2012 IEEE 4th International Conference on Cloud Computing Technology and Science

DFCloud : A TPM-based Secure Data Access Control Method of Cloud Storage in Mobile Devices

Jaebok Shin * Yungu Kim * Wooram Park *† Chanik Park *†
* Department of Computer Science and Technology
† Department of Creative IT Excellence Engineering and Future IT Innovation Laboratory
Pohang University of Science and Technology, Pohang 790-784, Korea
{zstormx, pi3wi2, wizrampa, cipark}@postech.ac.kr

Abstract— **Using the cloud storage services, users can access their data in any time, at any place, even with any computing device including mobile devices. Although these properties provide flexibility and scalability in handling data, security issues should be handled especially when mobile devices try to access data stored in cloud storage. Currently, a typical cloud storage service, Dropbox, offers server-side data encryption for security purpose. However, we think such method is not secure enough because all the encryption keys are managed by software and there is no attestation on the client software integrity. Moreover, a simple user identification based on user ID and Password is also easy to be compromised. Data sharing which is critical in enterprise environment is significantly restricted because it is not easy to share encryption key among users.**

In this paper, we propose *DFCloud*, a secure data access control method of cloud storage services to handle these problems found in the typical cloud storage service Dropbox. DFCloud relies on Trusted Platform Module (TPM) [1] to manage all the encryption keys and define a key sharing protocol among legal users. We assume that each client is mobile device using ARM TrustZone [2] technology. The DFCloud server prototype is implemented using ARM Fastmodel 7.1 and Open Virtualization software stack for ARM TrustZone. For DFCloud client, TPM functions are developed in the secure domain of ARM TrustZone because most ARM-based mobile devices are not equipped with TPM chip. The DFCloud framework defines TPM-based secure channel setup, TPM-based key management, remote client attestation, and a secure key share protocol across multiple users/devices. It is shown that our concept works correctly through a prototype implementation.

Keywords : cloud storage service; security; TPM; ARM TrustZone

I. INTRODUCTION

Today, many computing devices with high portability and strong computing power are emerging. Since anyone can have multiple such devices and can also access to wireless network easily, the needs for efficient methods to sharing or synchronizing his or her data among several devices have arisen. Using cloud storage services is one of the possible solutions and has become very popular nowadays.

Because of the nature of cloud storage service, user data are stored in and managed by the remote server. So there are several security problems such as data leakage, modification, or data loss. To be used widely, the service providers should guarantee the security level of their product otherwise cautious consumers including enterprises will hesitate to use services.

Data on cloud storage services can be leaked due to several reasons. We classified them into two main categories. One is server-side leakage and the other is client-side leakage. The server-side data leakage can be occurred by exploiting vulnerabilities of commodity hypervisors. According to a study of the National Vulnerability Database [7], there were 26 and 18 vulnerabilities in Xen [8] and VMware ESX [9] respectively. Malicious insiders or attacker can achieve arbitrary user data through these vulnerabilities via VM escape attacks [17], hypervisor rootkits [18] or executing malicious codes.

On the other hand, user's carelessness incurs client-side data leakage. If a user losses mobile devices that connected to the cloud storage service then anyone who picked up this device can access the user's security-sensitive data, or if a user accesses his or her cloud storage using untrusted devices, user's credentials or security-sensitive data can be intercepted by malicious programs: Keylogger programs, viruses, or malicious codes can reside in untrusted device and cause client-side data leakage through the network.

Many commercial cloud storage services protect user's data stored in server storages by introducing client-based or server-based data encryption [10][11]. When client-base encryption is used, it is ensured that user's data is encrypted before transmitting to server. All components related to encryption such as encryption process, library and data keys are hosted by client program. By using these components, client program generates one-time-use symmetric key for data encryption. This key will be encrypted using user's asymmetric public key and uploaded with encrypted data to the server. When the user want to download his or her data, the client program request this encrypted data key along with data and then decrypt the data key using user's private key and finally decrypt the data.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

In the case of server-based encryption, secure network link is established between server and client before the data transmission take place. User data are securely transferred to the server, encrypted in the server, and then stored in server's storages. Server key manager maintains data encryption keys assigned uniquely to each client. Amazon Simple Storage Service (Amazon S3[4]) currently supports both kind of encryption depending on user's choice. Dropbox, in other hand, uses server-based encryption method and SSL in transferring data between server and client.

