A Security-Enhanced vTPM 2.0 for Cloud Computing

Juan Wang[1,2], Feng Xiao[1], Jianwei Huang[1], Daochen Zha[1], Chengyang Fan[1,2(✉)], Wei Hu[1,2], and Huanguo Zhang[1,2]

[1] School of Computer, Wuhan University, Wuhan 430072, China
{jwang, f3i, jw.huang, daochenzha, cyfan, liss}@whu.edu.cn, 564545297@qq.com

[2] Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education, Wuhan 430072, China

Abstract. Virtual Trusted Platform Module is required in cloud due to the scalability and migration of virtual machine. Through allocating a vTPM (Virtual Trusted Platform Module) to a VM (Virtual Machine), users of VM can use the vTPM's crypto and measurement function, like using the physical TPM. However, current vTPM still faces some key challenges, such as lacking runtime protection for the vTPM keys and code, lacking the mechanism of vTPM keys management, and lacking the support for the new TPM 2.0 specification. To address these limitations, we design vTPM 2.0 system and then propose a runtime protection approach for vTPM 2.0 based on SGX. Furthermore, we present vTPM key distribution and protection mechanism. We have implemented vTPM 2.0 system and the security-enhanced protection mechanism. As far as we know, the vTPM 2.0 system based on KVM and its security-enhanced mechanism are designed and implemented for the first time.

Keywords: vTPM · Trusted computing · Intel SGX · KMC · Cloud security

1 Introduction

Security is currently the key factor of restricting the development of cloud computing. In the cloud computing environment, how to protect the integrity of cloud infrastructure is a basic requirement of cloud security. Trusted computing has been considered as a feasible way to protect the integrity of cloud infrastructure.

However, in cloud computing environment, a lot of virtual machines may be running in a physical machine. It is difficult to use hardware TPM (Trusted Platform Module) to build trusted virtual execution environment. Therefore, vTPM has been put forward and used in cloud [13, 14].

IBM designed and implemented vTPM system on a virtualized hardware platform [11]. They virtualized the Trusted Platform Module by extending the standard TPM command set to support vTPM lifecycle management and enable trust establishment in the virtualized environment. Hence, each virtual machine instance gets its own unique and virtual TPM. However, vTPM still faces some key challenges in cloud.

S. Qing et al. (Eds.): ICICS 2017, LNCS 10631, pp. 557–569, 2018.
https://doi.org/10.1007/978-3-319-89500-0_48

Firstly, current vTPM lacks the mechanism to ensure the security vTPM itself. vTPM is an emulated software TPM. Due to lacking the physical hardware protection, it is subject to greater security threats compared with entity TPM. Furthermore, physical TPM may not provide runtime protection for vTPM because its NVRAM is usually very small and cannot support multiple virtual machines. In addition, entity TPM incurs a large overhead in performance when multiple vTPMs run at the same time.

Secondly, current vTPM cannot support TPM 2.0 specification. The architecture of TPM 2.0 is different with TPM 1.2, for example the keys of TPM 2.0 are generated by three persistent hierarchies and also it can support all kinds of cryptographic algorithms through incorporating an algorithm identifier. Hence we need to design vTPM based on TPM 2.0 architecture so as to improve vTPM 1.2.

Aiming at these problems, we propose security-enhanced vTPM 2.0 system which can support TCG TPM 2.0 specification and the keys and private data can be protected using SGX keys and enclave. To the best of our knowledge, it is the first time that vTPM 2.0 based on KVM (Kernel-based Virtual Machine) has been proposed and implemented. In our system, we also propose a vTPM 2.0 key distribution and protection mechanism based on KMC (Key Management Center) [20, 21]. Our approach can achieve the key hierarchy, which is same as the physical TPM. In addition, it can avoid the problem that new physical platform always regenerates the certificate for vTPM and rebuilds the trust binding during the migration for each time. Moreover, the basic seeds of vTPM can be backed up by KMC. When the vTPM is damaged, the keys and data of vTPM can be easily recovered. We also implement our system on KVM and Skylake CPU and evaluate vTPM 2.0 performance. The result shows that the SGX-enhanced vTPM brings about 20% additional overhead compared with the vTPM 2.0 which lacks security protection.

The remainder of this paper is organized as follows. Section 2 provides related work. Section 3 introduces the background of TPM 2.0 and SGX. Section 4 describes the design of security-enhanced vTPM 2.0. Section 5 proposes the vTPM key distribution and protection mechanism. The security-enhanced vTPM 2.0 implementation is described in Sect. 6. Section 7 presents the evaluation of vTPM 2.0. Section 8 provides the conclusion.

