ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.3

Proffering Cloud Storage Secured using Scrambled Technique

E. Nanthini[1], G. Praiselin Lydia[2], V. Poongothai[3], Mrs.G.Indumathi[4]
UG Student[1,2,3], Assistant Professor[4]
Department of Computer Science and Engineering
Velammal Institute of Technology, India

Abstract:
Increase in the usage of Cloud computing, the security alarms in the form of confidentiality of user data by the domain and administrator. This paper proposes the secured cloud proffering with scrambled technique and uses the Hardware Security Module (temporary storage) to store automatic generating keys for encryption, decryption and authentication which isolates the cloud user data from potentially malicious domains or cloud administrators. Within the secured environment, the hardware security module provides essential security functionality with automatic generated keys for data in storage. Such restrictions prevent the cloud administrator from affecting the security of the Guest Users. This system not only defends against wide attacks bur also for the small TCB. This paper discusses about the software implementation of the proposed Proffering approach with safety unit, analyzes the security and presents its performance results. This system the three methods of cloud security .The connection is secured .There will be secured platform between the cloud user and cloud admin. By using the security key they verified the respective user of cloud. The data stored in the cloud will be encrypted.

1. INTRODUCTION

Although the use of cloud computing has been expanding due to its elasticity in resource provisioning, security concerns over the exposure of private data in such systems have restricted its further proliferation. In such cloud computing environments, physical systems are commonly virtualized, and a cloud user accesses a virtual machine (VM) which may share a physical system with other users. To enable such a sharing environment, several software components are needed. A virtual machine monitor (VMM; hypervisor) provides fundamental functionality such as sharing a physical CPU with guest VMs and isolating the memory of guest VMs. A privileged domain (privileged VM) provides management interfaces to manipulate guest VMs and device drivers, to perform I/O operations requested by guest VMs. Since it is such software components that provide isolation between guest VMs, the security of guest VMs cannot be guaranteed if software components are compromised. Existing cloud systems have been built on an assumption that a privileged domain is trusted. However, such an assumption is unacceptable by cloud service users. A privileged domain has a large body of code because it includes an entire OS supporting various device drivers and several software components for VM management. A large code implies that the system has a wide attack surface, making it easier to compromise the system [1] and hindering extensive security verification. Frequent patches for OS kernel and device drivers make security verification even more complicated. Furthermore, threats caused by cloud administrators have arisen lately [2], [3]. Cloud administrators have privileges not only to manage VMs through the privileged domains but also to manage cloud facilities such as network and storage devices. Accordingly, if cloud services are to be widely accepted, there must be a trusted cloud service capable of overcoming even the vulnerabilities of privileged domains and untrustworthy cloud administrators. To provide a trusted cloud service, the security of guest VMs should not be affected by a compromised privileged domain or a malicious administrator. We propose cloud architecture in which security-critical processes and data affecting the security of guest VMs are isolated from privileged domains and cloud administrators. In particular, the proposed architecture guarantees three aspects of cloud security. The first aspect is the connection between a cloud user and an allocated VM. Before a cloud user connects to an allocated VM, the cloud user wants to have a guarantee that: a) the allocated VM is launched from a correct and valid VM image, b) the VMM on which the allocated VM runs is valid, and c) the connection is secured. The second aspect is the security of guest VM images. To guarantee the confidentiality and integrity of guest VM images, the encryption and integrity checking of guest VM images are supported by a cloud system. It is here that a cryptographic environment for the encryption and integrity checking should be isolated from privileged domains. A cryptographic key and integrity (hash) data of VM images also should be isolated from cloud administrators to prevent potential security threats. The last aspect is a VM management operation. A cloud user sends a VM management operation such as start VM or shutdown VM to change the state of a VM. However, a cloud administrator can trigger a management operation or can disable a management operation triggered by a cloud user by ignoring the operation. Therefore, a current VM state could be different from what a cloud user thinks. Such inconsistency would cause a potential security threat. The proposed architecture consists of VMM and dedicated hardware providing security functionality to isolate guest VMs from privileged domains. Since cloud administrators could not access security processes and data, potential threats by cloud administrators could be mitigated. In summary, this paper makes the following contributions.

- The proposed system provides a secure connection scheme between a user and an allocated VM. It also enables a cloud user to verify the allocated VM as well as the VMM where the allocated VM runs before the first connection. Moreover, the connection between a cloud user and an allocated VM is secured by exchanging keys.

- The proposed system provides a secure storage scheme. To protect VM guest images from cloud administrators, the cloud system should isolate the cryptographic environment as well as a cryptographic key and integrity data. Since the cryptographic environment is not affected by privileged domains, VM images of guest VMs are protected in the proposed system.
- The proposed system provides a secure management scheme. Since a management operation is triggered by a cloud user, and the result of the operation is reported to the user, a cloud user knows the exact VM state. Therefore, cloud administrators cannot change the VM state at their discretion.

To show the feasibility of the proposed architecture, the paper introduces our implementation of the proposed hardware security module prototype in a PCI board with a low-cost embedded processor and a modified VMM supporting the hardware. The security analysis and evaluation of our prototype implementation show that the proposed system has a narrow attack surface and good I/O performance. In Section 2 of this paper is described necessary background, including our assumptions and threat model. The proposed system is illustrated in Section 3 and in Section 4 are described the implementation details of the proposed system. The security of the proposed system, including protocol verification and security analysis is discussed in Section 5. The evaluation results are shown in Section 6.

- In existing system, using I/O model of hypervisor with management tool they performed the cloud storage allocation for the cloud administrators. In this module the physical machines shared by multiple admin so they can easily access the others data.
- By I/O scheduler they perform the bidding operations based on the storage space required by an cloud administrator.
- The digital signature based on only the conventional system (i.e) using the DES algorithm.
- While digital signature proposed which provides encryption function on more complexing function it will more complex for users and easily attacked by malicious admin.
- In previous methods they used DES –Data Encryption Standard which uses 62 bit encrypt all data.
- Self service cloud computing method is efficient but in this method they cloud user and cloud admin have equal authority to change the data so it is insecure for the data. anybody can access the data which is stored in the database.