Both data encryption methods mitigate risk of server-side data leakage, but several risks of data leakage still exist. First, in case of the client-based encryption, the keys involved in the process of encryption are managed by software manner. If the TCB of client corrupted, attacker will intercept keys. Second, in case of the server-based encryption, user's data still remain as plain text in the server before encryption process. Therefore attacker can exploit vulnerabilities of servers to achieve user's data. Third, user authentications are just depending on user ID and password only. A user can access to every contents of cloud storage if he or she logged in with valid pair of user ID and password.

To handle these problems, we proposed a secure data access control method of cloud storage named DFCloud which stands for Data Firewall [12] Cloud. We designed and implemented DFCloud as follow:

1. DFCloud uses client-based encryption method to guarantee that server-side data leakage cannot occur.
2. DFCloud relies on Trusted Platform Module (TPM) functionalities to manage all the encryption keys and define a key sharing protocol among legal users.
3. We assumed that each client is mobile devices using ARM TrustZone technology and implemented security modules in the Secure World of the TrustZone environment to support hardware-based key management.
4. DFCloud performs remote attestation on each client to prevent data or credential leakages caused by malicious programs in client-side, before a data encryption key is used. The key can be used only if the whole client system is in "safe state".

II. REALATED WORK

Dropbox [3] is one of the popular cloud storage services. Mulazzani et al. analyzed the Dropbox's security drawbacks and proposed simple solutions [14]. Dropbox supports multiple devices per user and assigns a 128-bit unique host ID to each device. The host ID is stored in each device when the client software initialized and hereafter, client software never asks user's ID and password again unless the client software reinstalled. Due to the host ID authentication explained above, data leakage can occur either when Dropbox-connected device is lost or when the host ID is stolen by malicious programs.

Dropbox uses SHA-256 hash of file to identify already stored user data to reduce redundant data transmissions. When a client sends files, client software calculates hash values and sends them to the server. If the hash value is already stored by other user before, then the server simply links that files to the user's account and consequently, the user can access that file. If an attacker acquired the hash value of desired victim files, then attacker can receive the data freely.

The other security problem is related to server-side encryption. In the server of cloud storage service, user's data still remain as plain text before encryption process. Therefore attacker can exploit vulnerabilities of servers to achieve user's data.

There are number of previous works which tackles existing cloud storage services' security problems. Popa et al. implemented CloudProof [6] to verify untrusted service provider and control access to data among users. CloudProof offers ability to client for proving the service provider's malicious behaviors, cryptographic data protection, and key distribution. DEPSKY [19] also used encryption for data storing. DEPSKY uploads replicas of data on diverse clouds that forms a cloud-of-clouds to ensure availability, integrity, and confidentiality of data stored in cloud. Venus [20] is another framework addressing cloud storage service providers' security. The main focus of Venus is integrity and consistency of data on service provider. Venus introduced versioning method of each operation on data objects and by using this version, Venus maintains integrity and consistency of data objects.

Although these works can ensure server-side data security, they assumed that client is trustworthy; the key is still managed by software and not support verification client platform integrity. We claim that user data or key can be leaked by untrusted client platform, so DFCloud is the framework for making client trustworthy.

III. ASSUMPTION AND THREAT MODEL

Each client is an ARM TrustZone technology enabled mobile device for hardware-based key management. As malicious programs can reside on any mobile client, we assumed that client device is untrusted. Cloud provider also untrusted and can be compromised by attacker, including malicious insiders, who can exploit vulnerabilities of service provider and consequently leak the user data. We mainly focused on data leakage either on client-side or server-side, but not integrity, consistency, or availability of providers: these are also important, but we assumed that pre-existing solutions are working well on this.

