TSINGHUA SCIENCE AND TECHNOLOGY
ISSNll1007-0214 02/10 pp520–534
DOI: 10.26599/TST.2018.9010129
Volume 24, Number 5, October 2019

Cloud Virtual Machine Lifecycle Security Framework Based on Trusted Computing

Xin Jin, Qixu Wang*, Xiang Li, Xingshu Chen, and Wei Wang

Abstract: As a foundation component of cloud computing platforms, Virtual Machines (VMs) are confronted with numerous security threats. However, existing solutions tend to focus on solving threats in a specific state of the VM. In this paper, we propose a novel VM lifecycle security protection framework based on trusted computing to solve the security threats to VMs throughout their entire lifecycle. Specifically, a concept of the VM lifecycle is presented divided up by the different active conditions of the VM. Then, a trusted computing based security protection framework is developed, which can extend the trusted relationship from trusted platform module to the VM and protect the security and reliability of the VM throughout its lifecycle. The theoretical analysis shows that our proposed framework can provide comprehensive safety to VM in all of its states. Furthermore, experiment results demonstrate that the proposed framework is feasible and achieves a higher level of security compared with some state-of-the-art schemes.

Key words: virtual trusted computing; virtual machine lifecycle; trusted chain; security measurement; state monitoring

1 Introduction

Due to their strengths of flexibility and scalability, Virtual Machines (VMs) have become the main mode of tenant service deployment in cloud computing environments. As an independent computing entity, besides facing traditional security threats, VMs also confront some security threats specific to cloud computing; namely, those security threats that specifically target different states of the virtual environment (e.g., the co-resident attack[1], the data security threats[2–5], the attack to VM storage[6], and the attack to VM migration[7, 8]). Intuitively, in order to achieve a secure environment, comprehensive protection is required for each state of the VM; i.e., what this paper refers to as lifecycle security protection.

Recently, some promising approaches[9–12] have been proposed to solve the security problems facing VMs at different stages of the lifecycle. These approaches work by employing specific methods during the lifecycle of a VM, including (a) *authentication and data encryption*, utilizing isolation technology to protect the CPU and memory context of a VM; (b) *implementing an agent*, installing an agent in a VM to implement security features such as access control, encryption, and integrity verification; (c) *protecting the integrity of the image*, providing security for a VM by protecting the integrity of the image file; and (d) *scanning and updating*, the fine-grained scanning and updating of a VM leveraging the technique of optimized storage deployment.

However, the aforementioned approaches suffer from some major limitations. First, cryptography mechanisms can mitigate the security threats faced

• Xin Jin, Xiang Li, and Wei Wang are with the College of Computer Science, Sichuan University, Chengdu 610065, China. E-mail: xinjin_cn@163.com; 2016323040026@stu.scu.edu.cn; 15762254497@139.com.
• Qixu Wang and Xingshu Chen are with the College of Cybersecurity, Sichuan University, Chengdu 610065, China. E- mail: qixuwang@scu.edu.cn; chenxsh@scu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2018-10-10; accepted: 2018-11-10

by the VM, but guaranteeing the confidentiality and security of the keys that are used in storage and authorization is still an open problem. Second, these methods focus on protecting the security of a particular state of the VM but do not protect the VM across its full lifecycle. Third, some subtle but important states of the lifecycle (e.g., the states of migration and snapshot) are often ignored by existing schemes. Finally, the trusted state of the VM cannot be checked and verified, which means that a security assessment of the operating status of the VM cannot be realized.

Fortunately, trusted computing technology can overcome these problems by creating a trusted environment for VMs. Secure storage of keys is realized by storing them in the Trusted Platform Module (TPM). Moreover, the environment of trusted computing provides a basis for protecting a VM throughout its entire lifecycle. Furthermore, the security of the operating status for a VM can be checked and validated via trusted computing, thus achieving a proper security assessment.

In this paper, we study security threats to VMs, and aim to propose a comprehensive lifecycle protection scheme using trusted computing to achieve perfect security within a trusted chain from the hardware TPM to the VM. This builds on the fact that trusted computing principles have been adopted in many cloud platforms with high security requirements.

We first present a novel model of the VM lifecycle, in which different lifecycle states are divided up in accordance with the different states of the VM. We then discuss the conditions and requirements for the proposed lifecycle model to achieve soundness and security, showing that in order to meet these goals each state of the VM needs to concern itself with and to satisfy certain requirements.

Based on this model, we further study the security protection method for VMs and propose a novel solution using trusted computing. The main idea is to establish the VM as a trusted computing environment by extending the trusted chain from the hardware TPM to the VM initial launcher and then to the applications of the VM. At the same time, the security and legitimacy of the virtual Trusted Platform Module (vTPM) instance are ensured by establishing a physical trusted basis and implementing trusted associations with VM instances.

In a nutshell, the main contributions of this paper are as follows.

- We present a novel security protection framework for the lifecycle of a VM based on trusted computing, in which the states of the lifecycle are divided up by the different states of the VM. The proposed framework can provide a dependable level of security and reliability for each state. Simultaneously, a deep attestation method for the VM's running state is implemented.
- A comprehensive complexity and security analysis of the proposed framework in terms of the trusted VM lifecycle and security protection framework is provided. The analysis results show that the proposed framework can build a trusted environment and ensure the security and dependability of a VM throughout the lifecycle.
- To illustrate feasibility and availability, we conduct the experiments on a real OpenStack-based cloud computing environment. The results demonstrate that the proposed framework realizes the ability for centralized management of the trusted information of VMs, and provides complete security protection throughout the VM lifecycle.

The remainder of this paper is organized as follows. In Section 2, we review related work. We revisit the preliminaries in Section 3 and define the system model, security threats, and security goals in Section 4. Section 5 follows with a description of our proposed framework. We discuss and prove the security proposals in Section 6, and provide the implementation and analysis of our proposed framework in Section 7. Finally, we conclude our paper in Section 8.

2 Related Work

In recent years, many studies have been conducted on the protection of VMs against security threats in different operation states. We trace the related works organized into the different VM states that they are concerned with.

In the storage state, the main security threat is the leakage of the VM image file. Although encryption is a common method to protect the VM image files[6], in the case of a large VM instance it consumes a high level of resources. Moreover, before running the encrypted VM instance it must be decrypted, which equates to even more severe resource consumption. Meanwhile, data leakage can still be a problem if the decrypted image file is stolen. To solve this drawback, Schwarzkopf[12] provided an optimized storage method to improve the efficiency of VM image file updating and vulnerability detection. Nonetheless, this method does not consider the secure storage of private data in the running time of the VM.

In the running state, data leakage[13] is the major threat to the VM. Existing solutions (e.g., encryption[14], authentication and access control[15], and infrastructure protection[16]) cannot solve the fundamental problem behind the data leakage problem, which is the confidentiality and security of encryption keys. The methods proposed in Refs. [17, 18] can be used to guarantee the security of the VM's data transmission process. Intel Software Guard Extensions (Intel SGX) can provide a trusted environment within an untrusted cloud platform to protect private data in a VM[19, 20]. In addition, the Intel Trusted Execution Technique (Intel TXT) can dynamically protect the security of the Virtual Machine Monitor (VMM), i.e., the hypervisor[21]. Unfortunately, both of these techniques require the support of security hardware (e.g., CPU and Basic Input Output System (BIOS)), which makes them difficult to make use of in a virtualization environment. Meanwhile, the state of the application in the VM is also worth studying. Existing research proposes methods that focus on improving VM performance[22] or detecting VM behavior from outside of a VM[23], and rarely looks at protecting applications in VMs. Reference [24] proposed an attestation method for applications in a VM, but this requires deploying an agent, the security of which is difficult to guarantee.

