trust in a computing platform's capability to protect its information and functional assets, and to attest to those protections. Since many of the attestation steps described in the present document (refer in particular to clause 7) may be costly in terms of platform requirements for deployment and computational resources during runtime, to properly assess the security assurance it is necessary to map the results of the attestation procedures to a predefined scale, and decide whether it is sufficient to address the threat relevant for the service to be provided. This scale consists of a set of specific Levels of Assurance (LoA) that would be applicable to establish the trustworthiness of a particular set of components (e.g. system or platform), according to the nature of the requested service, the threats being considered, and the applicable policies at all levels, from legal requirements to commercial SLAs. This clause proposes a set of LoAs, including the verification applicable to achieve each one, and discusses issues related to the assurance on attestation mechanisms.

Six Levels of Assurance are defined, named with a number that characterizes each one in a relative scale of trust, where a higher number implies a higher degree of trust. These LoAs are:

1) LoA 0 indicates no integrity checking of any kind is required.

2) LoA 1 only requires the integrity of hardware and virtualisation platform (hypervisor) to be verified locally, based on signatures, during boot and application loading. No integrity evidence is provided. Integrity status is derived from platform state after the end of boot and application load processes (see clause 4.2).

3) LoA 2, in addition to the checks required by LoA 1, requires the remote verification of hardware and virtualisation platform integrity by means of remote attestation. Boot time and application load time measurements are employed (see clause 4.1.4).

4) LoA 3, in addition to the checks required by LoA 2, requires the local verification of VNF software packages, based on signatures, as they are loaded. Signatures should be verified for all packages loaded during VNF startup and subsequently, when new packages are loaded (i.e. during VNFCI boot and VNFCI application loading).

5) LoA 4, in addition to the checks required by LoA 3, requires the remote verification, by means of remote attestation, of VNF software packages. VNFCI boot time measurements and application load time measurements should be used.

6) LoA 5, in addition to the checks required by LoA 4, requires the remote verification of the infrastructure network deployed to support the VNF and the remote verification of the virtualisation layer and VNF software packages integrity state during run-time (i.e. post load time). As these measures can be implemented independently, LoA5 can be split in two non-hierarchical sublevels:

    a) LoA5a, when infrastructure network remote verification, by means of remote attestation, is employed in addition to the checks required by LoA4;

    b) LoA5b, when run-time integrity state remote verification of virtualisation layer and VNF software packages, by means of remote attestation, is employed in addition to the checks required by LoA4.

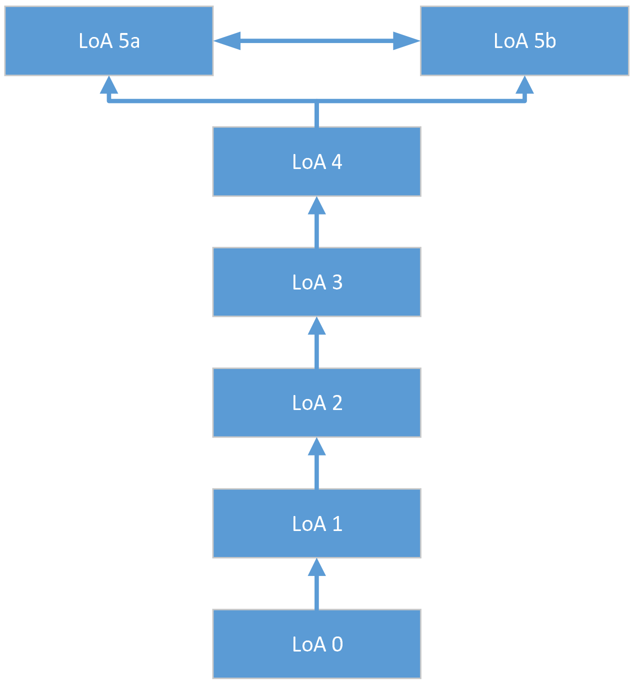


**Figure 8: LoA hierarchy**

In LoAs 2 to 5 procedures can be applied only at boot time ("boot LoA") or at periodic intervals of time ("continuous LoA with a period of X").

While LoAs 1 to 4 are achievable with current technologies, and the applicable operational procedures discussed in clause 7, LoA 5 is currently subject of research, with some initial results discussed in clause 8, together with other techniques that could enable potential higher LoAs.

## 5.1 Attestation and Assurance

### 5.1.0 Introduction

MANO operators and Service operators need assurance that their VNFs are running securely so in order to achieve this they need to know that the host platform is trustworthy and that the VNF is securely launched and continues to run securely in VNFCIs on the platform. Such secure platform has to provide ways for VNFs to verify trust-related information about the NFV platform. Security-mindful MANO operators and Service operators may wish to:

1) Receive or perform an attestation of the platform and setup of a VNFCI for the VNFM. For that, VNFM needs to be provided with a desired level of trust in hardware and/or the virtualisation layer during the initial VNFCI instantiation, as well as throughout the entire VNFCI life cycle (i.e. migration, cloning, suspension, and resumption).

2) Receive or perform an attestation of the VNFCI running on the NFVI node.

### 5.1.1 Platform Attestation

MANO operators and Service operators need to be provided with a desired level of trust in the setup of a VNFCI. e.g. the host platform, host platform identity, the virtualisation layer, the established layer during the initial VNFCI instantiation, as well as throughout the entire VNFCI life cycle (i.e. migration, cloning, suspension, and resumption) and a virtual TPM function within the VM. Such a secure platform has to provide ways for VNF custodians to verify trust-related information about the setup of the VM. It is important that the VM operations and data are isolated from the VM provider and other VM tenants.

There are two methods of attesting a secure VM launch to MANO operators and Service operators:

- the VM provider can deploy a trusted cloud and attest its trustworthiness to VNF custodians; or
- trustworthiness statements can be relayed to VNF custodians by an independent trusted third party. For this approach, the trusted third party needs to be able to collect and assess evidence of trustworthiness from the VM providers. Given the dynamic nature of the NFV and the diverse set of hardware, operating systems (OSs) and VM managers (VMMs) used, some basic platform assurance and certification provenance is needed.

### 5.1.2 Virtual Machine Attestation

Since the VNFCI is to be isolated from the actors (MANO operators and other Service operators) and Infrastructure (NFVI node and other hosted processes), the VM attestation function needs to be distinct and separate from the NFVI node attestation. An attestation of the VNF function within a VM may be performed by providing access to a hardware anchored virtual RTS and virtual RTR functions in a VM. There are two methods of attesting to the state of a VNF in a VM.