IV. ARCHITECTURE

A. Overview

DFCloud architecture (Fig. 1) consists of three entities; ~~a~~ client, DFCloud server and commodity cloud storage services. Client is a TrustZone enabled mobile device and separated into two worlds: Normal world and secure world. A user can access the data through the

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

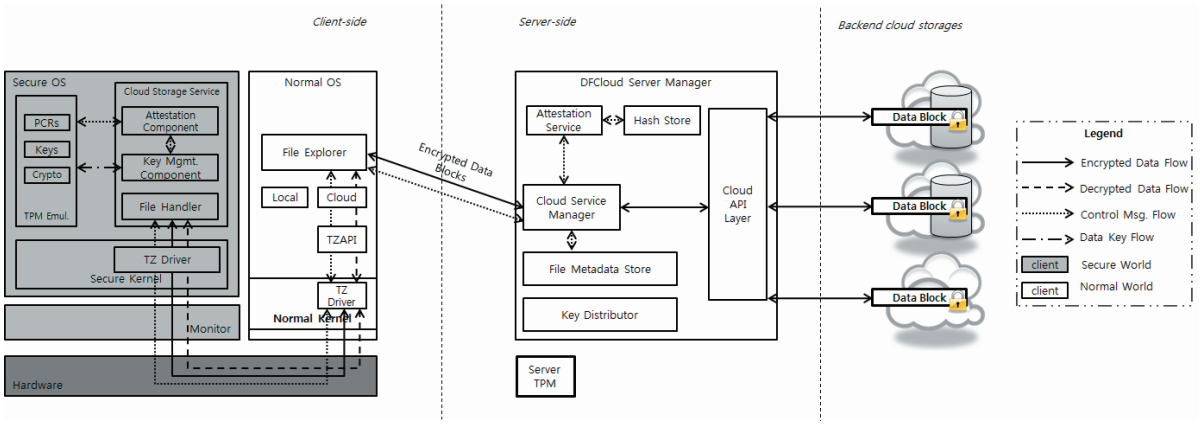


Figure 1. DFCloud Architecture

File Explorer which is a simple application running on the normal world; it that provides listing either local or cloud files, and basic file operations such as execute, delete, copy and move. It communicates with server for file list retrieval, uploading, downloading of encrypted data, or control messages.

DFCloud client's security components are running in Secure World. These components have responsibilities for file encryption/decryption, key management, and platform attestation. We designed to use TPM emulator instead of hardware TPM because the most of mobile devices are not designed to equip hardware TPM chip. Using the TrustZone technology, we can assume that DFCloud's security components are in same security level as hardware TPM chip. The Cloud Storage Service consists of three sub-processes. The File Handler manages encryption or decryption of data blocks. All data blocks or control messages including file list pass through the TrustZone monitor which performs context switching between Normal and Secure world. All data blocks transmitted to the Server are encrypted by the File Handler. We present about the other two sub-processes, Attestation Component and Key Management Component, in following sections.

The DFCloud Server acts as a proxy server that manages users and the third party cloud storage services. The DFCloud server has responsibility for maintaining File Metadata Store. The file metadata consist of file owner information and each data block's address that points the location on third party cloud storage service. When a client requests user's file list, the Cloud Service Manager will look up the Store, organize the result message and send to client's Cloud Service process. When a client requests to upload data blocks, the Cloud Service Manager forwards data blocks to third party cloud storage services and update data block's address to File Metadata Store. On the other hand, when a client requests to download data blocks, the Cloud Service Manager sends the information of data blocks' location in the third party cloud storage services. Then the client can access to desired data blocks directly from third party cloud storage services.

In the following sections we describe detailed protocols for secure cloud storage service.

