JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 5, No. 4, December 2011, pp. 331-337

Data Firewall: A TPM-based Security Framework for Protecting Data in Thick Client Mobile Environment

Wooram Park and Chanik Park*
Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea
wizrampa@postech.ac.kr, cipark@postech.ac.kr

Abstract
Recently, Virtual Desktop Infrastructure (VDI) has been widely adopted to ensure secure protection of enterprise data and provide users with a centrally managed execution environment. However, user experiences may be restricted due to the limited functionalities of thin clients in VDI. If thick client devices like laptops are used, then data leakage may be possible due to malicious software installed in thick client mobile devices. In this paper, we present Data Firewall, a security framework to manage and protect security-sensitive data in thick client mobile devices. Data Firewall consists of three components: Virtual Machine (VM) image management, client VM integrity attestation, and key management for Protected Storage. There are two types of execution VMs managed by Data Firewall: Normal VM and Secure VM. In Normal VM, a user can execute any applications installed in the laptop in the same manner as before. A user can access security-sensitive data only in the Secure VM, for which the integrity should be checked prior to access being granted. All the security-sensitive data are stored in the space called Protected Storage for which the access keys are managed by Data Firewall. Key management and exchange between client and server are handled via Trusted Platform Module (TPM) in the framework. We have analyzed the security characteristics and built a prototype to show the performance overhead of the proposed framework.

Category: Ubiquitous computing

Keywords: Trusted platform module; Data firewall; Virtual desktop infrastructure; Virtualization

I. INTRODUCTION

IT administrators have to handle various security issues, such as security-sensitive data leakage by malicious insiders, updating the secure system, and installing security patches. According to the most recent CSI Computer Crime and Security Survey [1], respondents have experienced the following types of attacks; malware infection (67%), laptop theft (34%), insider abuse (25%), unauthorized system access (26%) and unauthorized access or privilege escalation by insider (13%) in 2010 (Table 1).

Virtual Desktop Infrastructure (VDI) [2, 3] is one of the solutions to these issues. In VDI, the execution images including the OS and libraries are managed centrally in a server and each execution image (a.k.a., Virtual Machine [VM] image) is executed on a virtual machine in the server. Each client will access its own virtual machine running on the server through remote desktop protocols like MS RemoteFX [4], Citrix ICA [5], and VMWare PCoIP [6]. VDI assumes a thin client device model, which has limited functionalities, so the user performance of the VDI is highly dependent on the remote desktop protocol supported. There are two important benefits of VDI: central management of all client VMs and storage of each client's security-sensitive data in a server. All the client VM images are managed centrally in a server, thus enabling uniform installation of security patches etc. In addition, a user is not allowed to store any data in a local storage of a client device, but must store it in the remote storage of the server. Accordingly, VDI has been widely

Open Access **http://dx.doi.org/10.5626/JCSE.2011.5.4.331** **http://jcse.kiise.org**

Received 16 July 2011, **Accepted** 26 August 2011
*Corresponding Author

Table 1. Type of attacks experienced

Type of attack	2009 (%)	2010 (%)
Malware infection	64	67
Insider abuse	30	25
Unauthorized access or privileged escalation by insider	15	13
Laptop or mobile hardware theft or loss	42	34
Theft of or unauthorized access to system	30	26

adopted to centrally manage security-sensitive data, thus enabling secure protection of data. In order to overcome the limited functionalities of a thin client, VDI usually is based on thick client devices such as laptops.

However, it is still possible that the data stored in the central storage can be leaked by malicious insiders (e.g. server administrators), because they can have privileged control of the central server and storage. Moreover, data leakage could occur due to malicious software installed in mobile client devices and unauthorized access to the central server.

In this paper, we present *Data Firewall*, a security framework that protects security-sensitive data based on Trusted Platform Module (TPM) [7] and client-side virtualization. Data Firewall adopts *remote attestation* [8, 9] and *Protected Storage* [10] techniques based on TPM to prevent data leakage by malicious insiders, although they have privileged control. There are two types of execution domains defined in Data Firewall: *Normal VM* and *Secure VM*. In Normal VM, a user can execute any applications installed in the laptop in the same manner as before. A user can access security-sensitive data only in the Secure VM, for which the integrity (i.e., virtual machine image) should be checked in advance. All the security-sensitive data are stored in Protected Storage, for which the access keys are managed by TPM.