2 Related Work

Trusted computing technology is usually used to building trusted computing base of a computer system and protects the system integrity and confidentiality. Microsoft has leveraged trusted computing technology to implement trusted boot and its BitLocker has been used to disk encryption. Google chrome book [24] also has integrated TPM [23] chip to implement trusted boot and device anti-theft. Chen et al. [7] proposed cTPM which is practical, versatile, and easily applicable to cross-device trusted mobile applications. Santos et al. [6] provided a new trusted computing abstraction for designing trusted cloud services. Bates et al. [8] defined a provenance trusted computing base and created a trusted provenance-aware execution environment, collecting complete whole-system provenance. With the development of trusted computing, the limitations of

TCG TPM 1.2 architecture [25] are found, for example, cipher algorithms are not flexible. Hence TCG has proposed TPM 2.0 specification [1–5] and has published as ISO standard in 2015 [22]. Scarlata et al. [18] present a support for a variety of TPM model and different security properties of system framework based on Xen virtual machine hypervisor.

vTPM is a virtualized TPM which is generally used in virtualized environment, such as cloud computing platform, to build trust computing base. The most important work about virtualized TPM is that Berger et al. [9] from IBM designed and implemented a vTPM system on a virtualized hardware platform. They virtualized the Trusted Platform Module by extending the standard TPM command set to support vTPM lifecycle management and enable trust establishment in the virtualized environment. England and Loeser [15] extended hypervisor to add the vPCR (virtual PCR) and TPM context manager resource virtualization which allows guests operating systems to share hardware TPM. But the number of virtual machines on a physical machine is uncertain, their approach must meet performance bottleneck due to the limited memory space of TPM. In addition, Yang et al. [16] designed an Ng-vTPM framework. In Ng-vTPM framework, the EK and SRK are produced by physical TPM. The approach can protect the keys' security to some extent, but once the physical TPM is damaged, the keys of vTPM will not be used and recovered forever. Yan et al. [17] propose a secure enhancement named vTSE. The scheme utilizes the physical memory isolation feature of SGX to protect the code and data of vTPM instances, but they do not consider the vTPM keys recovery and cannot support TPM 2.0.

Current work is just support TPM 1.2. In our work, we design and implement the vTPM 2.0 on KVM. In addition, we provide runtime protection for code and private data of vTPM 2.0 using SGX. Furthermore, the vEK (virtual EK) and vSRK (virtual SRK) of vTPM 2.0 will be generated by a trusted party, KMC, and they are bound with VM UUID. Therefore, the keys are easy to be recovered once damaged.

3 Background

3.1 TPM 2.0

Trusted computing is a technology mainly used to protect the integrity and confidentiality of a system. It relies on TPM, which is a cryptographic coprocessor integrated in most commercial PCs and servers. TCG released TPM 1.2 specification in 2003. Currently, TCG has released TPM 2.0 specification [1–4] which has overcome some of the drawbacks of TPM 1.2. Compared with TPM 1.2, TPM 2.0 has many advantages.

TPM 1.2 can only support SHA1 and RSA, but TPM 2.0 can support all kinds of cryptographic algorithms, such as ECC, SHA256 and AES. Additionally, TPM 1.2 has only one key hierarchy: the storage hierarchy. TPM 2.0 has three persistent hierarchies: platform, storage, and endorsement, each with at least one root, such as EPS (Endorsement Primary Seed), SPS (Storage Primary Seed), and PPS (Platform Primary Seed). Furthermore, TPM 2.0 incorporates an algorithm identifier that would permit design of a TPM using any algorithm without changing the specification. Hence all kinds of cipher algorithms, such as Chinese commercial cipher algorithms SM2, SM3 and SM4 can be integrated easily. In addition, TPM 2.0 unifies the way all entities in a

TPM that are authorized. Besides the traditional password and HMAC authentication methods, the authentication method based on policy authorization has been added. It allows one or more authorization policies are used, which can enhance key's security. Last but not least, TPM 2.0 enhances the robustness. In the TPM 2.0 specification, some important things can be sealed to a PCR (platform configuration register) value approved by a particular signer instead of to a particular PCR value. Additionally, platform hierarchy allows OEM (Original Equipment Manufacturer) directly to use the function of TPM in BIOS without considering the OS' s support.

3.2 SGX

The Software Guard Extension (SGX) [26, 27] was introduced in 2013 by Intel Corporation. It protects a portion of the application's memory space and places code and data in this container that Intel calls enclave [12, 28]. Once the protected part of the application is loaded into the enclave, SGX will protect them from external process such as OS, drivers, BIOS, hypervisor and System Management Mode (SMM). Moreover, when the process terminates, its enclaves will be destroyed and the runtime data and code that are protected in the enclave will disappear. SGX also provides the seal function to encrypt the data and store it on the permanent media and then we can restore it in enclave when we need to use it again.