B. User Login and Remote Attestation

We assume that Core Root of Trust for Measurement (CRTM) is Secure World because current mobile devices do not meet TCG's specification. During the booting phase, the Attestation Component in Secure World measures the integrity of normal OS bootloader and normal OS image, based on the assumption above and extends result hash values to TPM emulator's Platform Configuration Registers (PCRs). Since the normal OS bootloader has no interface to communicate to Secure World, we designed the Attestation Component measures the normal OS and normal OS bootloader at the same time. After measurement of Normal World's initial image, booting sequence of normal OS proceeds. During the running time of normal OS, the Integrity Measurement Architecture (IMA) [13] measures integrity of libraries and processes at each time of loading. Then the IMA extends the result hash value to TPM emulator in Secure World by using TrustZone API (TZAPI). At every time of remote attestation, Client's File Explorer quotes PCR values by issuing the attestation TZAPI then the Attestation Component in Secure World returns attestation results to the File Explorer and File Explorer sends the attestation request. By DFCloud Server's Attestation

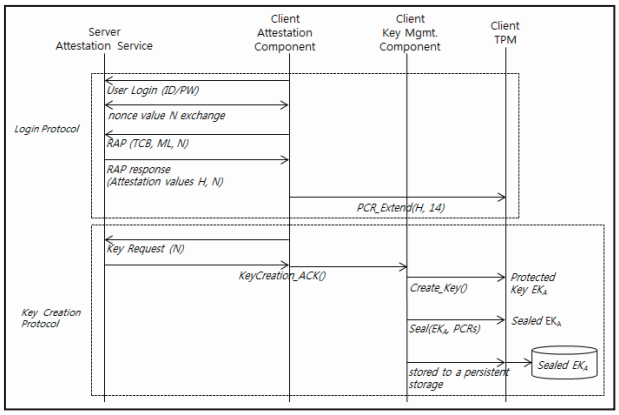


Figure 2. Login Protocol and Key Creation
RAP : Remote Attestation Protocol
ML : Measurement Log

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Service, these PCR values are used to verify the client system, whether it is in known safety state.

At the first time of use, a user should enter his or her credential such as a user ID and password pair. If the credential is valid one, then the user can just read the list of stored files: Actual file contents can be retrieved after achieving the data encryption key. To achieve key, the client is required to pass the attestation from the server and get an attestation value (Server's Attestation Service creates an attestation value if the client satisfies attestation requirements, and Attestation Service stores the mapping information between user ID and attestation value for later usage).

The Login Protocol (the first dashed box in Fig. 2) consists of verifying user's confidential, remote attestation of client and retrieving attestation result. First, a secure session is established between Server's Attestation Service and Client's Attestation Component. When the user sends login request, the two components exchange a fresh nonce ***N*** which is created by server-side Attestation Service if the user's credential is valid. Using this nonce ***N***, ***TCB*** (i.e., PCR values in client' TPM emulator) and ***ML*** (i.e., measurement log), the client-side Attestation Component creates attestation request. As receiving attestation request massage form client, server-side Attestation Service validates client's environment by comparing with Hash Store's contents. Finally client receives response message consists of Attestation value ***H***, and nonce ***N***. If the attestation passed, the Client's Attestation Component extends the Attestation value ***H*** into one of PCR's slot (e.g., PCR 14), otherwise if the attestation failed, client receives just fail message which means deny of service. When the attestation process passed successfully, now the client is ready for creating or using data encryption key.

C. Key Creation and Management

If the user has no pre-created data encryption key then the client's Attestation Component starts Key Creation Protocol (the second dashed box in Fig. 2). The Attestation Component sends a key creation request to the server, and the server updates user's status to present that the user created data encryption key in current device. The acknowledgement for key creation is finally forwarded to the client's Key Management Component then it starts creating user's new data encryption key - the client TPM emulator creates symmetric key $\boldsymbol{EK_A}$ which will be used in encryption and decryption process by the Cloud Service.

When the client does not use the key $\boldsymbol{EK_A}$, the key is stored in client's persistent storage in the form of 'Sealed data'. The Key Management Component requests to TPM emulator to seal the key $\boldsymbol{EK_A}$ using current PCR values that are including attestation value ***H***. In later when the key is requested by the Cloud Service to encrypt or decrypt the contents of data block, the attestation process will take place and acquire unsealed key $\boldsymbol{EK_A}$ after the unsealing process. If the attestation process fails then client cannot achieve unsealed key; therefore no one can access to the data.