II. BACKGROUND

In this section, we provide a brief survey of TPM, the integrity measurement approach for trusted computing in a virtualization environment, and VDI, which are used to develop Data Firewall.

TPM is a security chip, proposed by the Trusted Computing Group (TCG) [11]. It supports capabilities such as *remote attestation* and *sealed storage*. Remote attestation creates hash chains of the hardware and software platform configuration at boot time. This allows a Trusted Third Party (TTP) [12] to verify that the platform environment has not been changed. Sealed storage provides support for data encryption with a platform state in the TPM, called *sealing*. The TPM must be in the correct state in order for the data to be decrypted, called *unsealing*. Also, the TPM supports the TPM Attestation Integrity Key (AIK), which is a private key (AIK^{-1}) stored inside the TPM. It is only used to sign the current values of the TPM's Platform Configuration Registers (PCRs). A challenger can get the AIK public key from the AIK certificate.

The TPM contains a volatile storage for protecting data and

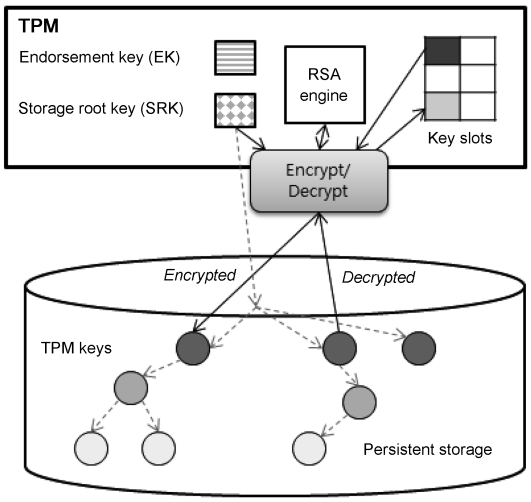


Fig. 1. Trusted platform module (TPM) key hierarchy. RSA: Rivest Shamir Adelman.

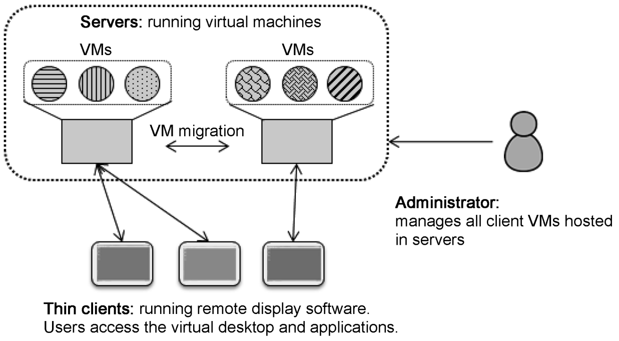

Servers: running virtual machines
VMs
VMs
VM migration
Administrator: manages all client VMs hosted in servers
Thin clients: running remote display software. Users access the virtual desktop and applications.

Fig. 2. Virtual desktop infrastructure (VDI). VM: virtual machine.

keys entrusted to it. This storage holds currently used keys, but it is very small, thus unused keys are encrypted with the storage key and stored in a persistent storage, such as a hard disk. The storage key is encrypted with the Storage Root Key (SRK), which is generated during the process of taking logical ownership of the platform and is embedded in the TPM. Fig. 1 shows the TPM Key hierarchy.

The Terra [13], proposed by Stanford University, was the first approach that tried to adapt attestation and VM identity in a virtualization environment for trusted computing. It provided a novel concept for well-defined remote attestation and authentication, and it involved building a certificate chain using TPM. Moreover, Terra defined two VM concepts, *open-box VM* and *closed-box VM*. Open-box VMs can run commodity operating systems and customize the appearance of today's general-purpose platforms. Closed-box VMs provide the functionality of implementation on a dedicated closed platform. Hypervisor protects the privacy and integrity of a closed-box VM's contents, and the closed-box VM can only start to run after attestation.

In VDI, operating systems and applications are running inside virtual machines that reside on servers. Operating systems inside VMs are referred to as virtual desktops. Users access virtual desktops and applications from a thin client as a terminal. They use applications as if they were loaded on local systems, but the difference is that they are centrally managed. Desktop administrative and management tasks are significantly reduced. Applications can be quickly added, upgraded and patched. Also, data is stored in the central storage, thus data is easier to manage [14]. Fig. 2 shows the simple VDI architecture.