The VNF custodian can deploy an attestation server with capabilities to:

- Perform an initial attestation of the platform on which the VNF is hosted (as described in clause 5.1.1).
- Perform an attestation of the VNF function running on the VM.

Trustworthiness statements can be relayed to MANO operators and Service operators by an independent trusted third party hosting an attestation server function on behalf of the MANO operators and Service operators.

# 6 Infrastructure Capabilities

## 6.0 Introduction

This clause briefly lists some of the components which may be provided by the infrastructure and may provide capabilities relevant to attestation. Note that there is no expectation that all of these components will be part of the NFVI - they may be spread through various parts of the deployment architecture.

It is not expected that any single component will be able to provide appropriate levels of attestation, and the various methods and techniques (including protocols) that can be used to combine their capabilities will be (and need to be) different to suit the requirements of different LoAs.

## 6.1 Roots of Trust

### 6.1.1 Overview

To establish a hardware-based chain of trust (i.e. to ensure that the system is in a good state), three Roots of Trust that are inherently trusted are required:

1) Root of Trust for Measurement (RTM)

2) Root of Trust for Storage (RTS)

3) Root of Trust for Reporting (RTR)

### 6.1.2 Root of Trust for Measurement

The RTM is responsible for taking measurements (pertaining to the integrity of the system components) and sending them to the RTS to establish a chain of trust. The RTM may be static or dynamic. Either way, the CPU is the root. In the static case, the platform configuration is measured once whenever the system powers on or reboots. The action is started by the CPU (in a defined state) executing a set of instructions known as the core RTM. Subsequently, the software components that are executed on the platform extend the CoT (see clause 4.1.2). In the dynamic case, system software and its configuration can be measured anytime upon request by the operating system. This is not subject to a system reboot but require the CPU to provide the dynamic core RTM (D-CRTM).

Some (hardware based) mechanisms is leveraged to implement a static or dynamic RTM. Some possibilities, according to NIST SP800-155 [i.9] are (this is by no means a comprehensive list):

- Platform start-up logic guaranteeing that the CRTM is the first code to be executed after a reset. Several methods on protecting the CRTM are also presented:
  - Factory sealed (immutable) CRTM (e.g. stored in ROM storage device).
  - CRTM stored on storage devices that provide hardware based access control mechanisms (e.g. set-until-reboot latches); this is a highly discouraged approach for hosts on which sensitive NFV workloads will be scheduled.
- Isolated execution modes of the CPUs (e.g. TEE).

### 6.1.3 Root of Trust for Storage

A RTS enables a platform to protect certain data from tampering, unauthorized disclosure or both. An implementation of a RTS is a prerequisite to building a Remote Attestation solution, as it provides confidentiality of Attestation Integrity Keys (see clause 4.1.3) and protects measurements against tampering.

Any component that provides HSM-like functionality is usually suited for use as an implementation of a RTS (e.g. TPM, HSM).

### 6.1.4 Root of Trust for Reporting

Some of the contents of a RTS (e.g. measurements) is reported through Remote Attestation. To ensure authenticity and integrity of the reported data, a RTR implementation is required.

As a RTR works in conjunction with a RTS, which provides both the data to be reported and the necessary cryptographic material (i.e. keys), the RTR is usually implemented by the same component as the RTS. Therefore, any component that provides HSM-like functionality is usually suited for use as an implementation of a both a RTS and a RTR (e.g. TPM, HSM).

### 6.1.5 Examples of implementation of Roots of Trust

#### 6.1.5.1 Trusted Platform Module

As defined by the Trusted Computing Group, a Trusted Platform Module (TPM) is a system component whose state is separate from the host system on which it reports. The TPM and the host system can interact only through the interface specified in TCG's Trusted Platform Module Library Specification [i.12]. The interface primarily allows the TPM to perform upon request certain operations on the data held within and ascertain the system state. As a result, it is possible to build a chain of trust connecting all pieces of firmware and software with hardware.

NOTE 1: A TPM is not the trusted computing base (TCB) of a system but a component that allows an independent entity to determine if the TCB has been compromised. A TCB is the collection of resources (hardware, firmware and software) responsible for enforcing the security policy of a system.

TPMs are implemented in hardware. A typical embodiment is as a separate chip attachable to a computer system through a low-performance interface (such as the Low Pin Count bus). The chip has a processor specialized in cryptographic operations and shielded storage (in terms of volatile and non-volatile memory). The host system cannot directly change the values in the TPM memory.

NOTE 2: The NFVI should support TPM 2.0 and should not support TPM 1.2. Among other things, TPM 1.2 supports only SHA-1 and RSA, while TPM 2.0 supports, in addition, a list of stronger algorithms (e.g. SHA256 and ECC) as well as cryptographic agility.

The TPM provides the RTS and RTR, but not the RTM.

The TPM can serve as the RTS given its externally inaccessible memory. In particular, the TPM memory includes a shielded area (with a limited size) to hold long-term secrets (e.g. the seeds for generating keys and the primary key for storage that can be used to wrap other secrets outside) and protect them from disclosure. It also includes a set of platform configuration registers (PCRs) to hold the measurements for different components (e.g. the BIOS, PCI ROM, boot loader, and host OS). Different sub-sets of registers could be used to hold static and dynamic measurements. The registers for static measurements may be reset only when the system powers up or receives a hard reset signal. Which PCRs to use for certain measurements is platform-specific. In general, PCRs are accessible only to the host system and access may be distinguished by privilege levels. Furthermore, PCRs may not be freely overwritten but only be reset to known values under specific conditions or be *extended*. (The value of a PCR after an *extend* operation is equal to a hash over both the existing value concatenated with the new value.) PCRs may also be used to gate access to other data stored in the TPM. For example, the TPM can only allow the use of a particular cryptographic key only if certain PCRs are in a specific state (i.e. a predefined set of PCRs have certain values, usually corresponding to software measurements considered trusted).

The RTR reports on the non-confidential contents of the RTS, most notably the contents of PCRs. Reports (or quotes) are signed. Since the confidence in a signature depends on the confidence in the signing key, the best way would be to use a manufacturer provisioned signing key and certificate specific to that TPM. The TPM specifications allow for such a key, called Endorsement Key (EK) and a certificate (Endorsement Certificate) to be deployed in the TPM during manufacturing. But using this key with the RTR creates a privacy concern. Any report recipients will know which TPM was in use.