DISADVANTAGE

- While resource sharing improves hardware utilization and service reliability, this may also open doors to side channel or performance interference attacks by malicious admins.
- Using VMWare GSX Server and a Debian Linux host OS, is not “suitably high assurance for a real TVMM [Trusted VMM]”.
- By using digital signature it may increase the difficulty of factoring and the high computational cost.

2. PROPOSED SYSTEM

To prevent the data from the malicious admin we are providing keys for the secure isolated path for the cloud administrators. The keys are stored in separately in cloud, the keys are automatically generated. The data are stored in encrypted form protected with security key in the proposed system we used AES-Advanced Encryption Standard which encrypts the data on more efficient way .For each process there will be an attestation using security key this will prevent the data from malicious admin**.** There will be a hash values for the every data to prevent the forgery and is verified by the device authority this will increase more secure connection between cloud administrator and cloud provider**.** Using the entrusted attestation, management and reporting and managing VSC**.** It will prevent the data from the malicious admin.The data are securely stored in database in the form of encrypted form While bidding admin detail will be secured. Some advantages are The proposed system the AES algorithm is used which is more efficient than the DES This system will reduce the difficulty of factoring and computational cost Admin cannot change the virtual machine status arbitrarily.

- The proposed system provides a secure storage scheme. To protect VM guest images from cloud administrators, the cloud system should isolate the cryptographic environment as well as a cryptographic key and integrity data. Since the cryptographic environment is not affected by privileged domains, VM images of guest VMs are protected in the proposed system.
- The proposed system provides a secure management scheme. Since a management operation is triggered by a cloud user, and the result of the operation is reported to the user, a cloud user knows the exact VM state.

3. SYSTEM ANALYSIS

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and another is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then the successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of the present system can lead diversion from solution. We propose a new cloud architecture protecting the security of guest VMs against potentially malicious cloud administrators by isolating security-critical processes. This isolation is accomplished by providing restricted interfaces to a privileged domain. This architectural isolation guarantees that: a) a secure connection exists between a cloud user and an allocated VM, b) a secure cryptographic environment exists for VM images as well as for cryptographic keys and integrity (hash) data, and c) there is a secure management of guest VMs. The proposed cloud architecture includes cloud nodes and a trusted 3rd party authority called device authority (DA). DA verifies each cloud node and lets a cloud user know whether a VM of the cloud user is running on a verified node. Each cloud node consists of a VMM called Trusted VMM (TVMM) and of a hardware security module called trusted cloud module (TCM) which are described in Fig. 2. TVMM is a security-enhanced VMM isolating several data for guest VMs from cloud administrators. TVMM encrypts the guest VM images with cryptographic keys delivered by TCM, and enables a secure connection between a cloud user and an allocated VM. TVMM also validates a management operation triggered by a cloud user.

TCM is a hardware based security module supporting an isolated environment, and performs several security processes, such as signing and validation checking, within the isolated environment. TCM also provides special storage for keys and data. The biggest difference from existing security hardware is the privilege on the storage. A single root user (a cloud administrator in a cloud service context) has a root privilege on the storage of existing security hardware, whereas a cloud user has a privilege only on their TCM data. Therefore, the data in storage can neither be accessed nor deleted, even by a cloud administrator. In particular, the proposed architecture provides the following capabilities.

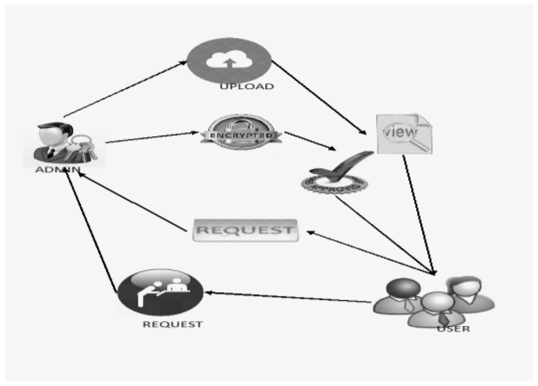


Figure.1 System Architecture

- Interface isolation. The proposed system provides restricted interfaces to a privileged domain to isolate security-critical processes and data from the privileged domain. It also provides security-critical interfaces only to a verified VMM.
- Secure connection. The proposed system guarantees a secure connection between a cloud user and an allocated VM.
- Secure storage. The proposed system guarantees a protected cryptographic environment for guest VM images.
- Secure management. The proposed system delegates a VM management privilege to a cloud user, and provides a way to check the completion of the management request.

3.1 Interface Isolation

The main goal of the propose architecture is to isolate security-critical processes and data from cloud administrators. Since the proposed architecture does not count on privileged domains, security processes are performed in TCM and TVMM. Therefore, the private data in TCM are accessed only by TCM itself and TVMM. On the other hand, a privileged domain also needs to access TCM for system managements. Accessing TCM from a privileged domain, however, jeopardizes the security of guest VMs. To resolve this dilemma, the proposed system supports dual interfaces which expose different interfaces, management and security-critical interfaces, to a privileged domain and TVMM respectively, as depicted.

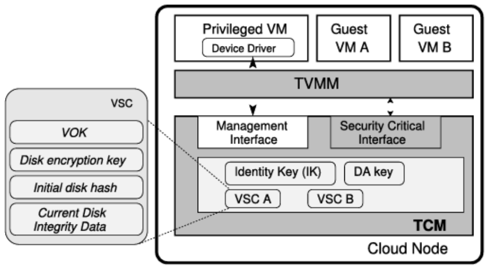


Figure.2. System Architecture

Management interface is a set of interfaces for initiating VM creation, destruction, and attestation. Therefore, the management interface is used by a privileged domain and includes following functions which are unrelated to accessing private data of guest VMs. Although the privileged domain initiates the management operations, any security sensitive operations are conducted within TCM, protecting the private data.