III. DATA FIREWALL FRAMEWORK

A. Assumptions and Threat Model

Assumptions: We assume that the target client and server are equipped with the TCG's hardware (i.e., BIOS with Core Root of Trust for Measurement [CRTM] and, TPM or Mobile Trusted Module [MTM] [15, 16]). Thus, target client and server can be booted through TCG's trusted boot process. Also, we assume that Kernel Virtual Machine (KVM) [17] is installed in each client or LiveDF.

Threat Model: Our framework aims to protect security-sensitive data in thick client mobile devices or central servers from malicious insiders and outside attackers. We assume that malicious insiders and attackers can use a privilege to access the central storage and the client root anytime. Furthermore, they can mount attacks via remote attestation and compromise the policy of Secure VM for access to the protected storage with the root privilege. Finally, a user has various client device types such as desktop, laptop and other mobile devices. We assume that the user can lose their mobile devices or that attackers can steal them in order to access security-sensitive data.

B. Architecture Design

Fig. 3 shows the architecture of Data Firewall. The server of Data Firewall has *Data Firewall Key Manager (DFKM)*, *Protected Storage*, a client information database, and TPM. DFKM

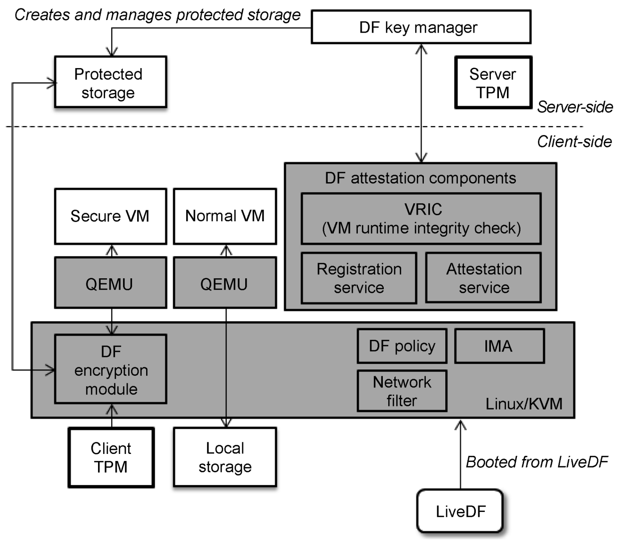


Fig. 3. The overview of the data firewall (DF) architecture. Trusted computing base (TCB) is shown in gray. The host system is booted from LiveDF. TPM: trusted platform module, IMA: integrity measurement architecture, KVM: Kernel virtual machine.

supports VM image management, remote attestation, and key management for Protected Storage. The server TPM creates a *Protected Key* for an encryption key of each Protected Storage.

The client of Data Firewall is booted from *LiveDF,* which is a bootable mobile device, and it has all the Data Firewall components of the client. The client TPM supports a trusted boot process with DF attestation components and Integrity Measurement Architecture (IMA) [9]. Furthermore, it stores the Protected Key migrated from the server TPM. After remote attestation, a user can access the encrypted data with the Protected Key through *DF Encryption Module* in Secure VM. Also, the user can execute any applications installed in the client in Normal VM.

We discuss how to build the trusted computing environment in Data Firewall, and how to access security-sensitive data protected from malicious insiders and attackers.

***1) Registration Process*:** A user has to create a LiveDF device through the user registration process of DFKM when using Data Firewall for the first time. DFKM registers a user's information (i.e., user ID, password, physical LiveDF ID, and LiveDF Key) with the client information database. Moreover, DFKM creates the user's Protected Key and LiveDF Key using the server TPM. Protected Key is a unique key for each Protected Storage. It is used to encrypt and decrypt the data in Protected Storage through the DF Encryption Module. Finally, DFKM installs Data Firewall components and a LiveDF key to the LiveDF. The LiveDF key is used for matching between the user, client and LiveDF. The user's client is booted from the LiveDF the first time, and then the registration service is activated. We define it as the client registration process. The registration service gathers the client's initial status (i.e., the measurement log and Platform Configulation Register [PCR] values in TPM). This information is created by CRTM, a trusted bootloader [18, 19] and IMA through a trusted boot process. We measure the Linux kernel, KVM, initial ramdisk, QEMU [20, 21], and DF attestation components for building TCB. We extend PCRs in the client TPM with these measurements and make the measurement log. The registration service sends the registration request to DFKM and gets a 160-bit nonce value (N). The service uses TPM Quote commands to create an DFResponse (Equation 1) and sends this response to DFKM, which decrypts an encrypted ID message in this response with LiveDF Key and, attests LiveDF ID and user ID/PW. Finally, DFKM registers this information to the user's information database and uses it for verifying the client next time.

