2-1-2015

Using trusted platform module for securing virtual environment access in cloud

Asser Sherif

The American University in Cairo

School of Sciences and Engineering

Using Trusted Platform Module for securing virtual environment access in Cloud

A Thesis Submitted to

Department of Computer Science and Engineering

in partial fulfillment of the requirements for

the degree of Master of Science

by Asser Sherif

under the supervision of Professor. Amr El-Kadi

December/2014

ACKNOWLEDGMENTS

I would like to thank my Thesis supervisor, Professor Amr El-Kadi for all his encouragement, patience and support who without his guidance and motivation I would have not been able to accomplish this work.

I would also like to thank my Thesis supervision committee members, Professor Sherif El-Kassas, Professor Sherif Gamal Ali and Professor Ali Fahmy for their support and insightful comments.

Thanks also go to my Family and friends who supported me throughout this long period and encouraged me all the time to get it done.

ABSTRACT

The American University in Cairo, Department of Computer Science and Engineering

Using Trusted Platform Module for Securing Virtual Environment Access in Cloud

Asser Sherif

Supervisor: Prof. Amr El-Kadi

Keywords: Trusted Platform Module (TPM), Machine Authentication, Virtualization Security, Multi-Tenancy, Malicious Insiders, Machine Identity

With the increasing usage of Cloud and the Virtualization technology, there comes also an increasing demand to ensure the security levels of all computing environments and components associated and accordingly in this work we propose a new machine authentication mechanism using Trusted Platform Module that can be used to provide a secure access to virtual environments in the cloud.

The proposed authentication module is aiming to contribute in providing a solution to Poor machine identity, Multi-tenancy as well as Malicious insiders known security problems in the cloud. It is targeting the access security to graphical user interface of virtual machines hosted on VirtualBox hypervisor in a Linux based environment through authenticating clients trying to connect using the client's Trusted Platform Module Public Endorsement key as a pre-authorized signature to the virtual environment in addition to the normal user name and password authentication of the connecting user.

Results obtained from the output of this work indicates that it is possible to authenticate the machines based on their Trusted Platform Module signatures and provide them access to VirtualBox environment only based on a pre-defined Access Control List with minimal one time overhead upon establishing the initial connection.

LIST OF ACRONYMS

ACL	Access Control List
AIK	Attestation Identity Key
CA	Certificate Authority
DAA	Direct Anonymous Attestation
EK	Endorsement Key
LTS	Long Term Support
PAM	Pluggable Authentication Module
PCA	Privacy Certificate Authority
PCR	Platform Configuration Register
RDP	Remote Desktop Protocol
SCP	Secure Copy
SRK	Storage Root Key
TCG	Trusted Computing Group
TNC	Trusted Network Connect
TPM	Trusted Platform Module
UUID	Universal Unique Identifier
VM	Virtual Machine
VMM	Virtual Machine Monitor
VRDP	VirtualBox Remote Desktop Protocol

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 BACKGROUND

In this chapter, we provide a background review on Cloud computing as well as Virtualization Technology with an overview on the security challenges related to the focus of this work.

We also provide an overview on the Trusted Platform Module and its components in addition to some related research activities conducted using TPM. We conclude this chapter with a review on the future directions of using TPM.

1.1 CLOUD COMPUTING

The term "Cloud Computing" has been used widely since 2008 to define a technology where users can access infrastructure and applications hosted remotely on network-enabled servers viewed as a pool of shared resources having the following characteristics [3],[4]-[6]:

- *Multi-tenancy:* based on the model of shared resources, computing environments are no longer dedicated to a single user but are shared across multiple users and sometimes organizations to provide access to offered services
- *Elasticity:* where the users can control the computing resources being used and increase or decrease them based on the current need
- *Scalability:* where cloud service providers have the ability to scale up their offered resources to a much higher level than what organizations can offer on a locally hosted data centers
- *On-Demand resources provisioning:* where users have the capability of self-provisioning their computing resources needs, resources are monitored and controlled to ensure optimization
- *Pay as you go:* where users or organizations will only pay for the actual resources utilization on the time they are used for, this is performed using the cloud metering capabilities for each offered service

Virtualization technology is a key driver for cloud computing as it provides the ability to pool resources and easily administer and monitor their utilization. We will be discussing this in more details in section 2.2.

1.1.1 Cloud Computing Service Models

Cloud Computing consists of three main service models upon which the offered cloud service depends. The services offered at a higher level depend on the services offered on the underlying levels of this service model [3]-[5].

Figure (1) illustrates the main three service models of the cloud-computing environment [7].

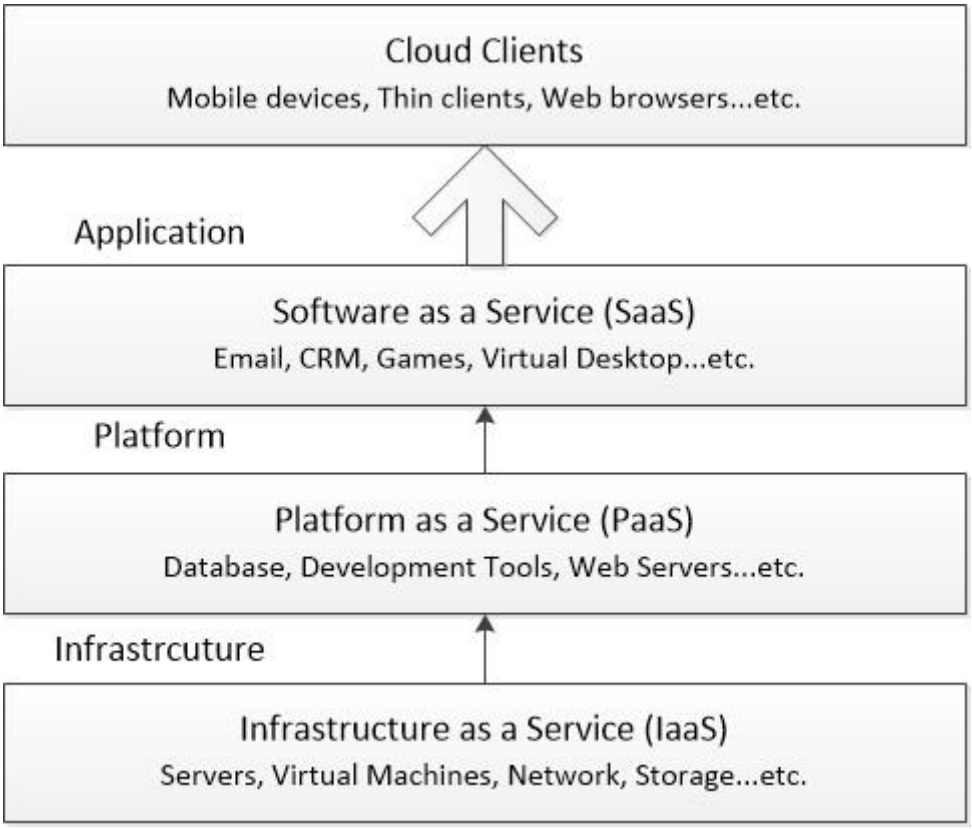


Figure 1 Cloud Computing Service Models

- Infrastructure as a Service (IaaS)

The underlying layer of the cloud computing service model where cloud service providers provide processing, storage, memory as well as networking resources to users and is mostly in a virtualized environment.

- Platform as a Service (PaaS)

The middle layer of the cloud computing service model where cloud service providers provide users (mostly developers) with the required platforms to be capable of deploying and managing applications using programming languages or tools that are offered in the cloud without in depth knowledge of the underlying infrastructure environment.

- Software as a Service (SaaS)

The upper layer of the cloud computing service model where cloud service providers provide users with the capability of executing applications without any knowledge of the development platforms or the infrastructure environment being used to host these applications. User can access this layer from any cloud client (Mobile devices, Browsers, Thin Clients or Terminal emulators).

1.1.2 Cloud Computing Deployment Models

Cloud computing has several Deployment Models [3]-[5], as illustrated in Figure (2) [7]:

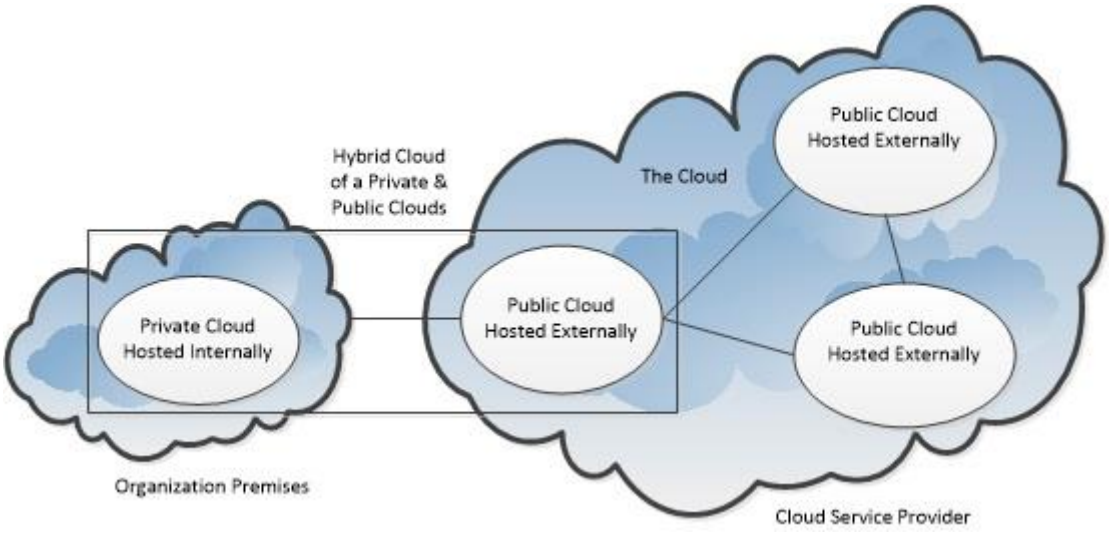


Figure 2 Cloud Computing Deployment Models

- Public Cloud

Cloud service providers offer cloud resources for public use servicing multiple tenants. The services are offered free of charge or using a payment model which is usually based on pay as you go calculations. The cloud resources are hosted in data centers that are owned and managed by the cloud service providers and are accessed over the internet and accordingly requires special security considerations.

- Private Cloud

Cloud resources that are accessed only by a single organization servicing multiple business units for this organization. The computing environment is usually hosted internally inside this organization's premises or sometimes externally at a service provider data center with a dedicated, secured communication channel only for the hosted organization. It can be managed either internally or using a third party.

- Hybrid Cloud

The formation of a cloud computing environment by combining two or more distinct clouds from any deployment model (Public or Private). This offers more flexibility in terms of scalability for organizations as they can expand their computing resources – even on temporary basis if needed - beyond their private owned cloud by gaining access to a public cloud environment.

- Community Cloud

Cloud resources that are accessed by multiple organizations that share a common interest or together are forming a community serving a particular concern. The computing environment is hosted internally in one or more of these organizations or externally at a service provider data center and it can be managed either internally from within the community or using a third party.

1.2 VIRTUALIZATION TECHNOLOGY

Virtualization technology is a key driver for Cloud computing where it depends on de-coupling the computing resources from the applications running on top of them including the operating systems. This concept allows for the creation of multiple virtual machines on a single physical machine and benefiting from a pooled resources model to ensure efficient utilization [5].

Virtualization is also a key enabler for multi-tenancy in the cloud; it provides the capability of hosting multiple computing environments for several organizations on shared computing resources. It also provides organizations with a possibility of data centers consolidation for a better utilization of existing resources as well as cost saving opportunities in the future [5], [8].

There are different types of Virtualization models (Hardware, Software, Memory, Storage and Network). In the scope of this work, we will be focusing on Hardware Virtualization where "Host" is used to refer to the physical machine providing the resources to create virtual machines while "Guest" would be used to refer to the virtual machine running on top of the physical host machine. "Hypervisor" or "Virtual Machine Monitor (VMM)" is software that is used to create and manage virtual machines on the physical host [7], [8].