D. Key Sharing across Multiple Devices

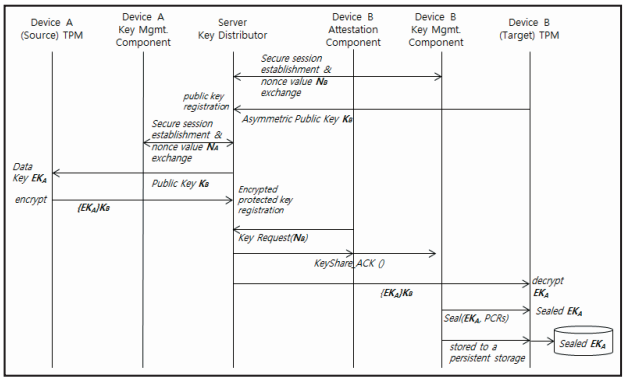


Figure 3. Key Sharing Protocol b/w Device A and B

A user can have multiple devices, so possible usage scenario is like this: A user is currently using device A and made account for DFCloud, made data key $\boldsymbol{EK_A}$, and uploaded files that are encrypted with $\boldsymbol{EK_A}$. If he wants to access that files on DFCloud using device B, he must have $\boldsymbol{EK_A}$ in device B to decrypt data.

Handling the challenge of key sharing across the multiple devices, DFCloud offers a key sharing protocol (Fig. 3). Server's Key Distributor, target client's Key Management Component and source client's Key Management Component are participants in the protocol.

First phase is exchanging the asymmetric public key $\boldsymbol{K_B}$ of device B (the target client) between two client devices via Server's Key Distributor. When device A receives $\boldsymbol{K_B}$, it unseals its sealed data encryption key to gain $\boldsymbol{EK_A}$ and finally encrypt $\boldsymbol{K_A}$ with $\boldsymbol{K_B}$ ($\{\boldsymbol{K_A}\}\boldsymbol{K_B}$). This encrypted data key will be stored in Server DB.

Next step is attestation of target device B as described in Section B. If the attestation process succeeds, the Key Distributor sends the device A's encrypted data key to the device B. Device B decrypt the encrypted data key using its private portion of asymmetric key and then seals the $\boldsymbol{EK_A}$ using current PCR values and stores in persistent storage.

E. Data Upload and Download

After login phase, data encryption key is stored in mobile device's persistent storage. The File Handler performs encryption or decryption of user data using the data encryption key. When a user performs copies a file from the File Explorer to local storage then a download session is established between File Explorer and Server's Cloud Service Manager. After that the File Explorer sends download request to Cloud Service Manager with file name. Searching the File Metadata Store, the Cloud Service Manager determines source backend cloud storage service where the actual user data stored, and forward the data's location request to that backend cloud storage. Then the source backend cloud storage service transmits user data's location to DFCloud server's Cloud Service Manager and the Cloud Service Manager forwards the location to client's File Explorer. Client's File Explorer can access to data blocks stored in backend cloud storage with obtained

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

address and performs downloading. At this point, the user cannot retrieve the plain text of data because every data are encrypted, so next step is decryption of data by sending encrypted data blocks to Secure World's File Handler. First, the File Explorer establishes data decryption session with Secure World's File Handler via TrustZone API. After receiving session establishing command, the File Handler initiates decryption process and returns acknowledgement to the File Explorer. Consequently, the File Explorer issues decryption file TZAPI command with encrypted data to the File Handler. The File Handler decrypts the received encrypted data with decryption algorithm and user data key which is unsealed by using current PCRs, and attestation value obtained in login process before. Finally, decrypted data are returned to the File Explorer and stored to intended path in local storage.

The data uploading sequence is reverse of the downloading sequence. When a user copies a local file to cloud (i.e. copies to the cloud folder in the File Explore), the File Explorer first establishes data encryption session with Secure World's File Handler. Performing the similar sequence to data decryption, the File Explorer finally gets encrypted form of intended file and establishes uploading session with Server's Cloud Storage Manager. Then the File Explorer send uploading request with file name and encrypted data. The Server's Cloud Storage Manager selects destination backend cloud storage service, stores received encrypted data to that cloud service and updates the File Metadata Store with uploading information including destination cloud storage and file information.

