TEE: A Virtual DRTM Based Execution Environment for Secure Cloud-End Computing

Weiqi Dai[1,2], Hai Jin[1], Deqing Zou[1], Shouhuai Xu[2], Weide Zheng[1], Lei Shi[1]
[1] Cluster and Grid Computing Lab, Services Computing Technology and System Lab
Huazhong University of Science and Technology, Wuhan, 430074, China
[2] University of Texas at San Antonio, San Antonio, TX 78249, USA

ABSTRACT
Cloud computing is believed to be the next major paradigm of computing because it will substantially reduce the cost of IT systems. Ensuring security in the cloud-end is necessary because customers' data are stored and processed there. Previous studies have mainly focused on secure cloud-end storage, whereas secure cloud-end computing is much less investigated. The current practice is solely based on *Virtual Machines* (VM), and cannot offer adequate security because the guest *Operating Systems* (OS) often can be easily breached (e.g., by exploiting their vulnerabilities). This motivates the need of solutions for more secure cloud-end computing. This poster presents the design, implementation and analysis of a candidate solution, called *Trusted Execution Environment* (TEE), which takes advantage of both virtualization and trusted computing technologies simultaneously. The novelty behind TEE is the virtualization of the *Dynamic Root of Trust for Measurement* (DRTM).

Categories and Subject Descriptors:
C.5 COMPUTER SYSTEM IMPLEMENTATION, C.5.0 General; B.0 GENERAL

General Terms
Security

Keywords
Virtual Machine Monitor (VMM), Dynamic Root of Trust for Measurement (DRTM), cloud computing, Xen hypervisor.

1. INTRODUCTION
Cloud computing customers (including enterprises and individuals) move their data and processing of the data into a cloud, which must ensure both secure storage and secure computing because a single attack could cause the compromise of multiple customers' data. To reduce cost, *Virtual Machines* (VM) have become essential for cloud computing and are indeed widely utilized in today's cloud-end computing practice. As shown in the example scenario of Figure 1, the cloud provides the customers the *illusion* of their current self-managed systems through the so-called *Trusted Virtual Domain* (TVD) [2]. While much progress has been made in secure cloud-end storage (e.g., [1]), the problem of secure cloud-end computing has yet to be tackled.

CCS'10, October 4-8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

Our contributions. We present the design, implementation and analysis of a novel system, called *Trusted Execution Environment* (TEE), for secure cloud-end computing as shown in Figure 1. TEE can support a spectrum of application needs, ranging from pure cryptographic libraries to full-fledged trustworthy software. Moreover, TEE can provide an *overlay* of TEE network that can be deemed as more trustworthy than the network of VMs within the same TVD (e.g., for cryptographic multi-party computation).

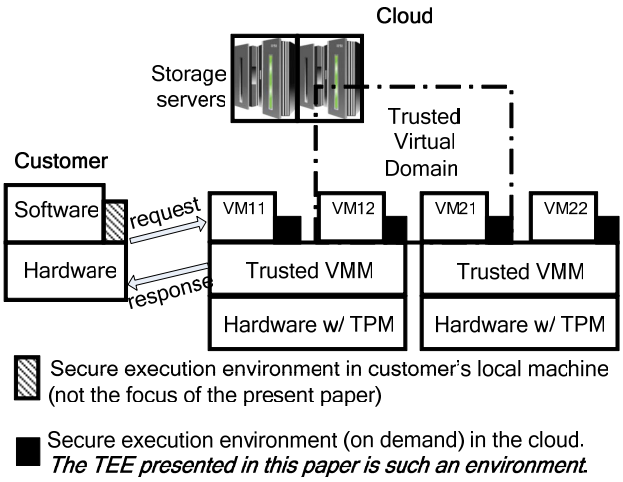

Figure 1. A scenario of secure cloud computing.