The concern is addressed in the TPM specifications by putting restrictions on the use of the EK (i.e. it cannot be used for signing) and using attestation keys (AKs or AIKs) as a level of indirection. Specifically, a user can generate attestation keys on the TPM, have a certificate authority (external if required) certify with the necessary proof that the attestation keys reside in a TPM, and use the keys for signing reports. The necessary proof in this case is generated by using the EK. In this manner, it is ensured that the certification authority is the only entity, which is aware of the connection between a specific TPM and the AIK that it uses to sign quotes. Furthermore, the report recipient will have no problem validating and trusting a quote signature produced by an AIK, even if it cannot identify the TPM that produced it. It only needs to establish a trust relation with the certification authority.

NOTE 3: TPM 2.0 supports key derivation from three persistent, randomly generated seeds for the platform, storage, and endorsement hierarchy, respectively. This approach allows multiple encryption and signing algorithms to be supported without using up the limited storage space in the TPM.

To provide assurance that the PCR values accurately reflect the state of the platform, a certificate (known as a platform certificate) that binds the RTR and the RTM (i.e. the platform) may be used.

#### 6.1.5.2 Hardware Security Module

HW security modules (HSM) provide physical and logical protection for data and in particular for cryptographic material, such as keys. In addition they offer highly secured cryptographic services with physical and logical protection.

HSMs are fully contained solutions for scalable cryptographic processing, key generation, and centralized key storage. As purpose-built appliances, they automatically include the hardware and firmware (i.e. software) necessary for these functions in an integrated package. They may be optimized for a specific or general purpose (e.g. performance, environment, portability, interface).

HSMs cooperate with services/applications via several secure means (interfaces, protocols).

Physical and logical protection of the appliance is supported by a tamper resistant/evident shell; and protection from logical threats, depending on the vendor's products, is supported by integrated firewall and intrusion prevention defenses. Some HSM vendors also include integrated support for two-factor authentication.

Security certification (e.g. PCI-HSM, FIPS 140-2, Common Criteria) is typically pursued by HSM vendors and positioned as a product feature. This security certification is a mean to establish the confidence in the HSM and provides the inherent trust, main feature of a root of trust.

The HSM can serve as the RTS given its externally inaccessible memory. In particular, the HSM memory includes a shielded area to hold long-term secrets and protect them from disclosure. It may also hold the static or dynamic measurements for different components (e.g. the BIOS, PCI ROM, boot loader, and host OS).

With the capability of the HSM to implement specific cryptographic functions, the HSM may provide the RTR. The RTR reports on the non-confidential contents of the RTS, most notably the measurements. The reports (or quotes) are signed using attestation keys generated in the HSM.

The Hardware-Mediated Execution Enclave technology, as defined in ETSI GR NFV-SEC 009 [i.6], could be used to host virtual HSMs that may be rooted in real HSMs.

#### 6.1.5.3 Hardware Co-Processors, Chipset, Processor Modes

Security co-processors, hardware chipsets, and specific processor modes can serve as hardware root of trust. These capabilities are inextricably bound to the platform hardware, as these are part of the processors and chipsets. In addition, these allow a simpler platform board design, while offering an integrated root of trust that can be securely provisioned.

One notable example of such technology is the HMEE, as defined in ETSI GR NFV-SEC 009 [i.6]. ETSI GS NFV-SEC 012 [i.3] defines requirements for hosts running sensitive workloads to use this technology in protecting secrets and workloads. This of course opens the possibility for using HMEEs to also protect virtual RoT. In any case, the implication on the composition of the CoT should be evaluated (i.e. it might reduce the complexity of the CoT) and the whole Remote Attestation process adapted to the specific architecture employed.

## 6.2 Measured Boot

The platform requires a Measured Boot process implementation, as described in clause 4.1.4.1. Having a Measured Boot process in place ensures that a CoT (Chain of Trust) is established and measurements for all software, executed as part of the boot-up procedure, are recorded.

The process of collecting the platform's software measurements is a pre-requisite for enabling remote trust validation through Remote Attestation. Conversely, establishing a CoT, based on which the measurements are afforded a trust level, depends on the availability of a Root of Trust for Measurement (RTM).

## 6.3 OS Measurement Architecture

A Measured Boot process (see clause 4.1.4.1) typically ends with the measuring and launch of the OS kernel. Any software launched after that point (e.g. user space application) will not be measured and will not be part of the CoT.

A Measurement Architecture (see clause 4.1.4.2), if implemented in the OS, can extend the CoT initiated by Measured Boot up to any user space application. It should be noted that only load time measurements are performed by the Measurement Architecture. No runtime measurements are performed. One example of such OS feature implementation is IMA (Integrity Measurement Architecture), available in the Linux Kernel.

## 6.4 Secure Boot

A Secure Boot process clause (see clause 4.2) allows enforcement of a strict integrity policy. A new state is not reached (e.g. new software not executed), if the appropriate integrity validation checks were not passed.

The Secure Boot process, also builds a CoT (see clause 4.2), but it does not record integrity measurements. It only uses the measurements to allow or block transition to a new state. Therefore, it does not, natively, provide support for attestation. The integrity trust level is inferred by the completion status of the boot-up process (successful launch means no integrity compromise was detected).

## 6.5 OS Enforcement of Integrity

A Secure Boot typically stops after the OS kernel is validated and executed. The process can be extended (with the proper OS level support) up to user space applications level (see clause 4.2). This allows validation of the integrity of any user space application at load time. No integrity enforcement is applied beyond load time (e.g. runtime - while the application is running). Examples of such OS features implementation include the IMA-Appraisal, EVM (Extended Verification Module), dm-verity, etc. available in the Linux Kernel.

## 6.6 Remote Attestation

Remote Attestation (RA), as described in clause 4.1.3, allows remote validation and assignment of a particular level of trust (within given contexts and constraints) for a platform, mainly based on the software load time integrity status. Dedicated features as described in following paragraph are required to be supported by the attested platform itself, as well as a dedicated (attestation) infrastructure to be setup, including an Attestation Authority with all its dependencies (e.g. software measurement reference values).

For the attested platform, availability of a RTM (see clause 6.2), a TPM/HSM (providing a RTS and RTR) (see clause 6.1) and a Measured Boot process (see clause 6.3) are pre-requisites for enabling RA. Additionally, an OS Measurements Architecture (see clause 6.4) can be leveraged, if available, to extend the measurements collecting process beyond the OS kernel and thus extending the CoT up to user space applications level.

The platform and the attestation infrastructure may already have an existing trust relationship, or this may need to be established first.

## 6.7 Other Capabilities

Other infrastructure capabilities relevant for the implementation of attestation procedures are listed hereafter:

- **Geolocation.** The ability of a particular component to assert its location. Geolocation usually refers to geographic (physical) location, but logical location may also be an important issue to consider.

- **Read-only file systems.** The ability for a file system to be mounted read-only. This may be a temporary measure to allow measurement at a particular point in the boot and attestation process, or a permanent measure to ensure that the integrity of certain data is maintained.

- **Encrypted file systems.** The ability for a file system to be encrypted before mounting. Typically, the keys to decrypt the file system are provided to the boot process as part of the measured boot and/or attestation procedure, based on measures of trust in the measured boot process.

## 6.8 Levels of Assurance to Capabilities Mapping

**Table 1: LoA to Capabilities mapping**

| | **6.1.2 RTM** | **6.1.3 RTS** | **6.1.4 RTR** | **6.2 Measured Boot** | **6.3 OS Measurement Architecture** | **6.4 Secure Boot** | **6.5 OS Enforcement of Integrity** | **6.6 RA** | **8.3 Run-time attestation** |
|---|---|---|---|---|---|---|---|---|---|
| **LoA 0** | | | | | | | | | |
| **LoA 1** | | | | | | H | H | | |
| **LoA 2** | H | H | H | H | H | H | H | H | |
| **LoA 3** | H | H | H | H | H | H+V | H+V | H | |
| **LoA 4** | H+V | H+V | H+V | H+V | H+V | H+V | H+V | H+V | |
| **LoA 5** | H+V+N | H+V+N | H+V+N | H+V+N | H+V+N | H+V | H+V | H+V+N | H+V |

H = Host required, V = VNFCI required, N = Infrastructure Network required (8.2).
NOTE: As discussed in clause 5.0, the technologies required to achieve LoA 5 are still a subject of research and currently not fully implementable, but going as close as possible to LoA 5 is highly desirable.

# 7 Operational Procedures

## 7.0 Introduction

Validation of a (physical or virtualised) platform's integrity state is not achieved through a one-time execution of a process. Instead it is based on the combined execution of multiple specific processes throughout the lifecycle of the platform. The combined result of these process will ultimately be associated to a certain trust level (Level of Assurance).

The processes will either be launched at the explicit request of administrators (e.g. during platform deployment) or will be executed automatically between the main attestation actors. The main actors (software components) involved in a platform attestation are:

- Attestation Agent - an agent running on the target platform, which listens for attestation requests from a remote verifier and provides, upon receiving such a request, a signed integrity measurements report.
- Remote Verifier - the software component (executing on a different platform than the target platform, see clause 4.1.1) responsible for requesting and validating the target platform's integrity measurements report.
- Trust Assessor - the software component that consumes the result of the integrity validation performed by the Remote Verifier, possibly combining it with other information to determine a suitable Level of Assurance and further actions.

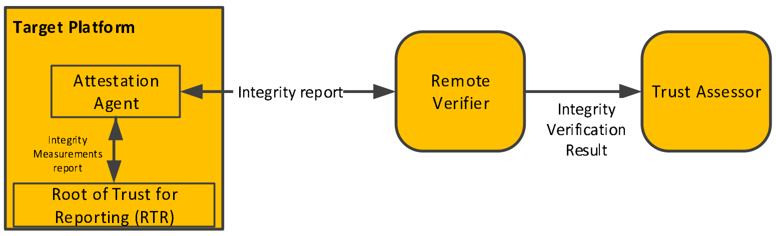

Target Platform
Attestation Agent
Integrity Measurements report
Root of Trust for Reporting (RTR)
Integrity report
Remote Verifier
Integrity Verification Result
Trust Assessor

**Figure 9: Main actors of a platform attestation**

A catalogue of binding between platform identifiers and their respective RoTs, which may be implicitly combined into a single ID, is required to allow the attestation process to be completed with a required level of trust in the RoT. The owner of the platforms will need to ensure that this is kept up to date with information regarding new, changed or removed platforms. The entries in the catalogue can contain further details on the capabilities of the RoT, e.g. RoT security assessment. The integrity of this catalogue is important and expected to be maintained. Mechanisms for updating of this catalogue are out of the scope of the present document.

Some processes need normally to be executed outside of the NFV environment, e.g. within a HBRT (as defined by ETSI GS NFV-SEC 012 [i.3]) manufacturer's infrastructure or hardware platform manufacturer's infrastructure.

## 7.1 Platform Deployment

### 7.1.1 Deployment Specific Processes

Building trust in a platform's integrity report, begins as early as the deployment of the platform. It should be noted that in this clause, deployment, especially for a physical platform, refers mainly to the steps taken for enrolling in an NFV environment (as opposed to physical deployment in a datacentre).

During this phase the most important processes are:

- Attestation Agent installation and measurements collection mechanism activation. These processes might be skipped, depending on the existing platform configuration (i.e. agent already installed and BIOS/UEFI, boot loader and kernel already configured for measurements collection). Conversely, they are launched manually by an administrator.
- Mutual key registration between Attestation Agent and Remote Verifier.
- Golden measurements (as defined by NIST SP800-155 [i.9]) registration with the Remote Verifier.

These processes apply both for physical and virtualised platforms, although their coverage differs depending on the type of platform (e.g. software installation will not be required in most cases for virtualised platforms).

### 7.1.2 Mutual Key Registration

#### 7.1.2.1 Attestation Key Generation

As described in clause 4.1.2, the Attestation Agent obtains a signature (quote) from the RTR thus providing cryptographic proof as to the origin and the integrity of measurements (i.e. un-tampered measurements originating from the RTS of the target platform). The RTR may be implemented in a trusted component, such as TPM, SE, HSM.

Before starting any attestation cycle, an Attestation Identity Key (AIK) pair is generated (i.e. not provisioned during manufacturing process) using the built-in RNG of the trusted component. The AIK is protected by the trusted component's RTS and used by the RTR to sign the measurements.

The AIK is not a permanent key and it is subject to destruction when reassigning the platform or changing its ownership. During the deployment phase, it is assumed that a valid AIK exists. This might imply a manual or automated process to be executed, which assigns the platform administrator and commands the trusted component to generate the required keys.

#### 7.1.2.2 Attestation Key Registration

The AIK public key or the root of its certificate chain is expected to be registered with the Remote Verifier, so it can properly determine the identity of the signer and authenticity of the signature.

Some specific mechanisms for performing this registration have been defined, which rely on an Endorsement Certificate, a credential issued during manufacturing process, but are heavily influenced by privacy concerns (see clause 6.1.1) (e.g. Privacy CA, Direct Anonymous Attestation [i.12]).

In a NFV environment, it is conceivable that the MANO stack, as the potential Trust Assessor, will actually benefit from precise platform identification, as opposed to group identification (as provided by already defined mechanisms). Custom mechanisms can be defined for this registration, which do not emphasize privacy protection allowing the Remote Verifier to precisely identify the platform that generated a certain quote.

Creating an Endorsement Certificate is a process owned and executed by the manufacturer of the trusted component containing the RTR before integration of the chip in the target platform. The presence of such a credential is mandatory for the registration mechanisms to work.

#### 7.1.2.3 Remote Verifier Secure Channel

From the integrity validation process perspective, there is no requirement towards ensuring the confidentiality of the data carried from the target platform to the Remote Verifier. The most important property of this data is its integrity, which is ensured through the quote mechanism described in clause 7.1.2.

However, an attacker might be able to extract valuable information from the attestation data, if it travels through the network unencrypted. This data contains hashes of the software that is running on the target platform. More so, it can reveal precise software versioning information, which would make a potential attacker's task much easier. Therefore, as a good practice, the exchanges between the target platform and Remote Verifier related to the attestation should also ensure confidentiality using a secure channel.

A mutually authenticated secure channel between the Remote Verifier and the target platform, would also protect the latter from availability targeted DoS (Denial of Service) type of attacks, based on repeated attestation requests. Repeated attestation requests can affect the availability of some services provided by the attestation targeted platform, as the quote generation can be a very time consuming process.

Installation of the trust anchors (i.e. root certificates) required for establishing the secure channel between the target platform and Remote Verifier should be performed through either an automated or manual process during the deployment phase. This will ensure that all subsequent communications related to attestation can leverage the security guarantees provided by the secure channel (e.g. authentication, confidentiality, integrity).

### 7.1.3 Golden measurements registration

Upon receiving the integrity measurements of a platform, the Remote Verifier is expected to validate the platform's software integrity status. Besides verifying the RTR quote, a mandatory step to ensure data was not tampered or falsified, the Remote Verifier will also need to compare the integrity measurements of different software components to known reference measurements also called golden measurements.

There is no defined mechanism for the reference measurements to be automatically acquired. Therefore, these measurements can either originate from an attestation enrolment cycle, i.e. an attestation cycle performed in a controlled environment (i.e. platform is functional, but not yet deployed) or from the software developer (solution developer) provided certificates.

Gathering of the reference measurements in a controlled environment is the preferred method for physical platforms. The Remote Verifier can store the measurements thus collected in the form of profiles that can be reused for all the platforms matching the same software configuration.

For virtualised platforms, providing the reference measurements as part of the metadata typically stored along the image in the image repositories, might be the preferred method. In this case the reference measurements would be registered with the Remote Verifier during the process of image deployment in the image repositories. Nevertheless, the method based on attestation in a controlled environment is applicable also for virtualised platforms.

Regardless of the type of mechanisms employed for the initial registration of the reference measurements with the Remote Verifier, it is assumed that there is also a way to update, in a secure (i.e. authenticated) manner, these measurements. This is used, for example, when an update of the software running on the platform is performed.

Following a software update, the measurements that will be recorded in the measurements log will differ from the previously registered set of reference measurements, but this should not be treated as an integrity compromise. Conversely, the reference measurements should be updated on the Remote Verifier in parallel with the software update performed on the target platform.

As the chain of trust is constructed from the RTM up to potentially any user space application, based on each software components measurement, a reboot of the target platform might be required following a software update. This would enable the CoT to be recreated using the new measurements, which would have been already registered with the Remote Verifier.

## 7.2 Attestation Cycle

### 7.2.1 Attestation flow

A typical attestation flow is composed of the following steps ( based on TCG Attestation, PTS Protocol: Binding to TNC IF-M [i.5]):

- Remote Verifier sends an attestation request to the target platform (Attestation Agent). This step also typically includes a random nonce in the exchange between the verifier and the target platform. This nonce can be integrated into the attestation evidence (e.g. TPM quote operation).

- Upon receiving the request the Attestation Agent will:
    - Assemble all the relevant integrity measurement information from the RTS into a log.
    - Obtain from the RTR a digital signature proving the authenticity and integrity of the measurements.
    - Assemble the measurement log and the RTR generated quote into an integrity (attestation) report.
    - Send the attestation report (containing also the RTR generated quote) back to the Remote Verifier.
- Upon receiving the attestation report the Remote Verifier will:
    - Verify the attestation report authenticity and integrity using the RTR generated quote.
    - Validate the received measurements (from the measurement log) against the reference (golden) measurements.
    - Store the validation result for subsequent consultation by the Trust Assessors.

### 7.2.2 Attestation intervals

An attestation cycle can be started by the Remote Verifier at fixed intervals based on a policy or upon request from the Trust Assessors (which might require fresh integrity information).

If a platform is configured to only collect boot time measurements (see clause 6.3), an attestation cycle after each boot cycle would be sufficient to validate those measurements (the measurements will not change until the next boot cycle).

Conversely, if a platform is also configured to collect (load time) measurements of applications (see clause 6.4) obtaining integrity information at regular intervals would be beneficial. This fresh integrity information can be leveraged, for example, by the MANO stack to avoid scheduling sensitive workloads on potentially compromised hosts. Another notable example would be the use of periodic attestation cycles of the host (hypervisor) to ensure a vRoT running on it was not compromised and thus the prior VM attestation cycles results can be trusted (see clause 4.4.2).

# 8 Analysis of the Evolution of Attestation Technologies

## 8.1 Network Service (NS) Attestation

Platform and executable component attestation may provide trust environment and trust building blocks for the network services constructing upon them. In addition to the attestation of platform and virtual container, the Network Service (NS) in Logical Abstraction layer of the NFV architecture needs also to be attested for a trust service provision.
A Network Service (NS) or a Service function Chain is a composition of Network Functions arranged according to one or more forwarding graphs. The description of a Network Service as used by the NFVO functions to deploy a network service instance includes references to the descriptions of the objects, like Virtualised Network Function (VNF), Physical Network Function (PNF), Virtual Link (VL), VNF Forwarding Graph (VNFFG) and optionally Network Forwarding Paths (NFP) contained in the VNFFG. Through specified interface, NFVO uses descriptor/template on-boarded from OSS/BSS to enable network service orchestration. In case that the classification and selection rules applied to NFP are not included in the Network Service deployment template, they can be provided to NFVO later to be assigned to an existing NFP. The authorized entity sending NFP rule to NFVO may be OSS/BSS.