In the migration state, the most common attacks are migration target deception and man-in-the-middle. To overcome these threats, Wan et al.[25] proposed a protocol to assure the security requirements of VM migration using the methods of property-based attestation and trusted channel. Sun et al.[26] presented a mechanism to protect the dynamic data migration of a VM by utilizing the centralized management platform, virtual security gateway, hypervisor access engine, and VM security agent. Besides, in some studies (e.g., Refs. [7, 8]), host optimization algorithms were proposed to settle the problems of energy consumption, communication delays, and migration cost. The premise of the implementation of these algorithms should be that the target host is secure.

In the states of deployment and deletion, practical solutions have yet to emerge for the security of VMs. Existing studies either focus on improving performance[27–29] or delegate the security responsibilities to the Cloud Service Provider (CSP)[4].

It is feasible to use trusted computing to provide comprehensive security protection for each operation state of a VM. Mao[9] proposed a trusted and secure computing environment to the user, employing a proxy VM and a trusted computing base to prevent unauthorized access. Barak et al.[10] presented an approach to implement policy extraction, user control, encryption, local integrity verification, and access control in the VM. Forrester et al.[11] proposed a method that can verify the integrity of a VM. However, all of these research solutions require deploying agents in the VM; even though first-hand data can be obtained, the security of the agent still cannot be effectively guaranteed.

In order to solve these aforementioned problems, we propose a security framework using trusted computing to provide comprehensive security protection for a VM throughout its lifecycle. Instead of deploying agents, the proposed framework uses Virtual Machine Introspection (VMI) technology to collect information in VMs, builds a trusted environment for VMs, and provides a trusted guarantee for each state of the VM lifecycle.

3 Preliminaries

In this section, we briefly revisit the preliminaries used to construct our framework, including trusted computing, virtual trusted computing, and trusted attestation.

3.1 Trusted computing

In the context of Trusted Computing Group (TCG) specifications, *trust* is meant to convey an expectation of behavior[30]. With the TPM technology of TCG, we can protect systems by the means of secure key storage, cryptographic protection, and integrity measurement[31]. The trusted computing platform has the functions of ensuring data integrity, data security storage, and platform remote attestation[32].

TPM ensures the integrity of the system data through a *Root of Trust for Measurement* (RTM) and *trusted chain*, protects data security using cryptographic technology, and stores trusted measurement information through Platform Configuration Registers (PCR). In order to prevent the PCR value from being tampered with, the TPM restricts the behaviors of the user operating the PCR, allowing only reset and extension operations. The reset operation clears the PCR value automatically when the machine is powered off or restarted, while the extension operation is done in the following way.

$$\text{NewPCR}_i = \text{Hash}(\text{OldPCR}_{i-1} || \text{NewHashValue}) \quad (1)$$

where the symbol || indicates the conjunction, the

OldPCR_{i-1} indicates the value of the PCR before i extension, the NewHashValue indicates the hash value to be extended, and the NewPCR_i indicates the PCR value after the extension.

Since the extension operation is irreversible, the PCR value obtained by first extending the measurement value A and then extending the measurement value B is different from the value of extending B first and then extending A.

In the system startup process of a trusted computing platform, the module to be loaded is measured and the measurement value is extended to PCR. In theory, the PCR can record an infinitely long measurement sequence in the manner of Function (1). Due to the iterative nature of the extension process, if the measurement value of a component is changed, the PCR values of the subsequent measurement will be affected.

The measurement components and the sequence are stored in the Stored Measurement Log (SML). The SML is saved in low-security file systems, while the PCRs are protected by the high-security TPM. However, since the two are interrelated, even if the SML is tampered with, the tampering behavior can be detected by the PCR. Through the extending operation of the PCR, the trusted relationship scope extends from the RTM to the BIOS, system hardware, boot loader, and operating system. The trusted relationship upward transfer process has been labeled as a "chain of trust", or trusted chain[33]. Through the above analysis, we can see that this trusted chain can be described through the SML and PCR extension process.

3.2 Virtual trusted computing

With the development of cloud computing, there is a need to use trusted computing technology to protect the data in VMs. A vTPM is required to provide TPM services to a VM running on top of a hypervisor. In 2006, Perez et al.[34] proposed the first vTPM architecture. Since it is implemented using a software simulation, the following conditions need to be satisfied in the vTPM implementation:

- A vTPM must provide the same usage model and TPM command set to an operating system running inside a VM as a hardware TPM provides to an operating system running directly on a hardware platform;
- A strong association between the VM and vTPM must be maintained across the lifecycle of the VM, which includes the migration of the VM (together with its associated vTPM) from one physical machine to another;
- A strong association between the vTPM and its underlying Trusted Computing Base (TCB) must be maintained; and
- A virtual TPM must be clearly distinguished from a hardware TPM because of the different security properties of these two types of TPM.

The vTPM architecture was first implemented in Xen. As the KVM architecture has become mainstream, vTPM implementation solutions based on KVM[35] architecture have emerged and are now widely used[24,36].

3.3 Trusted attestation

The goal of trusted attestation is to prove to a remote party that the operating system and application software are intact and trustworthy[37]. Trusted attestation usually includes three roles: a challenger, a target system, and an authoritative third party, e.g., a Certificate Authority (CA). First, the target system is equipped with a TPM and a trusted computing environment has been established by trusted chain. Then the challenger sends an attestation request to the target system, and the attestation information signed by the TPM is collected and sent back to the challenger. Since the signed key was generated from the TPM and certified by the CA, the challenger trusts the source of the attestation information. Finally, with the attestation information of the latest PCR value and the SML files, the challenger reproduces the PCR extension process to verify the trustworthiness of the target system. Trusted attestation technology can be used to testify the integrity of the target platform[38], checking each component of the trusted chain. The integrity of the trusted chain then characterizes the trusted status of the target system.

4 Problem Statement

In this section, we state the problem by formalizing the lifecycle model and system model, and by identifying the security threats and security goals.

4.1 Lifecycle model

As depicted in Fig. 1, the lifecycle model of a trusted VM consists of 6 main states: creation, storage, deployment, execution, exit, and deletion, and 3 additional states: suspending, snapshot, and migration.

(1) *Creation state*. The VM is created and the Operating System (OS) is installed in this state. A template can be used to speed up the creation process instead of creating everything from scratch. Once the

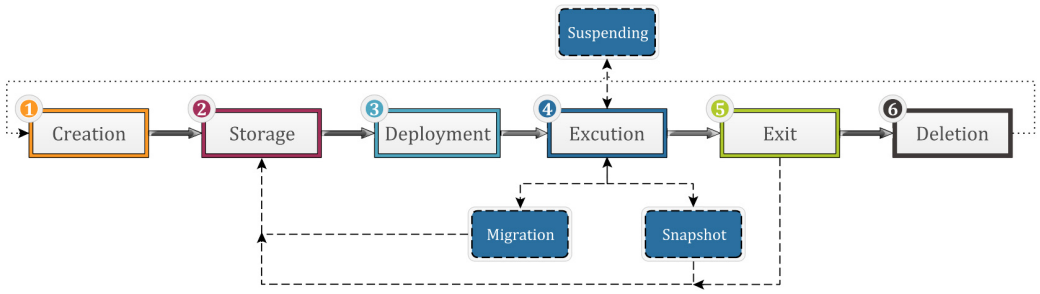


Fig. 1 Lifecycle of trusted VM.

VM is created, the associated vTPM instance is created as well.

(2) *Storage state*. The VM and vTPM instances are being kept in a dormant state on an appropriate storage system. In order to maintain confidentiality, the private data of the VM instance and vTPM instance are stored in encrypted form.

(3) *Deployment state*. Before loading the VM, the vTPM instance is decrypted and relationships of the VM to the vTPM and of the TPM to the vTPM are examined to verify the trustworthiness of the vTPM.

(4) *Execution state*. The trusted VM is executed to achieve its intended purpose. The guest operating system uses the vTPM as if it was a hardware TPM, and the persistent data of the vTPM is saved to the vTPM instance file continuously.

(5) *Exit state*. The trusted VM saves the persistent data generated by the execution state to the instance file and then the VM is shut down.

(6) *Deletion state*. In this state, all the contents of the trusted VM are deleted, including VM instances, vTPM instances, and VM security measures.

(7) *Suspending state*. The trusted VM gives up its CPU and memory resources after saving persistent data to the instance file. When needed, the trusted VM will regain the CPU and memory resources and resume execution.

(8) *Snapshot state*. The trusted VM first saves its operational state to a snapshot template (including the VM and associated vTPM). Then, when needed, the new trusted VM is created directly from the template files to quickly recover the state of the snapshotted trusted VM.

(9) *Migration state*. The operating environment of a trusted VM is transferred from one physical machine to another, along with its associated vTPM instance.

4.2 System model

The architecture of the VM lifecycle security framework consists of compute nodes and a cloud management center, as shown in Fig. 2. The compute nodes consist of the hardware infrastructure, trusted

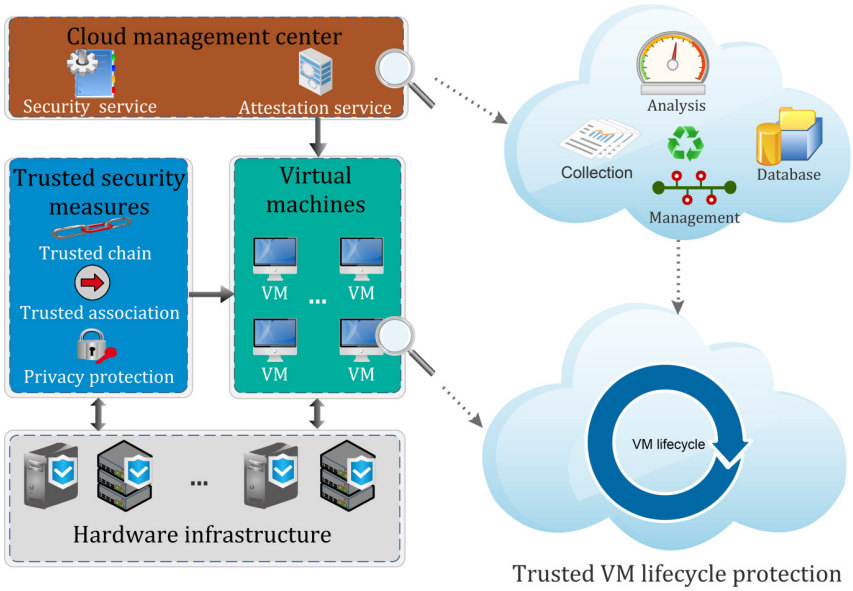


Fig. 2 Architecture of virtual machine lifecycle security framework.

security measures, and VMs. The cloud management center consists of the security and attestation services.

• *Compute nodes*. The compute nodes implement the basic functions of cloud computing through the VMs that are deployed on them. The TPM chip is deployed to each physical compute node, while vTPMs are deployed to each VM. The security measures based on trusted computing (i.e., a trusted chain from hardware TPM to VM, association between TPM and vTPM, association between vTPM and VM, and VM private data protection) are implemented to protect the security of the VM throughout the lifecycle.

• *Cloud management center*. The cloud management center is responsible for monitoring and managing cloud computing platforms. The platform administrator can issue a security strategy to manage the VMs. Meanwhile, the attestation service collects and analyzes the trusted information for each VM and the host OS. Comparing the analysis results with the trusted information database, the status of the VM and host OS can be verified and the results can be shown in the operation interface of the cloud platform administrator. This enables the administrator to monitor and manage VMs responsively.

4.3 Security threats

Security threats to the VM of the cloud may come from both external and internal attackers. External attacks include identity attacks, non-authorized logins, trojans and viruses, and channel attacks. Internal attacks include VM bypass attacks, vulnerability detection, and VMM privilege attacks. Attackers can inject trojans to control the VM to implement the attacks of vulnerability detection and VM escape to launch bypass attacks on other VMs. Attackers can use sniffers and implement channel attacks when the VM is communicating with external machines. If the VMM is compromised[2], an attacker can use VMM privilege attacks on VMs on the same host; in this condition, all protections in the VM are invalidated. Moreover, the VM can be stolen and reused in another environment, which can lead to a leakage of private data.

Since the cloud management center and physical compute nodes are maintained by a CSP, and normal users do not have access to the physical infrastructure of the cloud, we can assume that the cloud management center and physical cloud infrastructure are secure. Further, we can assume that CSP security management measures are relatively complete, and capable of preventing identity attacks, physical attacks, and internal attacks.

4.4 Security goals

To enable VM lifecycle protection under the aforementioned system model and resist the security threats, our framework should achieve the following security goals.

• Full lifecycle protection. Every state of the VM needs to be protected. We aim to provide methods for full lifecycle protection to ensure that the VM is always in a safe and trustworthy state. Therefore, an attacker cannot use any state of the VM to carry out an attack.

• Trusted environment. A trusted computing environment for VMs will be established. In these circumstances, any hostile action by an adversary can be detected in a short enough time to ensure the safety and credibility of the VM.

• Verifiability. The trusted status of the VM and host OS should be verifiable to ensure the correctness and trustworthiness.

5 Proposed Framework

In this section, we propose the framework of VM lifecycle security protection, which includes three main procedures: hardware security support, strong association establishment, and trusted status monitoring. The procedure of strong association establishment can continue to be subdivided into (1) association between TPM and vTPM, and (2) association between vTPM and VM.

5.1 Hardware security support

VMs are implemented by software simulation, which can leave them lacking in a security foundation. Authentication and cryptographic mechanisms can alleviate the security threat through access control and data encryption, but they can neither give a security foundation to the guest OS, nor can they verify the security status of the VM. Therefore, it is necessary to provide a hardware security foundation for VMs.

In order to provide security support for VMs, a trusted chain is built from the hardware TPM to the VM. The trusted chain consists of the core components of the host OS, the hypervisor, and the VM initial launcher. The collection and verification of trusted information in the trusted chain ensure the security and trust of the core components, thus ensuring the safety and trustworthiness of the environment under the VM.

The trusted chain can also be established in the VM, but if the root of the chain, i.e., the virtual Core Root of Trust Measurement (vCRTM) was to be destroyed, the trusted chain of the VM would become untrusted. To ensure the security of the vCRTM, we propose a method of extending the trusted chain of the host OS from the hardware TPM to the VM initial launcher. First, the hypervisor is measured to ensure its security, then the VM initial launcher is measured by the hypervisor. The measurement values are extended to the PCR and SML, which provide hardware security support to VMs.

The hypervisor and the VM initial launcher measurement method is shown in Fig. 3. The BIOS and vTPM simulators are measured by the hypervisor to ensure their integrity. The BIOS simulator corresponds to the vCRTM and vBIOS in the VM, and the vTPM emulation program corresponds to the vTPM device and the vTPM instance; this method protects the integrity of the VM initial launcher and provides base hardware security support to VMs.

When the trusted chain of the host OS is proved to be trustworthy, the initial launcher of the VM is also trustworthy, which is the foundation for establishing the VM's own trusted computing environment.