$$\begin{aligned} DFResponse = Sig_{AIK-1}\ &\{\text{N}|\text{PCR values}\}, \\ &\text{Measurement Log, AIK}_{cert}, \\ &\{\text{User ID/PW, LiveDF ID}\}_{LiveDF\ Key} \end{aligned} \quad (1)$$

2) Attestation Process: After the registration process, the attestation service is activated at the next boot time. We define this as client attestation. The attestation service gathers the client statuses during boot time. This is similar to the client registration process. The service sends the attestation request to DFKM and gets a 160-bit nonce value. It uses the TPM Quote command to create an DFResponse (Equation 1) and sends this response to DFKM, which verifies the current client status by comparing it with the initial client status. If the client is trusted,

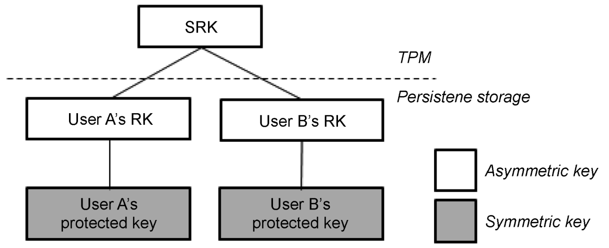


Fig. 4. Data firewall trusted platform module (TPM) key tree. SRK: storage root key, RK: roaming key.

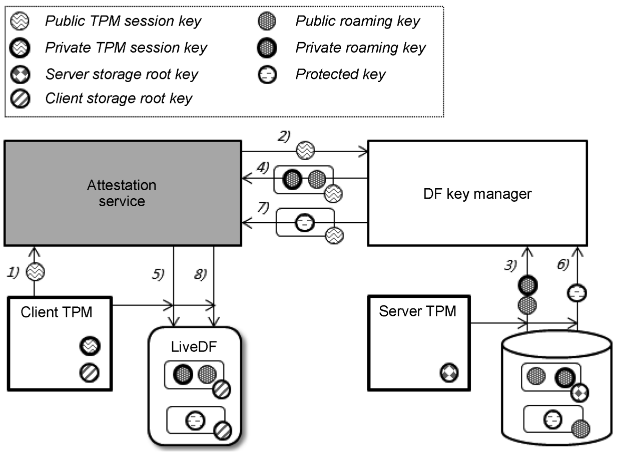


Fig. 5. Protected key migration. TPM: trusted platform module, DF: data firewall.

then DFKM sends the result, a DF policy for Secure VM, and a request for *TPM Key Migration*, which is the process of migrating a key from one TPM to another. The user's Protected Key is migrated from the server TPM to the client TPM. Fig. 4 shows our TPM key tree. SRK is the root of the TPM key and never leaves the TPM (Non-Migratable Key). We create a Roaming Key (RK) pair, which is a child key of SRK that is migratable and asymmetric. An RK pair is encrypted by the public SRK, and stored in a persistent storage. Furthermore, we create a Protected Key, which is a child key of RK that is migratable and symmetric. The Protected Key is encrypted by the public RK, and stored in a persistent storage.

Fig. 5 shows the Protected Key Migration process. After receiving the request, 1) the attestation service creates a TPM key pair, called the *TPM Session Key* Pair; 2) the service sends the TPM Session Public Key to DFKM; 3) DFKM loads the RK pair and the RK certificate from the server TPM; 4) DFKM rewraps the RK pair and the certificate with the TPM Session Public Key using the server TPM, and then they are sent; 5) The attestation service decrypts this data with the TPM Session Private Key using the client TPM and registers the RK pair with the client TPM. After RK migration; 6-8) the attestation service starts the Protected Key migration process between the server TPM and client TPM in the same manner.

After the Protected Key Migration process, the attestation service sets the network filter with the DF policy. Finally, it runs a Normal VM and a Secure VM with the DF policy using KVM and QEMU.