There are different levels of Hardware Virtualization as follows [7], [9]:

- *Full Virtualization*: A complete emulation of all the host hardware is available to each guest machine

- *Partial Virtualization*: Some of the host hardware is provided to the guest machines through emulation but not everything, accordingly some of the guest programs will need to be modified to run in this model

- *Paravirtualization:* Minimal hardware emulation, multiple guest machines can run at the same time on the host hardware in isolated domains but it requires modification of the guest operating systems to work efficiently with the Hypervisor

In addition, as illustrated in Figure (3), there are different types of Hypervisors [9]:

- *Type 1 (or native, bare metal):* Hypervisors will run directly on the host machine hardware to manage the guest machines and control the access to the host hardware resources

- *Type 2 (or hosted):* Hypervisors will run from within an operating system environment on the host machine as a second software level. The guest operating systems in this case will run at a third level above the hardware

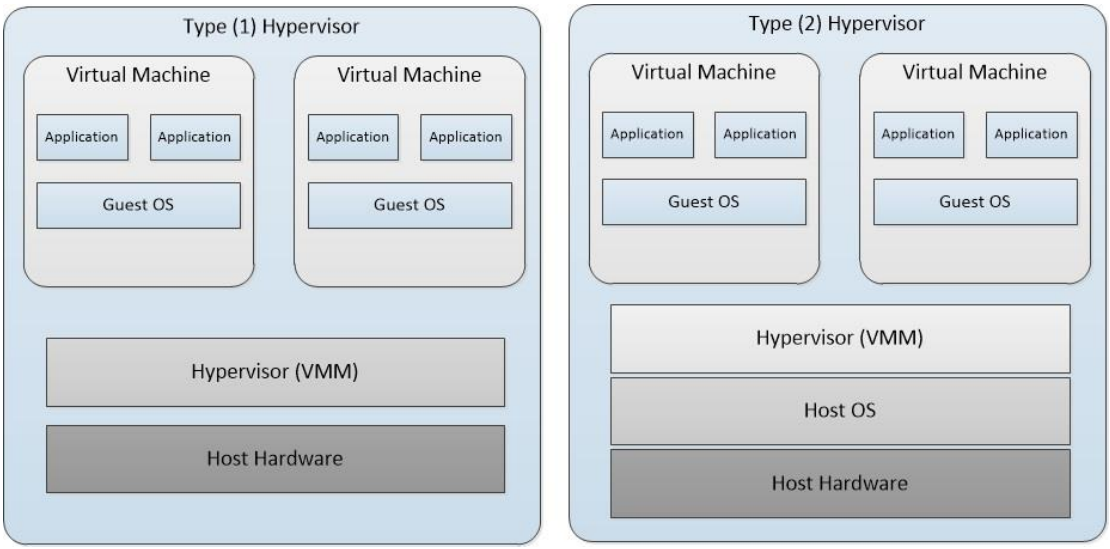


Figure 3 Hypervisor Types

1.3 IN-SCOPE SECURITY CHALLENGES

As with the case of traditional IT environments, cloud computing and virtualization technology usage also have their share of security issues and threats that change over time with the adoption of new technologies. The cloud and virtualization security is a very broad topic with many research studies conducted to cover each aspect of it. In this section, we will provide an overview on the security challenges related to the scope of this work.

1.3.1 Identity Management

Identity Management in cloud computing is an attempt to clearly answer the question of who is really using and offering the provided service. This spans several levels starting from the user id, the connected client devices and the service provider as well as any in between attributes and transactions to ensure the complete solution integrity [10].

Identity management from a user perspective – sometimes referenced as Identity Access Management - is a widely researched topic with focus on the digital identity of the user. It is mainly targeting how to authenticate and authorize the users before they access any service according to their defined roles or privileges.

Our focus in this work would be focusing on the identity of the client machines used by end users to access a virtual environment in the cloud. This will be associated with the normal user authentication processes to establish a user – machine complete authentication mechanism before the access to a virtual machine is established.

1.3.2 Multi-tenancy

CSA[1] defined Multi-tenancy as follows:

"Multi-tenancy in its simplest form implies use of same resources or application by multiple consumers that may belong to same organization or different organization. The impact of multi-tenancy is visibility of residual data or trace of operations by other user or tenant. Multi-tenancy in cloud service models implies a need for policy-driven enforcement, segmentation, isolation, governance, service levels, and chargeback/billing models for different consumer constituencies. Consumers may choose to utilize a public cloud providers' service offering on an individual user basis or, in the instance of private cloud hosting, an organization may segment users as different business units sharing a common infrastructure."

Multi-tenancy is mainly depending on Virtualization technology to enable the sharing of computing resources and at the same time ensures proper utilization. Multi-tenancy can be achieved at different levels of the cloud service models, IaaS, PaaS or SaaS [9].

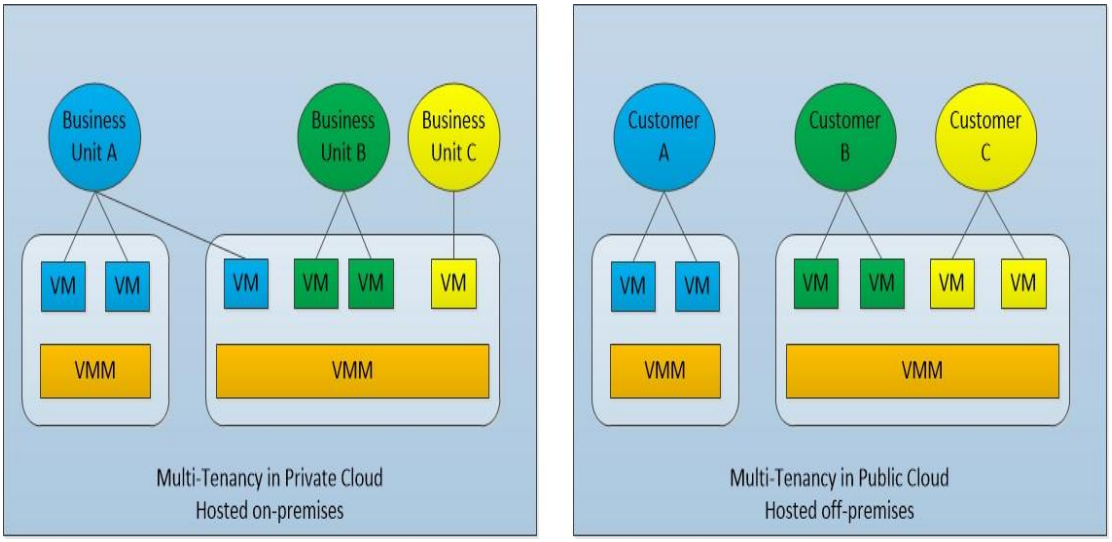


Figure 4 Multi-tenancy in Cloud

Figure (4) above illustrates the Multi-Tenancy concept in Private and Public cloud delivery models [6].

[1] https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

Our focus in this work will be on the Multi-tenancy in the IaaS model and the security related to the Virtualization environment since the co-existence of multiple tenants on the same physical servers requires strong security policies to ensure complete isolation and to protect against attacks.

The Cloud Security Alliance Group listed the Multi-Tenancy security threat being part of the shared technology vulnerabilities as number nine in the Top Nine Cloud Computing Threats in 2013 [11].

1.3.3 Malicious Insiders

The definition of Malicious Insiders as defined by CERT[2] is as follows:

"A malicious insider threat to an organization is a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems."

The threat of an insider using privileged access rights to perform un-authorized operations applies in the cloud environment and specifically in the IaaS service model where system administrators of a cloud service provider can have access to infrastructure as well as data storage and can compromise the security level if no proper security policies are implemented [11].

The Cloud Security Alliance Group listed the Malicious Insiders threat as number six in the Top Nine Cloud Computing Threats in 2013 [11].

1.4 TRUSTED PLATFORM MODULE

"Trusted Platform Module" refers to a specification and its implementation as an embedded cryptographic device which has been designed by the TPM Work Group, a part of the Trusted Computing Group (TCG) [12], [13].

TPM includes the following capabilities [14]-[15]:

- Storing of digital credentials such as passwords in a hardware-based vault
- Augmenting smart cards, fingerprint readers and fobs for multi-factor authentication
- Encrypting files and folders to control access
- Establishing state information to enable endpoint integrity

[2] http://www.cert.org/insider-threat/index.cfm

- Hash state information prior to hard driven shutdown for endpoint integrity
- Enabling more secure VPN, remote and wireless access
- Can be used in conjunction with Full Disk Encryption to restrict access to sensitive data
- Can be used for machine authentication and hardware encryption
- Can be used for signing, secure key storage and attestation

We can use TPM to provide low-volume public key cryptographic services to a machine using a private key stored on the TPM chip itself and to store credentials only to be used when the system is in some specific configuration. Using the TPM in addition to a trusted system BIOS, computers that are TPM enabled can boot into a trusted operating system and can verify the configuration of software that they are running to third parties [12]. Encryption and signing are well-known techniques but TPM can make them stronger by providing the capability of storing the keys in protected hardware storage [14].

One key capability that TPM provides to enable trusted computing is that it can securely store a series of measurements through a set of registers called Platform Configuration Registers (PCRs). A computer with an enabled TPM and trusted BIOS can boot up, take a series of measurements of the hardware and software on this computer and store them in the PCRs. The TPM can then cryptographically sign these PCR measurements and send them to a remote party to verify that the platform with that TPM has been booted up and measured correctly. TPM can also use these PCRs to bind keys to a platform in a specific state so that it may not be allowed to have access to the key if it is in a different state than it was when the key was created [12].

TPM can also be used for Full-Disk Encryption to protect Data using an early authentication mechanism before the OS is loaded [16]. A common implementation of this Encryption capability is used in Microsoft Windows BitLocker [17].

TPM is also used for "Remote Attestation" where third parties can authenticate a machine with enabled TPM remotely through the use of certificates, accordingly the authenticated machine can then be granted access to a cloud service [12].

An important authentication challenge with TPM is that an attacker who knows a legitimate user's password but does not have his physical machine will not be able to use the user account to access the resources he is permitted to access [16].

One limitation of TPM usage in protection is that it only provides a guarantee at software load time. Accordingly, if a legitimate application is loaded and then is modified in memory, the TPM will not be aware of this change. The TPM does not protect against any attacks that can use flaws in currently running software programs [18].

Various vendors manufacture these TPM chips and they are indented to be cheap devices that can be included on motherboards to enable trusted computing. The requirement of keeping the TPM cheap led to low requirements for hardware security, which entails that it might not cover all possible attacks against it [12].

1.4.1 TPM Components

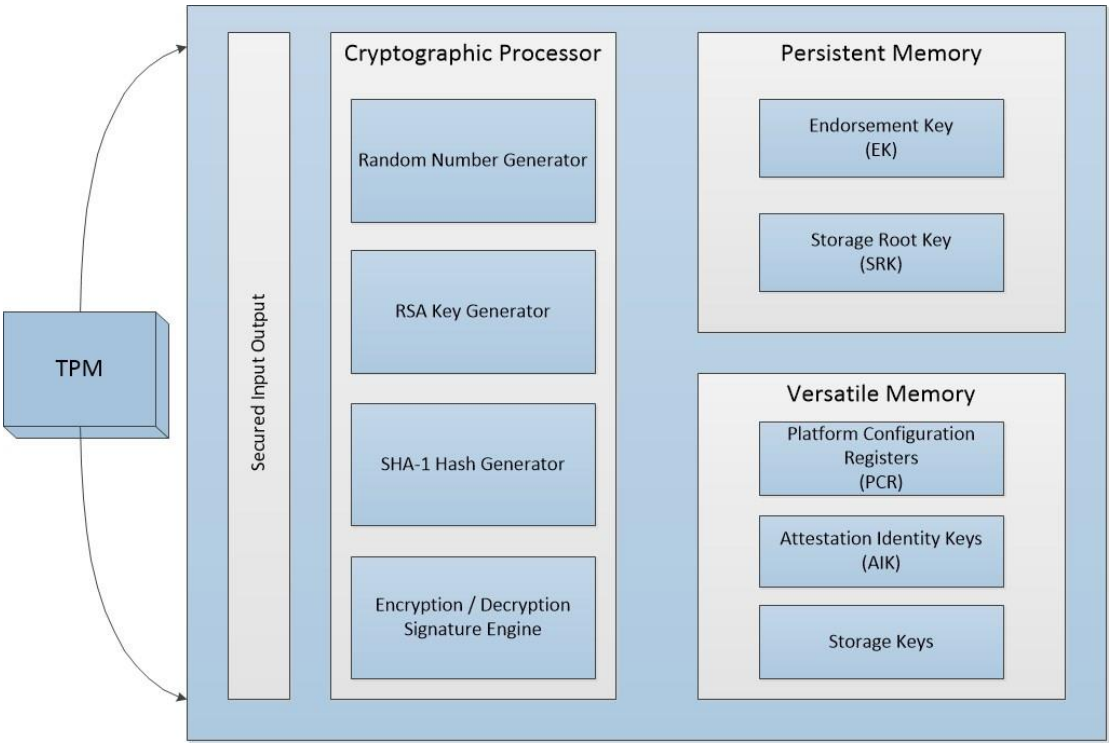


Figure 5 TPM Components

As illustrated in Figure (5) above, a TPM chip contains multiple components listed in details below [5], [13], [20].

- Cryptographic Processor

Has an RSA key generator used to create RSA key pairs, a random number generator, an encryption-decryption engine that provides the necessary functions for encryption and decryption of data on disks or signing PCR values. It also contains a Secure Hash Algorithm (SHA-1) used for hashing values that are stored in the PCR.

- Persistent Memory (Non-volatile Memory (NV))

Since there is a need to preserve certain values even when the TPM has no power, data stored in the persistent memory will stay unchanged when there is no power on the TPM chip.

The Endorsement Key (EK) is a pair of RSA keys that is installed when the TPM is manufactured. The public EK value is used to uniquely identify a TPM chip and will not change during the TPM chip lifetime.

The Storage Root Key (SRK) is also a pair of RSA keys that is used to encrypt other keys stored outside the TPM. SRK is the Root of Trust for Storage and can be changed when a new user takes ownership of the TPM chip on the system.

- Versatile Memory (Volatile Memory)

In contrast to persistent memory, data in the Volatile Memory will be lost (reset) when there is no power or when the computer reboots. Persistent memory limits the number of write accesses, but with versatile memory there is an unlimited number of writes allowed.

Versatile memory contains two main components:

  - Platform Configuration Registers (PCR)

There are at least 16 PCRs in a TPM chip to store platform configuration measurements. These measurements are normally hashed values (SHA-1) of applications running on a platform. PCR does not allow direct write access however data is stored on them using a process called extending the PCR.

  - Attestation Identity Keys (AIK)

To ensure that you are establishing a communication to a valid TPM enabled platform, remote attestation is performed using Endorsement Keys. Some arguments were made that this would uncover the privacy of the TPM owner due to the unique nature of the EK and therefore the commonly used method for remote attestation is done using AIK instead. We will be discussing in more details this argument within the scope of this work.

To summarize, Table (1) below presents the different keys that TPM uses along with their description:

Table 1 TPM Keys

TPM Key	Description
Endorsement Key (EK)	Unique RSA key pair installed during manufacturing and cannot be changed during the TPM lifetime
Storage Root Key (SRK)	RSA key pair used to encrypt other keys stored outside TPM, can be changed when TPM ownership is changed
Attestation Identity Keys (AIK)	Key pair created by TPM to be used for platform attestation instead of EK to protect privacy of the TPM client

1.5 TPM RELATED RESEARCH

In this section, we provide an overview on some of the researches that were conducted in relation to using TPM in access security.

1.5.1 C-MAS: The Cloud Mutual Authentication Scheme

In [21], a scheme is proposed to solve the problem of authentication between users and a cloud environment based on Trusted Computing Technology by the use of smart cards. The proposed mechanism establishes the authentication of both users and the cloud server provider, generates the session key and verifies the credibility of the platform of the cloud service provider.

As illustrated in Figure (6) below, the User, the TPM enabled cloud server the Certificate Authority can access each other over the internet. The user first register on the cloud server platform then uses his smart card to access the cloud server through his client once a valid attestation is performed using the certificate authority.

The certificate authority role is to manage the TPM public key of the cloud server that indicates its identity. The user, after being identified using his smart card communicate with the certificate authority and test the authenticity of the cloud service provider using the TPM public key [21].

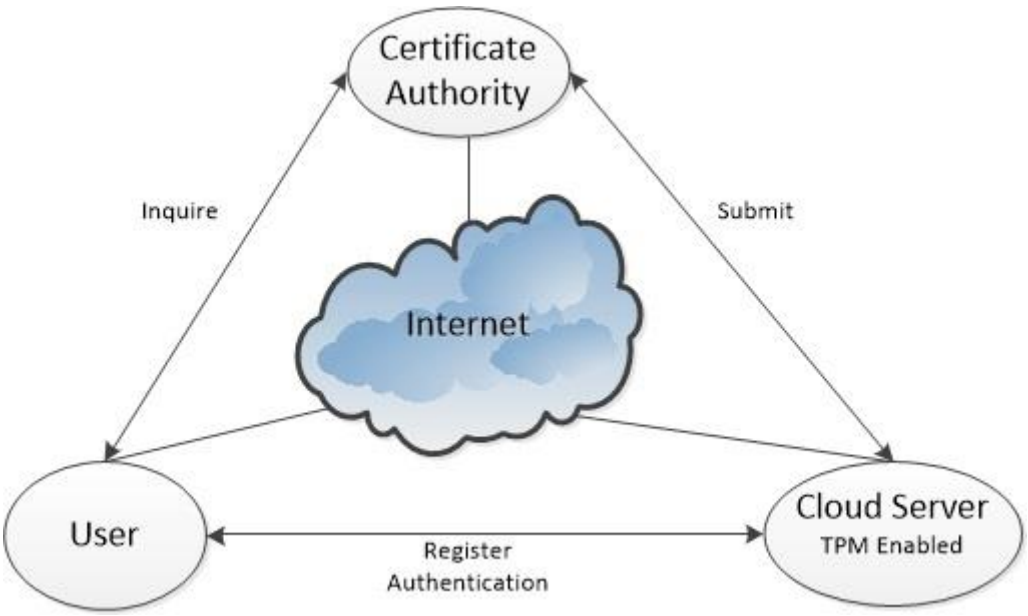


Figure 6 Cloud Mutual Authentication Scheme

This mechanism is used to verify the identity and integrity of the platform of the cloud service provider itself, but not the identity of the connected clients.

1.5.2 Enabling Trusted Distributed Control with Remote Attestation

In [20], TPM is used to measure the integrity of a system before it joins a network by building a trusted software stack that can be verified by a remote machine (Certificate Authority). This integrity check will allow a network to validate that a computer is booted with an uncorrupted software stack by measuring the PCR values in the TPM at boot time.

This feature is used to build an extension to the 802.1X network access control protocol and Extensible Authentication Protocol (EAP) that allows making authorization decisions based on the booted software stack of the computer instead of a key or a password based technique.

This implementation was done using Xen Hypervisor, Ubuntu 9.04 and a modified version of the Linux Grub Boot loader.

1.5.3 The Trusted Platform Module (TPM) and Sealed Storage

In [18], a model was presented that can provide a mechanism for creating sealed storage that can only be accessed by designated applications whenever needed. This mechanism is almost similar to the secure boot based on the last known good values of the TPM PCR's but this time it is used to seal and un-seal a storage device to ensure it is only being accessed if it is in a safe state.

This model depends on the fact that TPM provides a way to enable the correlation between data and the current platform configuration presented by the PCR values.

Two commands were introduced to Seal and Unseal a storage as follows [18]:

- Seal Command:

The Seal command will use a set of TPM PCR indices as input and encrypts the data provided using its Storage Root Key.

The output of the Seal command would be a cipher text in addition to a list of indices that are integrity-protected as well as the corresponding values of the PCR's at the time of issuing the Seal command.

The Seal command is also providing the possibility of associating the values of the PCR's that must exist before Unseal decrypts the data during the encryption time.

- Unseal Command:

The Unseal command will use the cipher text and the PCR indices created by the Seal command to verify the integrity of the corresponding PCR values and compare them against the current values of the PCR's.

Based on the output of the comparison, the TPM can decrypt the storage and return the output data required.

1.6 FUTURE DIRECTIONS FOR USING TPM

As discussed in earlier sections, the increasing usage of cloud technology requires an increased security measures as well to ensure a complete secure solution for end users and organizations. TCG[3] researches and other security vendors' commercial products will be providing an aid to secure the cloud more and more in the future. TPM in particular is playing an important role now in several security products to ensure this ultimate secure solution exists [14].

The overall secured solution starts from the ability of the TPM to check on the installed software of each cloud-based server and ensuring the integrity of the loaded software stack. Accordingly associating the output with something like TNC[4] to control the network access will provide the ability to restrict access to a network or a service [14].

In addition, we can also use TPM for self-encrypting of hard disk drives (BitLocker is an example of software that is starting to be widely used now by organizations to encrypt hard drives especially of employees laptops).

Another direction is Windows 8 operating system, Microsoft will be requiring having the TPM chip on all Windows PCs, phones and tablets. This will be moving the security checks to the platform's lowest level. There are future hopes and expectations from security experts that this will help to develop better security mechanisms that are based on TPM to prevent against the current boot-level intrusions faced. [22]

In addition to Microsoft, Google Chromebook is already including a TPM chip and is using it in measuring the firmware once the device is powered up. It determines if the firmware is changed since the last shutdown and if it did, it removes the faulty firmware and replaces it from a library of known good configurations. It then checks the firmware again and if it is ok, it continues to boot up. This is simply providing a self-healing capability that TCG is hoping can be extended to other vendors [22].

It is worth nothing, even though there is an increasing usage of TPM enabled devices for better security levels and data protection mechanisms, there are still some countries that do not allow the use of TPM for security purposes. The list of countries includes Russia, China & Morocco[5] [23].

---

[3] http://www.trustedcomputinggroup.org/

[4] http://www.trustedcomputinggroup.org/solutions/network_access_and_identity

[5] Based on personal work experience

2 INTRODUCTION

2.1 PROBLEM STATEMENT

To contribute in providing a solution for three Cloud Security problems, with a specific focus on the access security for Virtual environments.

The three problems within scope are:

- Poor Machine Identity

An attempt to answer the question of "Who is really using the cloud service?" This is targeting the identification of hardware clients trying to connect to cloud services and ensuring they are legitimate clients that are granted the correct access permissions to this cloud service [1].

- Multi-Tenancy

A standard security requirement from any cloud service provider that shared virtual resources that are accessed by users from different trusted organizations needs to be always isolated for each organization's users at different levels during provisioning, runtime, migration and de-provisioning [2].

- Malicious Insiders

A security problem where threats to an organization can be from Privileged insiders such as employees who change role but keep privileges or ex-employees whose access were not revoked after contract termination [2], our focus in this work would be on the client-side malicious insiders.

2.2 OBJECTIVE & MOTIVATION

The objective of this work is to Design, Implement and Verify a machine authentication and authorization mechanism using Trusted Platform Module for accessing remote Virtual Resources (Machines) in a Cloud Service. This mechanism will be used to identify the access privilege of a client machine in association with the normal user authentication and authorization mechanisms.

The Motivation behind this is the increasing usage of Cloud Services and Virtualization Technology by many organizations and accordingly the need to ensure a secure environment access. In addition, although Security is one of the most researched topics, client security in terms of authentication of Hardware machines is not a very popular one even though it can provide solutions to many security problems like the ones mentioned in the Problem Statement Section.

In addition, the use of Trusted Platform Module is increasing across several organizations and governments and is becoming a requirement for several software

providers to have a TPM chip to ensure hardware level security; we will be discussing this in more details in section 2.6 where we review the future directions in using TPM.

The output of this work should be able to provide the following benefits:

- Allow the restriction of virtual machines remote desktop access to specific physical client machines so it can contribute to the Multi-tenancy problem resolution.
- Guarantee for an organization that its cloud based virtual environment is only accessed through approved physical clients list so it will also contribute to the client-side malicious insiders problem by preventing the use of privileged accounts except from within an approved and monitored physical machines.
- In addition, it will be easy to identify the physical clients accessing a specific virtual machine at any given time and in case of threats, it will be possible to take corrective actions with those clients.

3 PROPOSED SOLUTION

In this chapter, we present the high-level Design, the Architecture and the Process Flow of the developed mechanism.

3.1 HIGH LEVEL DESIGN

The proposed solution must provide a mechanism to allow the identification of the client machines to the Hypervisor; in addition, it must also provide an authentication mechanism using the client machine's unique identifier along with the normal user authentication process. Furthermore, it should be able to identify the access rights of each client machine to allow or deny the access to virtual machines accordingly.

The solution is developed based on TPM Endorsement Keys being unique to each hardware platform and thus can be used to clearly identify a client machine.

The high-level design of the proposed mechanism is as follows:

- ACL database associated with the Hypervisor to add, remove and modify the hosted virtual machines and map them to the list of allowed TPM-enabled physical clients that are allowed to connect to their graphical user interface remotely

- ACL database administration module to add, remove or modify the TPM Endorsement keys of client machines allowed to access the remote desktop of virtual machines hosted on the Hypervisor

- Client machines authentication module using the client machine's TPM Key signature which in association with the normal user based authentication is able to either grant or deny access to the remote desktop of the virtual machines hosted on the Hypervisor

3.2 ARCHITECTURE

As illustrated in Figure (7) below, the architecture of the developed mechanism comprises several components:

- Linux Hosts

All physical machines are Linux Based running Ubuntu 12.04.4 LTS, used either as a Host for VirtualBox Hypervisor or as Client machines

- TPM

TPM chips used are version 1.2, Spec Level 2 and running on Dell Laptops of Models Latitude D630 and Precision M4500 both manufactured by Broadcom

- rDesktop

rDesktop version 1.7.1 is used for establishing RDP connections to virtual machines hosted on the VirtualBox Hypervisor

- VirtualBox

VirtualBox version 4.3.8 for Linux Hosts is used as the Hypervisor for hosting VM's. VirtualBox 4.3.10 Oracle VM VirtualBox Extension Pack is installed as well to provide the additional functionality of VRDP [8].

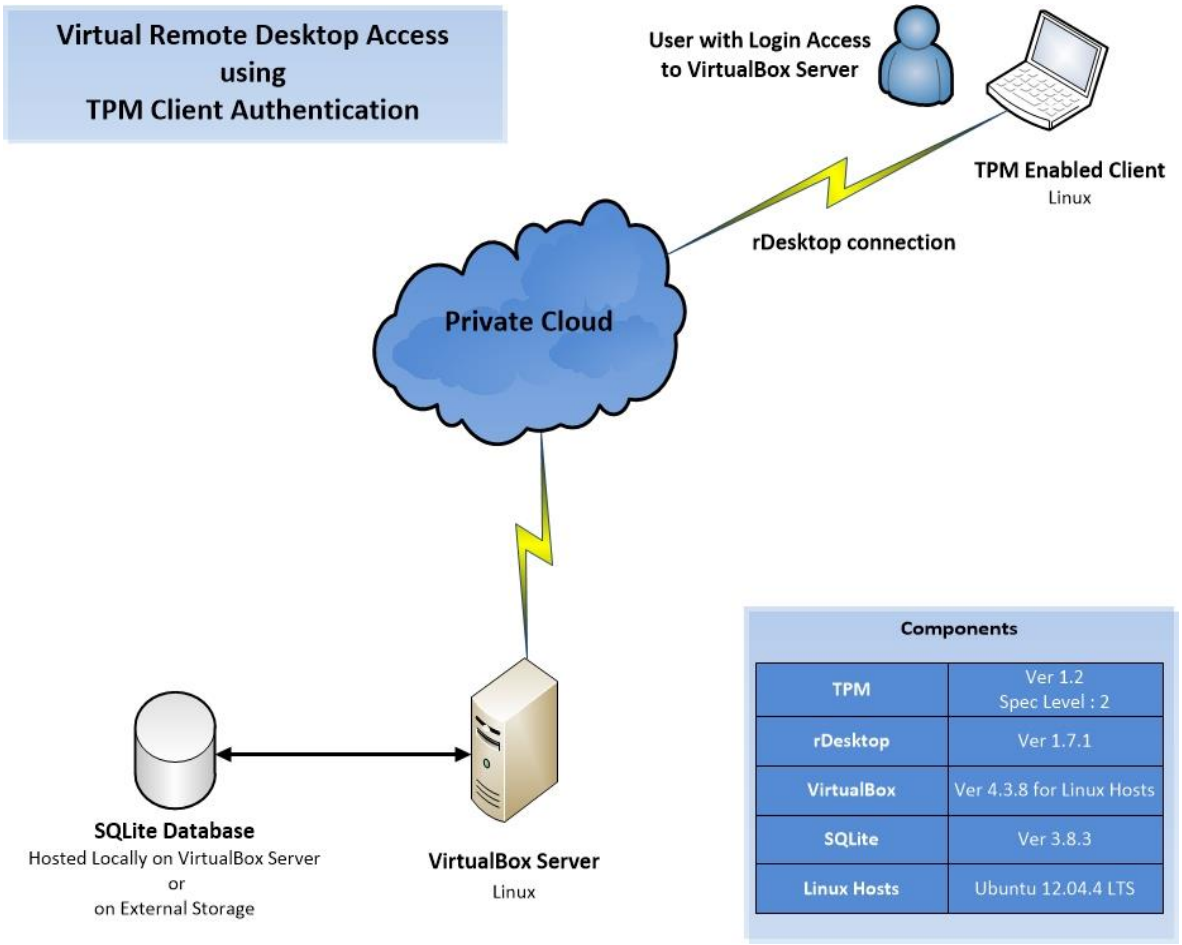


Figure 7 Architecture Overview

- SQLite

SQLite version 3.8.3 is used to host the ACL that is accessed from VirtualBox Hypervisor, physical clients and the Authentication Module. This Database can be hosted locally on the same physical host as the VirtualBox Hypervisor or can be on any external network storage as long as the path is accessible from all required components

- User Accounts

Users will need to have either access to the Physical machine hosting the VirtualBox Hypervisor through local or Domain accounts or access through local accounts to the destination VM directly, depending on the VirtualBox user authentication library to be used

In addition, each client machine owner will need to have his client TPM owner password to be used while establishing the RDP connection to the destination VM on VirtualBox.

3.3 PROCESS FLOW

In Figure (8), we provide a complete explanation of the mechanism process flow. As illustrated, the process starts with an initiation of an rDesktop connection from a user on a client machine to a VirtualBox Server to access the remote desktop display of a VM. This rDesktop request contains the following fields:

Table 2 rDesktop Connection Request Fields

Field Name	Description	Standard / TPM Related
User Name	This is required to authenticate the access for the remote connection request, either to the VirtualBox Server or to the Destination VM. (Depending on the VirtualBox User Authentication Library used) This is different from the Login User Name on the Destination VM once the remote Display connection is established	Standard Field
Password	For the user account used for the remote connection authentication	Standard Field
Client Machine TPM Public Endorsement Key	The new TPM authentication mechanism generates this key during the execution of the rDesktop connection request and sends it to the destination VirtualBox Server. The initiating user will be prompted to provide the TPM Owner Password to be able to generate this EK	TPM Related Field. This is a non-standard field for rDesktop and is used specifically for this new TPM authentication mechanism
Destination VirtualBox Server Name or IP address	To specify the destination server receiving the remote connection request	Standard Field
Destination VM Remote Desktop Server Port number	This is a unique port number assigned by VirtualBox for each VM that has VRDP enabled (can be modified as needed per VM)	Standard Field

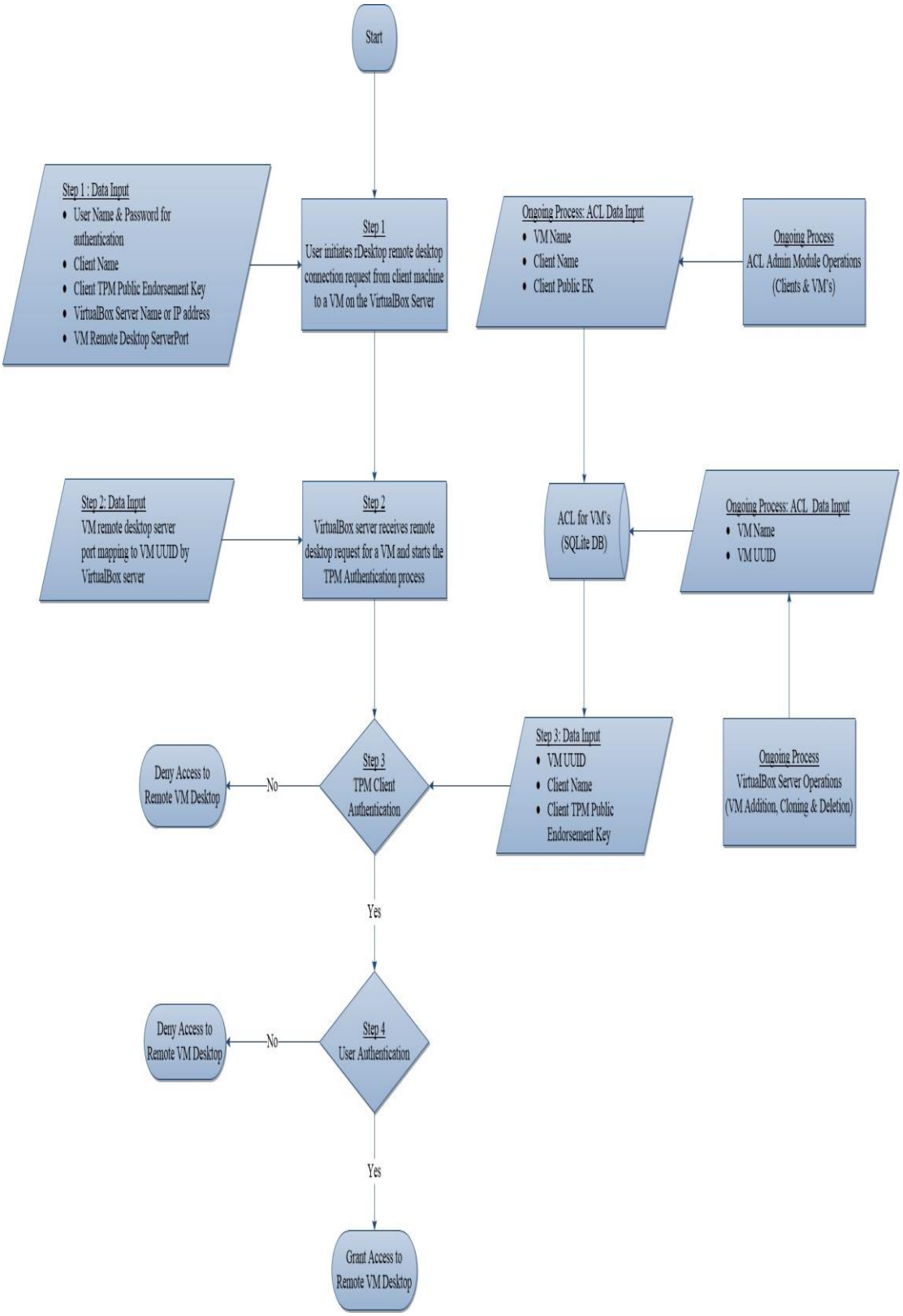


Figure 8 TPM Authentication Process Flow

The request is then received by the VirtualBox Server, which maps the received Remote Desktop Server Port to a UUID representing the Destination VM and passes

it on to the TPM Machine Authentication process. The client TPM public EK is also received by the VirtualBox server as part of this step.

In parallel, there is an ongoing process running all the time to create and update the ACL for the VM's hosted on the VirtualBox server. This process starts with the creation of the first VM on VirtualBox server, which triggers the creation of a SQLite Database that will be used for the ACL going forward. This Database is populated from VirtualBox upon the creation of any new VM or cloning of an existing VM with the following details:

- VM Name
- VM UUID

The VM Name is required to be present in the ACL prior to the addition of any client machines as the clients addition must be against an already existing VM Name. VirtualBox must also add the VM UUID in the ACL as this UUID is used to identify the destination VM upon receiving the rDesktop remote connection request from a client and accordingly used to check the ACL for client entry against this VM UUID.

The ACL is also updated by VirtualBox when a VM is deleted to remove the corresponding entries in the Database.

In addition, an ACL Administration Module is used to update the Database with Clients information and map the access permissions to VM's, the details provided to the ACL through this Administration module are:

- Client Name
- Client TPM Public Endorsement Key
- VM Name to be accessed by this Client

This ACL administration module should only be accessed by the System Administrator responsible for VirtualBox as it is the single point of entry to add and update clients, their corresponding Public Endorsement Keys and the VirtualBox VM's access list.

Having all this information about clients and VM's available in the ACL, the TPM Machine Authentication process then uses the UUID of the destination VM to check if the client trying to connect is listed in the ACL of this destination VM. If found, it will then compare the TPM public EK received along with the connection request with the one already existing in the ACL Database for this client machine and accordingly either deny the connection request or pass it on to the next level and start the User authentication process.

The User authentication process can be performed in one of two approaches supported by VirtualBox depending on the authentication module library used; both approaches will be discussed in more details in section 4.2:

- VBoxAuth (Linux PAM): User Authentication is done against the VirtualBox Server Host's Linux PAM system
- VBoxAuthSimple: User Authentication based on User Accounts and Passwords added to the extra data section of the settings file of each VM

Whether the Linux PAM or the VM extra data settings file approach are used for authenticating users, the user is either granted or denied access to the Remote Desktop of the destination VM and the process completes.

Figure (9) illustrates the sequence diagram for the use-case scenario starting from requesting a Virtual Machine to be created, requesting the TPM EK of a client machine to be added to the ACL then finally initiating the rDesktop remote connection to the VM and the associated machine TPM authentication followed by the user authentication.

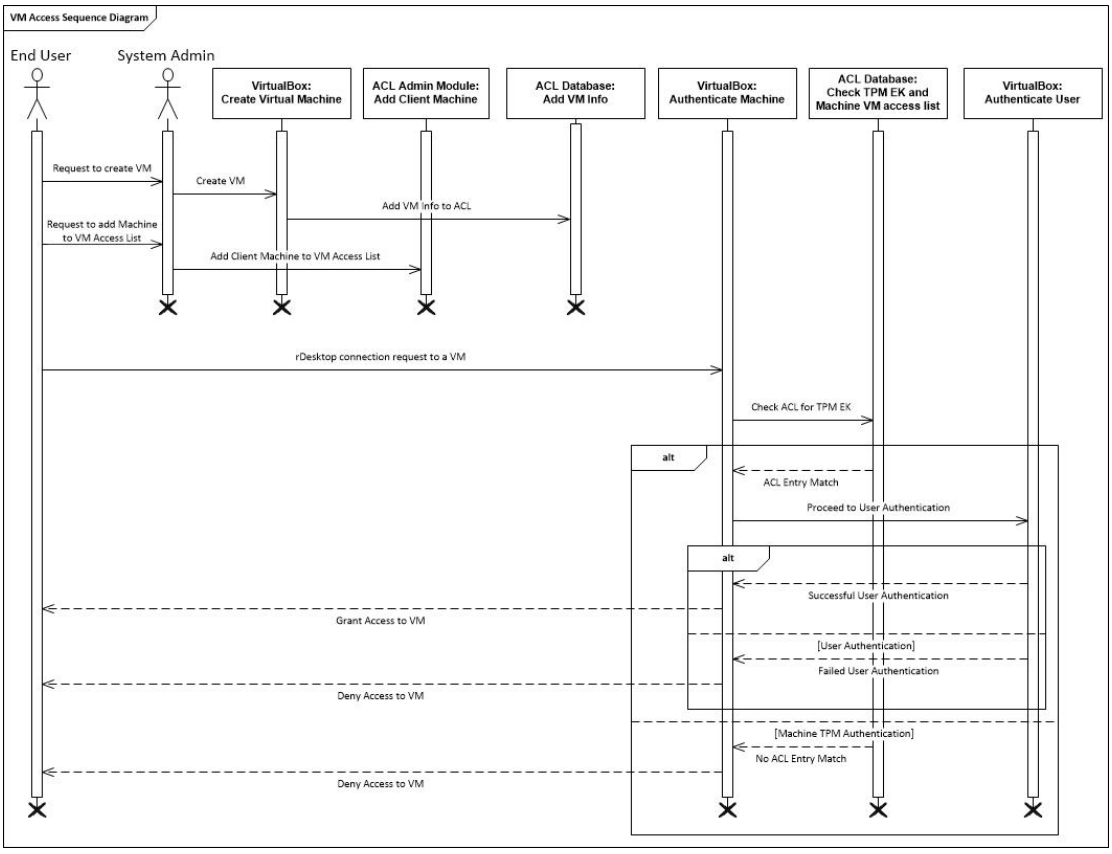


Figure 9 TPM Authentication Sequence Diagram

4 IMPLEMENTATION

In this chapter, we present the details of the implementation of the new TPM authentication mechanism, starting from the prerequisites and the environment setup to the technical details of how this new mechanism was developed and deployed.

4.1 PREREQUISITES

4.1.1 Server

The server used to host the Hypervisor is Linux based with Ubuntu Desktop 12.04 LTS "Precise Pangolin" 64-bit [24] as the operating system. We used VirtualBox 4.3.8 [25] for Ubuntu 12.04 to provide the Hypervisor functionality. In addition, we installed OpenSSH [26] to allow for SCP between the clients and the server.

The Database used to host the ACL contents is SQLite 3.8.4.3 [27].

4.1.2 Clients

All clients are Linux based with Ubuntu Desktop 12.04 LTS "Precise Pangolin" 64-bit [24] as the operating system. All clients must have a TPM chip with specification level 2 revision 1.2 [28].

To establish the remote Desktop connection from the client machines to the Hypervisor server, we used rDesktop 1.7.1 [29].

4.1.3 Accounts

For the system administrator to be able to setup the required environment including the Hypervisor on the server machine, an account with ROOT privilege is required.

On the clients' side, user accounts on client machines for normal login to the Operating system are required.

If VBoxAuth (Linux PAM) VirtualBox user authentication is used, user accounts on the server machine that the clients use to authenticate the connection to VirtualBox virtual machines are needed.

In case VirtualBox VBoxAuthSimple user authentication is used, user accounts in the individual VM's XML settings file are needed.

Users attempting to establish a connection to the Hypervisor use these user accounts to transfer the TPM Public EK through SCP in both methods, VBoxAuth (Linux PAM) or VBoxAuthSimple.

4.2 ENVIRONMENT SETUP

This section details the setup procedure required for the different solution components illustrated in Figure (7) in section 3.2.

4.2.1 VirtualBox 4.3.8

Since we need to modify the source code of VirtualBox to integrate the new TPM authentication module, the .deb Linux package is not used and instead we built VirtualBox from the source code directly using the following instructions [30].

To install the required prerequisites packages and additional packages:

```
>sudo apt-get install gcc g++ bcc iasl xsltproc uuid-dev\
zlib1g-dev libidl-dev libsdl1.2-dev libxcursor-dev\
libasound2-dev libstdc++5 libhal-dev libpulse-dev\
libxml2-dev libxslt1-dev python-dev libqt4-dev\
qt4-dev-tools libcap-dev libxmu-dev mesa-common-dev\
libglu1-mesa-dev linux-kernel-headers\
libcurl4-openssl-dev libpam0g-dev libxrandr-dev\
libxinerama-dev libqt4-opengl-dev makeself\
libdevmapper-dev default-jdk python-central\
texlive-latex-base texlive-latex-extra\
texlive-latex-recommended texlive-fonts-extra\
texlive-fonts-recommended

>sudo apt-get install ia32-libs libc6-dev-i386 lib32gcc1\
gcc-multilib lib32stdc++6 g++-multilib
```

Since we are building on a 64-bit system, there is a need to create some missing links to shared 32-bit libraries:

```
>ln -s libX11.so.6    /usr/lib32/libX11.so
>ln -s libXTrap.so.6  /usr/lib32/libXTrap.so
>ln -s libXt.so.6     /usr/lib32/libXt.so
>ln -s libXtst.so.6   /usr/lib32/libXtst.so
>ln -s libXmu.so.6    /usr/lib32/libXmu.so
>ln -s libXext.so.6   /usr/lib32/libXext.so
```

To start the build, we execute the following commands:

```
>cd source_code_root_dir
>./configure --disable-hardening
>source ./env.sh
>kmk all
>cd out/linux.amd64/release/bin/src/
>make
>sudo make install
```

The next two commands load VirtualBox module into the kernel and grant permissions to access it. This must execute before using one of the VirtualBox frontends:

```
>sudo modprobe vboxdrv
>sudo chmod 777 /dev/vboxdrv
```

The GUI frontend used is “VirtualBox” and invokes as follows:

```
>cd source_code_root_dir/out/linux.amd64/release/bin/
>./VirtualBox
```

Another VirtualBox frontend that invokes from the same path is “VBoxHeadless” [31] that does not depend on the X window system of the host, has no graphical user interface, and used to manage virtual machines that is only accessed through VRDP [32].

To make use of the functionality of VRDP, we must download and install VirtualBox extension pack [33] using the “VirtualBox” GUI frontend or using “VBoxManage” command [34]. “VBoxManage” invokes from the same path as “VirtualBox” and “VBoxHeadless”:

```
>VBoxManage extpack install Path_To_Extension_Pack
```

VirtualBox is now ready to setup virtual machines as needed with any operating systems supported; the following must be taken into consideration on the individual virtual machine level to be able to use the new TPM authentication mechanism:

4.2.1.1 Remote Display

4.2.1.1.1 VRDP Server

With the installation of the VirtualBox extension pack, the functionality of VRDP is now supported however, it is not enabled by default on each virtual machine except if “VBoxHeadless” is used, as there are no other means to access the VM except through remote display. Accordingly, to use the GUI frontend the VRDP server must be enabled on the individual virtual machine level either using the Display settings from the “VirtualBox” GUI frontend or using the “VBoxManage” command [32]:

```
>VBoxManage modifyvm "VM_Name" --vrde on
```

In addition to enabling the VRDP server, each VM requires a unique TCP port to use for listening to incoming remote connection requests. The default port is TCP 3389 that cannot be used for more than one VM. The port must be assigned per VM either

using the Display settings from the “VirtualBox” GUI frontend or using the “VBoxManage” command [32]:

```
>VBoxManage modifyvm "VM_Name" --vrdeport TCP_Port#
```

4.2.1.1.2 RDP Authentication

To control the access to virtual machines through RDP, VirtualBox provides three different RDP user authentication methods that we can configure per VM as needed [35].

- Null Authentication:

No authentication required, any user can connect through RDP to the VM

- Guest Authentication:

Authentication done against Guest accounts, this feature is not yet supported and still being tested by VirtualBox

- External Authentication:

VirtualBox uses External authentication libraries to perform user authentication, it comes with two built-in libraries as follows:

  - VBoxAuth

The default external user authentication library used by VirtualBox, it authenticates against the user credentials on the host system where VirtualBox is running. The user account does not have to match that on the destination VM.

On Linux Hosts, VBoxAuth.so uses the host’s PAM system to authenticate the users requesting RDP connections to a destination VM on VirtualBox.

On Ubuntu systems specifically, we need to configure the VirtualBox PAM authentication first before using the VBoxAuth.so library to allow it to access the host PAM services, using the below steps [36]:

i. Create a new PAM Service instead of the default “login” PAM service used by VirtualBox. For this, we create and edit a new PAM configuration file “vrdpauth”:

```
>sudo gedit /etc/pam.d/vrdpauth
```

ii. Add the below lines to the new “vrdpauth” PAM configuration file:

```
auth            required        pam_unix.so
account         required        pam_unix.so broken_shadow
```

iii. Set the value of the environment variable "VOBX_AUTH_PAM_SERVICE" to the new "vrdpauth" PAM services so that VBoxAuth.so uses it instead of the default /etc/pam.d/login service:

```
>export VBOX_AUTH_PAM_SERVICE="vrdpauth"
```

- VBoxAuthSimple

Additional library that provides user authentication against user accounts configured per VM in the "extradata" fields in each VM XML settings file.

Since this is not the default VirtualBox library for RDP external authentication, it has to be globally set manually for VirtualBox when needed as follows [37]:

```
>VBoxManage setproperty vrdeauthlibrary "VBoxAuthSimple"
```

Each VM where this external RDP authentication mechanism is required will need to first switch to external authentication then to configure user accounts in its XML settings file as follows [37]:

i. Configure the VM to use external RDP authentication, using the library already set by VirtualBox:

```
>VBoxManage modifyvm "VM_NAME" --vrdeauthtype external
```

ii. Use VirtualBox internal hashing mechanism to hash the password for the user who will be added to the XML settings file of the VM:

```
>VBoxManage internalcommands passwordhash "PASSWORD TO \
BE HASHED"

Password hash: XXXXXXXXXXXXXX
```

iii. Add the user name and hashed password to the "extradata" field of the VM XML settings file:

```
>VBoxManage setextradata "VM_NAME" \
"VBoxAuthSimple/users/USER_NAME" XXXXXXXXXXXXXX
```

In this work, we are using the External RDP Authentication and testing both VBoxAuth and VBoxAuthSimple libraries with the new TPM machine authentication mechanism.

4.2.1.2 Network

The virtual machine network adapter must be configured to use "NAT" [38] so that it cannot be accessed directly with an IP address and any request must be routed through the VirtualBox server itself. "NAT" configuration is the default VirtualBox settings when creating new virtual machines.

This network adapter configuration ensures that all the remote desktop connection requests are sent to the VirtualBox server IP and using the designated VM remote display server port (VRDP TCP Port) and accordingly, using one of the two VirtualBox external user authentication mechanisms that the new TPM authentication module integrates with, VBoxAuth and VBoxAuthSimple.

4.2.2 TPM

To be able to start using TPM on a client machine, an administrator or a user with sudo privilege must initialize the TPM chip on each client machine's OS. Also, need to install a TPM management tool "TrouSers" [39] to set the TPM owner and SRK passwords as well as to create the TPM Endorsement key pairs.

The following steps [40] ensure that the TPM drivers exist on the client machine and are loaded into the kernel then install the TPM management tools and test the successful installation:

i. Enable TPM Security in the client BIOS, making sure that TPM is activated

ii. Check that the "tpm_tis.ko" TPM module used with version 1.2 exists:

```
>ls -la /lib/modules/`uname -r`/kernel/drivers/char/tpm\
|grep tpm_tis

-rw-r--r-- 1 root root 30344 Jan 30 19:54 tpm_tis.ko
```

iii. Load the TPM generic Kernel modules and the tpm_tis module into the system Kernel:

```
>sudo modprobe tpm_bios
>sudo modprobe tpm
>sudo modprobe tpm_tis force=1 interrupts=0
```

iv. Install the TPM management tool "TrouSers":

```
>sudo apt-get install trousers
>sudo /etc/init.d/tcsd start
```

v. Test the successful installation of "TrouSers" and check the specification level and revision number of the installed TPM chip:

```
>sudo tpm_version

TPM 1.2 Version Info:
Chip Version:         1.2.7.11
Spec Level:           2
Errata Revision:      1
TPM Vendor ID:        BRCM
TPM Version:          01010000
Manufacturer Info:    4252434d
```

The next steps, take the ownership of the TPM on the client machine, create an Endorsement Key and try to retrieve the public part of that key to make sure we have a fully functional TPM. Steps are as follows [40]:

i. Take ownership of the TPM chip, this sets the owner and the SRK passwords and is only allowed once. If a change is needed then the TPM security must be disabled in the BIOS then re-enabled and set active and the below commands must be run again:

```
>sudo tpm_takeownership
 Enter owner password: XXXXXXXX
 Confirm password: XXXXXXXX
 Enter SRK password: YYYYYYYY
 Confirm password: YYYYYYYY
```

ii. Create Endorsement Key, this creates the private and public key pairs that are unique to the client and cannot be changed:

```
>sudo tpm_createek
```

iii. Test that we can retrieve the public Endorsement Key:

```
>sudo tpm_getpubek
Enter owner password: XXXXXXXX
Public Endorsement Key:
  Version:   01010000
  Usage:     0x0002 (Unknown)
  Flags:     0x00000000 (!VOLATILE, !MIGRATABLE,
!REDIRECTION)
  AuthUsage: 0x00 (Never)
  Algorithm:          0x00000020 (Unknown)
  Encryption Scheme: 0x00000012 (Unknown)
  Signature Scheme:  0x00000010 (Unknown)
  Public Key:
     8daeb10b 1735639b cb47b440 6a8abbc4 7364084c
     103c05a2 c9cf8a9e cb7f6242 2346e257 efe46735
     edbee3dd 44a1d899 4d5032ae 4b7829d0 595adaa6
     29bad9f7 5f560574 a21a163b 8aeae94b 54563f07
     1db34c39 3813a159 9386e533 c62cc451 c36107eb
     87deb0c9 de3d4a79 c0038cd8 0ff3340b 7f5fb33f
     5ed41de2 ce56f242 2b0dd5a2 6f29d0df 3aefac1d
     ef20c023 3ed3a9a3 f7892733 7f543157 74c05f04
     3717db0d d44e001a 6ed154fe d75f5e09 325a6ee5
     f960975a e70e35bc f4e5b093 f51e46bc 5a28295e
     bf4366b8 43ecf70c fb42fa60 fc3383cd 015cf308
     41a8930a 18eb3ec7 52798583 08067eae 13686186
     b6a09d26 5066cb1b bac686e9 05392a47
```

We will be using this Public EK as a unique identifier for each client machine in the new TPM machine authentication mechanism to control the access to VirtualBox.

4.2.3 rDesktop 1.7.1

Since we need to modify the source code of rDesktop to accommodate the changes required for the new TPM machine authentication mechanism, we build the rDesktop client from source code on any client machine using the below commands [41]:

```
>cd rdesktop1.7.1 source code directory
>./configure
>make
>make install
```

4.2.4 SQLite 3.8.4.3

To install SQLite on Ubuntu, the below command is used:

```
>sudo apt-get install sqlite3
```

4.2.5 OpenSSH

Since we will need to transfer the Public EK file from the client machine to the VirtualBox server host through SCP, we need to configure SCP for all users who will be using the new TPM authentication mechanism to connect without prompting for a password to ensure that the normal operations of the mechanism is not interrupted.

The below commands are used [42]:

i. Ensure that the VirtualBox server and all clients are running OpenSSH:

```
>ssh –V
OpenSSH_5.9p1 Debian-5ubuntu1.1,OpenSSL 1.0.1 14 Mar 2012
```

If not found, then install OpenSSH.

ii. Each user to generate RSA key pairs on his client machine using "ssh-keygen", providing a passphrase to use to encrypt his private key

iii. Each user to copy the contents of the public key file "~/.ssh/id_rsa.pub" on his client machine and append it to the ".ssh/authorized_keys" file under the his home directory on the server

iv. Permissions must be set to "644" on the "~/.ssh/authorized_keys" file on the server machine for each user

Each user will need to login from his client machine to the server using SSH and provide his passphrase, this will be required only once and afterwards no password prompt will be required.

4.3 MATERIAL AND METHODS

4.3.1 SQLite

In this section, we present the Database design and the fields used and how they are initially populated and afterwards modified whenever needed.

4.3.1.1 Database Design

Both the Database name and Location are variables declared in the VirtualBox Implementation File "VirtualBoxImpl.cpp" located under the path "VirtualBox-4.3.8/src/VBox/Main/src-server" as part of the new TPM authentication module.

Table 3 SQLite Database Properties

Database Name	"VirtualBox_Clients.db"
Database Location	"/home/assers" on the VirtualBox Server
Table Name	Clients

The creation of the table and fields uses a "CREATE TABLE IF NOT EXISTS" command that is also embedded in the VirtualBox Implementation File "VirtualBoxImpl.cpp".

Table 4 SQLite Table Fields

Field Name	Type	Primary Key	Purpose
ID	Integer	YES	Unique identifier for each client per virtual machine. Generated automatically by SQLite upon addition of new client entry
VM_Name	Text		Name of the virtual machine that a client will be permitted to connect to
UUID	Integer		Unique ID of the virtual machine, generated automatically by VirtualBox server upon the creation of a VM
Client_Name	Text		Name of the client to be permitted the connection to a VM
EK	Text		Copy of the TPM Public Endorsement Key of the client machine

4.3.1.2 Data Entry

Table (5) below lists how the SQLite database is created and how it can be modified.

Table 5 SQLite Data Entry

Action	When	How	Who
Database and Table Creation	With the creation of the first VM in VirtualBox	SQL "CREATE TABLE IF NOT EXISTS" command in "VirtualBoxImpl.cpp"	VirtualBox Hypervisor (New added Functionality)
Database Update with VM Information	With the creation or deletion of a VM in VirtualBox	SQL "INSERT" and "DELETE" operations in the "RegisterMachine" and "unregisterMachine" methods in "VirtualBoxImpl.cpp"	VirtualBox Hypervisor (New added Functionality)
Database Update with Client Information	As needed to add, modify and delete clients information	Using the ACL administration module that will be described in more details in section 4.3.2	Systems Administrator

4.3.2 ACL Administration Module

A BASH script developed to handle all Access Control List administration activities by modifying the SQLite Database entries as needed. A system administrator who has a privilege to modify the Database should only access this script. We used normal UNIX permissions to control this access as needed.

The BASH script code is presented in Appendix A and Table (6) below lists a full description of the different functionalities that can be performed by this script to administer the VirtualBox Database:

Table 6 ACL Administration Module Functions

#	Name	Description
1	Display Clients Database	Display all entries of the VirtualBox Clients Database including all registered virtual machines and their UUID along with the allowed clients physical machine list per VM and their corresponding TPM public endorsement key
2	Export Clients Database to CSV File	Provide a possibility to export the VirtualBox Clients Database entries to a CSV file for an easier way to manipulate the exported entries for reporting purposes. It prompts the user to provide a full path and file name to use to export the Database entries.

#	Name	Description
3	Add a new client	Add a new physical machine as an allowed client to one of the registered VirtualBox VM's. It prompts for a choosing an existing virtual machine from the Database and verifies that the provided virtual machine name exists then it prompts to provide a new physical machine client name and checks if an entry already exist for the same client name to the chosen virtual machine; it then prompts to enter the full path and file name for the copy of the client TPM public endorsement key and checks if the provided path and file name exists. It then truncates the TPM EK by removing un-needed lines to limit the EK entry in the Database only to the exact key to minimize the size of the field. Once all checks are completed, it then inserts the new client entry along with the corresponding TPM EK to a new entry in the Database table for the selected VirtualBox virtual machine.
4	Modify an existing client name (All Entries)	Modify the name of a physical machine client across all its entries in the Database. It prompts the user to provide the name of the client name that is required to be changed then verifies that entries in the Database exist for this client name. It then prompts for the new name of that client machine and verifies that this new name does not exist in the Database and accordingly changes the name across all entries to the new one. This functionality is required in case a physical machine name is changed and this change needs to be reflected in the VirtualBox Clients Database to ensure the access to the VM's is using the new client name.
5	Modify an existing client TPM EK (All Entries)	Modify the TPM EK entry for an existing physical machine client across all its entries in the Database. It prompts the user to provide the name of an existing client machine in the Database then

#	Name	Description
		verifies that it exists and then prompts the user to provide a full path and file name for the new TPM EK. It then verifies that this TPM EK file exists and then truncates it and use it to replace the existing TPM EK data in all entries for the selected client machine name. This functionality is only required in case the added TPM EK for a client machine was wrong and there is a need to replace it with the correct one.
6	Delete an existing client (Single Entry)	Delete a single entry of a physical machine client for a virtual machine. It prompts the user to provide a client name and verifies that it exists in the Database then prompts the user to select the virtual machine for which the client entry will be deleted and verifies that it also exists in the Database then it deletes the selected client name entry for that virtual machine. This functionality is required in case a physical machine is no longer allowed to access a specific virtual machine and accordingly it needs to be removed from the VM's allowed clients list.
7	Delete an existing client (All Entries)	Delete all Database entries of a physical machine client. It prompts the user to provide a client name and verifies that it exists in the Database then prompts for a confirmation from the user that all the entries of the selected client name will be deleted for all the virtual machines in the Database and deletes all the entries accordingly. This functionality is required when a physical machine is no longer allowed to access any of the registered virtual machines in VirtualBox and accordingly all its entries in the Database need to be deleted.
8	Clear all Database Entries	Delete all the entries in the VirtualBox Clients Database. Prompts for a confirmation before performing the delete operation and also checks if the Database is already empty and prompt the user accordingly.

#	Name	Description
		Once the Database is cleared then the virtual machines' information can only be added through the VirtualBox server upon creation of new VM's
9	Quit	Exit the BASH script

4.3.3 rDesktop

rDesktop is used to initiate the remote RDP connection from the client machine to the VirtualBox server. In order to develop the new TPM authentication module, we modified the rDesktop source code to incorporate the new TPM authentication functionality with the client connection string and accordingly send the required information regarding the physical machine name, user name and TPM public EK content to the VirtualBox server to complete the machine authentication process.

The changes in the source code were only in one file "rdesktop.c" located directly under the rDesktop installation folder. Appendix B presents the patch file that can be used to apply the modifications from the original source code file; this patch file was generated using Linux "diff" and "patch" commands.

The changes made to the original source code added the below functionalities to rDesktop normal operations:

i. Define a new command line argument "-w" to be used when TPM authentication is required. Without using this new arguments there will be no change in the normal rDekstop operations

ii. Upon using the "-w" argument, the TPM management command "tpm_getpubek" is used to generate the client TPM public endorsement key and redirects the output to a file under the user's home directory after appending the host name and the user name to it. The user is prompted at this step to provide the TPM owner password for this physical client machine

iii. The VirtualBox server Name or IP address is then extracted from the rDesktop connection string provided by the user

iv. Using Linux "sed" command, the generated TPM public EK file is truncated to remove all the un-needed lines so that we minimize the size of the file before sending it to the VirtualBox server. This matches the format of the TPM public EK for the clients added to the VirtualBox clients Database using the ACL administration module

v. Using Linux "scp" command, the new truncated TPM public EK is then transferred to the VirtualBox server under the "/tmp" directory and deleted from the initiating client machine and then the RDP connection request is processed normally

Exceptions handled to delete the generated TPM key in case of failure in sending it to the VirtualBox server. Also, to terminate the RDP request if the provided TPM owner password is not correct or if the "-w" argument is used from a physical client machine with no TPM hardware chip.

4.3.4 VirtualBox

To incorporate the new TPM authentication module with the VirtualBox normal operations, we modified three VirtualBox source code files to handle the new TPM authentication functionalities.

The changes made to the source code are classified into two main categories as listed below.

4.3.4.1 Populating the ACL with VirtualBox virtual machines info

As mentioned in section 4.3.1, the initial ACL Database creation is performed with the creation of the first virtual machine within VirtualBox. The changes done to handle this are added to the file "VirtualBoxImpl.cpp" located under the path "VirtualBox-4.3.8/src/VBox/Main/src-server".

In addition to the initial Database creation, the below functionalities are also added to the same source code file:

i. Upon registering a new virtual machine in VirtualBox, insert a new entry for the virtual machine into the ACL Database including the VM Name and VM UUID. This includes newly created virtual machines as well as virtual machines that are cloned from existing ones

ii. Upon un-registering a virtual machine in VirtualBox, delete all the corresponding entries for that virtual machines from the ACL Database

Appendix C presents the patch file that can be used to apply the modifications from the original source code file.

4.3.4.2 Handling the client machine TPM Authentication

To be able to add the TPM machine authentication prior to VirtualBox normal user authentication, we modified the two source files handling the possible "External" authentication methods for virtual machines as follows:

- VBoxAuthPAM.c

Located under the path "VirtualBox-4.3.8/src/VBox/HostServices/auth/pam", modified to perform the below functionalities:

i. Define the path to the ACL Database that will be used to check the allowed physical clients for each virtual machine along with their corresponding TPM public EK data

ii. Retrieve the UUID of the virtual machine accepting the RDP connection request from a physical client

iii. Compare the ACL Database entry for the physical machine trying to connect to a virtual machine with the received information with the request. This is done using the name of the physical machine, the user name initiating the RDP connection request and the TPM public EK file received with the connection request

iv. If a successful match is found in the ACL Database, TPM machine authentication is successful and VirtualBox then proceeds to the normal user authentication process using the Linux PAM service.

v. The received TPM file from the client machine is deleted after the TPM authentication process is completed whether it is successful or not

Appendix D presents the patch file that can be used to apply the modifications from the original source code file.

- VBoxAuthSimple.cpp

Located under the path "VirtualBox-4.3.8/src/VBox/HostServices/auth/simple", modified to perform the below functionalities:

i. Define the path to the ACL Database that will be used to check the allowed physical clients for each virtual machine along with their corresponding TPM public EK data

ii. Compare the ACL Database entry for the physical machine trying to connect to a virtual machine with the received information with the request. This is done using the name of the physical machine, the user name initiating the RDP connection request and the TPM public EK file received with the connection request

iii. If a successful match is found in the ACL Database, TPM machine authentication is successful and VirtualBox then proceeds to the normal user authentication process against the user accounts configured per virtual machine in the "extradata" fields in each VM XML settings file

iv. The received TPM file from the client machine is deleted after the TPM authentication process is completed whether it is successful or not

Appendix E presents the patch file that can be used to apply the modifications from the original source code file.

5 RESULTS

In this chapter, we present the testing plans for the newly developed TPM authentication mechanism along with the results achieved. We describe in full the details of how we conducted the tests with explanation of the additional tools used to automate the testing process.