5.2 Strong association establishment

5.2.1 Association between TPM and vTPM

Since the vTPM is simulated with software, with no hardware TPM to serve as underlying trusted support, the vTPM will fall into the vicious circle of one piece of software being used to verify another, and thus lose legitimacy as the secure root of the VM. Therefore, it is necessary to establish a trusted association between the hardware TPM and the vTPM.

The TPM-vTPM trusted association method is shown in Fig. 4. It introduces an intermediate level, namely the vTPM measurement list, between the vTPM and TPM. The information of vTPMs (including name and baseline measurement value) is saved in the vTPM measurement list. When a trusted VM is deployed, the vTPM loading operation can be implemented only if the vTPM measurement is equal to the baseline in the vTPM measurement list. The vTPM measurement list is usually in an encrypted state, and the encryption key is sealed by the TPM with multiple PCR state values as the seal condition. When the vTPM measurement list needs to be decrypted, the encryption key can only be unsealed when the corresponding PCR values are consistent with the seal operation.

The vTPM measurement list guarantees the integrity of the vTPM instance; the TPM guarantees the secure storage of the vTPM measurement list. In this way, the TPM to vTPM association is established.

5.2.2 Association between vTPM and VM

If the vTPM instance is not strongly associated with the VM instance and can be mounted by another VM instance, information leakage. Moreover, persistent information of the vTPM instance is stored in plaintext, which may lead to the disclosure of confidential information (e.g., the TPM owner password). The vTPM instance therefore should be stored confidentially and a strong association between the VM and vTPM should be established.

The VM-vTPM association and private data protection method is shown in Fig. 5. This method makes use of the capacity of the VM instance file (such as the QCOW2 storage format) to store custom information in the header label structure. The header label structure was designed to store the mapping information of the Universally Unique IDentifier

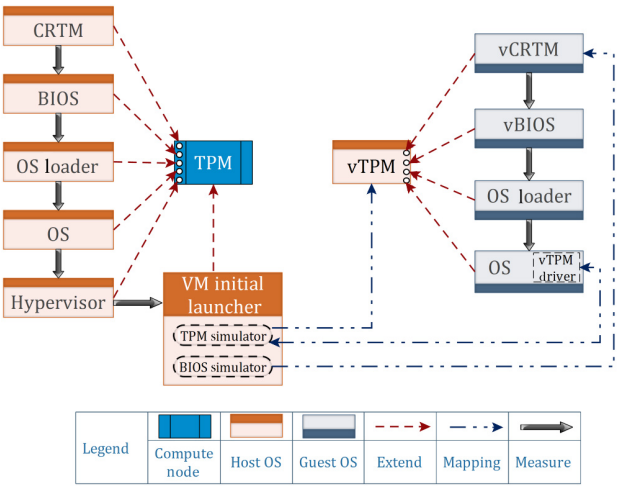


Fig. 3 Extend the host trust chain to the VM initial launcher.

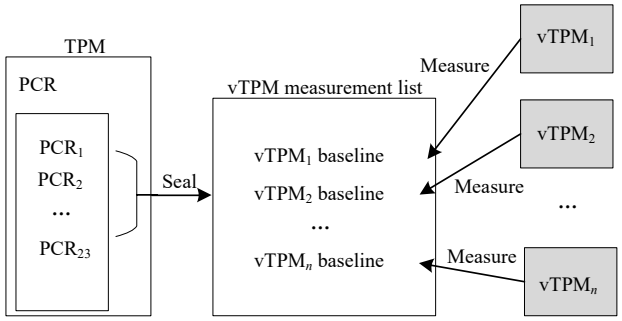


Fig. 4 Trusted association between hardware TPM and vTPM.

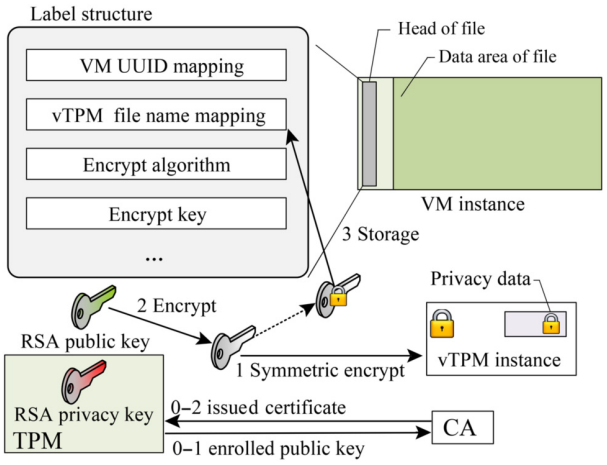


Fig. 5 VM-vTPM association and private data protection method.

(UUID) of the VM, the mapping information of the vTPM instance file name, the encryption algorithm of the vTPM instance, the encryption key of the vTPM instance, etc. With the information from the label, a strong association can be established between the VM and the vTPM instances.

The procedure for VM-vTPM association is as follows:

• The TPM generates non-migratable Rivest-Shamir-Adleman (RSA) key pairs, with the private part saved to the non-volatile area of the TPM, and the public key enrolled to CA to attain a certificate to represent the TPM;

• A symmetric key is used to encrypt the vTPM instance file, and this key can also be used to encrypt the private information of the vTPM instance in the execution state;

• The symmetric key is encrypted with the RSA public key, which establishes the relationship between the TPM and the VM; and

• The ciphertext is saved to a field in the label structure, and the label structure is saved in the header of the VM instance.

When starting the trusted VM, if the vTPM is not associated with the VM, the loading process would be halted.

5.3 Trusted status monitoring

In its execution state, the VM's trusted status can be affected by the underlying environment (e.g., hypervisor) and by active security breaches (e.g., a compromised kernel in the guest OS). Therefore, the trusted status of the running VM needs to be checked and verified at execution time. The TCG's deep attestation method verifies the host OS after the attestation of the guest OS, and the host OS Uniform Resource Identifier (URI) is provided by the guest OS, which can lead to spoofing attacks on the hypervisor[39]. The method for deep attestation based on Virtual Machine Introspection (VMI) is proposed in this paper. This method can collect the trusted information of VMs and the host OS at once, and ensure the authenticity of the information via the hardware TPM.

The VM trusted status monitoring method is shown as Fig. 6. First, the trusted chains are established in the VM and the host OS. Then, an agent is deployed in the hypervisor to collect the trusted information from the running VMs using VMI technology. Since the VMI technology can obtain VM trusted information without deploying agents, it is not subject to attacks on agents and is able to collect data from the VM without the perception of the user, which ensures the accuracy and reliability of the collected data.

Once the collected information of the VMs and host OS has been packaged and sent to attestation service, the trusted information is analyzed and compared with the baseline data in the database, thus verifying the integrity of the trusted chain and resulting in the trusted status of the VM (or host OS).

The verification results are shown on the cloud manager platform; the platform administrators can then safely manage VMs based on these results.

6 Framework Discussion

In this section, the framework is thoroughly analyzed in its advancements, security, and trustworthiness.

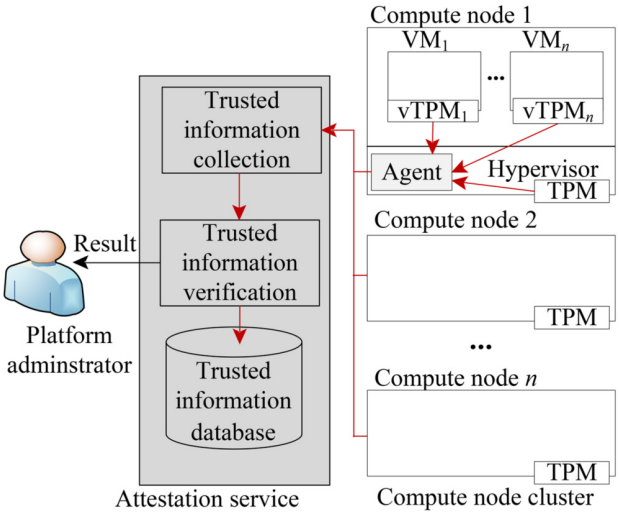


Fig. 6 VM trusted status monitoring method.

6.1 Advancements

The advancements made by the framework are as follows:

• Hardware security support. This method establishes a trusted chain from the hardware TPM to the initial launcher of the VM. It connects the internal trust chain of the VM to the host trust chain, and provides a physical foundation for the trusted chain of the VM.

• TPM-vTPM association. The vTPM measurement list method establishes the relationship between the TPM and the vTPM. It maintains a loose coupling between them, which is suitable for rapid restoration of the trusted chain after VM migration.

• vTPM-VM association. By making use of the characteristics of the custom label structure stored in the VM instance header, the correspondence between the compute node, VM instance, and vTPM instance is established. Moreover, the integrity of, confidentiality of, and private information within the vTPM instance are protected by encryption and decryption mechanisms.

• Trusted status monitoring. The VMI technology can collect the trusted status of all VMs on the same compute node in a single acquisition, which improves collection efficiency and ensures a correspondence between the VM and the host OS. This method can effectively prevent hypervisor spoofing attacks during deep remote authentication.

6.2 Security

The security analysis of the framework is as follows:

• Hardware security support. The main idea of this method is the TCG trusted chain extension mechanism. The extending operation of the TPM measuring the OS, the OS measuring the hypervisor, and the hypervisor measuring the VM initial launcher operation are in accordance with the TCG's trusted chain extension concept. Since the calculation and extension of measurements conform to the specifications of trusted computing, this method can be considered safe.

• TPM-vTPM association. This method applies the TCG's ideas on storage security. The vTPM instances are measured and the values stored in the vTPM measurement list, then the vTPM measurement list is encrypted by a symmetric key and the key is sealed by the hardware TPM with the PCR. Only where the PCR value is the same as it was at sealing time can the symmetric key be unsealed and the vTPM measurement list be decrypted. Since the measurement and sealing are basic trusted computing operations and the security of these operations has been validated, this method can be considered safe.

• vTPM-VM association. The main idea behind this method is the cryptography mechanism. Using the capacity to store a custom label structure in the VM instance file header, the vTPM instance can be encrypted by a symmetric key, the symmetric key can be encrypted by the RSA public key of the node, and the ciphertext cloud can be saved to the label structure. The label structure does not have any impact on the normal operation of the VM instance, and the confidentiality and integrity of the data in the label structure are protected by the hardware TPM; therefore, this measure can be considered safe.

• Trusted status monitoring. This method using the TCG's principle of remote attestation. Since the trusted information collection and verification process conforms to the trusted computing remote verification specification, this method can also be considered safe.

6.3 Trustworthiness

The trustworthiness of the framework can be proved by the integrity and confidentiality of the VM throughout its lifecycle.

• Creation state. The newly created vTPM instance is registered in the vTPM measurement list, the vTPM instance is encrypted by a symmetric key that was encrypted by the RSA public key of the node, and the vTPM measurement list is encrypted by a symmetric key that is sealed by the node's TPM. These methods protect the integrity and confidentiality of the VM and vTPM on creation.

• Storage state. When the trusted VM instance is being kept in a dormant state, the associated vTPM is encrypted with the key in the label structure of the VM instance. Since the relationship of the VM, vTPM, and compute node are maintained and cannot be changed without the privilege of the TPM, integrity and confidentiality are maintained in the storage state.

• Deployment state. The process of decrypting the vTPM instance proves the relationship between the VM, vTPM, and compute node. Only once the integrity of the vTPM instance is proved can the VM be loaded for execution, which ensures integrity and confidentiality on deployment.

• Execution state. A trusted computing environment is established to keep the VM in a safe and trusted

state, and the trusted information of the VM and host OS are collected and verified in real time by trusted status monitoring measures. These methods protect the integrity and confidentiality of the VM when it is in an execution state.

• Suspend state. The private information of the VM is encrypted and saved to the instance file, so confidentiality is maintained while the VM is suspended.

• Snapshot state. The vTPM template is encrypted by the key stored in the header label of the VM snapshot template. When a trusted VM instance needs to be created from a snapshot template, the new vTPM instance is copied from the vTPM template and the measured value is enrolled to the vTPM measurement list. A symmetric key is then used to encrypt the vTPM instance file and the node RSA public key is used to encrypt the symmetric key. Finally the ciphertext is saved to the label structure in the header structure of the VM that is copied from the VM snapshot template. These methods protect the integrity and confidentiality of newly created trusted VMs.

• Migration state. When the vTPM instance is encrypted and transmitted to the destination node, the migration process proves the authentication of the destination node and the integrity of the vTPM instance. The method of moving the encryption key from the source node to the destination node and resuming the trusted chain in the destination node maintains the integrity and confidentiality of the post-migration VM.

• Exit state. The baseline of the vTPM instance is updated to the vTPM measurement list and the vTPM instance is encrypted by the symmetric key in the label structure of the VM instance, then the vTPM measurement list is encrypted by a symmetric key that is sealed by the TPM. These methods protect the integrity and confidentiality of the VM on exit.

• Deletion state. Because the private information of the VM is protected by the vTPM, even if the VM persists after deletion no private information could be decrypted without the vTPM instance and the TPM. This protects the confidentiality of the VM after deletion.

In summary, the framework can ensure the trusted status of a trusted VM throughout its lifecycle.

7 Implementation and Evaluation

In this section, the framework is implemented in an OpenStack+KVM+QEMU environment, and its efficacy and performance impact are evaluated. For convenience of implementation, the objects are defined in Table 1. The prototype system was built using five servers, of which one is used as a control and network node and the other four act as compute nodes. The basic configuration of the servers is Xeon CPU E5-2623 v3 3.00 GHz/DDR4 16 G×8/10T/TPM1.2/ubuntu 14.04.

7.1 Main states of the lifecycle

The main security measures used during the lifecycle of a trusted VM are shown in Fig. 7.

To implement the security measures in the trusted VM lifecycle, some preparation work is required. First, the trusted chain of the host OS is created when the host OS is running. Then, the RSA key pairs are generated by the TPM, the RSA private key is stored in the TPM, and the RSA public key is used to identify the node. Finally, the vTPM measurement list is generated and encrypted by the hardware TPM.

In the creation state, a VM instance and the associated vTPM instance are created. Then, the measurement value of the vTPM instance is registered to the vTPM measurement list. Finally, the vTPM instance is encrypted by a symmetric key, the symmetric key is encrypted by the RSA public key, and the ciphertext is saved to the label structure in the VM

Table 1 Object name and description of the framework.

Object name	Description
X	Object (VM, vTPM, etc.)
TrustedChain	Trusted chain
PrivaceData	Private data (e.g., TPM_Owner)
vTPM_M_List	vTPM measurement list
Label$_X$	Label structure of X object
Key, key$'$	Symmetric encryption keys

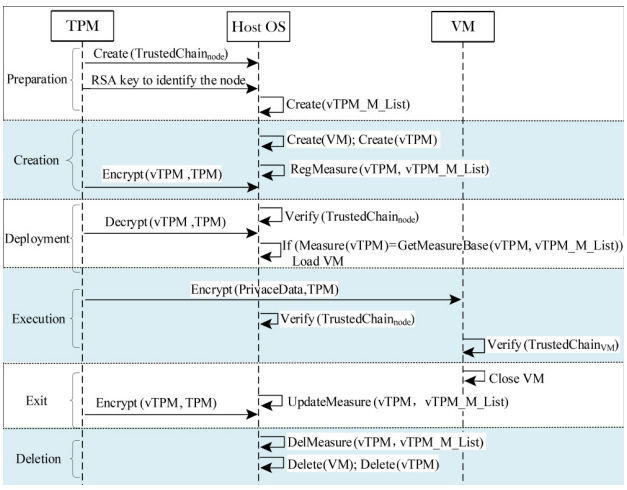


Fig. 7 Main security measures for a trusted VM lifecycle.

instance file header.

In the deployment state, the security conditions of the VM and vTPM instances are examined before loading. First, the trusted status of the host is verified, then the vTPM instance is decrypted by the TPM. Finally, the vTPM instance is measured and the measurement value is compared to the baseline of the vTPM in the vTPM measurement list; if the value matches, the follow-up operation can be performed, otherwise the follow-up operation is blocked.

In the execution state, a series of security measures are used to protect the safe operation of the VM. First, the VM's trusted computing environment is established, then the private information of the vTPM instance is encrypted. Finally, the trusted status of the VM and host OS is collected and validated in real time.

In the exit state, the vTPM instance is measured and the value is updated as a baseline to the vTPM measurement list, then the vTPM instance is encrypted by the TPM.

In the deletion state, the baseline of the vTPM, the VM instance, and the associated vTPM instance are deleted.

7.2 Snapshot state

The main work of snapshotting includes the *creating trusted snapshot template* and *creating trusted VM from snapshot template* steps. The method is shown as Algorithm 1.

Algorithm 1 Snapshot

procedure Trusted Snapshot
  function Snapshot_template (VM, vTPM)
    if VM is suspended & data saved finish
      $\mathrm{VM_{ST}} = \mathrm{VM}$
      $\mathrm{vTPM_{ST}} = \mathrm{vTPM}$
      $\mathrm{vTPM_{ST\text{-}en}} = \mathrm{Encrypt}\,(\mathrm{vTPM_{ST}},\ \mathrm{key})$
      $\mathrm{Label_{VM\text{-}ST}}:\ \mathrm{vTPMKey} = \mathrm{key}$
      resume VM
    end if
    return $\mathrm{VM_{ST}},\ \mathrm{vTPM_{ST}}$;
  end function

  function VM_from_template ($\mathrm{VM_{ST}},\ \mathrm{vTPM_{ST}}$)
    if Verify ($\mathrm{TrustedChain_{node}}$) = True **then**
      $\mathrm{key} = \mathrm{Label_{VM\text{-}ST}}:\ \mathrm{vTPMKey}$
      $\mathrm{vTPM_{ST}} = \mathrm{Decrypt}\,(\mathrm{vTPM_{ST\text{-}en}},\ \mathrm{key})$
      $\mathrm{VM} \Leftarrow \mathrm{VM_{ST}}$
      $\mathrm{vTPM} \Leftarrow \mathrm{vTPM_{ST}}$
      RegMeasure (vTPM, vTPM_*M*_List)
      Encrypt (vTPM, key′)
      $\mathrm{Label_{VM}}:\ \mathrm{vTPMKey} =$
        $\mathrm{Encrypt}\,(\mathrm{key}', \mathrm{RSA}_{\mathrm{node}}^{\mathrm{pub}})$
    else
      stop creation, return error information
    end if
    return VM, vTPM;
  end function
end procedure

In the *snapshot_template* function, when the VM is suspended and the persistent data of the VM has been saved to the instance file, the VM snapshot template file ($\mathrm{VM_{ST}}$) is copied from the VM instance file and the vTPM snapshot template file ($\mathrm{vTPM_{ST}}$) is copied from the vTPM instance. The vTPM snapshot template file is then encrypted with the symmetric key saved to the label structure of the VM snapshot template file header. Since the node for subsequent snapshot recovery is not determined, the key does not contain node information. After the vTPM snapshot is encrypted, the VM is resumed.

In the *VM_from_template* function, once the target node trusted status is verified, the vTPM snapshot template is decrypted by the symmetric key from the label structure of the VM snapshot template file header. The VM instance is then copied from the VM snapshot template and the vTPM instance is copied from the vTPM snapshot template. The vTPM instance is measured and the value is registered to the vTPM measurement list. Finally, the vTPM instance is encrypted by a symmetric key, the symmetric key is encrypted by the RSA public key of the target node, and the ciphertext is saved to the label structure in the VM instance file header.

7.3 Migration state

The case of dynamic migration on shared storage is taken as an example to describe the VM migration process, the method for which is shown as Algorithm 2.

Once the trusted status of the destination node is verified and the RSA public key of the destination node is sent to the source node, the source node verifies destination RSA key with the CA, and encrypts the symmetric key of the label structure with the destination RSA public key, which is used to encrypt the vTPM instance that is to be migrated.

Upon the vTPM instance being migrated and decrypted in the destination node, a blank VM and vTPM structure is created in the memory of the destination node. The in-memory data of the VM and vTPM are then transferred iteratively. When the remaining dirty in-memory data is lower than the iteration threshold, the VM is suspended in the source

Algorithm 2 Trusted VM dynamic migration

```
1: procedure Ddynamic_migration (VM)
2:   function Dynamic_migration (VM)
3:     if Verify(TrustedChain_des)=True then
4:       source_node ⇐ RSA_des^pub
5:       source_node encrypt the vTPM with a key
6:       source_node encrypt the key with RSA_des^pub
7:       transmit and decrypt the vTPM instance
8:       Iteratively transferring dirty data
9:       if dirty data < threshold then
10:        suspend the VM in source
11:        resume the VM in destination
12:        Transfer the rest memory data
13:      end if
14:      RegMeasure (vTPM, vTPM_M_List_des)
15:      DelMeasure (vTPM, vTPM_M_List_sou)
16:      delete (vTPM_sou)
17:    end if
18:  end function
19: end procedure
```

node and resumed in the destination node. The rest of the in-memory data of the migration VM is then transferred to the destination node.

After completion of the above procedure, the measurement value of the vTPM instance is registered to the vTPM measurement list of the destination node. The measurement value of the vTPM instance is then deleted from the vTPM measurement list in the source node, and the vTPM instance of the source node is deleted.

7.4 Framework evaluation

After implemented the proposed framework, experiments were done to analyze the performance. Through the experimental data, the performance impact from deploying the proposed framework was obtained and is shown in Fig. 8. Comparing the data for VM deployment, VM snapshotting, and VM migration, it can be seen that the performance impact of the proposed framework is within an acceptable range. Since the framework adds very little code to existing platforms and does not touch the core code of the operating system, the stability of the system can be guaranteed without a major impact on system performance.

A comparison was also made between the TCG's deep attestation method and the proposed trusted status monitoring method of the framework, with the results shown in Fig. 9. Compared to the TCG's verification process, which needs to verify the VM and the host OS at each attestation, our proposed method can simultaneously collect the trusted information of multiple VMs and the host OS on a single occasion. Therefore, as the number of virtual machines increases, the performance of this method is higher than that of the TCG.

The proposed framework was also compared with the existing VM lifecycle security framework, with the results shown in Table 2. The proposed framework analyzes the protection needs of each state of

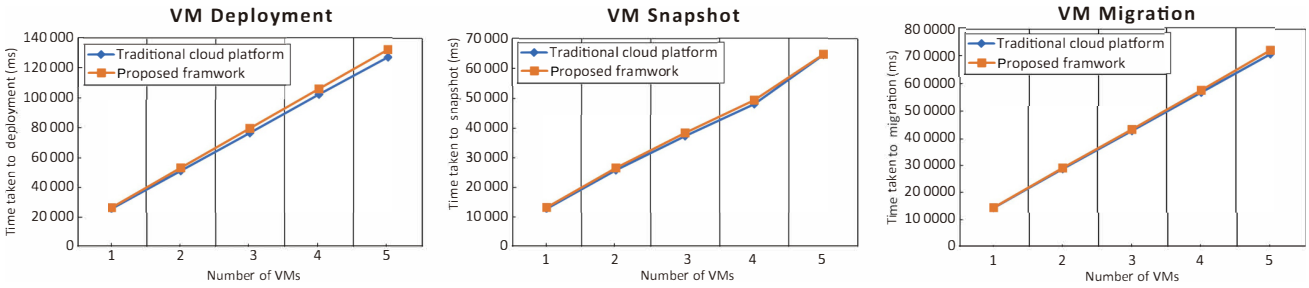


Fig. 8 Performance impact of deployment, snapshot, and migration.

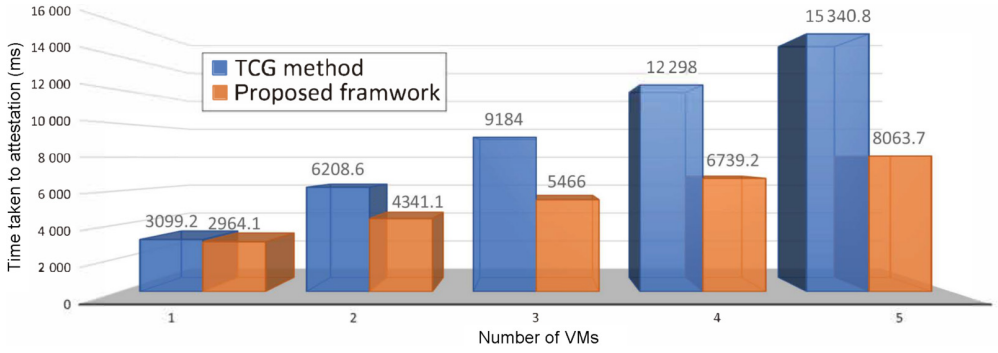


Fig. 9 Comparison of the TCG deep attestation and our attestation method.

Table 2 Comparison of the proposed framework with the existing VM lifecycle security framework.

Model	Protection						
	PI	CF	II	HP	VIL	VTV	HTV
Mao[9]	✓	✓					✓
Barak et al.[10]	✓	✓				✓	
Forrester et al.[11]	✓		✓				
Schwarzkopf[12]			✓		✓		
Proposed framework	✓	✓	✓	✓	✓	✓	✓

Note: Private Information (PI); Critical Files (CF); Instance Integrity (II); Hypervisor Protect (HP); VM Initit Launcher (VIL); VM Trusted Verify (VTV); and Host Trusted Verify (HTV).

the VM lifecycle and provides targeted protection. Compared with existing protection schemes, therefore, the proposed framework offers a more comprehensive range of protections.

The framework has the following advantages over existing VM lifecycle solutions.

• Hardware independence. Compared with Intel's SGX and TXT technologies, which require CPU or BIOS support, the framework can be completely implemented based on open source software, which makes it more broadly usable.

• System security protection. The framework builds a trusted computing environment for VMs and provides comprehensive security protection for the entire lifecycle of the trusted VM, ensuring that the trusted VM is in a safe and trusted status throughout the lifecycle.

• High compatibility. The framework can protect the integrity of the core components of the VM. It can be integrated with existing cloud platform security measures (e.g., access control, intrusion detection, virus detection, vulnerability recovery, hidden process detection, fine-grained logic isolation, hypervisor security, and cloud continuous assessment[40]) to achieve the best VM security protection.

• Real-time monitoring. This framework can be used to monitor the real-time trusted status of VMs in batches, which makes it possible to manage the trust status of a large number of VMs in the cloud.

8 Conclusion

This article proposes a VM lifecycle security protection framework based on trusted computing. The proposed framework can protect the integrity and security of VMs through their lifecycle using trusted computing technology. First, the trusted computing environment of the VM is established. Then, throughout the entire lifecycle of the VM, the security protection method is executed to maintain the integrity and security of the VM. Meanwhile, a status monitoring component is designed to confirm the trusted status by collecting and verifying the trusted information of the VM and host OS. Theoretical analysis shows the proposed framework's security advancements and trustworthiness. In order to verify the correctness and feasibility, a series of experiments were conducted, the results of which show that our proposed framework realizes full lifecycle security protection for VMs with scarcely any extra performance overhead. One interesting line of future work is to focus on analyzing, detecting, and describing abnormal behaviors in the execution state of a VM.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61802270 and 61802271), and the Fundamental Research Funds for the Central Universities (Nos. SCU2018D018 and SCU2018D022).

References

[1] Y. Han, J. Chan, T. Alpcan, and C. Leckie, Using virtual machine allocation policies to defend against co-resident attacks in cloud computing, *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 1, pp. 95–108, 2017.

[2] M. S. Dildar, N. Khan, J. B. Abdullah, and A. S. Khan, Effective way to defend the hypervisor attacks in cloud computing, in *Proc. 2017 2nd Int. Conf. Anti-Cyber Crimes*, Abha, Saudi Arabia, 2017, pp. 154–159.

[3] K. S. Tep, B. Martini, R. Hunt, and K. K. R. Choo, A taxonomy of cloud attack consequences and mitigation strategies: The role of access control and privileged access management, in *Proc. 2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, 2015, pp. 1073–1080.

[4] K. Scarfone, M. Souppaya, and P. Hoffman, *Guide to Security for Full Virtualization Technologies*. Gaithersburg, MD, USA: National Institute of Standards & Technology, 2011.

[5] Y. Yu, M. H. Au, G. Ateniese, X. Y. Huang, W. Susilo, Y. S. Dai, and G. Y. Min, Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage, *IEEE Trans. Inf. Foren. Secur.*, vol. 12, no. 4, pp. 767–778, 2017.

[6] S. M. N. Islam and M. M. Rahman, Securing virtual machine images of cloud by encryption through Kerberos, in *Proc. 2017 2nd Int. Conf. Convergence in Technology*, Mumbai, India, 2017, pp. 1074–1079.

[7] M. Masdari, S. S. Nabavi, and V. Ahmadi, An overview of virtual machine placement schemes in cloud computing, *J. Network Comput. Appl.*, vol. 66, pp. 106–127, 2016.

[8] F. Tao, C. Li, T. W. Liao, and Y. J. Laili, BGM-BLA: A new algorithm for dynamic migration of virtual machines in cloud computing, *IEEE Trans. Serv. Comput.*, vol. 9, no. 6, pp. 910–925, 2016.

[9] W. B. Mao, Method and apparatus for securing the full lifecycle of a virtual machine, US Patent 20130061293, March 7, 2013.

[10] N. Barak, A. Jerbi, E. Hadar, and M. Kletskin, System and method for enforcement of security controls on virtual machines throughout life cycle state changes, US Patent 9389898, July 12, 2016.

[11] R. J. Forrester, W. W. Starnes, and F. A. Jr. Tycksen, Method and apparatus for lifecycle integrity verification of virtual machines, US Patent 9450966, September 20, 2016.

[12] R. Schwarzkopf, Virtual machine lifecycle management in grid and cloud computing, http://archiv.ub.uni-marburg.de/diss/z2015/0407/pdf/drs.pdf, 2015.

[13] Top Threats Working Group, The Treacherous 12: Cloud Computing Top Threats in 2016, http://www.storm-clouds.eu/services/2017/04/the-treacherous-12-cloud-computing-top-threats-in-2016, 2016.