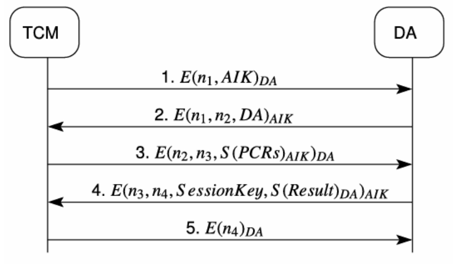
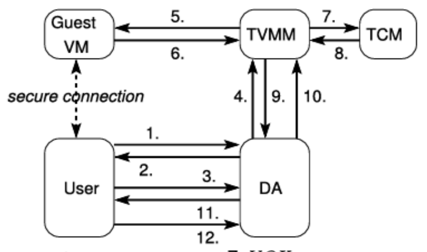

TCM
DA
1. $E(n_1, AIK)_{DA}$
2. $E(n_1, n_2, DA)_{AIK}$
3. $E(n_2, n_3, S(PCRs)_{AIK})_{DA}$
4. $E(n_3, n_4, SessionKey, S(Result)_{DA})_{AIK}$
5. $E(n_4)_{DA}$

Figure.3. Entrusted Connection

- Measurement and reporting. TCM supports an attestation function including system measurement and reporting as TPM does [6]. TCM stores system measurement values in special registers called platform configuration registers (PCRs) and returns a quoted blob. Since the measurement values are the calculated hash values of platform software stacks, an external entity is able to attest the platform where TCM is equipped.
- Entrusted attestation. After the system boots, only measurement and reporting and entrusted attestation functions are available to a privileged domain, but other functions remain unavailable. They become available only after verifying system states with the entrusted attestation performed via the management interface.
- Managing VSC. TCM maintains a data structure called VM security context (VSC), which contains security-critical data for each VM. VSC has cryptographic keys for disk encryption, and has the integrity data of VM images. Even though the keys and integrity data are read and modified via a securitycritical interface, allocation is accomplished via the management interface. Security critical interface is accessible only from the verified TVMM to provide security related functionality. To be specific, the following functions are provided.
- Accessing cryptographic keys and integrity data. TCM creates cryptographic keys for VM images and integrity data of the images, and stores them in VSC. The keys and integrity data are accessible from TVMM, which is verified with the entrusted attestation scheme before accessing the keys and integrity data.
- Signing initial guest image hash. To guarantee that a cloud user connects to a correct VM on the attested platform, the user should be able to check the initial guest image hash. The guest image hash is signed by the AIK of TCM to prevent forgery, and is verified by DA afterwards. This function is used for the secure connection between users and TCM, as described in the following section.
- Management request validation. A cloud user sends a management operation to a cloud node. When such an operation is not triggered by a valid cloud user, the VM state of a cloud user could be changed arbitrarily. Thus, this function is used for validating the owner of a guest VM as described in Section 3.4.

3.2 Secure Connection

To establish a secure connection between a cloud user and an allocated VM, there are three requirements. First, a cloud user should attest the platform where the allocated VM is running. Second, a cloud user should attest the allocated VM images. Third, a cloud user should receive a VM server key (e.g., SSH server key of the VM) and set a user key (e.g., user key of SSH server) securely in the allocated VM. To meet these requirements, we define a secure connection protocol (Fig. 4) and summarize keys in the protocol (Table 3). Since platform information and VM image information includes hash values that a cloud user does not know, the cloud user communicates to DA and DA translates such hash values into human recognizable information according to the protocol. DA communicates to TCM with a session key created by the entrusted verification protocol. During the entrusted verification process, DA attests the platform meeting the first requirement. To meet the second requirement, the proposed system stores the hashes of initial images during initial deployment, and the hashes are verified by DA afterward to ensure the initial integrity of VM images. In most cloud services, pre-made VM images are used to launch a new VM. Even though they exist as unencrypted images in the cloud repository, they are encrypted with a cryptographic key in VSC before the first launch. During the encryption process, the initial integrity hash value is calculated and stored in VSC. The hash value is delivered to DA in Step 9, and DA translates the hash value to VM image information such as OS version and software stacks. The VM image information is delivered to the cloud user in Step 10 and the cloud user is able to check the allocated VM images. To meet the third requirement, a VM server key and a user key are also exchanged securely with the proposed protocol. For major cloud service vendors, cloud systems create user keys and store them in their server. Afterwards, an allocated VM downloads the user key via HTTP.



1. $E(n_1, User)_{DA}$
2. $E(n_1, n_2, DA)_{User}$
3. $E(n_2, n_3, UK, VOK, AIK)_{DA}$
4. $E(n_3, UK, VOK, AIK)_{SK}$
5. UK
6. VK
7. VOK, n_3
8. $S(ImageHash, n_3)_{AIK}$
9. $E(n_3, n_4, S(Hash, n_3)_{AIK}, VK)_{SK}$
10. $E(n_4)_{SK}$
11. $E(n_3, n_4, ImgInfo, VK)_{User}$
12. $E(n_4)_{DA}$

Fig. 4. Secure connection.

Figure.4. secure connection

Accordingly, cloud administrators can access the stored user key. Contrary to such an approach, with the proposed protocol, a cloud user creates a pair of cryptographic keys and delivers the public part of the key to the allocated VM. The user key is denoted by UK in Fig. 4. Since the user key is asymmetric, the private part of the user key is and is never revealed. The VM server key denoted by VK is created by the OS of the allocated VM and is delivered to the cloud user via the protocol. Once a VM server key and a user key are exchanged, the communication between a cloud user andan allocated VM is secured by a publickey protocol such as the Needham Schroeder-Lowee protocol [14].

3.3 Secure Storage

Since a guest VM would write its private data to its VM images, the VM images should be protected. Even though the initial integrity of a guest VM image is guaranteed by the secure connection protocol, encryption and integrity checking processes are essential because a guest VM changes its own images after the VM starts. Moreover, such cryptographic processes should be isolated from a privileged domain. In the proposed system, TVMM is in charge of such cryptographic processes. However, the privileged domain reads encrypted data blocks from a storage device, and writes encrypted data blocks to the storage device, because the privileged domain has a storage device driver. As cryptographic operations are done in TVMM and the cryptographic keys are delivered via the security-critical interface from TCM, the privileged domain can access neither the cryptographic keys nor the cryptographic environment.