V. Implementation and Evaluation

A. Implemenation

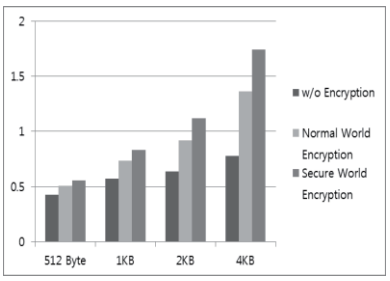
We built our prototype of DFCloud based on Open Virtualization' software for ARM TrustZone[15] and ARM Fastmodel[16] for emulating ARM Cortex-A15 that supports TrustZone technology. The Open Virtualization software is an open source project that provides Normal World's TrustZone API (TZAPI), TrustZone Driver, Secure monitor and custom secure OS. We built the Linux kernel (version 2.6.38) based Normal OS with IMA support by modifying the source code of IMA to invoke TZAPI instead of TPM_extend operation for extending measured hash values of loaded processes and libraries to Secure World. In Normal OS, DFCloud File Explorer is running and it performs file transmission to server, file reception from the server and viewing user's file list. The File Explorer uses TZAPI to communicate with Secure World's components.

Currently, the Secure OS implemented by Open Virtualization supports simple secure task which performs encryption and decryption of data. We modified this secure task to implement our Cloud Storage Service. When the File Explorer sends the file encryption or decryption requests, the Cloud Storage Service uses the user's 256-bit key and RC4 crypto algorithm, which Secure OS currently support, to perform crypto operations.

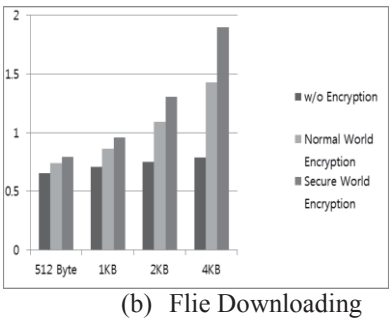
On the server side, we implemented DFCloud Server Manager running as application on the Linux, and maintain user information, user file metadata store and hash value store. We use Dropbox as a backend cloud storage service and the Cloud API Layer support wrapper function of Dropbox API.

B. Performance Evaluation

We measured our framework's performance overhead in downloading and uploading user's file. Since every user data are encrypted or decrypted in Secure World, we designed the experiment in three categories to evaluate crypto operation's overhead and Normal-Secure world switching overhead. For the same file, we conducted file download without crypto operations, with performing crypto operation in Normal World, and with performing crypto operation in Secure World. Current implementation support up to 512 bytes data encryption or decryption at one time, so using Secure World crypto operation, the world context switching occurs multiple times of 512 bytes to total file size; for example for 2KB file, world context switching occurs 4 times. The result (Fig.4 (a) for uploading time, Fig. 5 (b) for downloading time) shows that total execution time of uploading and downloading for various file sizes. Current cloud storage services support big size data transfer, but out current implementation do not support big files due to message size restriction of data exchange between Normal World and Secure World through Secure Monitor using TZAPI. At this moment, we examined performance overhead of file with small size such as 512 Bytes, 1KB, 2KB, 4KB.



(a) File Uploading



(b) Flie Downloading

Figure 4. File upload/download performance
Every execution time is represented in second.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Due to world switching time increases proportional to file size, the result shows that differences between Normal World encryption and Secure World encryption increase as file size increase (in our experiment, the difference increased as twice). The crypto operation's execution time is also increased proportional to file size.

Our prototype is designed to simply demonstrate the concept of secure cloud storage service for mobile devices using ARM TrustZone technology; we didn't perform any performance optimization yet. Several optimization techniques will be useful to mitigate significant performance overhead introduced in current implementation, such as using shared memory between Secure World and Normal World, using designated processor core to each World etc. We remain performance optimization of DFCloud in our future work.