5.1 TESTING ENVIRONMENT

The setup of the testing environment for this new mechanism is as follows:

- *VirtualBox Server:*

Dell Precision M4500 Intel Core i7 Quad core processor with 8 GB RAM and a TPM chip version 1.2, Spec level 2 manufactured by Broadcom running Ubuntu Desktop 12.04 LTS "Precise Pangolin" 64-bit as the OS

- *Client:*

Dell Latitude D630 Intel Core 2 Duo Dual Core processor with 4 GB RAM and a TPM chip version 1.2, Spec level 2 manufactured by Broadcom running Ubuntu Desktop 12.04 LTS "Precise Pangolin" 64-bit as the OS

- *Network:*

10/100 Ethernet shared (not dedicated) Local Area Network

5.2 TESTING APPROACH

To be able to measure the performance of this new mechanism, two tests were conducted for each of the VirtualBox Authentication Mechanisms, one using the developed TPM Authentication module and one without it. These tests are conducted by initiating a request to establish a remote desktop display connection to a VM hosted on the VirtualBox server and measuring the time taken to authenticate and connect.

The VirtualBox authentication mechanisms tested are:

- VBoxAuth (referred to afterwards as PAM): Using the Linux PAM system to authenticate the users requesting remote desktop connection to a VM, users must have login accounts on the VirtualBox Server host machine

- VBoxAuthSimple: User Authentication handled by using the “extradata” fields in each VM’s XML settings file where users & passwords can be added individually for each VM hosted on the VirtualBox Server

We used the comparisons between the differences in the average run times for five cases to illustrate the efficiency of the developed mechanism:

- Difference between PAM authentication with and without TPM authentication
- Difference between VBoxAuthSimple authentication with and without TPM authentication
- Difference between PAM & VBoxAuthSimple authentication using TPM authentication
- Difference between PAM & VBoxAuthSimple authentication without using TPM
- Overall comparison between using and not using TPM authentication

In addition, we calculated the standard deviation for each of the four authentication combinations of test runs (PAM with TPM authentication, PAM without TPM authentication, VBoxAuthSimple with TPM authentication and VBoxAuthSimple without TPM authentication) to measure the fluctuation amongst the individual runs itself.

5.3 TPM MODULE TESTING

Since the normal run time of a single connection request from a rDesktop client to the VirtualBox server is less than one second, accordingly to be able to test the performance of the new mechanism correctly a number of remote desktop connections to a Windows 8 VM are executed sequentially to compute the overall average time to connect with and without the TPM authentication module. Each test conducted by using runs of counts (10, 30, 50, 70, 100, 200 & 300) for each of the two VirtualBox user authentication mechanisms.

The format for connecting to VirtualBox VM using rDesktop when TPM authentication is required is in the form:

```
rdesktop –u USERNAME –p – -w \
VIRTUALBOX_SERVER:VM_VRDP_PORT
```

Where:

- *USERNAME* is the account used to connect to the VirtualBox server whether we are using the PAM or VBoxAuthSimple user authentication mechanisms
- "-p -", is directing rdesktop to prompt the user for a password to use to establish the connection to the VirtualBox server and again this is whether we are using the PAM or VBoxAuthSimple user authentication mechanisms
- "-w", is the new added option to specify using TPM authentication with this connection request

- *VIRTUALBOX_SERVER* is the VirtualBox server host name or IP address
- *VM_VRDP_PORT* is the remote desktop unique port number for the destination VM as configured on VirtualBox server

Since TPM authentication integrates now with rdesktop as part of the new mechanism and it changes the standard connection string format in case TPM authentication is required to establish the connection so accordingly, it prompts for providing the TPM Owner Password on the client machine trying to connect to the VirtualBox VM to be able to generate the TPM EK. This prompt requires an input from the user before rdesktop establishes the connection to the VirtualBox server.

To eliminate the need for a user intervention and automate the connection request without any user input prompts, "Expect" [43] tool is installed on client machines and used as shown in the below script to provide the TPM Owner password when prompted [44].

```
#!/usr/bin/expect

#Define user name to use to establish the connection
set user "USERNAME"

#Define VirtualBox Server the VM Remote Desktop
#unique port number
set machine " VIRTUALBOX_SERVER:VM_VRDP_PORT"

#Define TPM owner password on the client machine
set ownerpassword "TPM_OWNER_PASSWORD"

#Define user password
set password "USER_PASSWORD"

#Spawn the rdesktop connection command
spawn rdesktop -u $user -p $password $machine

#Provide TPM Owner password when prompted and close
while {1} {
expect {
      "Enter owner password:" {send "$ownerpassword\r"}
}
}
```

Figure 10 "Expect" Script

The *USER_PASSWORD* depends on which VirtualBox user authentication method is used. If using PAM, the user name and password authenticates against the Linux PAM system while if VBoxAuthSimple is used, the user name and password authenticates against the VM's XML file extradata content. However, in the case of VBoxAuthSimple, we also need the same user name created on the VirtualBox server not only on the VM level and ensure that seamless SCP access exists to that account

to copy the generated TPM EK securely from the client to the VirtualBox server. The Host and VM accounts do not need to have the same password.

To be able to run multiple connections using the "Expect" script, "GNU Parallel" [45], a GNU tool used to execute commands or scripts in parallel either on one or more computers, is used to run a number of sequential instances of the connection script with various counts against each of the VirtualBox user authentication mechanisms. This is required so that we can compute the average run time over a sizable set of connections. The counts used are (10, 30, 50, 70, 100, 200 & 300).

The command used to run the test is as follows [46]:

```
seq RUN_COUNT | parallel --joblog LOG_FILE \
PATH_TO_EXPECT_SCRIPT
```

Where:

- *RUN_COUNT* is the count of times the script is needed to run, for this testing the run count is set for each of (10, 30, 50, 70, 100, 200 & 300) and the output results measured for each
- *LOG_FILE* is the full path to the output log file that records the timing of each job in the parallel run
- *PATH_TO_EXPECT_SCRIPT* is the full path to the Expect script containing the connection string and passwords used to establish the remote desktop connection to the destination VM

For each test per authentication module, we examined VirtualBox log files (VRDP.log and VBoxAuthSimple.log) to ensure that all connection requests to the destination VM were successful.

5.4 TEST RESULTS

The detailed output results obtained from the test runs are listed in Appendix F to illustrate the run times for individual jobs as well as the average run time for each of the batch counts.

5.5 TEST RESULTS ANALYSIS

To summarize the obtained test results in Appendix F, Table (7) below presents the average run times per count for each authentication category followed by comparison graphs and corresponding analysis for the findings.

Table 7 Test Results Summary

Test Results Summary				
# Runs	Avg. Run Time - TPM PAM	Avg. Run Time - No TPM PAM	Avg. Run Time – TPM VBoxAuthSimple	Avg. Run Time – No TPM VBoxAuthSimple
10	0.98	0.21	1.05	0.22
30	1.02	0.2	0.87	0.23
50	0.91	0.19	0.86	0.21
70	0.90	0.19	0.88	0.17
100	0.96	0.22	0.87	0.19
200	0.90	0.19	0.88	0.21
300	0.89	0.21	0.94	0.16
Avg.	0.94	0.20	0.90	0.20

5.5.1 Difference between PAM authentication with and without TPM authentication

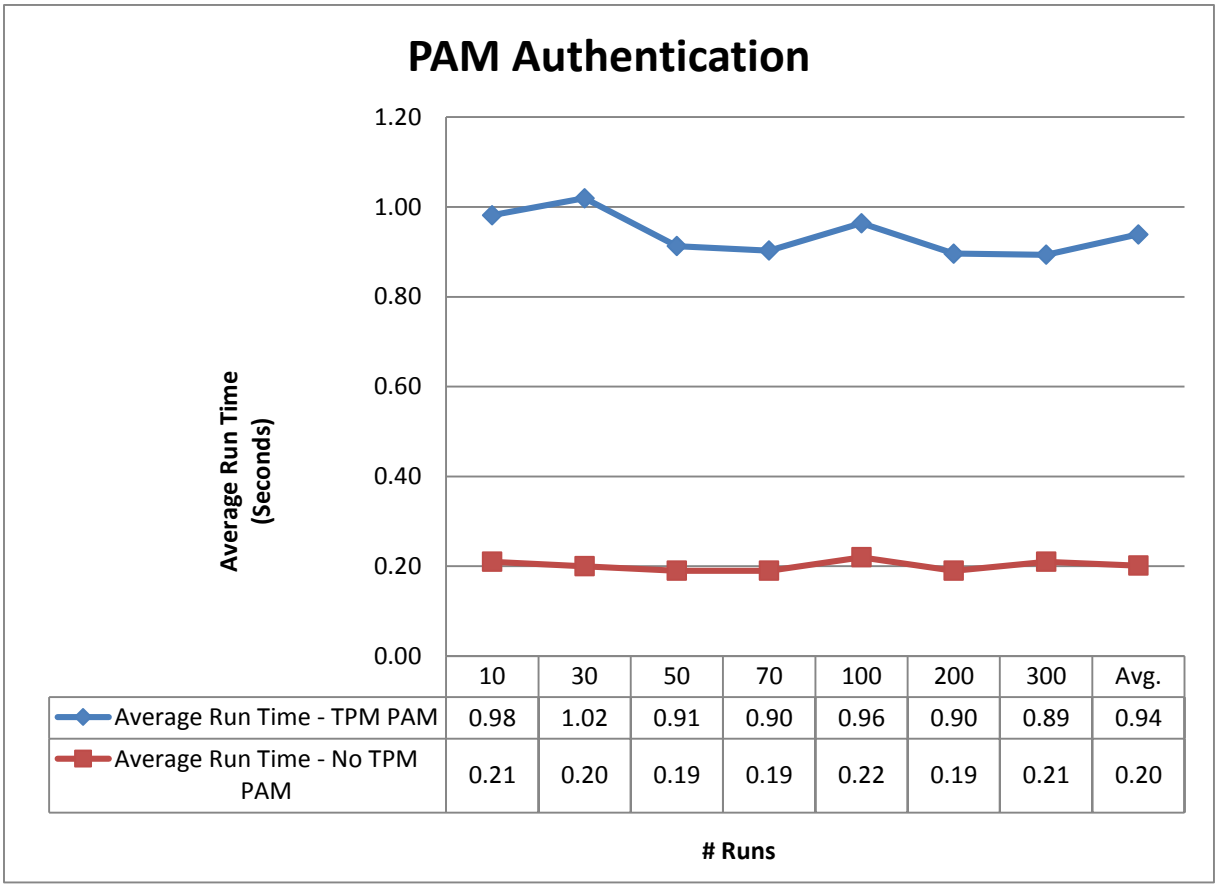
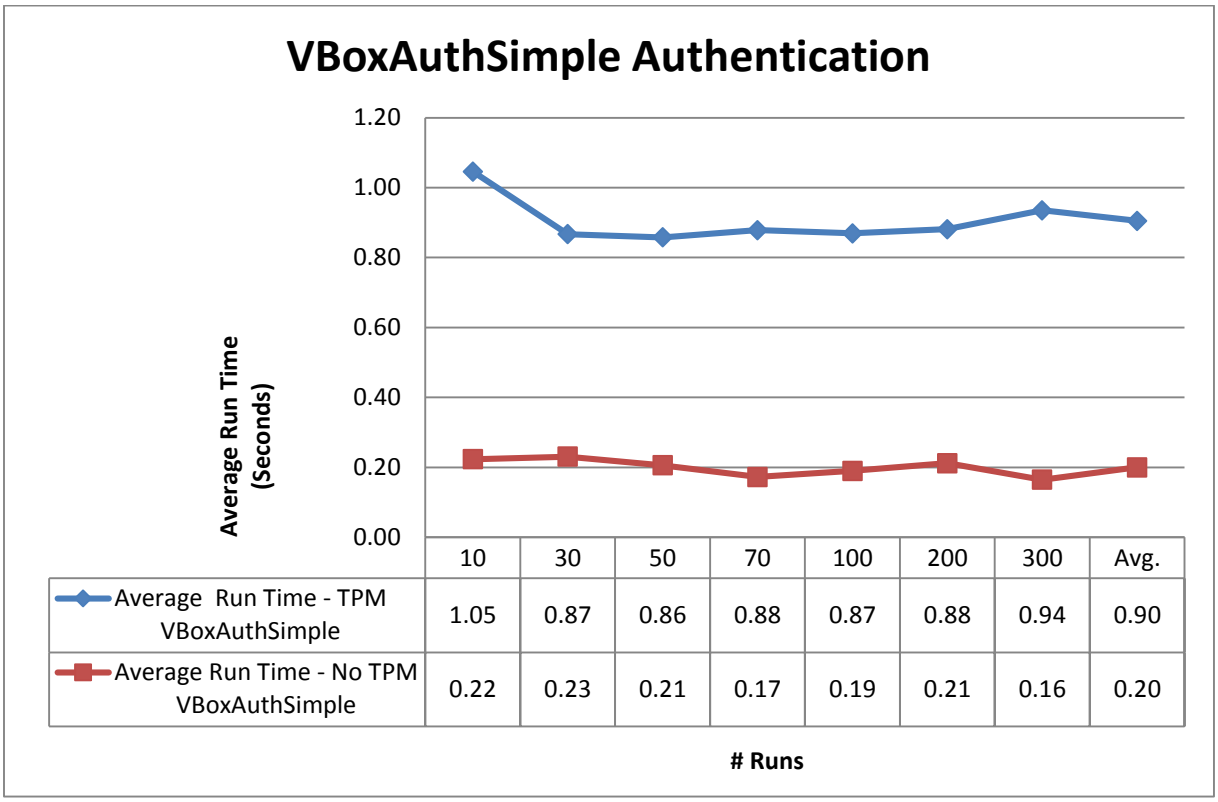


Figure 11 PAM Authentication with and without TPM

Figure (11) above shows the difference between the average run times when using VirtualBox PAM authentication with and without TPM authentication. Using TPM Authentication with PAM introduced a delay of 0.74 seconds (Original average runtime using PAM alone is 0.2, added cost for TPM authentication is 370%).