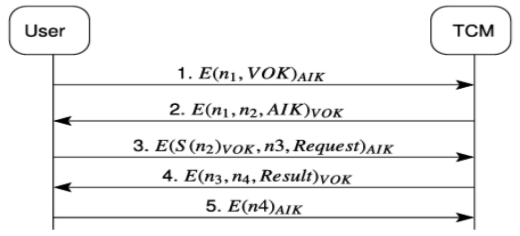

User
TCM
1. $E(n_1, VOK)_{AIK}$
2. $E(n_1, n_2, AIK)_{VOK}$
3. $E(S(n_2)_{VOK}, n3, Request)_{AIK}$
4. $E(n_3, n_4, Result)_{VOK}$
5. $E(n4)_{AIK}$

Figure.5. Secure Management

Even though cloud administrators are able to violate the integrity of VM images or to rollback encrypted VM images, they cannot manipulate the integrity data in VSC. Since the integrity data are protected by TCM and TVMM, they cannot be accessed via management interface. In the worst-case scenario, cloud administrators would boot a cloud node with a malicious VMM and try to access integrity data via the security-critical interface. However, the function to access integrity data is unavailable as far as the system states are not verified with the entrusted attestation scheme. Accordingly, with this approach, cloud administrators cannot alter integrity data.

3.4 Secure Management

The proposed architecture provides a secure management scheme guaranteeing cloud administrators cannot change a VM state arbitrarily. In native cloud architecture, cloud administrators can start guest VMs arbitrarily and can ignore a shutdown or delete operation requested by a cloud user. With such manipulated management operations, a cloud user could have an illusion that the allocated VM has stopped or been deleted while the VM is actually still running. Since the user would not maintain the OS of the VM anymore, the OS is vulnerable, especially to zero-day attacks. To make matters worse, a cloud user cannot get any guarantees that the data in VM are erased completely, even by a VM delete operation, and thus, the VM with private data of the cloud user would remain in the cloud server forever. To prevent illegal management operations from cloud administrators, a management operation should be performed not by a cloud administrator but by cloud users. The users also should check the real VM state. To achieve this goal, the proposed cloud architecture delegates a management privilege to a cloud user, and provides a way to check the result of a management operation by the cloud user. Thus, we present a secure management protocol as described in Fig. 5. A cloud user has a VM Owner Key (VOK) to show the ownership of a VM. The VOK is set in VSC by the secure connection protocol. Once the VOK is set, a management operation can be performed

3.5 SOFTWARE DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software. The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate an employee's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective.

3.5.1 SECURITY SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls.

CLIENT SIDE VALIDATION

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

- ➢ VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
- ➢ Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
- ➢ Tab-indexes are set according to the need and taking into account the ease of user while working with the system in Fig.1.

SERVER SIDE VALIDATION

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted.

Some of the server side checks imposed is

- ➢ Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
- ➢ User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
- ➢ Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User-name, passwords and permissions are controlled o the server sidein Fig 1..
- ➢ Using server side validation, constraints on several restricted operations are imposed.

4. IMPLEMENTATION

A *module* is a bounded contiguous group of statements having a single name and that can be treated as a unit. In other words, a single block in a pile of blocks.

Guidelines for Modularity

- ➢ Make sure modules perform a single task, have a single entry point, and have a single exit point. Isolate input-output (I-O) routines into a small number of standard modules that can be shared system-wide.
- ➢ Isolate system-dependent functions (e.g., getting date or time) in the application to ease possible future conversions to other computer platforms or to accommodate future operating system revisions.

4.1Modules

4.1.1) Consignment Module

Admin should commerce the auction once the auction is started then users who are all interested in the bidding can bid in the particular time period .Admin should upload the bid .The bidding offers will added to the home page of the webpage .Admin should upload the full details about the bidding offers

4.1.2) Put On Trail Module

Clients who are all interested in the bidding can apply for the bidding. For that the client should be a member for some organization they have separate accounts respectively .They have to register in the organization .if client interested in the bidding they have to login into their valid account an then update the rate what they wish for the current offer.

4.1.3) Confirmatory module

This Module plays an vital role in bidding once the client updates the offers their details should be encrypted for the secure bidding this should prevent the bidding rate which claimed by the single client .every clients details is encrypted in storage .people who are all applied for the bid details is encrypted using AES algorithm this algorithm used to encrypt the data in 128 bit which is most efficient way for the security purpose

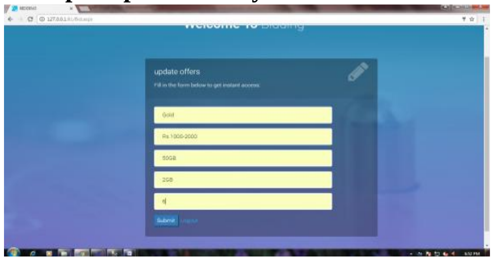
4.1.4) Cue Contemporaries Module

Encrypting the client details will helpful but they have to find the which user claimed for the highest bid rate .To know this for each client separate private key is generated and stored in the cloud storage once the client entered their details the data will be encrypted and the private key is generated and send to the admin via email

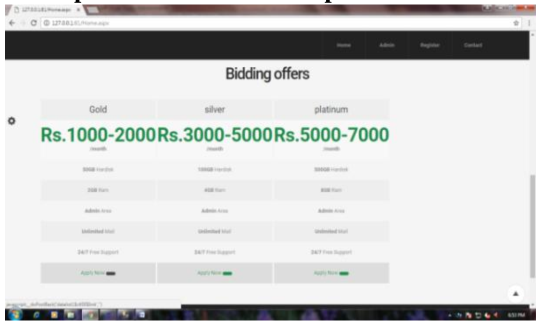
4.1.5) Decipherment Module

Once the bid time is completed the every clients details in storage in encrypted format to view this admin have the private key with them using private key they have to decrypt the details of the client if the private key is matched then the details of the client who claimed the offer for higher rate will be decrypted and then showed to the admin who uploaded the particular bidding offers other admin can't see the details of the client who bagged the offers.