Based on above analyse, NS is deployed by OSS/BSS and NFVO jointly and in most cases may be initiated by OSS/BSS and implemented by NFVO. Attestation mechanisms are suggested to be provided to fill the gap between OSS/BSS and NFVO so that NS may be correctly on-boarded according to NS Catalogue within NFVO and instantiated to satisfy service providers' expectation.

One alternative flow to attest NS may be:

- The NS verifier retrieves from the NFVO NS catalogue the NS information for the attested NS.
- The NS verifier challenges OSS/NMS for the relevant NS instance information.
- The NS verifier assesses whether or not the NS instance information from OSS/NMS is consistent with the expected NS information retrieved from the NFVO NS catalogue.

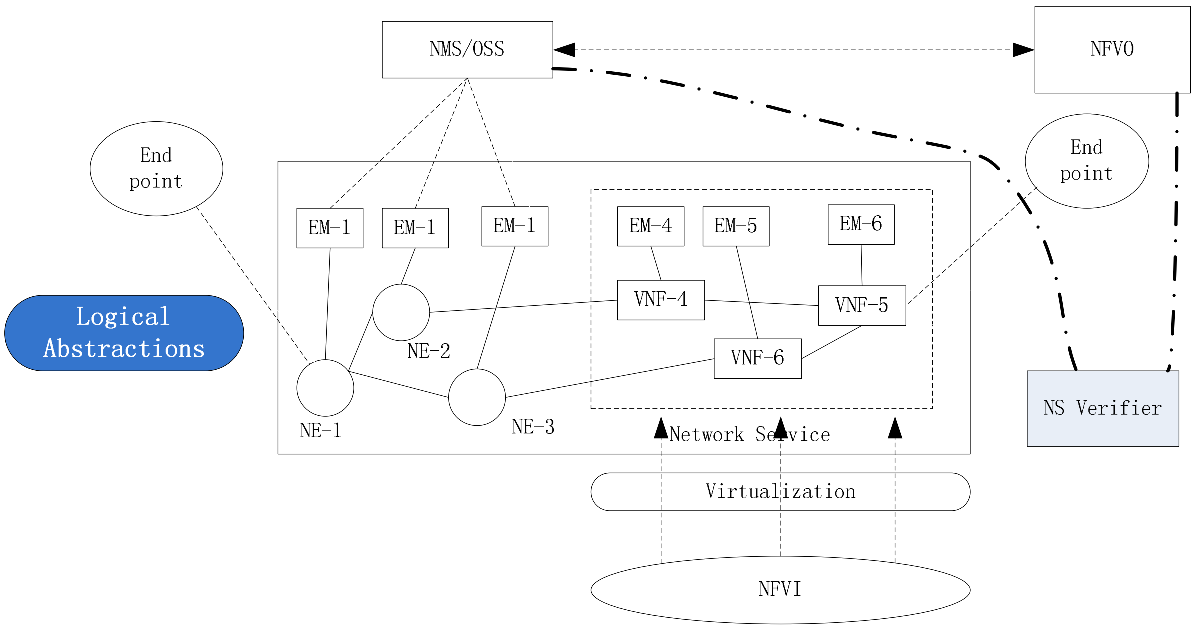


**Figure 10: Network Service Attestation**

## 8.2 Infrastructure Network Attestation Using a SDN Verifier

In addition to the platform and executable component attestation, the infrastructure network needs to be attested also to assess the enforcement of the VNF chaining. Whilst traditional attestation is sufficient for the traditional networking paradigm (network element with a fairly static configuration), the dynamicity brought by SDN makes attestation of SDN-aware network element more challenging. Since the dynamic part of a SDN-aware network is driven by the SDN controller but enforced by the different network elements, the network attestation architecture needs to bridge this gap.

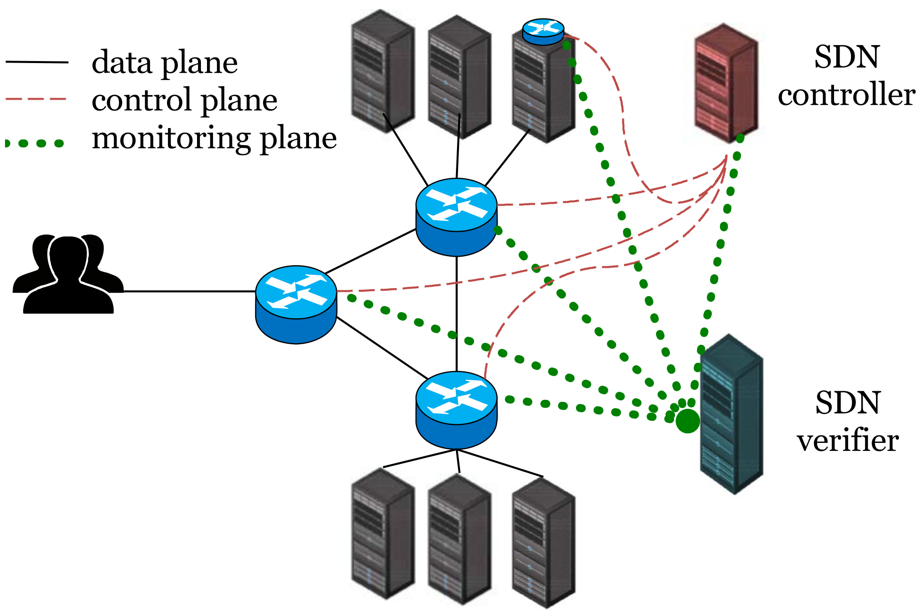
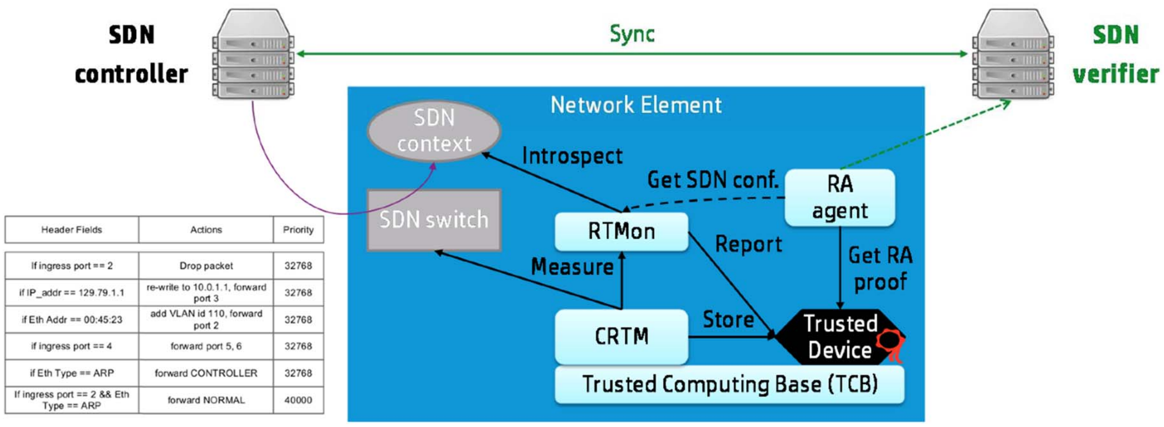


**Figure 11: Infrastructure network attestation architecture**

One architecture design choice for infrastructure network attestation is to introduce a SDN verifier and extend network element's attestation features to also report on the currently enforced SDN configuration. Upon request of a network element attestation, the SDN verifier performs the following steps:

- The SDN verifier retrieves from the SDN controller the SDN configuration for the attested network element.

- The SDN verifier challenges the attested network element for its SDN configuration.
- The SDN verifier assesses that the SDN configuration enforced by the network element is consistent with the expected SDN configuration retrieved from the SDN controller.



| Header Fields | Actions | Priority |
|---|---|---|
| If ingress port == 2 | Drop packet | 32768 |
| if IP_addr == 129.79.1.1 | re-write to 10.0.1.1, forward port 3 | 32768 |
| if Eth Addr == 00:45:23 | add VLAN id 110, forward port 2 | 32768 |
| if ingress port == 4 | forward port 5, 6 | 32768 |
| if Eth Type == ARP | forward CONTROLLER | 32768 |
| If ingress port == 2 && Eth Type == ARP | forward NORMAL | 40000 |

**Figure 12: Leveraging a TPM for SDN attestation**

For consistency across the NFVI, the SDN verifier and the attestation mechanism (the Root of Trust for Monitoring - RTMon) on the network element can leverage a TPM to monitor the SDN configuration. More precisely, the network element implements a regular measured boot - using the Core Root of Measurement and a TPM as the trust anchor, and is extended with the RTMon which is responsible for the monitoring and reporting of the run-time SDN configuration. The Remote Attestation (RA) agent is present only to proxy the requests by the SDN verifier.

## 8.3 Perspectives for Run-Time Attestation

Current attestation technologies focus on boot or load time attestation via remote attestation (see clause 4.1.2), and local attestation (see clause 4.2), which are typically based on TPM or software signing.

Normally, after a system is running, certain parts of the configuration may change from runtime operation. For example, after some operation time the VNF memory may contain other (binary) data, e.g. coming from shared objects, loaded/unloaded modules (e.g. LKM), relocated code (even if the code is position independent, its address references and/or offset tables, etc. may have changed), or run-time patches. Even if all modifications are coming from non-malicious actions (normal OS behaviour), there would be many different "hash" values, which could be reported to an external "attestation entity". Now there is a need for a way to predict what changes will take place, as described in DRIVE [i.13], if such a mechanism is not available, the attestation at run-time will then have to be restricted to a static part of the system (e.g. core kernel functions, read-only memory).

Regardless of how these issues get addressed, a run-time attestation solution will need to be designed based on the basic principles of attestation: extending a RTM anchored CoT (see clause 4.1.2) and making use of appropriate RTS and RTR for measurements storage and reporting (see clause 4.1.2). Therefore, a run-time measurement collection architecture, as a component of a run-time attestation solution, needs to extend the boot time CoT, similarly to the load time measurement collection architecture (see clause 6.3). Also, any measurement log created by such run-time measurement component (similar to the SML) needs to be anchored in a RTR to ensure its integrity, which can also be proven to a remote verifier (e.g. TPM quote) (DRIVE) [i.13].

Run-time attestation an open issue at the publication of the present document and is subject of intense academic and non-academic research. There are many aspects that need to be investigated for making run-time attestation a feasible approach, from time aspects like periodicity or the frequency of attestation checks after boot-time (or the application of event-driven verifications) to the actual run-time measurement and authentication technologies (software signing, applicability of TPM, etc.), and including operational modes (failure categorization and actions), impacts on the infrastructure, and the specific aspects related to the dynamic nature of the intended verification (what and how is allowed to change in memory, for example).

## 8.4 Attestation using HMEE based technology

The Hardware-Mediated Execution Enclave (HMEE) technology provides hardware-enforced isolation of both code and data. This HMEE can act as a root of trust and can be used for attestation. In such a scenario, the remote verifier will request a quote from the respective HMEE, possibly, through an attestation agent. The HMEE will provide the signed measurement data (quote) to the remote verifier. The remote verifier can then verify the signature and the quote with its own stored data. In this way the HMEE based attestation provides remote verifier the assurance that this is the right application executing in the right platform.

The quote is signed by using an attestation key. This attestation key can be anchored to the platform specific key thus providing a binding between the platform and the HMEE. The HMEE and the attestation infrastructure may already have an existing trust relationship, or this may need to be built up during startup of an HMEE.

Figure 13 depicts a simplified overview of the attestation process where an attestation agent is used.