In addition to providing security attributes of memory isolation and protection. SGX architecture also supports attestation function. In SGX, attestation [30] is to prove the identity of the platform, and supports two attestations, local attestation between enclaves and remote attestation by a third party.

4 Design of Security-Enhanced vTPM 2.0

In this section, we design security-enhanced vTPM 2.0 architecture. Specifically, each VM can get its unique vTPM 2.0 devices with TPM 2.0 functionality.

As shown in Fig. 1, our security-enhanced vTPM 2.0 architecture includes the following basic components: vTPM 2.0 management, Libtpms 2.0, NVRAM files, tpm2driver, tpm_tis and SeaBIOS. The vTPM 2.0 management module implements vTPM management, such as vTPM creating, command processing interface etc. Libtpms 2.0 is the main module which can provide emulated TPM function in hypervisor. NVRAM files save the seeds, keys, PCRs and other private data. The module tpm2driver provides the interfaces to access TPM 2.0 hardware device. Tpm_tis emulates the hardware interface of TPM interface specification. SeaBIOS is a virtual

BIOS served for guest OS. In our system, the key modules of vTPM 2.0 including Libtpms 2.0, NVRAM, are sealed by SGX keys and isolated in a SGX enclave.

Libtpms 2.0 a TPM emulator, is the key module of vTPM 2.0 which is able to provide all the TPM 2.0 functions and command sets. Due to supporting multiple virtual machines on the same physical platform to access TPM 2.0 resources independently without impacting each other, we design Libtpms 2.0 as a shared software library located in host operating system. Libtpms 2.0 consists of tpm module, platform module, Crypto Engine module, and include module. TPM module implements the

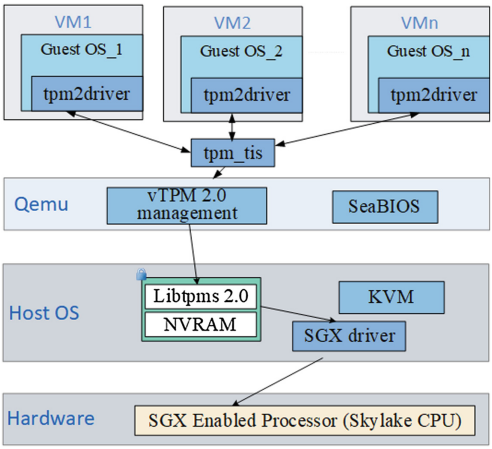

Fig. 1. Security-enhanced vTPM2.0 architecture

reset of NVRAM, the initialization of TPM components and the process of command. Platform module implements the creation of NVRAM file, the set of locality and the management of power state of TPM. The Crypto Engine module packages the realization of crypto algorithms provided by TPM and implements them through calling interfaces provided by OpenSSL.

Because Libtpms 2.0 module undertakes the core function of vTPM 2.0, its code and process need to be protected. We isolated the module into a SGX enclave. When Libtpms 2.0 is loaded, the SGX enclave is created and then Libtpms 2.0 program is measured to validate its integrity. Once the integrity is not tampered with, Libtpms 2.0 code will be executed in the enclave EPC (enclave page cache). Hence the program is protected in runtime and only itself can access the code in the enclave. The untrusted part of vTPM 2.0, such as vTPM 2.0 management module, just can call the function of Libtpms 2.0 through enclave call (ecall) and out call (ocall).

NVRAM is like TPM memory. Due to lacking the isolated physical NVRAM of TPM, it is designed as a separated file which saves the keys, PCR values, seeds and other private data. When a vTPM 2.0 device is created, a NVRAM file will be also created. Because the important data of vTPM is saved in NVRAM file, it is vital for vTPM security. Hence, we leverage SGX sealing to protect NVRAM. To preserve some secret data in an enclave for future use, SGX offers a sealing function. Sealing can encrypt the data inside an enclave and store them on a permanent medium such as a hard disk drive, so the data can be used the next time. When sealing data, there are two options available: sealing to the current enclave using the current version of the enclave measurement (MRENCLAVE) or sealing to the enclave author uses the identity of the enclave author (MRSIGNER). In this work, we use both mechanisms. The private data of NVRAM is sealed by the seal key which is generated in the corresponding enclave. When the NVRAM file is loaded into RAM, it will be unsealed and isolated in an enclave. Therefore the software except for Libtpms 2.0, including OS, drivers, BIOS and hypervisor cannot access the data of NVRAM.

Tpm2driver is generally used to provide the interfaces to access TPM 2.0 hardware device. Tpm_tis emulates the hardware interface of TIS (TPM Interface Specification) in QEMU and implements the interfaces to call Libtpms 2.0. SeaBIOS also plays an important part in the process of creating a VM on KVM platform. Apart from implementing the whole standard calling interfaces as a typical x86 hardware BIOS, SeaBIOS is extended to support TPM by initializing the vTPM 2.0 when creating a VM. This includes allocating a fixed virtual memory address in which the vTPM communicates with the lower operating system and resetting all the registers of vTPM.