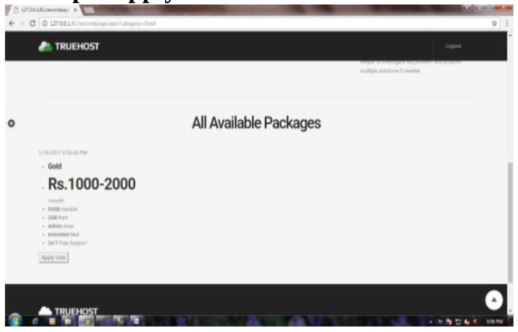
Step1: upload bid by admin



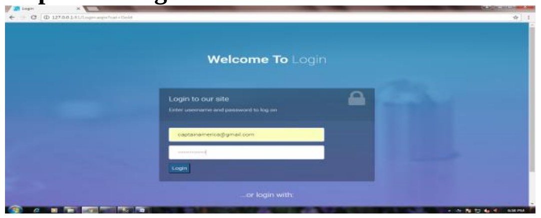
Step2: use can view the upload bid



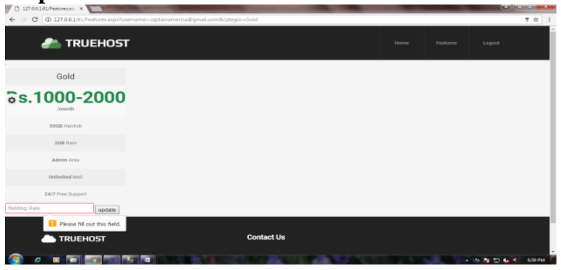
Step3:Apply now button is disable here



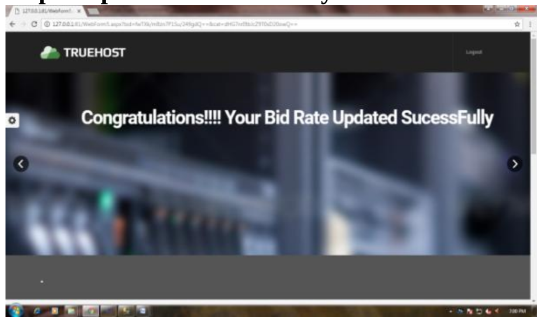
Step4:User login



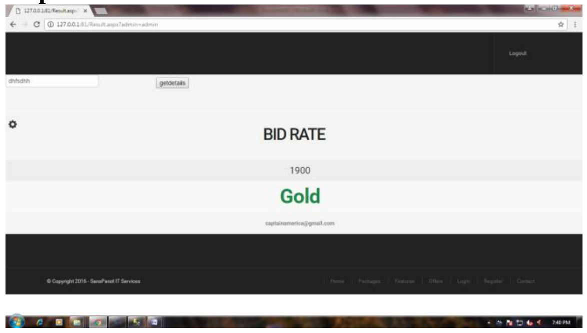
Step 5:



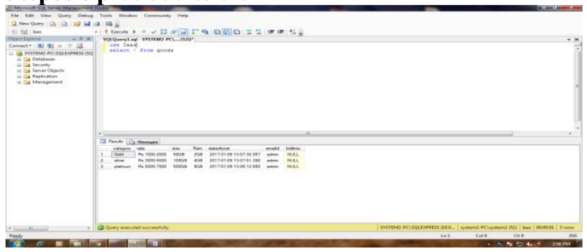
Step6: Uploaded successfully



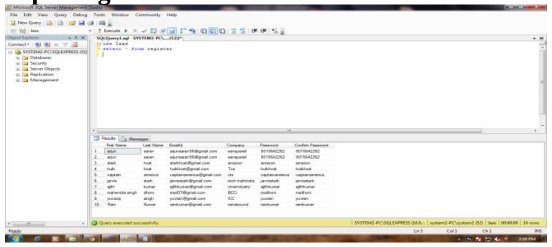
Step7:admin



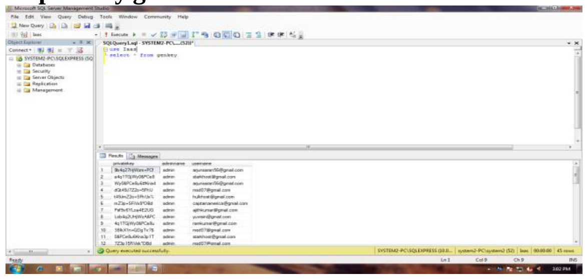
Step8: uploaded table



Step8:Register table



Step9: Key generated table



4.2 Secure Storage

TVMM stores the initial integrity hash of VM images in TCM, and performs cryptographic operations. We modified Xen 4.1.3 for Type I TVMM implementation. Xen has a privileged domain called dom0. Henceforth, dom0 refers to the privileged domain. Here, we describe an initial integrity measuring process followed by cryptographic operations in TVMM.

4.2.1 Initial Integrity of Guest VMs

We built a deployment tool consisting of an application and a kernel module for dom0. Since VM images in a cloud repository exist in unencrypted text, the tool initiates encrypting and measuring processes. The deployment tool sends a VSC creation request to TCM first, and TCM creates VSC with a cryptographic key for cryptographic operations. After loading the first block of VM images, the tool sends a hyper-call to TVMM to encrypt and measure the block. TVMM calculates the hash of the block and encrypts the block. The tool repeats such a process from the first block to the last block of VM images. After measuring the hash of the last block, TVMM stores the initial integrity hash in TCM. Since the tool exists in dom0, the tool can be compromised and caused to function improperly. However, in such a case, the initial integrity is checked by DA and the abnormality is reported to a cloud user with a secure connection protocol. Once the initial integrity is determined to be valid by DA, I/O operations are protected while VM runs as follows.

4.2.2 I/O Operations

In the native architecture, a block-back driver (blkback) exists in dom0, and a block-front driver (blkfront) exists in a guest VM for block I/O operations. When blkfront reads a data block, blkback in dom0 asks VMM to map a memory block of the blkfront, and VMM grants the request if the operation is permitted. After blkback reads the requested block with block I/O operations, blkback asks VMM to unmap the memory block, and VMM unmaps the memory block. In our implementation, TVMM encrypts and decrypts data blocks. Thus, we modified the Xen blkback and blkfront drivers and added hypercalls in TVMM for cryptographic operations. The detailed processes of a read operation are depicted in Fig. 7 Before starting an I/O operation, TVMM gets a cryptographic key from TCM. TVMM also gets the top hash data from TCM with regard to integrity check, and the other integrity data are delivered by dom0. TVMM calculates the top hash of the integrity data delivered by dom0 and compares the top hash data stored in TCM. If the calculated top hash and stored top hash data are the same, the integrity data delivered by dom0 are valid. After validating the integrity data, TVMM manages the integrity data using Merkle tree [17]. During a read operation, blkfront marks a memory block with a sector number to read, and asks blkback to read a block with the sector number. Then, blkback reads an encrypted block and loads the block into the memory of dom0. TVMM decrypts the encrypted block directly to the memory of a guest VM and checks its integrity with the generated Merkel tree. TVMM also checks the requested sector number written by blkfront to prevent dom0 from loading a different block. Finally, blkfront reads the decrypted block from the memory. During a write operation, all operations are done in reverse order, except for updating integrity data in the Merkel tree and encrypting the data. read operation, blkfront marks a memory block with a sector number to read, and asks blkback to read a block with the sector number. Then, blkback reads an encrypted block and loads the block into the memory of dom0. TVMM decrypts the encrypted block directly to the memory of a guest VM and checks its integrity with the generated Merkel tree. TVMM also checks the requested sector number written by blkfront to prevent dom0 from loading a different block. Finally, blkfront reads the decrypted block from the memory. During a write operation, all operations are done in reverse order, except for updating integrity data in the Merkel tree and encrypting the data.

4.3 Dual Interface

TCM has two different memory-mapped regions for management and security-critical interfaces. Before starting the dom0 construction process, TVMM reserves a memory region for the security-critical interface, and thus, dom0 cannot access the region.

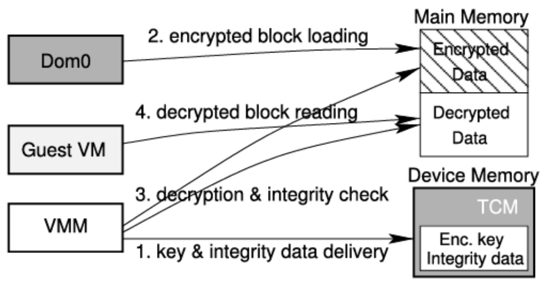

Main Memory
Encrypted Data
Decrypted Data
Device Memory
TCM
Enc. key
Integrity data
Dom0
Guest VM
VMM
2. encrypted block loading
4. decrypted block reading
3. decryption & integrity check
1. key & integrity data delivery

Figure.7. Encryption

The reserved region is only accessible by TVMM. On the other hand, accessing a memory region for the management interface from dom0 is granted by TVMM, allowing dom0 to access the management interface only as an ordinary PCI device.

4.4 Device Authority and Protocol Delivery

We implemented DA with a simple Python [18] web-service daemon. Cryptographic functions such as a signing and a verifying signature were implemented using the Openssl library [15] and TrouSerS [19]. Since TVMM and TCM do not have device drivers for network communication, protocols are initiated by dom0. Dom0 communicates to TVMM via hypercalls and delivers packets to DA via HTTP protocol. Such a process is performed by shell scripts in dom0. Since the protocols are verified as explained in the next section, dom0 cannot compromise the protocols. Instead, dom0 is able to deny to initiate the protocols or to deliver packets to DA. In such cases, TVMM cannot launch guest VMs because TCM denies any access to VSC, and thus, cannot get a cryptographic key. Therefore, even though a compromised dom0 could postpone starting guest VMs temporarily, it cannot harm the confidentiality and integrity of guest VMs.

5. DISCUSSION

In this section, we analyze the security provided by the proposed system and discuss the correctness of the proposed protocols with a verification tool. Afterwards, we discuss other issues such as I/O model analysis and management issues, and finally, we compare the proposed system to prior work. Other issues such as I/O model analysis and management issues, and finally, we compare the proposed system to prior work.

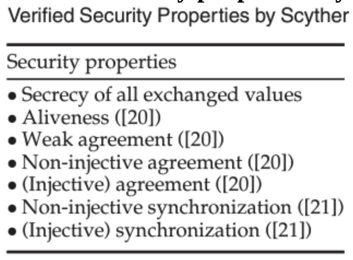
5.1Security Analysis

The proposed system protects three aspects of VM security:a connection between a cloud user and an allocated VM, the security of guest VM images, and a VM management operation. First, the connection is secured with the secure connection scheme. The connection keys shared between a cloud user and an allocated VM are exchanged via the secure connection scheme. Since the secure connection scheme exchanges only the public part of keys and the private part of a user key is confined, cloud administrators cannot login to guest VMs illegally. Therefore, the proposed system defends against A6. A cloud user can also verify the initial integrity of an allocated VM and the platform integrity on which the allocated VM runs via the secure connection scheme. Therefore,the proposed system defends against A4andA5. Second, guest VM images are secured by isolating the cryptographic environment from dom0. During the domain building process, dom0 loads initial boot code for a guest VM. After checking the integrity of the boot code [11], TVMM prevents dom0 from accessing the memory of the guest VM [3]. Since cryptographic operations are performed in TVMM and dom0 cannot compromise TVMM, the proposed system defends against A1. Moreover, since cryptographic keys and integrity data are isolated from cloud administrators by TCM, the proposed system defends against A2 and A3. Finally, a VM management operation is secured by a secure management scheme. With the proposed scheme, a VM management operation is triggered by a cloud user rather than a cloud administrator, and the result of the management operation is securely reported to the cloud user. Thus, the proposed system defends against A7. A limitation of this study is that the proposed system does not guarantee the availability of guest VMs. With the privileges of cloud administrators, the proposed protocols can be disabled by deleting executable files in dom0 or reconfiguring the firewall to block the network connectivity between cloud users and cloud nodes. Such denial-of-service (DoS) attacks cannot be prevented by the proposed system. However, the use of a cloud monitoring system can mitigate the problems. For examples, a cloud monitor system could report the availability of guest VMs to cloud providers and cloud users routinely. However, since building such a system is out of scope of this paper, the details are not discussed here.

5.2 Protocol Verification

We propose the use of three protocols, namely, entrusted attestation, secure connection, and secure management. When a protocol has a defect, a malicious agent (e.g., intruder) is able to take control of communication and thus the security of guest VMs cannot be guaranteed. To check whether a protocol is designed correctly, a number of automatic protocol verification tools have been proposed (e.g., AVISPA [22], Scyther [23], Casper [24], and Proverif [25]). To verify a protocol, such tools perform a formal verification process and report whether the protocol is designed correctly. Among the various verification tools, we chose Scyther [23] because it has been widely used in recent work [26], [27], [28], and is able to verify a strong security property such as synchronization [21]. Specifically, we verified the security properties in Table.

Table. Verified security properties by scyther



Verified Security Properties by Scyther

Security properties
• Secrecy of all exchanged values
• Aliveness ([20])
• Weak agreement ([20])
• Non-injective agreement ([20])
• (Injective) agreement ([20])
• Non-injective synchronization ([21])
• (Injective) synchronization ([21])

We attached the protocol descriptions for Scyther in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/ 10.1109/TSC.2015 .2392099. The protocols were verified by Scyther 1.1.3 with an unbounded number of runs. For the secure connection protocol, we simplified the protocol description. Since Step 5, Step 6, Step 7 and Step 8 in Fig. 4 are performed within TCB, such steps were removed with the Scyther description. 5.3 I/O Model Analysis To analyze potential threats, we describe the detailed I/O operation process first. The detailed I/O operation processes between the native and our proposed model are different, as depicted Fig. 8. In the native model, when the OS of a guest VM wants to read or write data blocks, bulk front of the OS marks a grant table showing that the I/O operations are permitted in the marked pages ,and sends I/O requeststodom0. When dom0 receives the requests, dom0 maps the memory pages of the guest VM to the memory of dom0. During mapping operations, physical-to-machine (P2M) and machine-to physical (M2P) tables are updated to enable dom0 to access the page of guest VMs. Afterwards, dom0 performs I/O operations with the mapped pages, and unmaps the mapped pages by updating P2M and M2P tables, to prevent dom0 from persistently accessing the memory. In the proposed architecture, I/O operations are performed as follows. When a guest OS wants to read or write data blocks, blkfront of the OS marks a grant table with a sector number showing that the I/O operations are permitted with the specified sector number, and sends the I/O requests to dom0. In dom0, blkback handles read and write operations differently. When a guest OS read data blocks, dom0 loads the encrypted blocks to the memory of dom0 via block read operations, and TVMM directly decrypts the loaded blocks to the memory of the guest VM. When a guest OS writes data blocks, TVMM directly encrypts the requested blocks to the memory of dom0, and dom0 performs block write operations with the encrypted blocks. As a result, dom0 could not access the plain (decrypted) I/O blocks. When VMM encrypts and decrypts, VMM also checks the permission of the requests as in the native architecture, and additionally checks the specified

6.ALGORITHMS

In the proposed system the AES algorithm used to transfer the private key for encryption and decryption via email. A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow. The features of AES are as follows −Symmetric key symmetric block cipher128-bit data, 128/192/256-bit keys, Stronger and faster than Triple-DES Provide full specification and design details, Software implementable in C and Java

Pseudo code for AES:

```
Cipher(byte in[16], byte out[16], key_array round_key[Nr+1])
begin
 byte state[16];
state = in;
AddRoundKey(state, round_key[0]);
for i = 1 to Nr-1 stepsize 1
do
SubBytes(state);
ShiftRows(state);
MixColumns(state);
AddRoundKey(state, round_key[ i]);
 end for
```

SubBytes(state);
ShiftRows(state);
AddRoundKey(state, round_key[Nr]);
end

Steps Involved:
6.1. Key Selection: The sender and receiver agree upon a 128 bit key. This key is used for encryption and decryption of images. It is a symmetric key encryption technique, so they must share this key in a secure manner. The key is
represented as blocks k[0],k[1]...k[15]. Where each block is 8bits long (8*16=128 bits).

6.2. Generation of Multiple keys: The sender and receiver can now independently generate the keys required for the process using the above explained Modified AES Key Expansion technique. This is a onetime process; these expanded keys can be used for future communications any number of times till they change their initial key value.

6.3. Encryption: Encryption is done in spans, where we process 16 pixels in each span. For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: Sub Bytes, Shift Rows, Mix Columns and Add Round Key.

6.4. Decryption: The decryption process is similar as encryption, but we use Inverse Sub Byte Transformation. The whole AES structure is sketched in Figure 8**.**

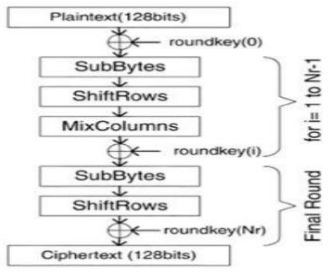


Figure.8. AES Algorithm
7. SYSTEM FEATURES

7.1DESCRIPTION AND PRIORITY
The proffering secured cloud storage with safety using scrambled technique system maintains information on details of Bidding information's. Of course this paper has high priority because Admins can view the details of Bidding Rates. Some admin may leak the bidding rates for their own profit to overcome this we introduced an scrambled technique each admin have separate private key using these key only they can access their own data so other admins can't view the data.

7.2 STIMULUS/RESPONSE SEQUENCES
Search for Bidding displays the available bidding offers and make a bid value with corresponding rate after bidding they value is uploaded into database

7.3 DATABASE
Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network. They have client sites and others are server sites. All data resides at the server sites. All applications execute at the client sites. The term client/server refers primarily to an architecture, or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).

7.4 SOFTWARE INTERFACES
Operating system-We have chosen Windows XP operating system for its best support.
Database-To save the flight records, passengers records we have chosen SQL+ database.
Asp.Net-To implement the project we have chosen Vb.Net language for its more interactive support.

7.5 **COMMUNICATION INTERFACES**
This paper supports all types of web browsers. We are using simple electronic forms for the application and displaying purposes etc.

8. PERFORMANCE REQUIREMENTS

8.1 Overall Description
This section describes various parts of the system and aims to convey a general understanding of the system and its requirements.