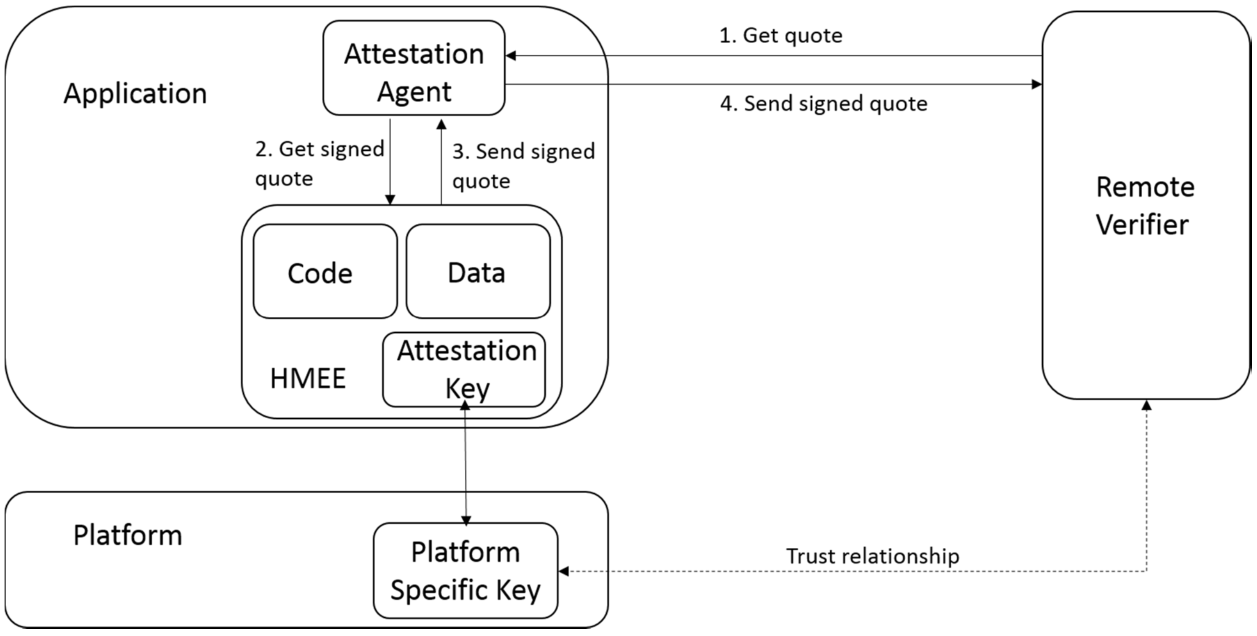


**Figure 13: A simplified overview of the attestation process using HMEE**

Note that the quote includes HMEE measurement and optionally some arbitrary piece of data. This arbitrary piece of data may contain the public key or the certificate of the enclave. Once the enclave gets verified, and then the remote verifier can use this certificate to communicate securely with the enclave. This can be further used to provision a secret within the enclave that can be used by the application without leaking the secret to anybody.

# Annex A: Possible Proof of Concepts

Proofs-of-Concept on the following aspects are encouraged, with the idea of addressing the issues identified in the present document, and to provide guidance on the highlighted technology perspectives:

- Implementation of attestation technologies on different virtualisation constructs, and especially on container frameworks
- Integration of attestation mechanisms with existing MANO stacks
- Provision of Roots-of-Trust on different virtualisation techniques: VM, containerization, unikernels, etc.
- Alternatives for remote verification in virtualised environments
- Incorporation of levels of assurance in NFV descriptors and lifecycle management
- Network service attestation
- Run-time Attestation

# Annex B: Authors & contributors

The following people have contributed to the present document:

**Rapporteurs:**
Dr. Diego Lopez, Telefonica
Mihai Serb, Huawei

**Other contributors:**
Anthony Rutkowski, Yaana
Mike Bursell, Intel
Alec Brusilovsky, TCG/Interdigital
Zhaoji Lin      , ZTE
Dr. Huilan Lu, Alcatel-Lucent
Wei Lu, Nokia
Anne-Marie Prade, Gemalto
Bruno Chatras, Orange
Kapil Sood, Intel
Esa Salahuddin, Cisco
Kazi Wali Ullah, Ericsson

# Annex C: Change History

| Date | Version | Information about changes |
|---|---|---|
| 2014-12 | V0.0.1 | Initial ToC and Skeleton |
| 2015-03 | V0.0.2 | Editor notes on clause contents and contribution on infrastructure |
| 2015-07 | V0.0.3 | Initial contribution from TCG<br>First version available at the ETSI portal open area |
| 2016-04 | V0.0.4 | Contributions on attestation procedures (concepts, technologies, existing models)<br>Initial definition of LoAs<br>Contribution on capabilities (TPM)<br>Contributions on technologies analysis (NS attestation, gap analysis outline) |
| 2016-09 | V0.0.5 | Contributions on capabilities description update<br>Contributions on Trustworthy Platform Configuration |
| 2016-09 | V0.0.6 | Contribution on defining Levels of Assurance<br>Contribution on requirements for runtime attestation |
| 2017-02 | V0.0.7 | Contribution on reorganizing clause 6.1<br>Contribution on defining RTS & RTR as capabilities (6.1.3, 6.1.4)<br>Contribution on defining HSM as an example of RoTs implementation (6.1.5.2)<br>Multiple contributions on defining the attestation specific operational processes (7.1, 7.2) |
| 2017-02 | V0.0.8 | Apply GR template<br>Resolve pending editor's notes, grouping references in clause 2.2<br>Contribution on follow-on PoCs (annex)<br>Contribution on RoTs (4.1.1) |
| 2017-05 | V0.0.9 | Integrate editorial changes from contribution NFVSEC(17)000036<br>Contribution NFVSEC(17)000043 - references update |
| 2017-05 | V0.0.10 | Editorial changes (disable track changes) |
| 2017-06 | V0.0.11 | NFVSEC(17)000044r2 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000068r1 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000045r1 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000047r1 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000066 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000067r1 (r1 implements the changes agreed in NFVSEC#101 - meeting report NFVSEC(17)000084)<br>NFVSEC(17)000069r1 (r1 implements the changes agreed in NFVSEC#101 - meeting report NFVSEC(17)000084)<br>NFVSEC(17)000082 (meeting report NFVSEC(17)000084)<br>NFVSEC(17)000080 (meeting report NFVSEC(17)000084)<br>Editorial changes:<br>- Spelling & formatting errors<br>- Diagrams renumbering |
| 2017-07 | V0.0.12 | NFVSEC(17)000091r1 (meeting report NFVSEC(17)000089)<br>NFVSEC(17)000093r1 (meeting report NFVSEC(17)000089)<br>Editorial changes:<br>- Apply new (2017-07-10) GR template<br>- Update references formatting to comply with ETSI editing rules |
| 2017-09 | V0.0.13 | Changes to 4.1.1.3 and 6.1.5.2 as agreed in NFVSEC#107 (meeting report NFVSEC(17)108001)<br>Editorial changes:<br>- Fill "Authors & contributors" annex |

## History

| Document history | | |
|---|---|---|
| V1.1.1 | October 2017 | Publication |
| | | |
| | | |
| | | |
| | | |