C. *Security anlaysis*

We mainly focused on data leakages that can occur in either client-side or server-side. DFCloud exploits client-side encryption technique to mitigate server-side data leakages such as malicious insider attack or exploiting vulnerabilities of server platform. DFCloud's mobile client supports hardware-based key management by using ARM TrustZone and running TPM emulator on Secure World. Since every cryptographic function is processed only in Secure World, any malicious programs cannot access data key. Due to remote attestation protocol for verifying the client, we ensure that malicious behaviors cannot occurs on Normal World. Therefore, a user can access to cloud storage's contents in secure mobile environment and store user data to the remote server in encrypted form using securely created and managed data encryption key.

VI. CONCLUSION

In this paper, we presented DFCloud, a TPM-based secure data access control method of cloud storage in mobile devices. There are several security issues in cloud storage services, among these issues we mainly focused on data leakages that can occur in either client-side or server-side. DFCloud exploit client-side encryption technique, remote attestation for client platform, and hardware based key management to build a secure access environment. DFCloud also support secure key sharing protocol across the multiple devices or users.

We implemented our prototype on ARM Fastmodel to emulate ARM Cortex-A15 core and Open Virtualization's software stack in environment setup. The performance overhead is quiet high, but if we adopt some optimization techniques such as shared memory between two World, then we can reduce overhead introduced in our current implementation.

ACKNOWLEDGMENT

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2012-H0301-12-3002) and this work was supported by IT Consilience Creative Program of MKE and NIPA (C1515-1121-0003)

REFERENCES

[1] TPM. http://www.trustedcomputinggroup.org/resources/tpm_main_specification

[2] ARM, "ARM Securtiy Technology, Building a Secure System using TrustZone Technology", 2009

[3] Dropbox. http://www.dropbox.com

[4] Amazon Simple Storage Service. http://aws.amazon.com/s3/

[5] Trusted Computing Group (TCG). http://www.trustedcomputinggroup.org

[6] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang. Enabling Security in Cloud Storage SLAs with CloudProof. In Proceeding of the 2011 USENIX Annual Technical Conference, June 2011.

[7] National Vulnerability Database. http://nvd.nist.gov/.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In Proceedings of the 19th ACM SOSP, October 2003.

[9] VMware ESXi: Bare Metal Hypervisor. http://www.vmware..com/products/esxi

[10] Fraunhoger SIT, "ON THE SECURITY OF CLOUD STORAGE SERVICES", 03, 2012 http://www.sit.fraunhofer.de/content/dam/sit/en/studies/Cloud-Storage-Security_a4.pdf

[11] Amazon S3, "Using Data Encryption" http://docs.amazonwebservices.com/AmazonS3/latest/dev/UsingEncryption.html

[12] W. Park, C. Park, "Data Firewall: A TPM-based Security Framework for Protecting Data in Thick Client Mobile Environment." Journal of Computing Science and Engineering, Vol. 5, No. 4, December 2011

[13] 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.

[14] M. Mulazzani, S. Schrittwieser, M. Leithner and M. Huber, "Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space," in USENIX Security, 2011.

[15] Sierraware, "Open Virtualization for TrustZone Overview", 2011, http://www.openvirtualization.org/Open-Virtualization-for-ARM-TrustZone.pdf

[16] ARM, "ARM Fast Model Reference Manual", http://infocenter.arm.com/help/topic/com.arm.doc.dui0423m/DUI0423M_fast_model_rm.pdf

[17] CVE-2008-0923. http://eve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0923

[18] The Blue Pill Project. http://bluepillproject.org/.

[19] A. Bessani, M. Correia, B. Quaresma, F. Andre, P. Souuusa. DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds. In Proceedings of the ACM SIGOPS EuroSys Conference, April 2011.

[20] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, D. Shaket. Venus: Verification for Untrusted Cloud Storage. In Proceedings of the ACM Conference on Computer and Communications Security Workshop, October 2010.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE