Towards a Trusted Launch Mechanism for Virtual Machines in Cloud Computing

Juan Wang[1,2](✉), Xuhui Xie[1], Qingfei Wang[1], Fei Yan[1,2], Hongxin Hu[3], Sijun Zhou[1], and Tao Wang[1]

[1] School of Computer, Wuhan University, Wuhan 430072, Hubei, China
[2] Key Laboratory of Aerospace Information Security and Trust Computing, Ministry of Education, Wuhan 430072, Hubei, China
jwang@whu.edu.cn
[3] Delaware State University, Dover, DE 19901, USA
hhu@desu.edu

Abstract. Although cloud computing enables us to dynamically provide servers with the ability to address a wide range of needs, this paradigm also brings forth many new security challenges. The security of virtual machines (VM) is one of such critical challenges for cloud computing. However, existing techniques for VM security, such as Terra, tboot and TXT, mainly focus on the security of VM running environment. There is a lack of protection mechanism for VMs themselves in clouds. In this paper, we propose a trusted launch solution for virtual machines (TLVM), including four systematic mechanisms, image encryption, measurement, attestation and security-enhanced authentication, for protecting VMs in clouds. We also discuss a proof-of-concept implementation of our approach. Our experimental results demonstrate the feasibility of our solution to protect the whole launch process of a VM.

Keywords: Cloud security · VM · Measurement · Attestation

1 Introduction

The emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional information technology due to the reason that it provides an extensible and powerful environment for growing amounts of services and data. However, the security of current cloud infrastructures is a key challenge, probably hindering the development of cloud computing.

For infrastructure as a service (IaaS) in cloud computing, virtual machines are leased to users. Some sensitive user data is stored in the virtual machines. Once the data is leaked outside of the virtual machines, it will damage the interests of users. Thus, how to protect the security of virtual machines is crucial in IaaS. However, the existing techniques for VM security like Terra [1], TXT [2] and tboot [3] mainly focus on the security of VM running environment, such as the security of host and virtual machine monitor (VMM) [4,5]. There is a lack of protection mechanism for VMs themselves in clouds.

To address such a critical problem, we propose a trusted launch solution for virtual machines (TLVM). In TLVM, four systematic mechanisms including image encryption, measurement, attestation and trusted-enhanced authentication are used to protect virtual machines. The image encryption mechanism can prevent illegal users to start a VM. The measurement and attestation mechanisms can protect the integrity of a VM. The trusted-enhanced authentication mechanism can achieve two-way authentication between a user and a VM. Consequently, the overall solution can protect the whole launch process of a VM in cloud computing.

The remainder of this paper is organized as follows. In Sect. 2, we describe our goals and the framework of TLVM. In Sect. 3, we present the detailed design of our systematic mechanism for protecting VM security. Section 4 presents the implementation of TLVM. Section 5 discusses the experimental evaluation of our solution. Finally, Sect. 6 concludes this paper.

2 Trusted Launch of Virtual Machines

2.1 Trust and Attack Model

In our trust and attack model, an administrator is able to copy VM images to outsides of a trusted domain. The trusted domain comprises trusted nodes. The trusted nodes including hosts and VMM can be achieved by tboot, TXT and dynamic measurement technology, such as SICE [8] and TEE [5]. Furthermore, the attackers including the administrators of IaaS can tamper with VM images. In addition, since a user can not trust the identity of the VM, the user suffers from the VM phishing attack.

A VM instance is considered trusted in the current attack model if and only if it fulfills the following criteria:

(1) The VM image used for the instance is not tampered with.
(2) The VM instance is launched on a trusted domain.
(3) The identity of the VM is trusted.

In above criteria, the second one can be insured by tboot, TXT and dynamic measurement [9,10] technology, such as SICE and TEE. These methods are not our major focus in the paper. Instead, we investigate the trust issue for VM launch.

2.2 Overview

Trusted Computing [6] is a technology developed and promoted by the Trusted Computing Group. With Trusted Computing, a computer will consistently behave in expected ways, and those behaviors will be enforced by computer hardware and software. Enforcing those behaviors is achieved by building a trusted chain based on trusted base. Five main key technologies: endorsement key, secure input and output, memory protected execution, sealed storage, and remote attestation, can be used to ensure the trust of a protected platform.

Based on the trusted computing technology, TLVM provides a security protection mechanism for virtual machines in a cloud computing platform which can guarantee the confidentiality, integrity and authentication of a user's VM. In TLVM, we add the modules of image encryption, measurement and attestation in VMM. Furthermore, the authentication module based on Usbkey and trusted platform module (TPM) is added in VMs. These mechanisms provide a systematic solution of secure launching a VM.

2.3 Trusted Launch Process of a VM

The framework of TLVM and secure launch process of a VM are depicted as Fig. 1. The system is composed of a cloud management center including a user management module, a VM management module, a key management and attestation server, a host where virtual machines are running, an Usbkey administrator and users. The launch process of a VM is shown as following.

(1) A user requests to register in the cloud management center.

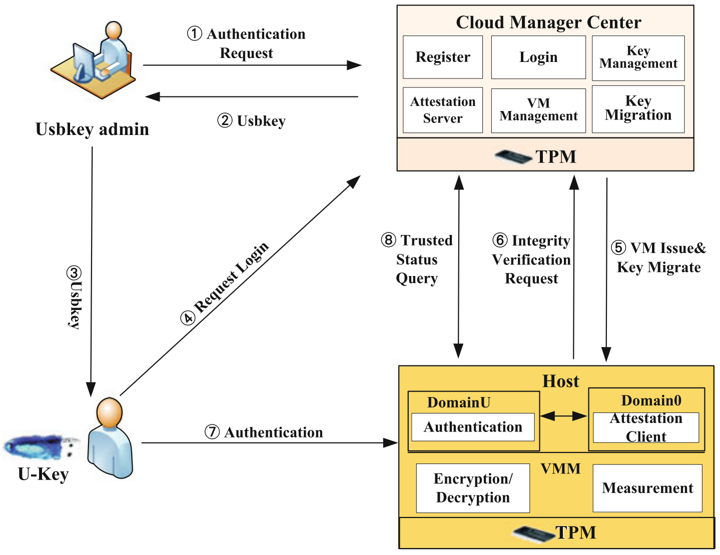

Fig. 1. Trusted launch process of a VM

(2) An Usbkey administrator requests a certificate, a private key and a symmetric key to the cloud management center server. The cloud management center generates a certificate, a private key and a symmetric key and then writes them to an Usbkey. Meanwhile, the symmetric key which is encrypted with the storage

root key (SRK) of the TPM on the cloud management server will be saved to the cloud key management center.

(3) The Usbkey administrator issues the Usbkey to the user.

(4) The user logins to the cloud management center and applies for a virtual machine. The cloud manager center creates a virtual machine for the user.

(5) The encrypted virtual machine image is launched on a host to run. Meanwhile, the symmetric key will be migrated to the host by TPM key migration command and then be protected by the local TPM.The VMM image encryption and decryption module will decrypt the image with the migrated symmetric key. Then the virtual machine image will be measured by measurement module.

(6) The measurement result including measurement value and report signed by TPM in the host will be sent to remote attestation server to verify the integrity of the image.

(7) The user will login to the VM using the Usbkey. The VM will communicate to the host and get the migrated key. The VM and the user will mutually authenticate by the symmetric key in the Usbkey and the migrated key.

(8) When the user logins the VM. The trust query module in VM will communicate with the attestation server and get the VM's measured result. Then the trusted status will be shown on the VM's desktop.

2.4 Detailed Design

In TLVM, the encryption/decryption mechanism of image, measurement, attestation and security-enhanced mechanism based on TPM are added. The detailed design of them is described as following.

2.4.1 Encryption Mechanism of VM Image

In clouds, a virtual machine's images are possible to be started by unauthorized users. For example, administrators possibly copy a virtual machine image to outside of trusted domains and then start the virtual machine. In order to protect the user's VM from unauthorized starting, the strongest method is full virtual machine disk image encryption [11], which makes it difficult to recover the image for unauthorized users. However, it is obviously a time-consuming process. To achieve a tradeoff between security and performance, we only encrypt the main disk information of a virtual machine image. For a common disk file, we could encrypt the master boot record (MBR), Boot, and some logic partitions. But in a cloud environment, some virtual machines may have the same MBR or boot. It is easy to copy a VM to attack other VMs. Thus, we have designed an image file encryption which is based on the file system and user's configuration. We first get the user's symmetric key which is protected by TPM, and get the partition and file system type information from the MBR, then read the user's encryption configuration. Finally, we encrypt the file contained in the configuration, and then encrypt key information of a file system, such as index structure.

The encryption process of a VM image is summarized as follows:

(1) Decrypt the symmetric key which is migrated to the host where VM is running when the VM launches and is protested by the TPM's SRK in the host.

(2) Get the partition and operating system's file system. For every image, partition table is in the end of MBR. We can get the partition information such as initial location, size, file system etc.

(3) Load the user's crypto configuration. Every VM has a crypto configuration which has been protected by the VM's symmetric key. Every operating system has a default configuration.

(4) Check whether the image file has been encrypted. In order to avoid repeating encryption, we have set a flag to identify the image file's crypto status. If the status shows that the image file has been encrypted, the crypto process will be broken.

(5) Encrypt the file included in a configure file. In order to balance the encryption performance and security, we have provided elastic encryption. The configuration includes some kernel file which may be common for the same operating system. For windows 7, it must include the boot, registry, bootmbr, ntldr, boot.ini, winlog.exe and some important files. These files are invisible for users, and users can also add other important files to encrypt.

(6) Encrypt import partition for certain file systems. Different file systems have different organizations of the files. For a Linux system, we search every partition, and get the basic information about the partition from super block and traverse all block groups. Then, we encrypt group descriptors table,block bitmap, inode bitmap, inode table of every block group. Because the image files are in general large, we only rewrite the encrypt section to the image file.

2.4.2 Measurement Mechanism of VM

The measurement mechanism of VM provides integrity measurement of a VM before it starts. The integrity of a system is a semantic concept that indicates whether the system has been modified in an unauthorized manner. To measure an entire system is very expensive in practice. One efficient way is to associate the integrity semantics with some important files. In our mechanism, we measure the most important files of different systems and some files defined by users. For Linux systems, we have measured the boot, grub, kernel, kernel modules, binaries shared libraries and dynamic libraries. For Linux users, we should measure the data and applications. If the VM is a web server, we should measure httpd, mod_access.so, and libjvm.so.

The measurement process of a VM image is described as follows:

(1) Get the configuration file of measurement by VM's UUID. If the configuration file does not exist, a default one will be used.

(2) Get the partition and operating system's information.

(3) Mount every partition and measure the files defined in the configuration file. Use sha1 algorithm to compute the hash value of each file. All hash values will be saved in a measurement log file.

(4) Iterate the hash values in the measurement log as the final measurement values.

(5) Call the attestation client module to send and verify the measurement values.

2.4.3 Attestation Mechanism

Remote attestation [7] does the integrity verification of a system. It can prove whether system data is tampered with. It also provides a credible platform status report to a verifier. For remote attestation, TPM is the trusted root of the report. It helps ensure the report deriving from the current integrity measure values.

There are some differences between the remote attestation of VM and the general remote attestation protocol. The general process of remote attestation is shown in Fig. 2.

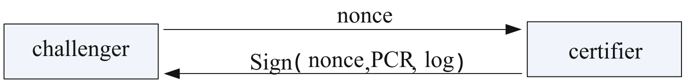

Fig. 2. Remote attestation protocol

A platform (Challenger) sends a challenging message and a nonce to another platform (Certifier), asking for one or more PCR values in order to verify the status of the platform. Certifier uses attestation identity key (AIK) to sign PCR values specified by the challenger, attached to the corresponding measure log entries and AIK certificate, sending to the challenger. Then challenger verifies the value. The validation process includes three steps, (1) re-compute the hash value according to the measure log; (2) verify the AIK certificate; and (3) match the signature value with the expected value.

For VM attestation, the measurement values cannot store in TPM's PCR, because the number of current TPM PCR is only 24, but there are a lot of VMs on a host and the number of VMs is not fixed. Moreover, the measurement values need to be sent to attestation server when a VM starts. The process of remote attestation in our mechanism is shown in Fig. 3. After measuring of VM, the measurement module invokes the attestation client, transfer the measurement value and log to the attestation client. Attestation client receives them, and then triggers the attestation server to send a nonce to the attestation client. After the attestation client gets the nonce, it uses SHA1 algorithm to calculate a hash value of the measurement value, then loads a private signing key from TPM to sign the hash value, forming an integrity report. The report includes nonce, measurement value, hash value, signature value and some other information about the VM. Finally, the attestation client sends the integrity report and measurement log to the attestation server. The attestation server checks nonce, verifies hash and signature, and then judges the platform's credibility by comparing the signature and the expected value.

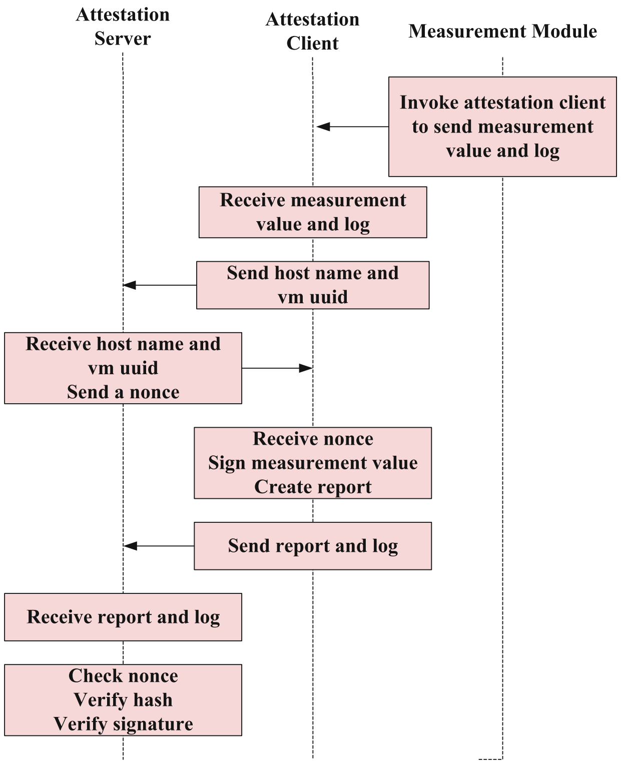
Attestation Server
Attestation Client
Measurement Module
Invoke attestation client to send measurement value and log
Receive measurement value and log
Send host name and vm uuid
Receive host name and vm uuid Send a nonce
Receive nonce Sign measurement value Create report
Send report and log
Receive report and log
Check nonce Verify hash Verify signature

Fig. 3. Process of remote attestation

2.4.4 Trusted Authentication Mechanism of VMs

When the integrity of a VM is achieved by measurement and remote attestation, the VM will be started. However, currently VMs can authenticate users in clouds, but users can not authenticate VMs. Hence, a user easily suffers from a VM phishing attack. To address the issue, we propose a mutual authentication mechanism between VMs and a user based on migrated symmetric key from the TPM of cloud management center and Usbkey. When a user sends request to a VM to login, a VM agent will generate a random N1 and encrypt with the symmetric key K2 migrated from the TPM of cloud management center. The VM agent sends the encrypted N1 to the user. The user decrypts the N1 with the symmetric key K1 issued by the cloud administer. The Usbkey of user side also generates a random N2 and encrypts the sequence (N1, N2) with K1 sent to the VM agent. The VM agent receives the sequence and gets the random N1. If the random N1 is same with the original random N1, the VM can ensure the user is legal. Furthermore, the VM agent uses K2 to encrypt random N2 and sends this random to the user. The user compares the N2 with the initial random N2. If they are same, the user can believe that the VM is legal. If one of them fails, the manual authentication will terminate.

When the mutual authentication succeeds, the user logins to the VM. The VM pops up a web page which accesses the attestation server and gets the integrity verification result of VM image, displaying the trusted status of the VM. According to this measurement verification result, a user can judge that if the VM has been tampered with.

3 Implementation

We have implemented our proposed mechanisms based on Xen and Eucalyptus. The detailed implementation environment of our system is shown in Table 1.

Table 1. Implementation environment of the system

Name	Configuration
Cloud platform	Eucalyptus 3.1
VMM	XEN 4.1.2
Host	HUAWEI RH2288
Domain0 OS	SUSE Linux Enterprise 11SP2
Virtual-Machine OS	Windows 7, ubuntu 10.10
TPM	STM 1.2.7.0
TSS	Trousers 0.3.10
Database	PostgreSQL9.0

For implementing the crypto of a virtual machineimage file, the VM's image should be decrypted before the VM starts and encrypted when the VM shuts down. We also consider some exceptional situation, for example the host where the VM is running suddenly powers off when a VM image is being encrypted. In that situation, we should be able to recover the VM image file when the host powers on. We have modified the source code of Xen to support our approach. The number of modified code is about 1000 lines. We adopted 256-bits Advanced Encryption Standard (AES) as our crypto algorithm.

The principle of encrypting a VM image file system is as following. For Linux system, the file system is ext2, ext3 or ext4 and the basic storage cell of ext is block. One partition has a server block group. Every group block has a number of blocks. The group descriptors table, block bitmap, Inode bitmap and Inode table are stored in front of each block group's description information. The block group 0 has a super block. The super block includes a number of information about this partition such as block size, block number, block group number, and file system flag, node number. The super block has several backups in some block groups. We encrypted the above image file descriptor. For Windows systems, the file system is NTFS. In NTFS, the most important structure is Master File Table (MFT). It includes some system files and file area. Every file has a record in MFT. For every partition it has a backup for MFT. We can encrypt the files to prevent an attacker from opening the VM image file.

For measuring the VM image, such as Linux and Windows 7 virtual machine images, we modified the source code of Xen mainly in XendXm and Python. The amount of modification is about 2000 lines of code. Xen supports two types of virtualization technologies: full-virtualization and para-virtualization. Different types have different implementation mechanisms and image files. We measure the VM image based on the full-virtualization. We should first obtain the OS type, and load the measurement nodes based on the OS type. All measurement nodes are stored serially in a XML configuration file. The XML file can be modified by the manager, if the manager wants to measure more files. Then, we should read the partition information from MBR and measurement each partition. In addition, we firstly mount the boot files and measure them. Furthermore, other partitions are mounted and measured. We store all measurement log and hash values in a XML file. The hash value is iterated as the final hash value.Finally, the measurement module will call attestation module to sign and send the final value and measurement the log file to attest the measurement values.

The attestation module will additionally send the measurement values to the attestation server to verify its integrity. The remote attestation implementation includes two main modules: client signature module and server verification module. First, the client computes hash of the measurement value outside of TPM. Then, the client invokes an interface to implement the signature in TPM. We sign the hash value by using "*Tspi_Hash_Sign*". The sign key is loaded by the VM UUID. Then, the signature value is returned to form an integrity report.

The attestation server receives the integrity report "report.xml" from the attestation client through the web service. Then the attestation server desterilizes report.xml into a java content tree. We use JAXB provided by java to implement unmarshalling process. Marshaller class marshals the given object to a given javax.xml.transform.result. Result is a tagging interface that basically represents an XML output abstraction. The Unmarshaller class reads from the given javax.xml.transform.source, and returns the object read. As with Result, source is a tagging interface that has three concrete implementations. After parsing the report, the server checks the nonce, verifies the hash, uses public signing key to verify the signature. The public signing key is stored in cloud management center. When the server wants to verify the signature, it will ask the cloud management center for the public signing key. The attestation module contains about 4700 lines of code.

Finally, the user and the VM will mutually authenticate when the VM is started. For implementing the mutual authentication based on Usbkey and TPM, we customized a new Credential Provider (CP) for VM images. In the new CP, we added functions, like data encryption, data decryption, key transmission, etc. For the user side, we also added authentication client program to implement data encryption and decryption. Usbkey provider provides Usbkey functions interfaces. In order to ensure the security of the new function, we encapsulated our program to the system library file. We used Openssl to protect their data transmission to guarantee the security of the communication between VM agents and users. The total code of mutual authentication based on TPM and Usbkey including the Usbkey driver and API is about 6400 lines.

4 Evaluation

We have measured the runtime performance of our TLVM solution at the implementation environment shown in Table 1. We have launched the VM of Windows 7 img with 10 G 100 times. The total average time of TLVM is 20.076 s. The average time of each stage is show in the Table 2.

Table 2. Performance of TLVM for a VM

	Decryption	Measurement	Attestation	Authentication
T(s)	4.714	1.382	6.277	7.663

The total time of decrypting VM img is about 4.717 s, including decrypting key with TPM SRK taking about 3.852 s, saving temp file taking 0.223 s, decrypting the img disk file system taking 0.267 s, decrypting file specified in configuration file taking 0.368 s.

The total time of measurement is 1.382 s, including mounting the image file taking 0.186 s, measuring the kernel taking 0.241 s, measuring dynamic link libraries taking about 0.432 s, measuring boot files taking 0.164 s and measuring drivers taking 0.459 s.

The attestation time is about 6.277 s. Generating measurement report based measurement files takes 0.569 s. Signing the measurement hash value will cost 3.872 s. Verifying the signature and comparing the value with baseline values takes 0.684 s. The remaining is the time of communication with the attestation server.

The authentication time with a Usbkey is about 7.663 s including the gentgeneration time of nonces, communication time and encryption/decryption time.

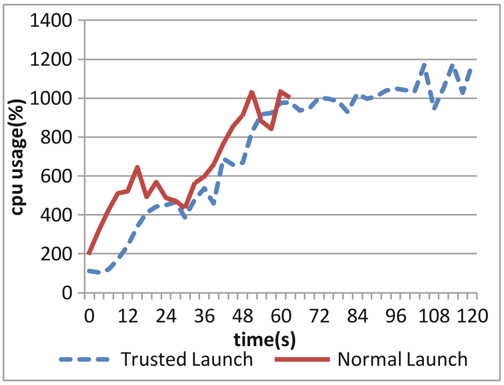

Fig. 4. The performance of multiple VM launched

Because the input PIN speed of different people vary greatly, we have ignored the time of inputing PIN.

In order to show the trusted launch performance impact on the host, we have launched 22 virtual machines with windows 7 image simultaneously. The tested host is HUAWEI RH2288 with 24 core CPUs. Generally, a VM runs in a CPU core. As shown in Fig. 4, normally starting 22 virtual machines takes 63 s, and the CPU usage increase quickly. Trusted launch time is about 120 s. The CPU usage increases slowly. Taken as a whole, the performance overhead is acceptable.

5 Related Work

Santos et al. [12] presented the design of a trusted cloud compute platform, which ensures VMs are running on a trusted cloud node. In the paper, a trusted coordinator and a trusted virtual machine monitor are leveraged to ensure VMs are running on a trusted cluster. The limitation of such a solution is that a trusted coordinator (TC) which is inside an external trusted entity beyond the cloud platform like VeriSign is needed, which makes it hard to be adopted in practice. In addition, they only gave a framework and protocols, but did not implement them. Mudassar Aslam et al. [14] proposed a secure VM launch protocol using Trusted Computing. However, they still focused on the integrity of VM running environment, not the security of the VMs themselves. The difference between our approach and their work is that their work mainly focus on building a trusted execution environment for VMs, such as the trust of host and VMM. But our approach considers the trust of VMs themselves.

Schiffman et al. [13] proposed a centralized verification service called cloud verifier (CV). A user can request to CV to verify the trust of a VM and a host. Nicolae Paladi et al. [15] provided a protocol to ensure the launch of a VM instance. But their method lacks the measurement and remote attestation mechanism for VMs. Moreover , they can not provide a proof-of-concept implementation. In contrast, our TLVM mechanism provides a comprehensive protection approach for VMs. Furthermore, the implementation and evaluation of our approach are given.

6 Conclusions

In this paper, we proposed a trusted launch solution for virtual machine which includes four systematic mechanisms, image encryption, measurement, attestation and security-enhanced authentication, are used to protect a virtual machine in clouds. Image encryption can prevent unauthorized users to initialize a VM. Measurement and attestation mechanism can protect the integrity of a VM. The security-enhanced authentication can achieve two-way authentication between a user and a VM. The mechanisms provide a systematic solution of secure launching a VM. We also discuss a proof-of-concept implementation of our approach. Our experimental results demonstrate the feasibility of our solution to protect the whole launch process of a VM.

Acknowledgment. This work is sponsored by National Natural Science Foundation of China (61173138 and 61103628) and the Huawei Technologies Co., Ltd. collaborative research project.

References

1. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. ACM SIGOPS Operating Syst. Rev. **37**(5), 193–206 (2003)
2. Intel Corp. Intel Trusted Execution Technology. http://www.intel.com/technology/security/
3. Intel Corp. Trusted Boot (tboot). http://sourceforge.net/projects/tboot (2007)
4. Azab, A.M., Ning, P., Wang, Z., Jiang, X., Zhang, X., Skalsky, N.C.: HyperSentry: enabling stealthy in-context measurement of hypervisor integrity. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 38–49. ACM (2010)
5. Dai, W., Jin, H., Zou, D., Xu, S., Zheng, W., Shi, L.: TEE: a virtual DRTM based execution environment for secure cloud-end computing. In: Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010), New York (2010)
6. Challener, D., Yoder, K., Catherman, R.: A Practical Guide to Trusted Computing. Pearson Education, Indianapolis (2008)
7. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of 11th ACM Conference on Computer and Communications Security, ACM Press (2004)
8. Azab, A.M., Ning, P., Zhang, X.: DSICE: a hardware-level strongly isolated computing environment for x86 multi-core platforms. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS ’11), pp. 375–388. ACM, New York (2011)
9. Suh, G.E., Clarke, D., Gassend, B., et al.: Hardware mechanisms for memory integrity checking[R]. MIT LCS TR-872 (2003)
10. Maheshwari, U., Vingralek, R., Shapiro, W.: How to build a trusted database system on untrusted storage. In: Proceedings of the 4th USENIX Symposium on Operating System Design and Implementation (2000)
11. Tomonori, F., Masanori, O.: Protecting the integrity of an entire file system. In: First IEEE International Workshop on Information Assurance (2003)
12. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards trusted cloud computing. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, HotCloud2009. USENIX Association, Berkeley (2009)
13. Schiffman, J., Moyer, T., Vijayakumar, H., Jaeger, T., McDaniel, P.: Seeding clouds with trust anchors. In: Proceedings of the 2010 ACM Workshop on CloudComputing Security, CCSW 2010, pp. 43–46. ACM, New York (2010)
14. Aslam, M., Gehrmann, C., Rasmusson, L., Bjorkman, M.: Securely launching virtual machines on trustworthy platforms in a public cloud - an enterprise’s perspective. In: Leymann, F., Ivanov, I., van Sinderen, M., Shan, T. (eds.) CLOSER, pp. 511–521. SciTePress, Copenhagen (2012)
15. Paladi, N., Gehrmann, C., Aslam, M., Morenius, F.: Trusted launch of virtual machine instances in public iaas environments. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 309–323. Springer, Heidelberg (2013)