From a technical perspective, the novelty of our work is the virtualization of DRTM. This is a non-trivial task because, in particular: how should we deal with the problem that vTPM cannot determine the origin of TPM commands (because in the Xen hypervisor we experimented with only locality 0 is used)? We solve this by extending the locality control and management of vTPM. We have implemented the design reported in the poster, using Xen para-virtualization.

2. RELATED WORKS
There are investigations that exploit VMMs for secure execution environment. Why do we have to virtualize DRTM and TEE? In comparison to the investigations that exploit either virtualization or trusted computing, TEE takes advantage of both in a non-trivial fashion because of the following. First, unlike Flicker [4], TEE does not directly use DRTM; instead, it is based on vDRTM, so when a sensitive application executes in the hardware-protected environment, other customers' VMs running on the same platform will not be frozen. Second, execution environment solely based on vTPM is not sufficient for our purpose because (1)

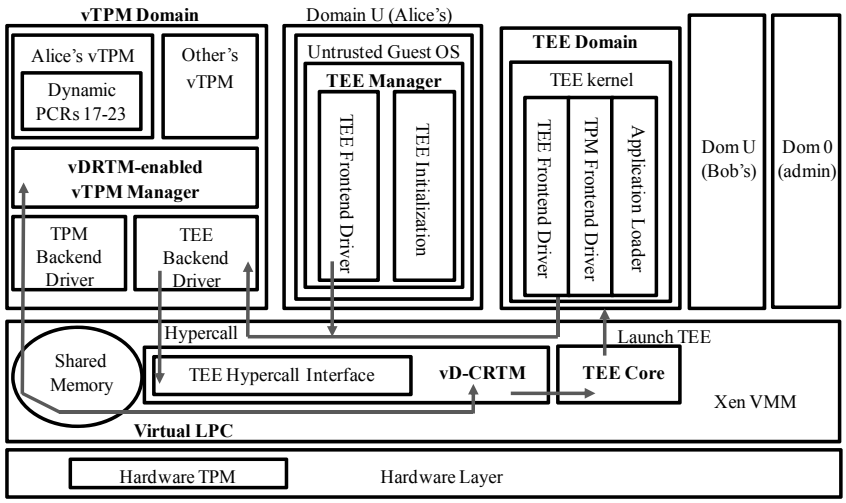

Figure 2: TEE architecture has two parts: vDRTM (consisting of vDRTM-based vTPM Manager, Virtual LPC, and vD-CRTM) and TEE (consisting of TEE Manager, TEE Core, and TEE Domain). Arrows represent control flows.

vTPM is fundamentally based SRTM and thus cannot provide a secure system environment without rebooting the VM, and (2) vTPM cannot tell that a command is from an untrusted guest OS or from a trusted execution environment. These are actually the reasons why DRTM is introduced and explain why we should utilize DRTM. Third, integration of vTPM and Terra [3] is also not sufficient for our purpose because of the following. (1) TEE can be invoked on demand but Terra is not designed to operate in this fashion. (2) TEE can implement fine-grained protection and attestation but Terra cannot achieve this without paying price because of the following dilemma. If one runs a single sensitive application in a single VM, it is difficult to share data between the VMs that are associated with their own vTPM (e.g., how should one vTPM unseal the data sealed by another vTPM? This causes a flexible or scalable key management problem, which TEE does not suffer from).

3. TEE ARCHITECTURE

In order for a customer to run (on demand) a sensitive application in a secure execution environment that mimics the one enabled by DRTM, we virtualize DRTM. The resulting system architecture consists of two parts: vDRTM and TEE.

3.1 vDRTM

Table 1: Locality in DRTM and in vDRTM

localities	used by (in DRTM)	used by (in vDRTM)
locality 4	Trusted hardware (DRTM)	vDRTM
locality 3	Auxiliary components	Auxiliary components
locality 2	" Runtime" environment for Trusted OS	TEE Kernel during launch
locality 1	Trusted OS	TEE Domain after launch
locality 0	Legacy environment for SRTM	Untrusted guest OS

It mimics the functions and services of DRTM while allowing multiple invocations, and has three components. Recall that TPM 1.2 uses five localities to distinguish the origin of a TPM command so as to determine whether it is trusted. As high-lighted in Table 1, we define five localities in vDRTM that are in parallel to their counterparts in DRTM. This is primarily implemented by vDRTM-enabled vTPM Manager, which ensures that only vD-CRTM is able to reset vTPMs PCRs 17-19. In particular, locality 4 is the only origin of command that can reset PCRs 17-19. This means that guest OS and TEE cannot reset these PCRs because they cannot use locality 4. Note that during the launch of TEE kernel, it can use locality 2 and thus can release keys and data in the corresponding vTPM. TEE kernel and sensitive application can use locality 1, and thus cannot release the keys and data associated to locality 2. Guest OS (locality 0) cannot release cryptographic keys associated to locality 1.

vDRTM-enabled vTPM Manager (in vTPM Domain). It supports the control of locality, and is obtained by modifying the Xen vTPM manager. It can prevent guest OS (locality 0) from accessing sensitive data sealed by sensitive application under locality 1. We use Algorithm 1 to control the localities.

Algorithm 1 vtpmd locality processing

```
1: if received command origin == 1 then
2: if received TOSPresent == 0 then
3: set localityModifier = 000
4: end if
5: set TOSPresent as the received value
6: end if
7: if TOSPresent = 1 then
8: set localityModifier as the received locality bits
9: end if
```

vD-CRTM (in VMM). vD-CRTM is responsible for receiving and authenticating the hypercalls for launching TEE, measuring the TEE kernel and application (possibly input as well) that will run in TEE domain, commanding vTPM to reset PCRs 17-19, extending the aforementioned measurement into PCR 17, and finally hands over the control to TEE core.

Virtual LPC. vDRTM must facilitate the communication between the vDRTM-enabled vTPM (in vTPM Domain) and vD-CRTM (in VMM), in a fashion similar to the communication between TPM and DRTM. A straightforward method is to establish a channel between vD-CRTM and each vTPM instance, which however will waste a significant amount of resource.

Because vTPM manager can communicate with the vTPM instances, we build a virtual LPC between vD-CRTM and vTPM manager, which is then multiplexed by all the vTPM instances. Our virtual LPC is implemented via shared memory between Xen and vTPM manager. After vTPM manager asks Xen to share one page of memory with it, virtual LPC works as follows (see also Figure 3).

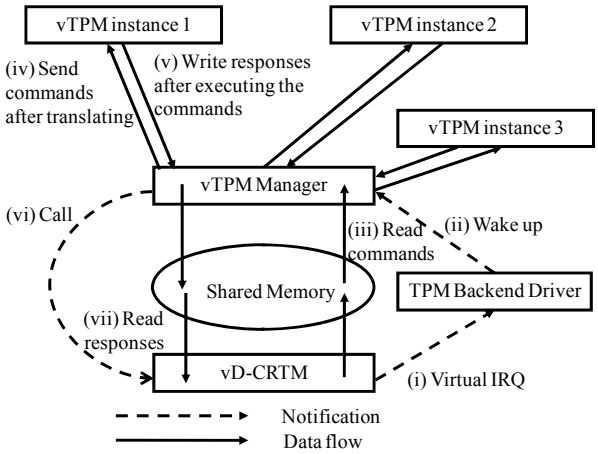

Figure 3: Virtual LPC

3.2 TEE

TEE consists of the following three components (see Figure 2).

TEE Domain. It is the trusted execution environment in which a sensitive application runs on top of TEE kernel, which can be obtained by extending a desired or needed software system (ranging from pure cryptographic libraries to full-fledged OS as long as the system is deemed as trustworthy). The extension is to incorporate three modules:

1. TPM Frontend Driver. It allows TEE to have access to a vTPM. It is obtained by modifying the Xen TPM Frontend Driver.
2. TEE Frontend Driver. Through TEE Backend Driver (in vTPM domain), it allows TEE to send exit request to TEE core (in Xen VMM) so as to destroy the TEE domain and unpause the paused guest OS.
3. Application Loader. It loads (and decrypts, if needed) the sensitive application, and encrypts the output of the sensitive application (if any).

TEE Manager (in guest OS). It is the interface between a customer and TEE, and consists of the following two modules.

1. TEE Initialization. This initializes the cryptographic mechanisms for authenticating the customers and the associated cryptographic keys.
2. TEE Frontend Driver. It loads both the TEE kernel and the sensitive application into the guest OS memory space, and forwards to the vD-CRTM the request for launching TEE as well as the relevant parameters.

TEE Core (in VMM). It uses Xen's pause command to pause the guest OS in which the TEE manager initiated the request for launching TEE, launches the TEE domain (TEE kernel, which then loads the sensitive application), destroys the TEE domain after the sensitive application exits, and uses Xen's unpause command to unpause the guest OS.

3.3 Running example

Let us look at a running example (see Figure 4).

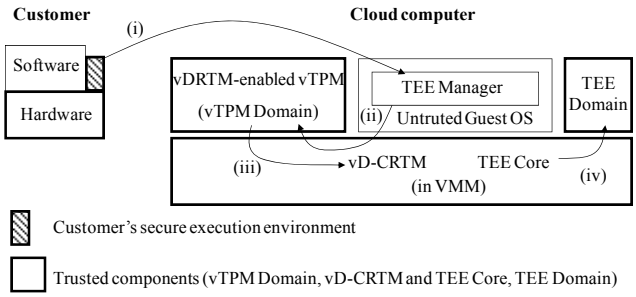

Figure 4: High-level control flow for launching TEE

1. A customer sends to TEE Manager in a VM (or guest OS) a request as well as parameters about the (encrypted) sensitive application.
2. TEE Manager allocates memory for the sensitive application and TEE Kernel, and requests the TEE Backend Driver (in vTPM Domain) to launch the TEE Kernel.
3. TEE Backend Driver (in vTPM Domain) uses a hypercall to request TEE Core to launch TEE.
4. TEE Core pauses the VM that issued the request. vD-CRTM authenticates the request and measures TEE Kernel and sends the measurement to vTPM Manager, which extends the measurement into PCR 17 in the VM's associated vTPM; TEE Core launches TEE Kernel, which loads the application. TEE Domain is then scheduled by Xen (as a replacement of the paused VM).

We note that the above process corresponds to the case that a customer runs a whole sensitive application in TEE domain. It can be easily extended to accommodate the fine-grained protection that a portion of an application needs to run in TEE domain. In this case, we let the application contact the customer-end trusted execution environment, which then invokes TEE domain in the cloud computer in the same fashion as illustrated above. The involvement of customer-end trusted execution environment is actually important because the calling application is not protected from the guest OS, which makes it possible that a malicious guest OS runs the sensitive portion of the application many times.

4. REFERENCES

[1] K. Bowers, A. Juels, and A. Oprea. Hail: A high-availability and integrity layer for cloud storage. Cryptology ePrint Archive, Report 2008/489, 2008. http://eprint.iacr.org/.

[2] A. Bussani, J. Griffin, B. Jansen, K. Julisch, G. Karjoth, H. Maruyama, M. Nakamura, R. Perez, M. Schunter, A. Tanner, L. van Doorn, E. V. Herreweghen, M. Waidner, and S. Yoshihama. Trusted virtual domains: Secure foundation for business and it services. Technical report, 2005.

[3] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. SIGOPS Operating System Review (SOSP'03), 37(5): 193-206, 2003.

[4] J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. Flicker: an execution infrastructure for tcb minimization. In ACM Eurosys'08, pp.315-328, 2008.