[14] M. Henson and S. Taylor, Memory encryption: A survey of existing techniques, *ACM Comput. Surveys*, vol. 46, no. 4, p. 53, 2014.

[15] I. O. Nunes and G. Tsudik, KRB-CCN: Lightweight authentication & access control for private content-centric networks, arXiv preprint arXiv: 1804.03820, 2018.

[16] C. Alcaraz and S. Zeadally, Critical infrastructure protection: Requirements and challenges for the 21st century, *Int. J. Crit. Infrastruct. Prot.*, vol. 8, pp. 53–66, 2015.

[17] D. J. Chen, N. Zhang, R. X. Lu, N. Cheng, K. Zhang, and Z. G. Qin, Channel precoding based message authentication in wireless networks: Challenges and solutions, *IEEE Network*, vol. 33, no. 1, pp. 99–105, 2018.

[18] N. Zhang, N. Cheng, N. Lu, X. Zhang, J. W. Mark, and X. M. Shen, Partner selection and incentive mechanism for physical layer security, *IEEE Trans. Wirel. Commun.*, vol. 14, no. 8, pp. 4265–4276, 2015.

[19] X. P. Liang, S. Shetty, L. C. Zhang, C. Kamhoua, and K. Kwiat, Man In The Cloud (MITC) defender: SGX-based user credential protection for synchronization applications in cloud computing platform, in *Proc. 2017 IEEE 10th Int. Conf. Cloud Computing*, Honolulu, HI, USA, 2017, pp. 302–309.

[20] M. Plauth, F. Teschke, D. Richter, and A. Polze, Hardening Application Security using Intel SGX, in *Proc. 2018 IEEE Int. Conf. Software Quality, Reliability and Security (QRS)*, Lisbon, Portugal, 2018, pp. 375–380.

[21] W. Arthur, D. Challener, and K. Goldman, Platform security technologies that use TPM 2.0, in *Proc. A Practical Guide to TPM 2.*0, Berkeley, CA, USA, 2015, pp. 331–348.

[22] J. X. Li, D. S. Li, Y. M. Ye, and X. C. Lu, Efficient multi-tenant virtual machine allocation in cloud data centers, *Tsinghua Sci. Technol.*, vol. 20, no.1, pp. 81–89, 2015.

[23] X. M. Ye, X. S. Chen, H. Z. Wang, X. M. Zeng, G. L. Shao, X. Y. Yin, and C. Xu, An anomalous behavior detection model in cloud computing, *Tsinghua Sci. Technol.*, vol. 21, no. 3, pp. 322–332, 2016.

[24] X. Jin, X. S. Chen, C. Zhao, and D. D. Zhao, Trusted attestation architecture on an infrastructure-as-a-service, *Tsinghua Sci. Technol.*, vol. 22, no. 5, pp. 469–477, 2017.

[25] X. Wan, X. F. Zhang, L. Chen, and J. X. Zhu, An improved vTPM migration protocol based trusted channel, in *Proc. 2012 Int. Conf. Systems and Informatics*, Yantai, China, 2012, pp. 870–875.

[26] D. G. Sun, J. Zhang, W. Fan, T. T. Wang, C. Liu, and W. Q. Huang, SPLM: Security protection of live virtual machine migration in cloud computing, in *Proc. 4th ACM Int. Workshop on Security in Cloud Computing*, New York, NY, USA, 2016, pp. 2–9.

[27] N. T. Hieu, M. Di Francesco, and A. Y. Jääski, A virtual machine placement algorithm for balanced resource utilization in cloud data centers, in *Proc. 2014 IEEE 7th Int. Conf. Cloud Computing*, Anchorage, AK, USA, 2014, pp. 474–481.

[28] F. L. Pires and B. Baran, A virtual machine placement taxonomy, in *Proc. 2015 15th IEEE/ACM Int. Symp. Cloud and Grid Computing*, Shenzhen, China, 2015, pp. 159–168.

[29] Z. Zhou, Z. G. Hu, T. Song, and J. Y. Yu, A novel virtual machine deployment algorithm with energy efficiency in cloud computing, *J. Cent. South Univ.*, vol. 22, no. 3, pp. 974–983, 2015.

[30] ISO/IEC, ISO/IEC 11889–1: 2015 Information technology —Trusted platform module library—Part 1: Architecture, Geneva, Switzerland, 2015.

[31] Trusted computing group, *Trusted Computing: An Effective Approach to Cybersecurity Defense*. Beaverton, OR, USA: TCG, 2013.

[32] Trusted computing group, *TPM Main Part 1 Design Principles, Specification Version 1.2, Revision 116*. Beaverton, OR, USA, TCG, 2011.

[33] D. Challener, K. Yoder, and R. Catherman, *A Practical Guide to Trusted Computing*. London, UK: Pearson Education, 2007.

[34] R. Perez, R. Sailer, and L. Doorn, vTPM: Virtualizing the trusted platform module, in *Proc. 15th Conf. USENIX Security Symp.*, San Diego, CA, USA, 2006, pp. 305–320.

[35] C. H. Devassy, M. Prasad, and V. Anil, *Mastering KVM Virtualization*. Birmingham, UK: Packt Publishing Ltd., 2016, pp. 155–156.

[36] Y. Shi, B. Zhao, Z. Yu, and H. G. Zhang, A security-improved scheme for virtual TPM based on KVM, *Wuhan Univ. J. Nat. Sci.*, vol. 20, no. 6, pp. 505–511, 2015.

[37] S. Berger, K. Goldman, D. Pendarakis, D. Safford, E. Valdez, and M. Zohar, Scalable attestation: A step toward secure and trusted clouds, in *Proc. 2015 IEEE Int. Conf. Cloud Engineering*, Tempe, AZ, USA, 2015, pp. 185–194.

[38] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*. Springer, 2015, pp. 156–161.

[39] Trusted computing group, *Virtualized Trusted Platform*

Architecture Specification, Specification Version 1.0, Revision 0.26. Beaverton, OR, USA, TCG, 2011.

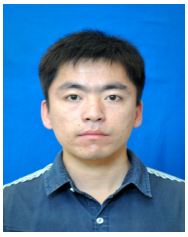
[40] X. Li, X. Jin, Q. X. Wang, M. S. Cao, and X. S. Chen, SCCAF: A secure and compliant continuous assessment framework in cloud-based IoT context, *Wirel. Commun. Mob. Comput.*, vol. 2018, p. 3078272, 2018.

Xin Jin is a PhD candidate at Sichuan University. He received the BEng degree from Liaoning Shihua University in 1999 and the MS degree from Chongqing University in 2006. His research interests include trusted computing, virtuaization secutity, and cloud computing security.

Qixu Wang is currently an assistant researcher in the College of Cybersecurity at Sichuan University. He received the BS degree from Southwest University of Science and Technology in 2009, and the PhD degree from University of Electronic Science and Technology of China in 2017. His current research interests include cloud computing security, wireless network security, data privacy protection, and trusted computing.

Xiang Li is a PhD candidate at Sichuan University. He received the BS degree from Hainan University in 2009 and the MS degree from Chongqing University of Posts and Telecommunications in 2012. His research interests include information security, cloud computing security, and cloud service assessment.

Xingshu Chen received the PhD degree from Sichuan University in 2004. She is now a professor of the College of Computer Science and Cybersecurity Research Institute at Sichuan University. She is the member of China Information Security Standardization Technical Committee. Her research interests include cloud computing, cloud security, distributed file system, big data processing, network protocol analysis, and new media supervision.



Wei Wang is currently a master student in the College of Computer Science of Sichuan University. He received the bachelor degree from Sichuan University in 2016. His current research focuses on trusted computing virtualization technology in cloud computing.