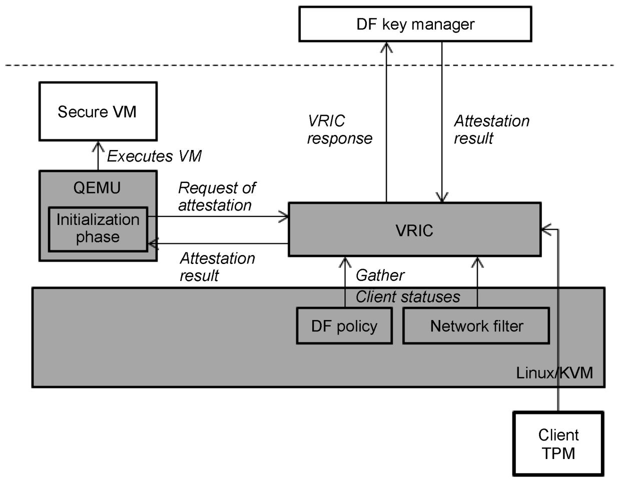


Fig. 6. The VM runtime integrity check (VRIC) process. DF: data firewall, VM: virtual machine, KVM: Kernel VM.

***3) Virtual Machine Runtime Integrity Check (VRIC)*:** We need to ensure that we run a Secure VM with a trusted DF policy. We make a *VRIC* module and add some codes for creating a link between QEMU and VRIC to the initialization phase of QEMU. VRIC is similar to the attestation service, but it is only activated at the running VM. Secure VM is started, and then QEMU enters the initialization phase. In this phase, QEMU checks the VM options and sets the configuration of the VM. Before setting up the configuration of the VM, QEMU sends the VM options to the VRIC module and waits until the completion of attestation. After receiving the attestation request from QEMU, VRIC sends a request to DFKM and gets a 160-bit nonce. Next, VRIC extends PCRs in the client TPM with the DF policy of Secure VM and the configuration of the network filter. VRIC uses the TPM Quote command to create a VRIC response (Equation 2) and sends this response to DFKM.

$$\begin{gathered} \mathit{VRICResponse} = \mathit{Sig}_{AI\,K-1}\,\{\mathrm{N|PCR\ values}\}, \\ \text{Measurement Log, AIK}_{cert}, \\ \{\text{DF Policy, Network Filter Configuration}\}_{LiveDF\ Key} \end{gathered} \tag{2}$$

DFKM makes a comparison between the current and initial client status. Furthermore, it compares the current client DF policy and network filter configuration to the initial configuration in the client information database. The DF policy and network filter configuration includes the VM configuration, such as allowed networks and devices for VM. If the current status is trusted, then DFKM sends the result and VRIC returns a success message to QEMU. This process is shown in Fig. 6.

***4) Protected Storage*:** In Normal VM, the VM image is booted from the operating system pre-installed in a client device, and a user can execute any applications installed in the device. All the executions in Normal VM are based on the physical local storage. In Secure VM, the VM image is booted from a VM image stored in the Data Firewall server for a specific client. A user in Secure VM can access the security-sensitive data only after the VM image is attested according to the DF policy and VRIC by the Data Firewall server. As previously explained, all the security-sensitive data are encoded by the DF Encryption

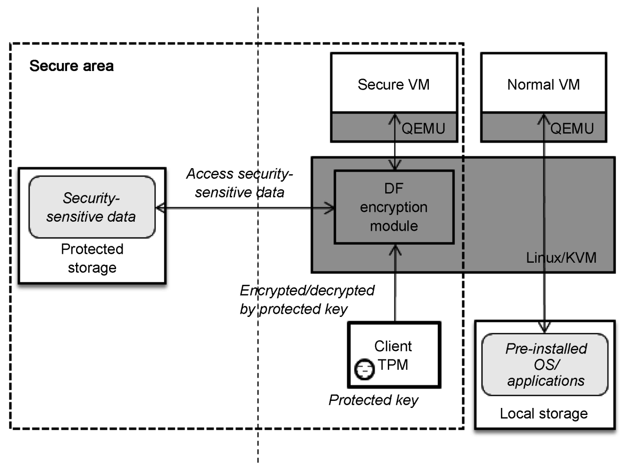


Fig. 7. Protected storage. A user in secure virtual machine (VM) can access the security-sensitive data. A user in Normal VM can only access the local storage and cannot access the secure area (dash box). DF: data firewall, KVM: Kernel VM, TPM: trusted platform module.

Module and stored in the space called Protected Storage. A symmetric key called Protected Key is created for a client or a group of clients by the Data Firewall server, and the key is distributed to the DF Encryption Module of each legal client. Key management and distribution are based on TPM both in clients and the server. This process is shown in Fig. 7.

IV. ANALYSIS OF DATA FIREWALL

A. Security Characteristics

We evaluate how our framework achieves the goals of protecting security-sensitive data from various types of attacks experienced (Table 1).

***1) Malware Infection and Unauthorized System Access*:** First, we ensure that the adversary cannot compromise the TCB of our framework to influence its output. The Root of Trust is started from the CRTM in our framework. The Linux/KVM, DF attestation components, and QEMU are measured during the TCG's trusted boot sequence. The integrity of each component is extended to TPM's PCRs. Their integrity can be attested using the TPM's standard attestation process. Thus, any attempt to compromise the TCB of our framework will be detectable, regardless of the root privilege of the adversary. Moreover, this can prevent malwares installed in mobile devices used as Data Firewall clients. Because malwares should compromise the TCB of our framework, they would be detectable during the attestation process. We also ensure that the adversary cannot compromise the LiveDF. This has all the Data Firewall components of the client and LiveDF Key, thus it has to be protected from any attacks. LiveDF has a unique LiveDF ID and LiveDF Key. LiveDF ID is a physical device ID, such as information about the USB buses in the system. LiveDF Key is created by the server TPM and it is also unique. We register this information with the user information database with the initial status of the client in the user registration process. Thus any attempt to compromise LiveDF will be detectable during the attestation process. Finally, we can detect the attack compromising DF policy and the configuration of the network filter also. The DF policy and the configuration of the network filter are attested by VRIC when starting a VM. VRIC extends PCRs in the client TPM with this information in order to ensure that the adversary cannot compromise this information. DFKM can attest whether the configuration of the client is trusted or not. Additionally, VRIC gathers this information and reports it to DFKM periodically in order to prevent the Time-of-Check-to-Time-of-Use (TOCTTOU) attack. If the result of the request is false or the connection is closed due to various problems, then the VRIC stops the Secure VM immediately.

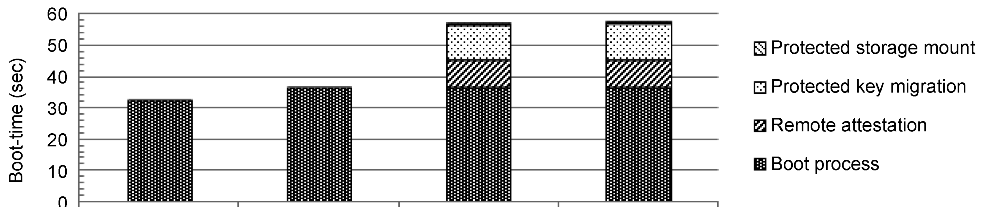
2) Insider Abuse and Mobile Hardware Theft: We can protect security-sensitive data in Protected Storage from malicious insiders and outside attackers. Each block of Protected Storage is encrypted by Protected Key and a user can only access the data through DF Encryption Module. The adversary cannot gain access to this key, because it is managed by the TPM and can only be used after the attestation process. We also use the TPM Key Migration technique for protecting against Protected Key leakage on the network. If a user loses their laptop or other mobile devices are used as Data Firewall clients, the adversary cannot access the data, because the Protected Key is only migrated from the server TPM after the attestation process at every boot time. Moreover, a malicious insider can gain unauthorized access to the Data Firewall server. However, Protected Storage can only be accessed through DF Encryption Module with Protected Key, which can only be obtained using a trusted Data Firewall client. Thus any attempt to leak security-sensitive data and Protected Key can be prevented.

B. Performance Evaluation

We implemented a Data Firewall prototype on a thick client mobile device, a DFKM, Protected Storage server, and a USB device for LiveDF. A laptop has a 2.4 GHz Intel Core 2 Duo processor with Intel-VT support and 3 GB RAM for a thick client mobile device. Windows XP is installed on this mobile device for a Normal VM. A server has three 2.83 GHz Intel Xeon processors, 3 GB RAM, and 1 TB hard-disk. LiveDF is a 16 GB USB device. We use 2.6.35.12 Linux/KVM with IMA as the target hypervisor and 0.14.0 QEMU. A Normal VM and a Secure VM consists of Windows XP, one vCPU, and 1 GB RAM. These systems are connected over a 100 Mb Ethernet switch and an 802.11g wireless access point, because a user's mobile devices or desktops are connected to a wired/wireless network in order to access security-sensitive data in their enterprise.

We evaluate three aspects of Data Firewall's performance overhead: 1) the boot time of Data Firewall, 2) the performance overhead of VRIC, and 3) the performance overhead of Protected Storage on Secure VM.

***1) Boot Time*:** We measure the boot time of Data Firewall using BootChart [22]. We examine three cases, kernel booting without IMA and the attestation service, with only IMA, and with IMA and the attestation service. Fig. 8 shows the experi-



	w/o IMA, attestation service	w/IMA	w/IMA, attestation service (wired)	w/IMA, attestation service (wireless)
Protected storage mount	0	0	0.8355	0.8621
Protected key migration	0	0	11.2783	11.5293
Remote attestation	0	0	8.4880	8.6152
Boot process	32.5461	36.7989	36.8188	36.8201

Fig. 8. Average boot time (sec). This is calculated ten times. IMA: integrity measurement architecture.

mental results. The average boot time is calculated ten times. The first measurement shows the time needed to boot the host system. The second measurement shows the performance overhead of IMA. The third measurement shows the performance overhead of the attestation service in Data Firewall. The attestation service consists of three components: remote attestation, Protected Key migration, and Protected Storage mount. The remote attestation and Protected Key migration techniques have high performance overhead, because they use various TPM commands, such as TPM Quote. However, they are background processes, thus a user can access a user's client during these processes. Measurements in both wired and wireless networks are almost identical, because the size of attestation messages and Protected Key between the client-server is small.

2) The Performance Overhead of VRIC: VRIC is only activated when starting a VM. The VM starts to run, and then VRIC gathers the current client status. This is similar to the remote attestation process of the attestation service. The experimental results indicate that the performance overhead of VRIC is 8.4912 seconds on average. This performance overhead only affects the initialization phase of VM, but not the VM running time. Therefore, a user can execute applications or access data with almost the same performance in the virtualization environment.

3) The Performance Overhead of Protected Storage: We measure the performance overhead of Protected Storage in the Secure VM using IOMeter [23]. We compare the results of running the benchmark on the Secure VM with Protected Storage and with iSCSI network storage. We compare our secure storage system with other types of network storage, such as iSCSI, because we assume that Protected Storage is located on a central server. In this experiment, Protected Storage in the central server is based on iSCSI. Fig. 9 shows the measured results in terms of average I/O response time for the IOMeter benchmark with 100% random I/Os and 8 outstanding I/Os per target. Protected Storage has a performance overhead, because every block is encrypted and decrypted with Protected Key. However, if the performance difference is less than 10%, this constitutes low performance overhead.

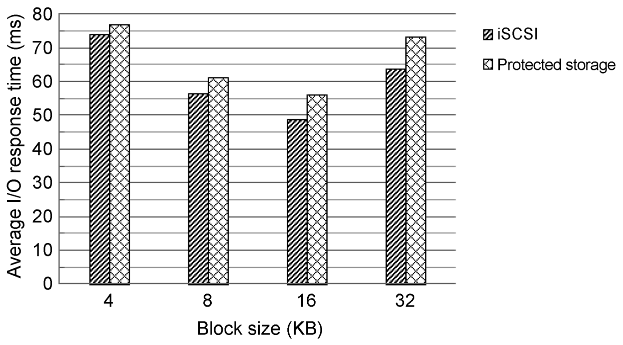


Fig. 9. Average I/O response time comparison for 33% writes and 67% reads of IOMeter benchmark.

V. CONTRIBUTION

Our contribution in this work is to develop a security framework for protecting security-sensitive data from many types of attacks. IT administrators can handle various security issues even if the attacker is a malicious insider in this framework. Despite the fact that our system sets a strong security policy for a user's client, the user can execute any applications installed in a user's client in the same manner as before in this framework.

VI. CONCLUSION

In this paper, we presented Data Firewall, a TPM-based security framework Data Firewall to centrally manage VM images and protect security-sensitive data in mobile environments.

The Data Firewall is based on the three major functional components of Protected Storage, remote VM image attestation, and key management/distribution protocol for TPM. We have analyzed the security features of the proposed framework and evaluated the execution overhead through a prototype.

