INTERNATIONAL STANDARD

ISO/IEC 27070

First edition
2021-12

Information technology — Security techniques — Requirements for establishing virtualized roots of trust

Technologies de l'information — Techniques de sécurité — Exigences relatives à l'établissement de racines de confiance virtualisées

Reference number
ISO/IEC 27070:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Trusted computing is a kind of security technology based on hardware trusted modules, which aims to ensure that a computer behaves as expected. The trusted computing technology has been developing fast since its establishment in the 1980s.

The emergence of cloud computing provides a new application scenario for trusted computing technology. Trust can be established in VMs using a RoT on the physical machine and a virtualized RoT in the VM and mechanism to bind them together to provide assurance they are on the same machine. The trusted migration of a VM could use trusted computing to establish trust in the state of the source and destination physical machines (including their VMM software) and components involved in the migration process. In the cloud computing environment, a single physical RoT only provides limited resources and computing efficacy, which is not enough for the large number of VMs on one server. To address this issue, virtualized RoTs are used. Using virtualization technology to create multiple virtualized RoTs on a single physical platform, providing a virtualized RoT for each VM, combined with cryptographic technology to support secure and trusted migration of VMs, thereby building a trusted cloud computing environment. The establishment procedure of virtualized RoTs consists of multiple steps, and any security problem in any step diminishes the trustworthiness of virtualized RoTs, resulting in an inability to establish trust using the virtualized RoTs.

The goal of the document is to provide a unified approach to virtualize RoTs based on hardware trusted modules.

Information technology — Security techniques — Requirements for establishing virtualized roots of trust

1 Scope

This document specifies requirements for establishing virtualized roots of trust.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
attestation key
AK
particular type of *trusted module* (3.7) signing key that has a restriction on its use, in order to prevent forgery

3.2
endorsement key
EK
key that is used in a process for the issuance of *attestation key* (3.1) credentials and to establish a platform owner

3.3
integrity measurement
process of calculating the hash value of the measured object using the cryptographic hash algorithm

3.4
root of trust
RoT
component that needs to always behave in the expected manner because its misbehaviour cannot be detected

Note 1 to entry: The complete set of roots of trust has at least the minimum set of functions to enable a description of the platform characteristics that affect the trust of the platform.

[SOURCE: ISO/IEC 11889-1, 3.59, modified — The abbreviated term has been added.]

3.5
remote attestation
RA
process of evaluating integrity measurements generated using a *root of trust* (3.4) for measurement, storage and reporting to establish trust in a platform remotely

3.6
sensitive information
information is sensitive that the *trusted module* (3.7) does not allow access to the information without proper authority

Note 1 to entry: An example of sensitive information in a trusted module is the private of an asymmetric key.

3.7
trusted module
TM
module for trusted computing providing integrity measurement, integrity report, cryptographic service, random number generation, secure storage functions and a set of platform configuration registers

Note 1 to entry: There are several implementations of trusted module, such as TPM, TCM, etc.

3.8
virtual machine
VM
virtualized hardware environment in which an operating system can execute, but whose functions are accomplished by sharing the resources of a real data processing system

3.9
virtual trusted module
vTM
component associated with a single *virtual machine* (3.8) that provides the functionality described in a *trusted module* (3.7)

3.10
virtual platform configuration register
vPCR
one or more platform configuration registers within a *virtual trusted module* (3.9)

4 Symbols and abbreviated terms

BIOS	basic input/output system
CPU	central processing unit
CRTM	core root of trust for measurement
GPT	globally unique identifier partition table
KEK	key encryption key
MBR	master boot record
OS	operating system
PCR	platform configuration register
PCA	privacy certificate authority
PI	platform initialization
ROM	read-only memory
SRK	storage root key
TPM	trusted platform module

TCM	trusted cryptography module
TSS	trusted software stack
UEFI	unified extensible firmware interface
VMM	virtual machine monitor
vCRTM	virtual core root of trust for measurement
vRTM	virtual root of trust for measurement
vRTR	virtual root of trust for reporting
vRTS	virtual root of trust for storage
vSRK	virtual storage root key
WK	work key

5 Functional view

5.1 Overview

This clause provides a neutral architectural view of functional components required by trusted computing activities in the cloud computing environment. It also presents the functional and security requirements for key components.

Figure 1 shows a framework of the required functional components, where specific types of functions are grouped into each layer.