When a VM sends a TPM command, tpm2driver of the VM will firstly talk to the tpm_tis frontend emulated by QEMU to deliver the TPM request. Then the tpm_tis frontend in QEMU delivers the request to Libtpms 2.0 driver, the driver will call the Libtpms 2.0 shared library to process the TPM command and return the results. This method does not have any limit of the numbers of VM (as long as the hardware resources permit). All that a user needs to do is to configure an exclusive NVRAM used to save all the persistent state and data for the vTPM 2.0 in each VM. During the VM migration, the corresponding NVRAM is migrated along with the VM and then the VM can continue to use vTPM 2.0 resources on the new platform.

5 The Key Distribution and Protection Mechanism of vTPM 2.0

When vTPM 2.0 is protected using SGX enclave, its keys and PCRs are encrypted by the CPU supported SGX. Once a virtual machine with vTPM 2.0 device is migrated, the vTPM 2.0 device also need to be migrated. However, the SGX keys cannot be migrated. Hence, the trust chain between vTPM and physical CPU will be broken during the vTPM migration. In addition, the current method cannot support key recovery. Once vTPM is damaged, the keys of vTPM will be lost. To solve the problems, we propose a vTPM 2.0 key distribution and protection mechanism based on KMC and Intel SGX.

In our method, the primary seeds of vTPM 2.0 including EPS (Endorsement Primary Seeds), SPS (Storage Primary Seeds) and PPS (Platform Primary Seeds) will be generated by KMC and then distributed to vTPM 2.0 by encrypted and secure channel. The primary seeds will be encrypted and saved in KMC and this process is carried out in the Enclave safe. Meanwhile, the primary seeds in a virtual machine will be encrypted by SGX key on host. Once physical CPU or vTPM is damaged, KMC can recover the primary seeds and then recover vTPM keys. The key distribution and protection process of vTPM 2.0 is described as shown Fig. 2. The encrypted communication channel is established using the SGX remote authentication feature. In order to achieve encrypted communication channel, it needs to introduce a special quoting enclave which is used to generate the credential that reflects enclave and platform status. When the KMC wants to authenticate a VM, the VM first executes the EREPORT instruction to generate the REPORT structure and then use the report key of quoting enclave to generate a MAC, along with the REPORT send to quoting enclave. Then the quoting enclave packs them into a quote structure QUOTE and signs it with EPID. Finally the quoting enclave sends QUOTE and signature to KMC.

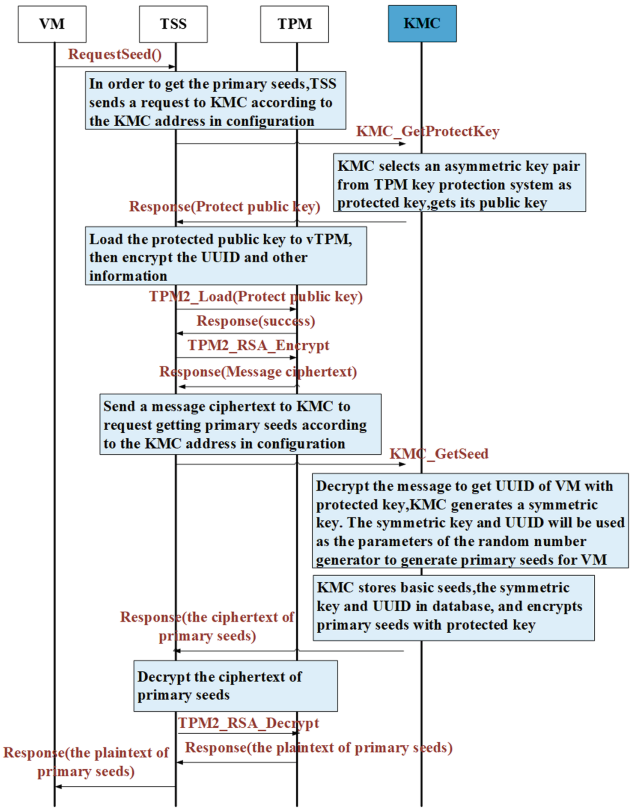

Fig. 2. The key distribution and protection process

1. A virtual machine communicates with TSS (Trusted Software Stack) and calls API (Application Programming Interface) to request getting primary seeds at startup process. Its UUID (Universally Unique Identifier) is as a parameter of the request.
2. In order to get the primary seeds, TSS sends a message to KMC according to the KMC address in configuration.
3. KMC selects an asymmetric key pair (e.g. RSA) from key protection system as a protected key, gets its public key and returns the protected public key to TSS.
4. TSS calls TPM2_Load to load the protected public key to vTPM, then calls TPM2_RSA_Encrypt to encrypt them and sends the cipher text to KMC, requests KMC to send basic seeds.
5. After KMC gets the encrypted request, it will use TPM interface to decrypt the cipher text to get the request with the private key. Then KMC will generate a symmetric key. Furthermore, the symmetric key and UUID will be used as the parameters of the random number generator to generate basic seeds for the virtual machine.

6. KMC stores the basic seeds, the symmetric key and UUID in database, and encrypts basic seeds and other information with protected key, then sends back the cipher text to vTPM TSS.
7. TSS calls TPM2_RSA_Decrypt to decrypt the cipher text, and returns the basic seeds to the virtual machine.

Compared with previous method, our approach can achieve the key hierarchy, which is the same as the physical TPM. In addition, it can avoid the problem that during the migration for each time new physical platform always regenerates the certificate for vTPM and rebuilds the trust bindings. Moreover, the basic seeds of vTPM can be backed up by KMC. When the vTPM is bad, the keys and data of vTPM can be recovered.

6 Security-Enhanced vTPM 2.0 Implementation

We implement our security-enhanced vTPM 2.0 on QEMU/KVM and Skylake CPU. Our Libtpms 2.0 module is mainly based on Windows TPM2 emulator from Microsoft. It only supports Windows operating system, hence we migrate it from Windows to Linux. During the migration, we rewrite all files.

Furthermore, the TPM 2.0 module is added into QEMU virtual machine. Firstly, we extract TPM libraries from TPM 2.0 emulator. According to source code analysis for TPM 2.0 and name it Libtpms 2.0. Secondly, we add TPM 2.0 module in QEMU virtual machine to call functions in library Libtpms 2.0. Then TPM 2.0 interface tpm_libtpms2.c is added in TPM module of QEMU. Furthermore we implement the TPM 2.0 interfaces defined in TPMDriverOps data structure with those functions.

The modules of TPM 2.0 interfaces in QEMU are mainly divided into two parts: the initialization module and the command process module. The initialization module includes device initialization, memory initialization, NVRAM initialization and so on. The command process module is responsible for receiving the TSS commands from VM, executing them and returning results.

In order to support TPM 2.0 device in VM, we update the device driver to TPM 2.0 in guest OS. Apart from modifying tpm.c, tpm.h, the Kconfig and the Makefile, the updating work is mainly in the new added tpm2_tis.c, which is the core file to realize TPM2.0 driver.

Firstly, we rewrite the entry function of loading and unloading driver. Then we begin to write the driver initialization function tpm2_tis_init(). This function will finish the register of TPM 2.0 device, including allocating its virtual memory address space, setting the default timeout, waiting delay, locality and all the internal flags of TIS and doing device self-testing. Besides, we write tpm2_tis_recv() and tpm2_tis_send() functions to send the TPM commands and receive the results. Finally, we write a couple of TPM 2.0 device attributes such as endorseauth, ownerauth, PCRs, phenable, shenable, ehenable. Users can access these attributes directly under /sys/class/misc/tpm0/device.

We also implement the protection mechanism of vTPM 2.0 based on SGX and KMC. When a virtual machine is created, a request seed message will be sent to KMC through encrypted communication channel. KMC then creates a RSA key pair and

sends the public key to QEMU. QEMU furthermore sends VM UUID encrypted by public key to KMC through security channel. KMC creates an AES key by local crypto chip. The AES key and UUID are used as the parameters of the random number generator to generate primary seed for virtual machines. Meanwhile the basic seeds, UUID and AES key will be encrypted by local crypto chip and then stored in KMC database. The encrypted seed will be reply to QEMU. The Libtpms 2.0 module in QEMU, QEMU will create vEK, vSRK, and other root keys for the vTPM.

For a vTPM, QEMU allocates a memory file to save nonvolatile data. The vEK, vSRK, and other root keys are saved to this file named NVRAM. In order to protect the keys security, the NVRAM file is sealed and isolated by SGX keys and enclave. The keys are also backed up to KMC. In addition, the Libtpms 2.0 is compiled into a static library so as to be loaded and run in the SGX enclave. We also add the ecall and ocall in the vTPM management module and the Libtpms 2.0 module in order to implement the communication with them.

7 Evaluation

7.1 Function and Performance

Firstly we conduct function test of the vTPM 2.0. For the test we use a server with an Intel Skylake processor i7 6700 CPU, 8 G memory and 500 G hard disk. The host OS is Ubuntu 16.04 and the guest OS is Ubuntu 14.04. Once the virtual machine has successfully loaded the tpm2_tis.ko module, there will be a device named tpm0 under /sys/class/misc/, indicating that the TPM device is successfully emulated in the VM.

In vTPM 2.0, we have implemented the support of multiple virtual machines. We create five VMs in one host and conduct the testing of authorization policies setting, key derivation, digital signature and verification, encryption and decryption using SM2 algorithm and RSA algorithm respectively. In order to make sure that the vTPM 2.0 in five VMs have different primary seeds and primary keys, we make a comparison of the primary keys between two different VMs during the primary key derivation process. The result is shown in Fig. 3, proving that the vTPM 2.0 state in different VMs is independent and will not influence each other.

We have measured the runtime performance of the SGX-enhanced vTPM 2.0 and the vTPM 2.0 which lacks security protection. We calculate the time of calling TPM interfaces to create RSA and SM2 signature keys, conduct RSA and SM2 signature, and verify RSA and SM2 signature. These tests have been done for twenty times and average time is calculated so as to make the results more precise. Figure 4 shows the comparison results. The result shows SGX-enhanced vTPM brings about 20% additions overhead.

7.2 Migration

We also carried out the single VM and multiple VMs live migration test. [19] Migration channel using SSH RSA public key encryption, we record the start time and end time, and compute the time cost of migration. In addition to the normal time

```
Cmd sent: TPM2_CreatePrimary

80 02 00 00 00 43 00 00 01 31 40 00 00 01 00 00
00 09 40 00 00 09 00 00 00 00 00 00 04 00 00 00
00 00 1a 00 01 00 04 00 03 00 72 00 00 00 06 00
80 00 43 00 10 08 00 00 00 00 00 00 00 00 00 00
00 00 00


Response Received:
Response size: 438

80 02 00 00 01 b6 00 00 00 00 80 00 00 00 00 00
01 9f 01 1a 00 01 00 04 00 03 00 72 00 00 00 06
00 80 00 43 00 10 08 00 00 00 00 00 01 00 cf b0
db c5 c7 2c 8b 3d e7 c1 a8 c4 f4 85 9a eb 05 00
67 54 dc af 78 d7 08 06 58 34 63 0b 26 e3 2d ed
93 d3 26 68 d4 ec b8 f6 f8 44 b4 6f 8a f8 86 1f
64 39 8c df 20 cf b8 5f 02 1c f0 a0 42 05 55 90
d6 4e 4d 07 c6 ab 1a 89 27 e4 25 39 4a dd 00 a0
cf 62 67 b9 23 c1 64 01 bd 5b 6b 11 03 d8 aa 27
2d 2e da 25 08 03 98 f4 aa 1d 46 7c 9a 11 a9 19
4f 34 dd 63 34 f7 a9 4d 15 a4 d7 41 9d 3a af 39
2b c4 1b 23 8a 86 da 4e 2d 25 0e ce b8 c7 f0 b0
53 a7 e6 82 d0 87 4d 12 2c c9 3f 41 77 64 28 40
cf e4 44 0d a7 ac 60 98 24 47 29 51 b7 81 ca 24
a3 1d ea 77 38 00 84 91 be fe 1c 45 ce d6 86 aa
60 66 f5 6f e0 d3 8b c8 d8 7f 79 b6 88 6b 27 2d
30 96 d0 d8 ba 2d 57 42 91 88 21 d4 c7 c3 67 72
80 7e 48 b1 91 63 93 6e f1 d4 5c cc 65 d9 b0 d3
```

```
Cmd sent: TPM2_CreatePrimary

80 02 00 00 00 43 00 00 01 31 40 00 00 01 00 00
00 09 40 00 00 09 00 00 00 00 00 00 04 00 00 00
00 00 1a 00 01 00 04 00 03 00 72 00 00 00 06 00
80 00 43 00 10 08 00 00 00 00 00 00 00 00 00 00
00 00 00


Response Received:
Response size: 438

80 02 00 00 01 b6 00 00 00 00 80 00 00 00 00 00
01 9f 01 1a 00 01 00 04 00 03 00 72 00 00 00 06
00 80 00 43 00 10 08 00 00 00 00 00 01 00 bd 4e
44 fa c1 63 d2 2d 94 01 16 bf 23 18 67 e1 4f 51
8a b6 45 bd 2a 6d 21 11 c7 aa c8 2b e3 2e bf 38
f8 a4 fa 75 ed 73 9f 40 cd 12 7d 1e f3 b5 41 3d
c1 3a 52 93 b0 90 47 d6 bb 3c b6 3b 5d 00 29 00
4f c7 39 ab 2f 9d f1 29 3d ab 45 7b d6 f9 6e d1
11 0c dc e0 fd cb 20 fd cd 33 1f c4 0e 06 d5 30
cd 23 2f 2c cb 73 98 8e 1e 4e 72 19 31 66 08 5d
cc d0 4e 8e 4c bc a9 d7 d2 09 55 fa 9c b2 e7 4b
04 ce 55 f9 bc 29 f1 2c 1a ae c3 4e 71 9c 11 b2
ca 62 08 9e 2f a6 75 80 e3 f2 9b 60 6a 2d c2 f2
cd a7 46 32 36 3c d5 f5 75 d8 0c 57 ba 1f a4 12
56 9a 65 7b c1 cf d6 03 04 ad 08 1f 68 9b fb b3
dc c9 15 49 17 86 98 59 ac 2f d1 07 9e 47 46 59
09 4d 97 32 a5 49 40 78 6f e8 1b cc 46 f3 7d 5a
1f 94 c7 60 b8 f0 72 35 b2 77 3b 23 ad fe 44 2d
```

Fig. 3. TPM2_CreatePrimary result in two different VMs

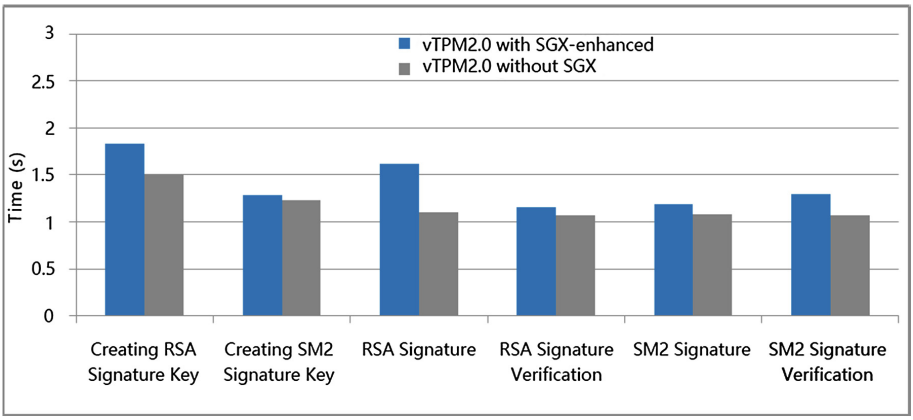

Fig. 4. The performance of vTPM with SGX-enhanced

needed for migration, VM with SGX-enhanced vTPM migration time also includes four parts: (1) unseal NVRAM from enclave; (2) migrate vTPM state; (3) the destination host decrypts NVRAM; (4) use new SGX to seal NVRAM. VM without vTPM does not include the four parts.

For a single VM migration, the VM image is Ubuntu 14.04 64-bit and the hardware resources allocated for the VM are 1 VCPU, 1024 MB RAM and 20 G Disk. For multiple VMs (ten units) concurrent migration, the allocated hardware resources for each VM are 1 VCPU, 1024 RAM, and 6 G Disk.

Our test is divided into four parts altogether: single_VM, single_VM_no_vtpm, multi_VM, multi_VM_no_vtpm. We test 100 times respectively and calculate the average value, the result is shown in Fig. 5.

We can know that the time-consuming of a single VM migration with SGX-enhanced vTPM is more than 5 s as compared to the single VM migration without it. When ten VMs migrate, the value is less than 10 s.

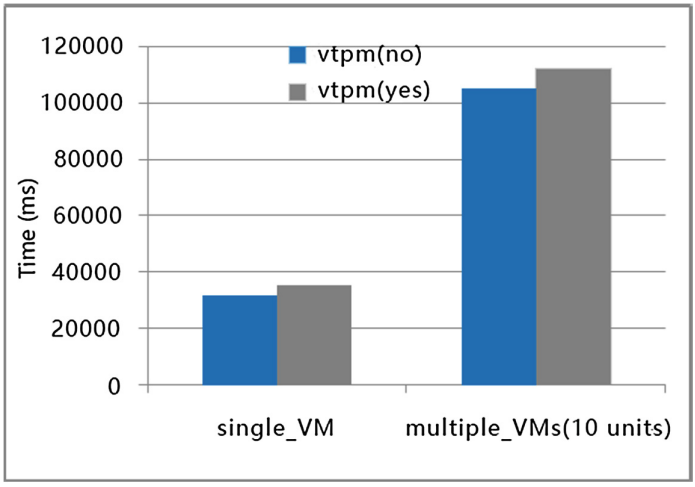

Fig. 5. The average time of VM migration

8 Conclusion

In the environments of cloud computing and NFV (Network Function Virtualization), vTPM is more and more used for protecting the security of virtualized machines and virtualized network function. TCG also presented TPM 2.0 specification to overcome the shortcomings of TPM 1.2. In this paper, we design a security-enhanced vTPM 2.0 system. Our approach cannot only support TPM 2.0 specification and KVM hypervisor, but also the keys and private data of vTPM 2.0 are statically and dynamically protected using Intel SGX. In addition, a vTPM key distribution and protection mechanism base on KMC are proposed, which can more conveniently support vTPM key recovery and vTPM migration. Moreover, we implement the security-enhanced vTPM 2.0 system and evaluate its performance.

Acknowledgment. This work is sponsored by the National Basic Research Program of China (973 Program) granted No. 2014CB340600, National Natural Science Foundation of China granted No. 61402342, 61173138 and 61103628, and the Huawei Technologies Co., Ltd. collaborative research project.

References

1. Trusted Computing Group. TPM Rev 2.0 Part1. Architecture. Family 2.0, Level 00. Revision 16 Jan 2014
2. Trusted Computing Group. TPM Rev 2.0 Part2. Structures. Family 2.0, Level 00. Revision 16 Jan 2014
3. Trusted Computing Group. TPM Rev 2.0 Part3. Commands. Family 2.0, Level 00. Revision 16 Jan 2014
4. Trusted Computing Group. TPM Rev 2.0 Part4. Supporting. Routines. Family 2.0, Level 00. Revision 16 Jan 2014
5. Trusted Computing Group. Trusted Platform Module Specification Family 2.0, Level 00. Revision 00.99 (2014)
6. Santos, N., Rodrigues, R., Gummadi, K.P., Saroiu, S.: Policy-sealed data: a new abstraction for building trusted cloud services. In: Proceedings of 21th USENIX Security Symposium on USENIX Security Symposium (2012)
7. Chen, C., Raj, H., Saroiu, S., Wolman, A.: cTPM: a cloud TPM for cross-device trusted applications. In: Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (2014)
8. Bates, A., Tian, D., Kevin, R.B.: Trustworthy whole-system provenance for the Linux Kernel. In: Proceedings of 24th USENIX Security Symposium on USENIX Security Symposium (2015)
9. Berger, S., Cáceres, R., Goldman, K.A., et al.: vTPM: virtualizing the trusted platform module. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15, p. 21. USENIX Association (2006)
10. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU based attestation and sealing. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, vol. 13 (2013)
11. Sadeghi, A.-R., Stüble, C., Winandy, M.: Property-based TPM virtualization. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 1–16. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85886-7_1
12. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using innovative instructions to create trustworthy software solutions. In: HASP@ ISCA, pp. 11–17 (2013)
13. Garfinkel, T., Pfaff, B., Chow, J., et al.: Terra: a virtual machine-based platform for trusted computing. ACM SIGOPS Operating Syst. Rev. **37**(5), 193–206 (2003)
14. Krautheim, F.J., Phatak, D.S., Sherman, A.T.: Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing. In: Acquisti, A., Smith, S. W., Sadeghi, A.-R. (eds.) Trust 2010. LNCS, vol. 6101, pp. 211–227. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13869-0_14
15. England, P., Loeser, J.: Para-virtualized TPM sharing. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 119–132. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68979-9_9
16. Yang, Y., Yan, F., Mao, J.: Ng-vTPM: a next generation virtualized TPM architecture. J. Wuhan Univ. (Nat. Sci. Ed.) **2**, 103–111 (2015)
17. Yan, F., Yu, Z., Zhang, L., et al.: vTSE: a solution of SGX-based vTPM secure enhancement. Adv. Eng. Sci. **49**(2), 133–139 (2017)
18. Scarlata, V., Rozas, C., Wiseman, M., Grawrock, D., Vishik, C.: TPM virtualization: building a general framework. In: Pohlmann, N., Reimer, H. (eds.) Trusted Computing. Vieweg+Teubner (2008)

19. Danev, B., Masti, R.J., Karame, G.O., et al.: Enabling secure VM-vTPM migration in private clouds. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 187–196. ACM (2011)
20. Zhang, Q., Zhao, S., Qin, Y., et al.: Formal analysis of TPM 2.0 key management APIs. Chin. Sci. Bull. **59**(32), 4210–4224 (2014)
21. NIST, Recommendation for Key Management–Part 1: General (Revision 3), Special Publication 800–57
22. http://www.trustedcomputinggroup.org/media_room/news/392
23. http://www.infineon.com/cms/en/product/security-ic/trustedcomputing/channel.html?channel=db3a30433efacd9a013f10d2a7264daa
24. http://www.chromebookblog.com/tag/tpm-chips-for-chromebook/
25. Arthur, W., Challener, D.: Practical Guide to TPM 2.0 Using the Trusted Platform Module in the New Age of Security. Willey (2015)
26. Mckeen, F., Alexandrovich, I., Berenzon, A., et al.: Innovative instructions and software model for isolated execution (2013)
27. Intel Software Guard Extensions, https://software.intel.com/en-us/sgx
28. Sinha, R., Rajamani, S., Seshia, S., Vaswani, K.: Moat: verifying confidentiality of enclave programs. In: ACM Sigsac Conference on Computer and Communications Security, pp. 1169–1184 (2015)