In summary, the proposed framework is shown to prevent data leakage by any type of attack including a malicious insider's attack using TPM capabilities such as remote attestation and key management. Moreover, it has low execution overhead, thus ensuring its practicality in a real-world situation.

ACKNOWLEDGEMENTS

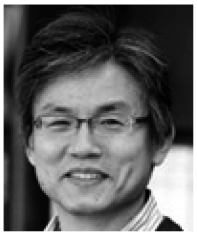
This research was supported by the Ministry of Knowledge Economy (MKE), Korea, under the Information Technology Research Center (ITRC) support program supervised by the National IT Industry Promotion Agency (NIPA) (NIPA-2011-C1090-1131-0009), and POSCO.

REFERENCES

1. R. Richardson, CSI Computer Crime and Security Survey 2010/2011, New York, NY: Computer Security Institute, 2011.
2. Citrix Systems Inc., "XenDesktop," http://www.citrix.com/xendesktop.
3. VMware Inc., "VMware View 5," http://www.vmware.com/products/view/overview.html.
4. Microsoft, "Windows Server 2008R2 Remote Desktop Services Features," http://www.microsoft.com/windowsserver2008/en/us/rds-remotefx.aspx.
5. Boca Research Inc., Citrix ICA Technology Brief, Boca Raton, FL: Boca Research Inc., 1999.
6. VMware Inc., VMWare View 4 with PCoIP, 2009.
7. Trusted Computing Group, "Trusted Platform Module," http://www.trustedcomputinggroup.org/developers/trusted_platform_module.
8. B. Kauer, "OSLO: improving the security of trusted computing," *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, Boston, MA, 2007, pp. 1-9.
9. R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," *Proceedings of the 13th Conference on USENIX Security Symposium*, San Diego, CA, 2004, pp. 16-16.
10. J. Choi, W. Park, and C. Park, "A framework of secure access to iscsi network storage based on TPM," *Proceedings of the 2009 Fall Conference on Korean Institute of Information Scientists and Engineers*, Seoul, Korea, 2009, pp. 5-9.
11. Trusted Computing Group, http://www.trustedcomputinggroup.org/.
12. "Trusted third party," http://en.wikipedia.org/wiki/Trusted_third_party.
13. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: a virtual machine-based platform for trusted computing," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 193-206, 2003.
14. VMware Inc., VMware View 3: Virtual Desktop Infrastructure White Paper. Palo Alto, CA: VMware Inc., 2008.
15. Trusted Computing Group, TCG Mobile Trusted Module Specification Version 0.9 Revision 1, Beaverton, OR: Trusted Computing Group, 2006.
16. J. E. Ekberg and M. Kylanpaa, Mobile Trusted Module (MTM): An Introduction, Helsinki, Finland: Nokia Research Center, 2007.
17. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," *Proceedings of the Linux Symposium*, Ottawa, Canada, 2007, pp. 225-230.
18. "GRUB TCG Patch to support Trusted Boot," http://trousers.sourceforge.net/grub.html.
19. H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetho, S. Yoshihama, and T. Nakamura, Trusted Platform on Demand (TPod). Research Report RT0564, Kanagawa, Japan: IBM Japan, Ltd., 2004.
20. F. Bellard, "QEMU, a fast and portable dynamic translator," *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, Anaheim, CA, 2005, pp. 41-46.
21. I. Habib, "Virtualization with KVM," *Linux Journal*, no. 166, p. 8, 2008.
22. Z. Mahkovec, "Bootchart," http://www.bootchart.org/.
23. J. Sievert, "Iometer: The I/O Performance Analysis Tool for Servers," http://www.intel.com/design/servers/devtools/iometer/index.htm.

Wooram Park

Wooram Park received his B.S. degree in Computer Science and Engineering from Korea University, Korea in 2006. He is currently a M.S. & Ph.D. candidate in Computer Science and Engineering at POSTECH in Korea.



Chanik Park

Chanik Park received his B.S. degree in Electronics Engineering from Seoul National University, Korea in 1983,his M.S degree from Electronics and Electrical Engineering (Computer Engineering) from KAIST, Korea in 1985, and his Ph. D. degree in Electronics and Electrical Engineering (Computer Engineering) from KAIST, Korea in 1988. He has been a professor in Computer Science and Engineering at POSTECH in Korea since 1989.