8.2 Communications Interfaces
The most obvious communications interface that this system relies on is the internet, the World Wide Web is what the user interface is based around, which relies on the TCP/IP protocol. Other types of servers also support the system. An FTP server for example is used to upload content to the website, and a database server used to store user information. Web pages hosted on the flame web server are capable of communicating with these other servers, this is extremely important in the case of the database server, because the system will be saving to and retrieving from the database frequently.

8.3 Memory Constraints
Secondary storage constraints – The storage constraints imposed by the web servers being used will not affect the operation of the system as there is ample storage for the system's needs. The secondary storage available in the oracle database may be a constraint if the system is used frequently and the database becomes large, this factor is beyond the control of the software developers. Secondary storage uses for the system on the end users machine will be very minimal, only a small amount of data will be stored on the local machine, so this is not a concern for the system. Primary storage constraints – All servers being used should have more than enough primary memory to cope with the resources that the system will require. The users machine will need to have sufficient primary memory to load the pages, any relatively modern system should be adequate.

8.4 Operations
Normal operations of the system include various users accessing the system at any one time, and thousands of queries being conducted by users on the database every day. Maintenance procedures will take place each day, such as database and website backups and tests. These procedures will be run during a non-peak usage time in order to reduce the possibility of using excess resources and slowing the system down. Only in extreme

cases will the web server need to be taken offline in order to perform this operation.

8.5. ASP.NET and ASP.NET AJAX

ASP.NET is the web development model and AJAX is an extension of ASP.NET for developing and implementing AJAX functionality. ASP.NET AJAX contains the components that allow the developer to update data on a website without a complete reload of the page.

8.6. ADO.NET

It is the technology used for working with data and databases. It provides access to data sources like SQL server, OLE DB, XML etc The ADO.NET allows connection to data sources for retrieving, manipulating, and updatingdatatng data.

9. CONCLUSION

In modern technology also security is consideration for the cloud. Even though the security threats by cloud administration are feasible and critical cloud service providers are mainly consideration with security threate from external attacks rather than internal attacks. the viewpoint hinders the proliferation of cloud computing inthis paper we presented a cloud system architecture consisiting of private key and those key are stored in separately, the data of cloud users are protected eveb with a compromised privilieged domain or malicious cloud administration .the proposed system providers security functionalities against wide attack vector and achieves a small TCB.it shows a reasonable I/O performance, proving its feasibility.

10. FUTURE ENHANCEMENT

In our paper we have stored private key separately in the cloud due to insecure environment the keys can be hacked to prevent these once the key generated and send to the mail then the key is cleared from the cloud this make way to more efficient for the security purposes .

11. REFRENCES

[1]. D. G. Murray, G. Milos, and S. Hand, "Improving xen security through disaggregation," in Proc. 4th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ., 2008, pp. 151–160.

[2]. N. Santos, K. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in Proc. Conf. Hot Topics Cloud Comput., article 3, 2009.

[3]. L. Chunxiao, A. Raghunathan, and N. Jha, "Secure virtual machine execution under an untrusted management OS," in Proc. IEEE 3rd Int. Conf. Cloud Comput., 2010, pp. 172–179.

[4]. S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy, "Self-service cloud computing," in Proc. ACM Conf. Comput. Commun. Security, 2012, pp. 253–264.

[5]. J. Kong, "Protecting the confidentiality of virtual machines against untrusted host," in Proc. Int. Symp. Intell. Inf. Process. Trusted Comput., 2010, pp. 364–368.

[6]. TCG architecture overview, version 1.4. [Online]. Available: http://www.trustedcomputinggroup.org/resources/

[7]. B. D. Payne, M. D. de Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in Proc. 23rd Annu. Comput. Security Appl. Conf., 2007, pp. 385–397.

[8]. C. Li, A. Raghunathan, and N. K. Jha, "A trusted virtual machine in an untrusted management environment," IEEE Trans. Serv. Comput., vol. 5, no. 4, pp. 472–483, Fourth Quarter 2012.

[9]. Y. Xia, Y. Liu, H. Chen, and B. Zang, "Defending against VM rollback attack," in Proc. 42nd IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops, 2012, pp. 1–5.

[10]. Amazon elastic compute cloud (EC2). [Online]. Available: http:// aws.amazon.com/ec2/

[11]. J. Choi, J. Park, J. Seol, and S. Maeng, "Isolated mini-domain for trusted cloud computing," in Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., 2013, pp. 194–195.

[12]. A. Baldwin, C. Dalton, S. Shiu, K. Kostienko, and Q. Rajpoot, "Providing secure services for a virtual infrastructure," ACM SIGOPS Oper. Syst. Rev., vol. 43, no. 1, pp. 44–51, Jan. 2009.

[13]. A. J. Baldwin and C. I. Dalton, "Trusted key management for virtualized platforms," US Patent Appl. 12/242,104., Sep., 2008.

[14]. G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using FDR," in Proc. 2nd Int. Workshop Tools Algorithms Construction Anal. Syst., 1996, pp. 147–166.
[15]. Welcome to the OpenSSL project. [Online]. Available: http:// www.openssl.org/

[16]. Software-based TPM emulator. [Online]. Available: http:// sourceforge.net/projects/tpm-emulator.berlios/

[17]. R. C. Merkle, "A digital signature based on a conventional encryption function," in Proc. Conf. Theory Appl. Cryptographic Techn. Adv. Cryptol., 1988, pp. 369–378.

[18]. Python. [Online]. Available: https://www.python.org/ [19] TrouSerS: The open-source TCG software stack. [Online]. Available: http://trousers.sourceforge.net/ [20] G. Lowe, "A hierarchy of authentication specifications," in Proc. 10th IEEE Workshop Comput. Secur. Found., 1997, pp. 31–43.

[19]. IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 9, NO. 3, MAY/JUNE 2016 ATrusted IaaS Environment with Hardware Security Module Jinho Seol, Student Member, IEEE, Seongwook Jin, Student Member, IEEE, Daewoo Lee, Jaehyuk Huh, Member, IEEE, and Seungryoul Maeng