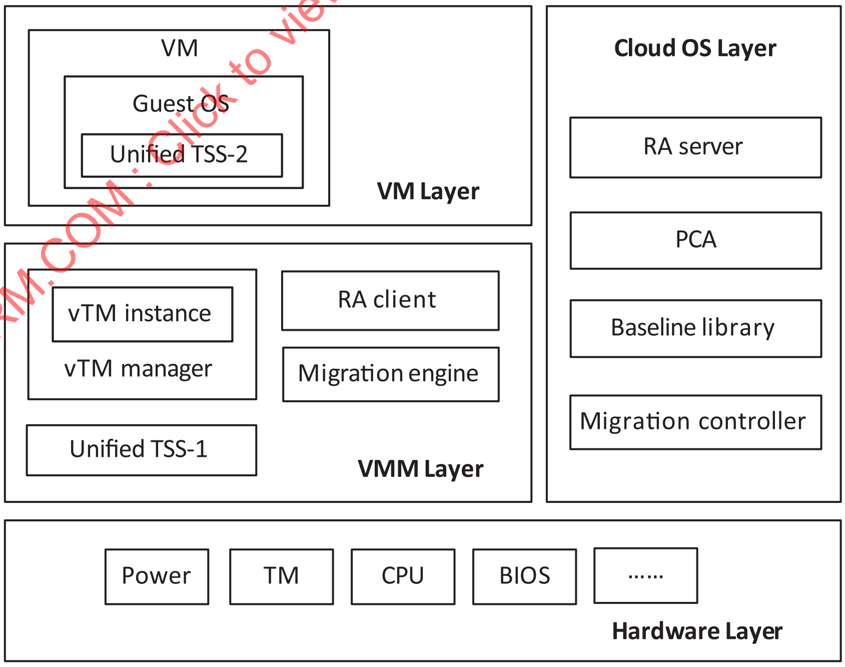

Figure 1 — Framework of functional components

5.2 Hardware layer components

5.2.1 General

At the bottom of the architecture, the hardware layer that includes hardware resources and devices is the base for building a trusted computing platform. This layer provides a RoT for the physical machine that typically offers trusted platform services for the VMM layer.

NOTE The VMM layer is also known as the hypervisor layer.

This layer also includes the CRTM, the initial set of instructions executed for establishing a new chain of trust for integrity measurement.

The hardware layer components include but are not limited to:

— power: Powering the computer system for booting and running. The Trusted Module (TM) can control power such that it can increase security by turning off the power if verification of the boot fails against a policy;

— TM: A trusted module on a special co-processor or chip with capabilities that include but not limited to integrity measurement, integrity reporting, generation of signatures for measured integrity values, key management, secure storage, identity verification, etc.;

  The TM shall support a root of trust for measurement, implement a root of trust for reporting and provide a root of trust for storage. See TM standards for details.

— BIOS/UEFI: A firmware with capabilities of initializing the platform, starting an OS loader and providing runtime services to the OS;

— CPU: The operating centre of the computing system.

The power, BIOS/UEFI and CPU have no special functional or security features to support virtualized RoTs. Hence, this document only lists the functional and security requirements for the TM in 5.2.2 and 5.2.3.

5.2.2 Functional requirements of key components

A TM shall provide the following functions.

— Support the integrity measurement, storing, generation of signatures for measured integrity values and reporting measured values for platform.

— Support key generation for use as signature keys.

— Support cryptographic algorithms, such as hash algorithm, encryption/decryption algorithm, but are not limit to any specific sets of algorithms.

— Protect integrity measurements in the PCR.

5.2.3 Security requirements of key components

A TM shall meet the following security requirements.

— Ensure the security of a TM itself.

— Ensure the security of confidential information, such as keys.

— Provide the secure storage area to store an SRK to ensure the security of the key information.

5.3 VMM layer components

The VMM layer provides virtualization services to VMs and ensures that VMs can operate independently.

This layer also provides virtualized RoTs for VMs, which includes virtual vRTM, vRTR and vRTS.

The VMM layer components include but are not limited to:

- VMM: It virtualizes the underlying hardware platform to enable multiple VMs to run in isolated environments and share the hardware resources;
- unified TSS-1: It provides a unified interface for upper layer applications to utilize TM functions without considering heterogeneous TSS implementations specifically;
- vTM manager: It establishes and maintains the binding list between each vTM instance and VM. It is responsible for creation, instantiation, deletion, start, stop and migration of the vTM instance associated with each VM;
- vTM instance: It emulates a hardware TM. Each vTM instance imitates interfaces and functions of the TM;
- RA client: It retrieves and transmits the integrity evidence of host and VMM layer. It does not leak the sensitive information during communication with RA server;
- migration engine: It provides capabilities of package, serialization and protection for vTM state data during transmission. It guarantees that only one vTM instance is active during transmission and the vTM instance is removed once it has been successfully migrated.

The VMM, unified TSS-1 and RA client have no special functional or security features to support virtualized RoTs. Hence, this document only lists the functional and security requirements for vTM manager, vTM instance and migration engine in 5.3.1 and 5.3.2.

5.3.1 Functional requirements of key components

5.3.1.1 vTM manager

A vTM manager shall provide the following functions.

- Create a new vTM instance along with the new VM creation.
- Establish a one-on-one correspondence between a VM and a vTM instance to ensure that each vTM instance only provides services for a dedicated VM.
- Instantiate a previous (saved) vTM instance when the VM reboot.
- Initialize the vTM instance that is expected to be reset during VM reboot.
- Maintain the non-volatile vTM instance when power on/off or VM reboot.
- Delete the vTM instance once the associated VM has been removed or migrated.
- Protect the vTM instance confidentiality and integrity.
- Bind the vTM instance exclusively with a VM's lifecycle until the VM is migrated to another platform.

5.3.1.2 vTM instance

A vTM instance shall provide the following functions.

- The vTM instance implementation shall be functionally compatible with the TM.

- The vTM instance implementation shall ensure that only the authorized VM associated with the vTM instance has access to the vTM instance's external interfaces.
- The vTM instance shall be designed in such a way that the VM is unable to directly access the internal state of the vTM instance unless through its external interfaces.
- vTM instances associated with different VMs shall have capabilities for independent updates, if updatable.

5.3.1.3 Migration engine

A migration engine shall provide the following functions.

- Listen for a vTM migration notification from the migration controller.
- Notify the vTM Manager that the vTM instance of a particular VM is to be migrated and the vTM manager shall package up the vTM state information.
- Transfer a vTM state package to the destination platform over the network.
- Transfer the received vTM state package to the vTM manager that resides at the destination site for instantiation.
- Return successful completion status to the sender migration engine that resides at the source site.
- Notify the vTM manager at the source site, once a transferred vTM state package has been successfully received by a destination platform.
- Instruct the vTM Manager to delete the associated vTM instance and notify the migration controller of migration completion.

5.3.2 Security requirements of key components

5.3.2.1 vTM manager

A vTM manager shall meet the following security requirements.

- Data encryption keys within the vTM manager should be protected by the TM.
- Ensure the integrity of the vTM manager based on chain of trust.
- Encrypt the content of the vTM state prior to storing it on persistent storage to prevent secrets from being stolen or modified.
- Decrypt the vTM state when it leaves the persistent storage and verify its integrity prior to re-instantiating it in the VM platform.

  NOTE For example, the vTM manager recalculate the hash value of the decrypted vTM state and compare the recalculated hash value with the original hash value of vTM state to ensure the integrity of the vTM state.

- Provide an access control mechanism for the TM to prevent the vTM instance from accessing any sensitive information or restricted resources in the TM.
- Provide a mechanism to authenticate the vTM identity.

  NOTE For example, the PCA verifies the virtual EK of a vTM before issuing the identity certificate.

5.3.2.2 vTM instance

A vTM instance shall implement the TM security requirements and meet the following additional security requirements.

- During the start-up and shut-down processes, the sensitive information within a vTM instance shall be protected against tampering or leakage.
- When the vTM instance is no longer used, the data within the vTM instance shall be encrypted and stored.
- During the operation of the vTM instance, an access control mechanism shall be provided to prevent unauthorized access to the VM state data and other sensitive information stored in the vTM instance.
- Ensure that the VM can only access to the vTM instance through the trusted interface.
- vTM instance failure should not prevent the VM running.
- Generate keys to protect sensitive information following security best practices and transmit keys via a secure channel, such as HTTPS, TLS, etc.
- When deleting a vTM Instance after the associated VM instance has been removed or migrated, the associated cache entries and other residual data traces should be eliminated.

5.3.2.3 Migration engine

A migration engine shall meet the following security requirements.

- Provide encryption and decryption mechanism for vTM state.
- Provide anti-rollback mechanism to prevent the vTM state from rolling back during the migration process.
- After the completion of the migration, the migration engine shall free the memory used for any local vTM instance to avoid duplicate instance.

5.4 VM layer components

At the top layer of the architecture, the VM layer is the location where VMs reside. Each VM operates in a compartment isolated from each other. The resources for the VM layer are provided by the VMM layer, including virtualized CPU, virtualized memory, virtualized devices and vTM instances.

The VM layer components include but are not limited to:

- VM: It provides a necessary environment for software execution, including an operating system and applications;
- guest OS: The operating system that runs within the VM;
- unified TSS-2: It provides a unified interface for VMs to utilize vTM instance functions.

  NOTE TSS-1 is generally deployed in the host OS and provides an interface for the host to access a TM. TSS-2 is generally deployed in the guest OS and provides an interface for the VM to access a vTM.

The VM, guest OS and unified TSS-2 have no special functional or security features to support virtualized RoTs. Hence, this document does not list requirements for them.

5.5 Cloud OS layer components

5.5.1 General

The cloud OS layer components include but are not limited to:

- RA server: It sends a challenge to the RA client and receives measured values from the RA client, then determines the trust of the VM layer, the VMM layer, and the hardware layer;
- PCA: It is a trusted third party that provides authentication services for interactions between the RA server and the RA client. It provides capabilities of verifying the legitimacy of an AK from the RA client, signing an AK certificate, issuing an AK certificate to the RA client and verifying the validity of an AK certificate sent by the RA client to the RA server. Figure 2 shows the relationship between the PCA, RA client and RA server;

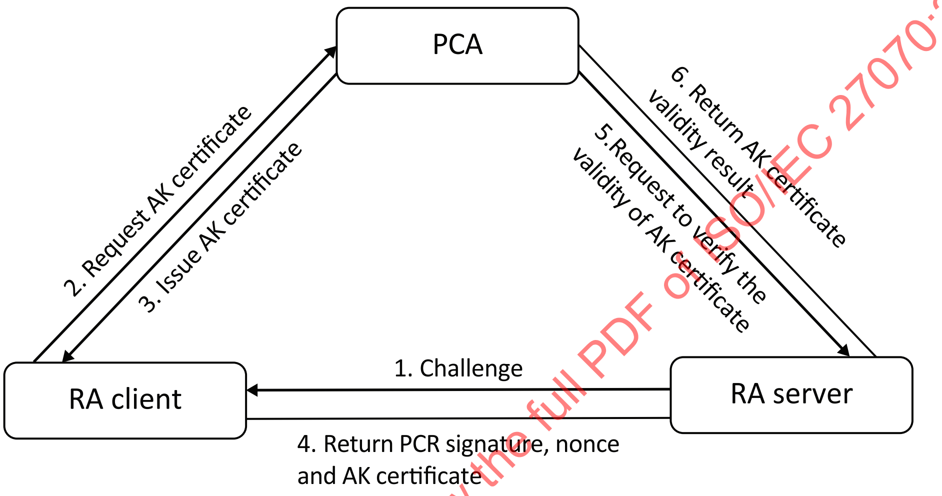
PCA
RA client
RA server
1. Challenge
2. Request AK certificate
3. Issue AK certificate
4. Return PCR signature, nonce and AK certificate
5.Request to verify the validity of AK certificate
6. Return AK certificate validity result

Figure 2 — Relationship between the PCA, RA client and RA server

- baseline library: It provides capabilities of storing and maintaining some important data, such as the integrity measurement results, baseline values, etc.;
- migration controller: It establishes a migration policy and instructs the migration engine to perform which operations.

The RA server, PCA and baseline library have no special functional or security features to support virtualized RoTs. Hence, this document only lists the functional and security requirements for migration controller in 5.5.2 and 5.5.3.

5.5.2 Functional requirements of key components

A migration controller shall provide the following functions.

- Determine the scenario when a vTM instance needs to be migrated to a new platform.
- Instruct the migration engine to initiate the migration.
- Listen for notification of a vTM instance migration completion.

5.5.3 Security requirements of key components

A migration controller shall meet the following security requirements.

- The migration controller shall be authenticated by an external entity to ensure that it is trusted.

- Ensure that migration controller is protected from network vulnerabilities and runs in an environment that is remotely verifiable.
- The migration controller shall support polices allowing a vTM that is cryptographically protected by an underlying TM to be moved to another platform.
- The migration controller shall have access to information about the trustworthiness of each system so it can optionally enforce restrictions on where the migration is allowed to occur.

6 Activity view

6.1 General

This clause describes how the functional components defined in Clause 5 are used to implement the essential trusted computing activities such as transitive trust, integrity measurement, remote attestation, data protection and vTM migration. These activities are required in the cloud computing environment supporting virtualized RoT. The logical relationship between the activity view and the functional view is illustrated in this document (see Annex A).

6.2 Transitive trust

6.2.1 General

Transitive trust is a process that the RoT is used to establish the trust of an initial executable function, and trust in that function is then used to establish the trust of the hardware platform, OS, application, which extends a chain of trust to the entire computing system.

In this transitive trust activity, the CRTM measures the hardware layer as the start point. After the hardware layer is measured, the chain of trust is extended to the VMM layer via measuring the components of the VMM layer. After the VMM layer is measured, the chain of trust is extended to the VM layer. The measured values of hardware layer and VMM layer are extended into PCRs within the TM, and the measured values of VM layer are extended into vPCRs within the vTM. In this document, the PCR definitions are presented in Table 1. The vPCR definitions in the virtualization environment are similar to the PCRs.

Table 1 — PCR definition

PCR index	PCR usage
0	SRTM, BIOS, host platform extensions, embedded option ROMs and PI drivers
1	Host platform configuration
2	UEFI driver and application code
3	UEFI driver and application configuration and data
4	UEFI boot manager code (usually the MBR) and boot attempts
5	Boot manager code configuration and Data (for use by the boot manager code) and GPT/partition table
6	Host platform manufacturer specific
7	Secure boot policy
8~15	Defined for use by the static OS
16	Debug
23	Application support

NOTE Static OS is the operating system that is loaded during the initial boot sequence of the platform from its platform reset.

Figure 3 shows a general transitive trust activity to illustrate where measurements should be processed as well as the relationship between components.

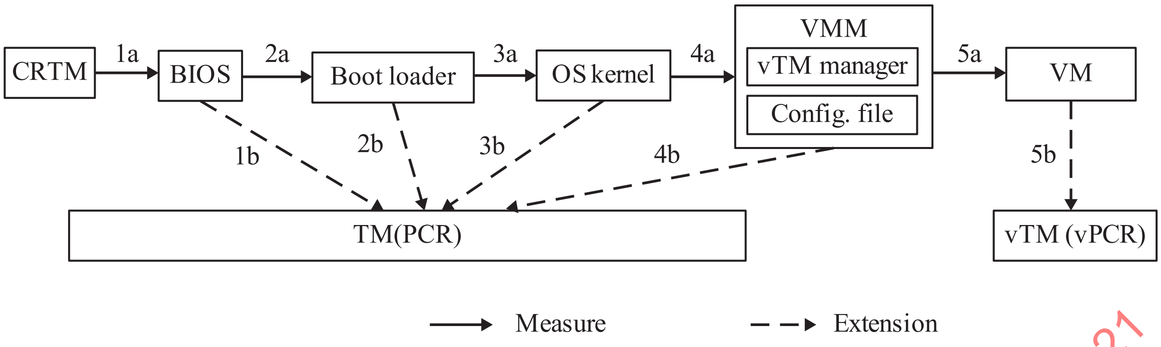

Figure 3 — Flowchart of transitive trust activity

This transitive trust activity consists of three steps:

- transitive trust in host;
- transitive trust in VMM;
- transitive trust in VM.

6.2.2 Transitive trust in host

- After the system is powered on, the CRTM measures the BIOS, and stores the measured result into the PCR.
- The BIOS measures the boot loader, and stores the measured result into the PCR.
- The boot loader measures the OS kernel, and stores the measured result into the PCR.
- The OS kernel measures the VMM, and stores the measured result into the PCR.

6.2.3 Transitive trust in VMM

The VMM measures the vTM manager and other files, and stores these measured results into PCRs.

6.2.4 Transitive trust in VM

The transitive trust chain in VM is built based on the vTM instance, which is similar to the transitive trust in host.

NOTE The vCRTM implementation is out of scope.

6.3 Integrity measurement

In integrity measurement activity, the CRTM first measures a component and then generates a measurement event. A measurement event consists of measured values and event description. The measured values are hash digests that represent the embedded data or program code in a component. The measured values are stored in PCRs and the event description is stored in a measurement log. Figure 4 shows the process of integrity measurement with two components.

All PCRs are reset to their default initial condition on TM reset or TM restart. Other than reset, the only way to change a PCR value is to extend it. The extend operation on a PCR is defined as:

$$PCR[n] \leftarrow Hash(PCR[n-1] || digest)$$

where *Hash*() is a collision resist hash function defined in ISO/IEC 10118-3.

After extending, the PCR value is a unique for specific order and combination of digest values that were extended. Once the one of measured values in the sequence is changed, the subsequent extended results are changed. In particular, the measured value of the BIOS is stored in PCR[0].

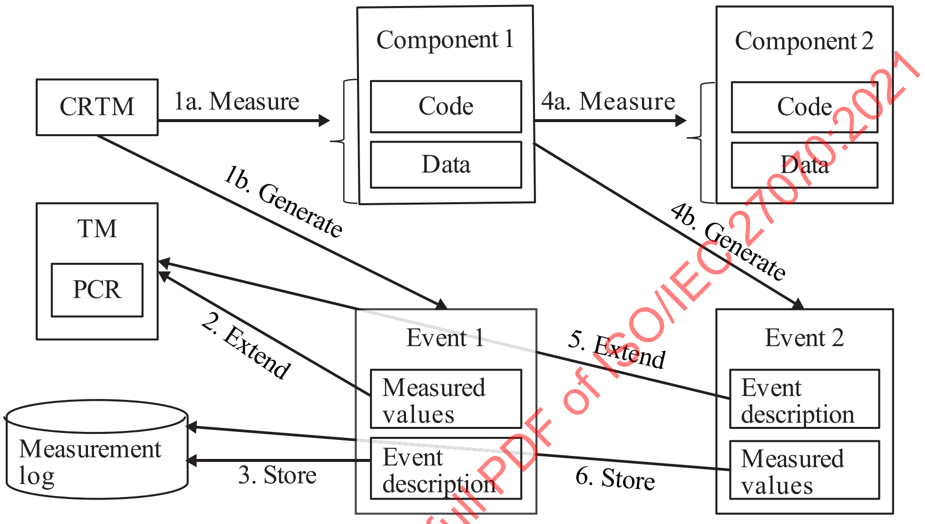

Figure 4 — Flowchart of integrity measurement activity

6.4 Remote attestation

The remote attestation activity involves three major components: the RA server, RA client and PCA. The RA client running in the VMM is responsible for listening for an integrity challenge. After receiving this integrity challenge from the RA server, the RA client executes the attestation process to ensure that a VM is running on a trusted environment. A trusted entity maintains an asset inventory of EK certificates to confirm the AK exists in a TM. The remote attestation activity is illustrated in Figure 5.

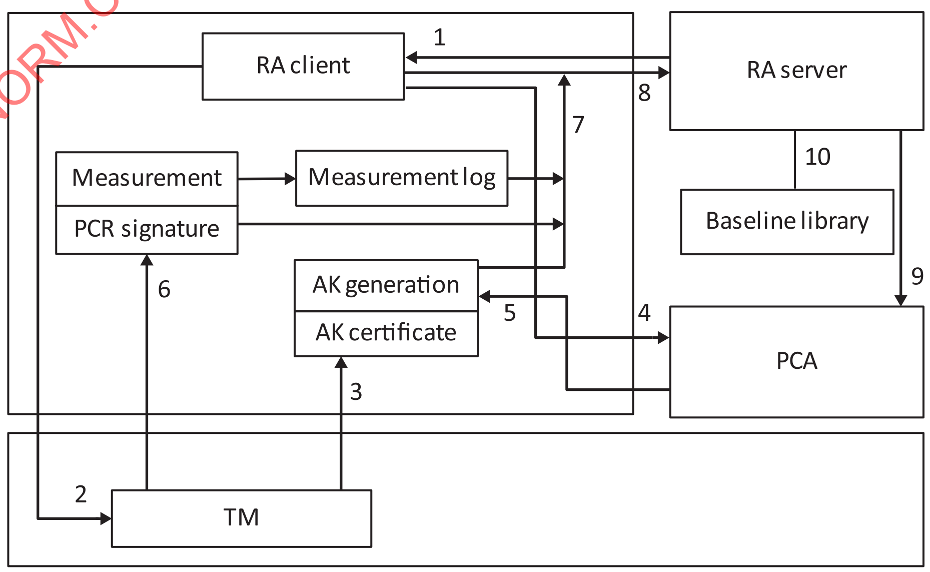

Figure 5 — Flowchart of remote attestation activity

The general remote attestation activity process is as follows.

- The RA server sends a remote attestation request and a random value (nonce) to the RA client.
- The RA client requests the TM to generate a new AK.
- The RA client sends the AK certificate request to the PCA.
- The PCA verifies the validity of the request and issues the AK certificate to the RA client.
- The RA client requests the TM to sign the PCR values and the nonce by using the private key in the AK certificate, then sends the measurement log, signature and AK certificate to the RA server.
- The RA server requests the PCA to verify the validity of the AK certificate sent by RA client, then the PCA returns the verification result.
- The RA server uses the public key in the AK certificate to obtain the PCR value, and recalculates the PCR value according to the measurement log, finally compares the recalculated PCR value with the PCR value sent by the RA client to determine the integrity of host.
- The RA server checks the integrity of the measurement log sent by the RA client by referring to the baseline values in the baseline library.

6.5 Data protection

6.5.1 General

In the cloud computing environment, the TM/vTM can protect sensitive data stored in a host and a VM, which processes the data in specific protection ways based on a key management mechanism. The encryption and decryption operations are executed inside the TM/vTM.

In this subclause, there are three typical keys as follows.

- SRK/vSRK: A key is located at the bottom of the key hierarchy and is generated by a storage primary seed in the TM/vTM. It is used to encrypt the upper layer key to build a hierarchical key protection system in TM/vTM.
- KEK: A key is used to encrypt the work key. It is also generated in the TM/vTM and it is the sub-key of the SRK/vSRK.
- WK: A key is used to encrypt sensitive data. It can be generated in the TM/vTM or imported into the TM/vTM. It is the sub-key of the KEK.

  NOTE There can be many WKs imported into the TM from an external source. If all of them are directly managed by the SRK of TM, the computational overhead of TM will be increased. In order to improve the efficiency of data encryption and decryption, and to strengthen the security of key management, it adds a middle layer for key management mechanism. To be specific, the WK is a symmetric key used to encrypt user data, and the KEK is an asymmetric key used to encrypt the WK by its public key. The private key of KEK is encrypted by the SRK. The SRK cannot be used for encryption and signing other non-TM state data.

Data protection activity consists of two ways:

- data binding;
- data sealing.

6.5.2 Data binding

Encrypt sensitive data (e.g. symmetric key) by using a TM/vTM internal key. The data binding is illustrated in Figure 6.

The general data binding process is shown as follows.

- The TM/vTM uses the SRK/vSRK to encrypt the KEK.
- The TM/vTM uses the KEK to encrypt the WK.
- The TM/vTM uses the WK to encrypt the sensitive data to generate the encrypted data.

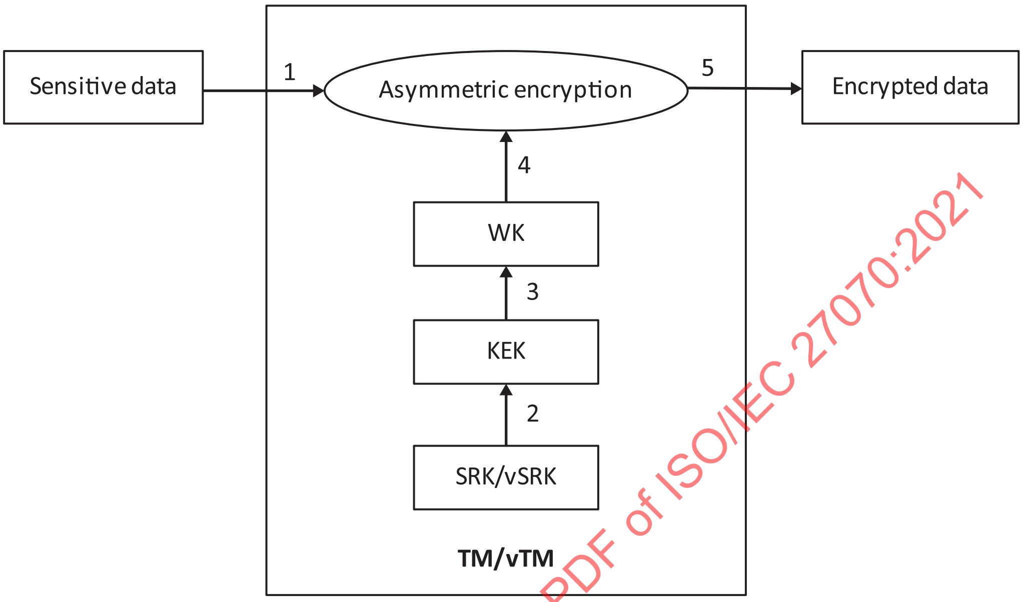

Figure 6 — Flowchart of data binding activity

6.5.3 Data sealing

Encrypt sensitive data (e.g. symmetric key) by using the TM/vTM internal key and PCR values. Once the computer status changes (e.g. software/firmware tampering), the sensitive data cannot be decrypted. The data sealing is illustrated in Figure 7.

The data sealing process is shown as follows.

- The TM/vTM uses the SRK/vSRK to encrypt the KEK.
- The TM/vTM uses the KEK to encrypt the WK.
- The TM/vTM binds the sensitive data and PCR value to use the WK to generate the encrypted data.

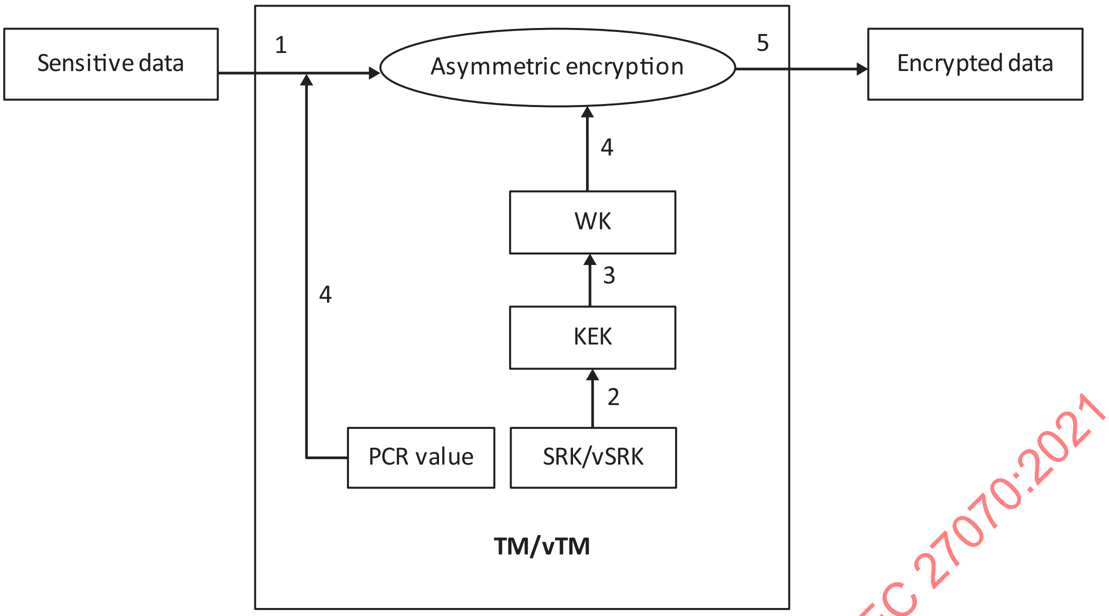

Figure 7 — Flowchart of data sealing activity

6.6 vTM migration

When migrating a VM from the source side to the destination side, a number of steps are required to migrate the associated vTM instance. Source side and destination side should verify the measured values of each side to establish trust, and the measured values should be the values of total VMM layer. It is critical that the sensitive data housed within the associated vTM are well protected during the migration. The vTM migration activity is illustrated in Figure 8 and these steps are initiated by the migration controller.

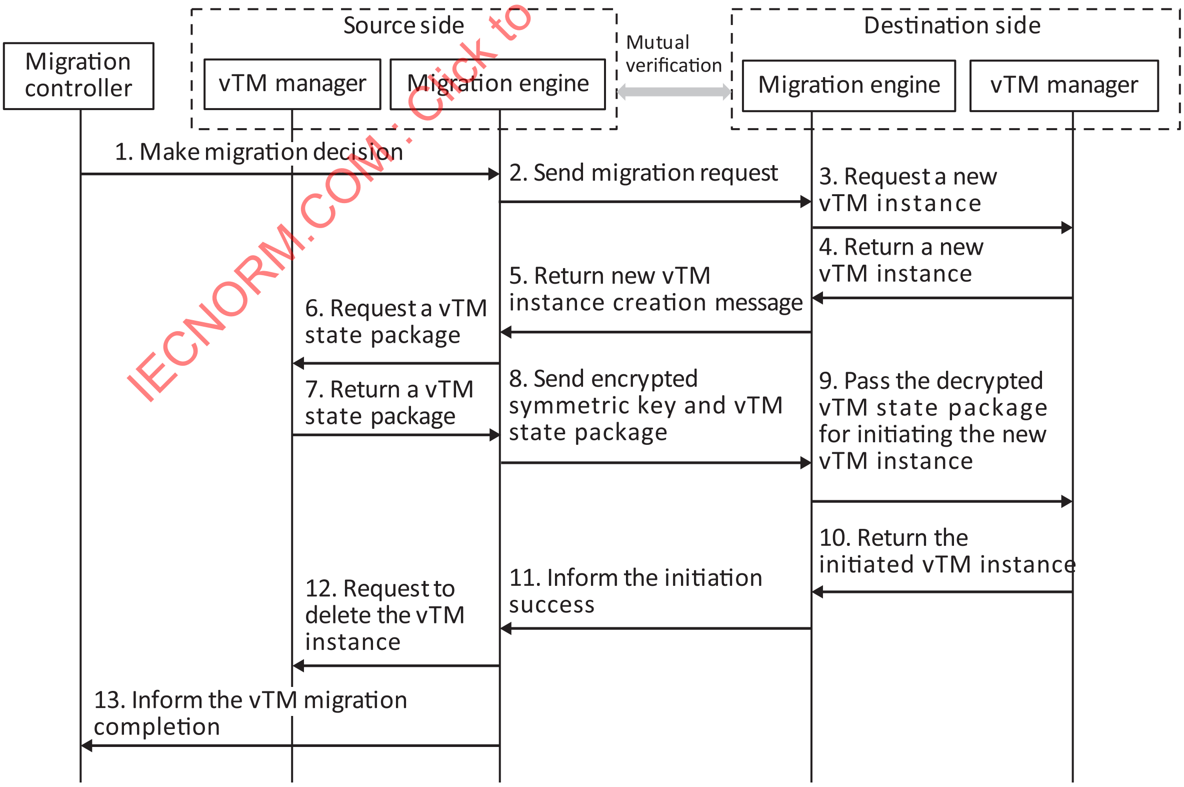
Source side
Mutual verification
Destination side
Migration controller
vTM manager
Migration engine
Migration engine
vTM manager
1. Make migration decision
2. Send migration request
3. Request a new vTM instance
4. Return a new vTM instance
5. Return new vTM instance creation message
6. Request a vTM state package
7. Return a vTM state package
8. Send encrypted symmetric key and vTM state package
9. Pass the decrypted vTM state package for initiating the new vTM instance
10. Return the initiated vTM instance
11. Inform the initiation success
12. Request to delete the vTM instance
13. Inform the vTM migration completion

Figure 8 — Flowchart of vTM migration activity