This is due to the time taken to generate and transfer the TPM Public EK between the client and the VirtualBox Server, in addition to the time taken to read the content of the file and compare it to the client's entry in the ACL then accordingly decide whether to proceed or not with the user authentication using Linux PAM.

5.5.2 Difference between VBoxAuthSimple authentication with and without TPM authentication



	10	30	50	70	100	200	300	Avg.
Average Run Time - TPM VBoxAuthSimple	1.05	0.87	0.86	0.88	0.87	0.88	0.94	0.90
Average Run Time - No TPM VBoxAuthSimple	0.22	0.23	0.21	0.17	0.19	0.21	0.16	0.20

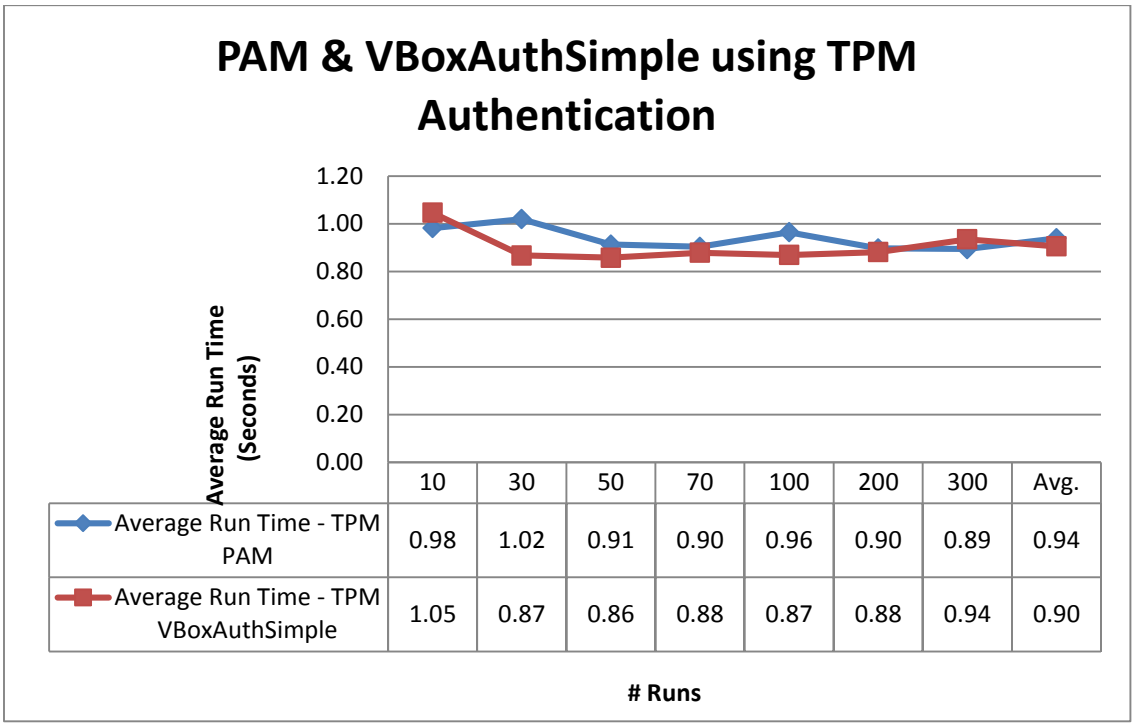
Figure 12 VBoxAuthSimple Authentication with and without TPM

Figure (12) above shows the difference between the average run times when using VirtualBox VBoxAuthSimple authentication with and without TPM authentication. Using TPM Authentication with VBoxAuthSimple introduced a delay of 0.7 seconds (Original average runtime using VBoxAuthSimple is 0.2, added cost for TPM authentication is 350%).

This is due to the time taken to generate and transfer the TPM Public EK between the client and VirtualBox, in addition to the time taken to read the content of the file and compare it to the client's entry in the ACL then accordingly decide whether to proceed or not with the user authentication using the client VM's extradata settings.

5.5.3 Difference between PAM & VBoxAuthSimple using TPM authentication

Figure (13) below shows the difference in the average run times for using TPM authentication with PAM and VBoxAuthSimple. This comparison shows that both mechanisms while using TPM almost have the same performance (VBoxAuthSimple is slightly better than PAM with just 0.04 seconds which can be negligible).

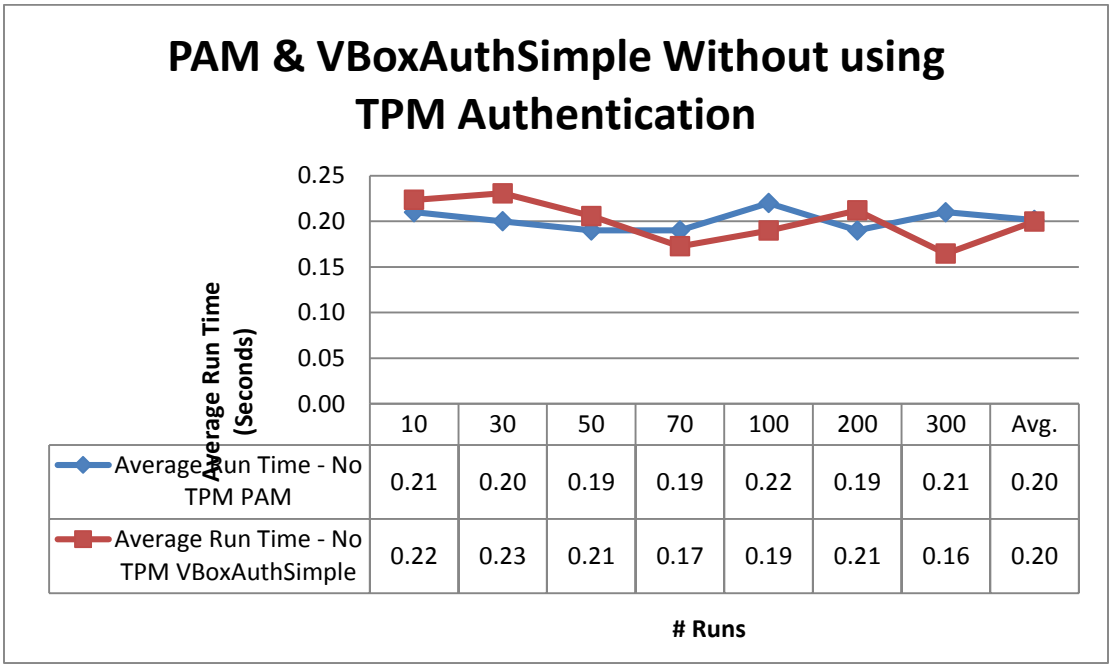


	10	30	50	70	100	200	300	Avg.
Average Run Time - TPM PAM	0.98	1.02	0.91	0.90	0.96	0.90	0.89	0.94
Average Run Time - TPM VBoxAuthSimple	1.05	0.87	0.86	0.88	0.87	0.88	0.94	0.90

Figure 13 PAM and VBoxAuthSimple using TPM Authentication

5.5.4 Difference between PAM & VBoxAuthSimple without using TPM

Figure (14) below shows the comparison between the average run time of the two VirtualBox authentication mechanisms without the use of the new TPM module. As shown, both mechanisms have the same performance on average of 0.2 seconds.



	10	30	50	70	100	200	300	Avg.
Average Run Time - No TPM PAM	0.21	0.20	0.19	0.19	0.22	0.19	0.21	0.20
Average Run Time - No TPM VBoxAuthSimple	0.22	0.23	0.21	0.17	0.19	0.21	0.16	0.20

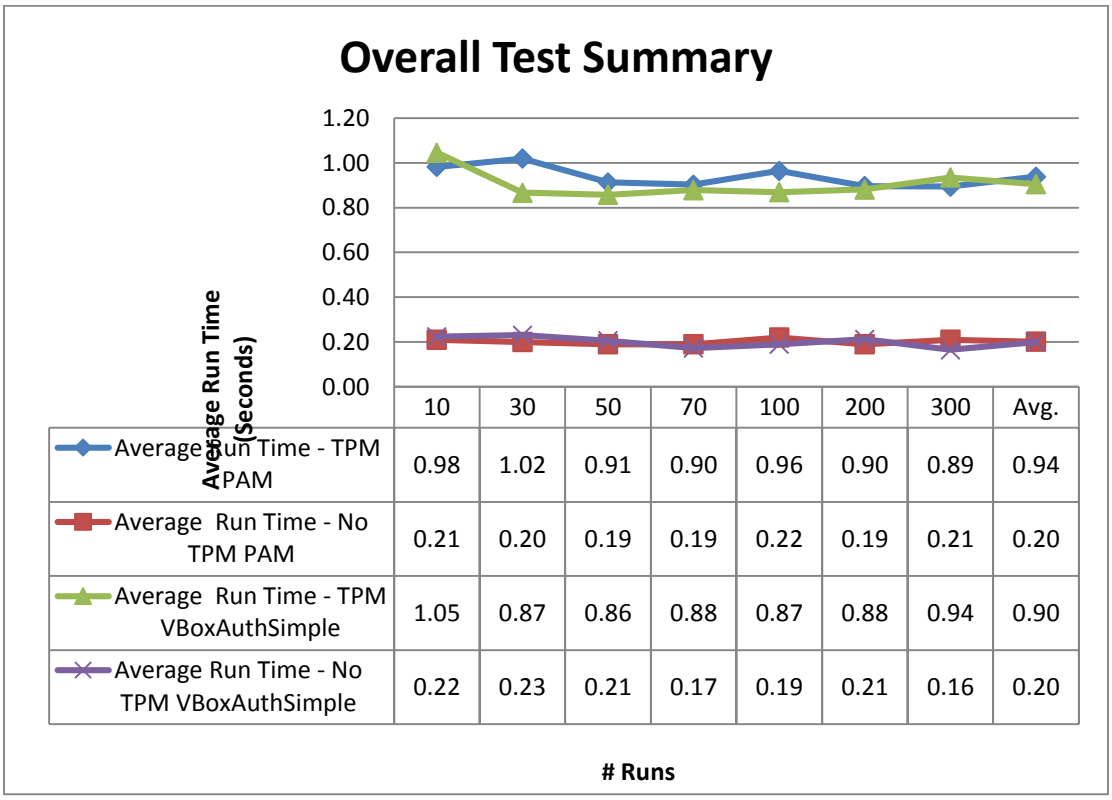
Figure 14 PAM and VBoxAuthSimple without using TPM Authentication

5.5.5 Overall comparison between using and not using TPM authentication

Figure (15) below shows the overall test summary for the four different authentication mechanisms combinations, it is noticeable that the introduced delay due to the TPM authentication mechanism is almost the same when used with different VirtualBox user authentication mechanisms. The time taken to generate the TPM Public EK and transfer over the network then read it and compare its content with the clients' entries in the ACL is almost constant regardless of the user authentication method used.

It also shows that the difference in the average runtime between using TPM with PAM and using TPM with VBoxAuthSimple is only 0.04 seconds. While there is no difference in the average runtime between using PAM and VBoxAuthSimple without TPM, which emphasizes that by using TPM with any of these user authentication modules the difference in the average runtime between them is almost equal whether TPM is used or not. Accordingly, TPM authentication can be used with any of the VirtualBox user authentication mechanisms since the overhead introduced is almost the same for both.

As mentioned earlier in section 5.2, a note to be taken into consideration for VBoxAuthSimple is that even though the actual user authentication is done on the VM level and not on the VirtualBox host level still, the user account used must also exist on the VirtualBox server as it will be used by SCP to transfer the file from the client to the Server for the machine authentication to take place. This user account does not have to have the same password on both levels.



	10	30	50	70	100	200	300	Avg.
Average Run Time - TPM PAM	0.98	1.02	0.91	0.90	0.96	0.90	0.89	0.94
Average Run Time - No TPM PAM	0.21	0.20	0.19	0.19	0.22	0.19	0.21	0.20
Average Run Time - TPM VBoxAuthSimple	1.05	0.87	0.86	0.88	0.87	0.88	0.94	0.90
Average Run Time - No TPM VBoxAuthSimple	0.22	0.23	0.21	0.17	0.19	0.21	0.16	0.20

Figure 15 Overall Test Summary

5.5.6 Standard Deviation Comparison between using and not using TPM

Figure (16) below shows the standard Deviation calculation between the different run times for each category of the four authentication combinations. It shows that the standard deviation between the individual runs is higher when using TPM for authentication. This is due to the fact of introducing dependency on TPM EK file transfer between the client and the server machines over the network so accordingly the time taken to establish a connection will vary for each time a user submits an initial connection request based on the network utilization at that time as one of the affecting factors.

However, the measured standard deviation is not that high with a maximum of 0.07 seconds when using TPM with VBoxAuthSimple.

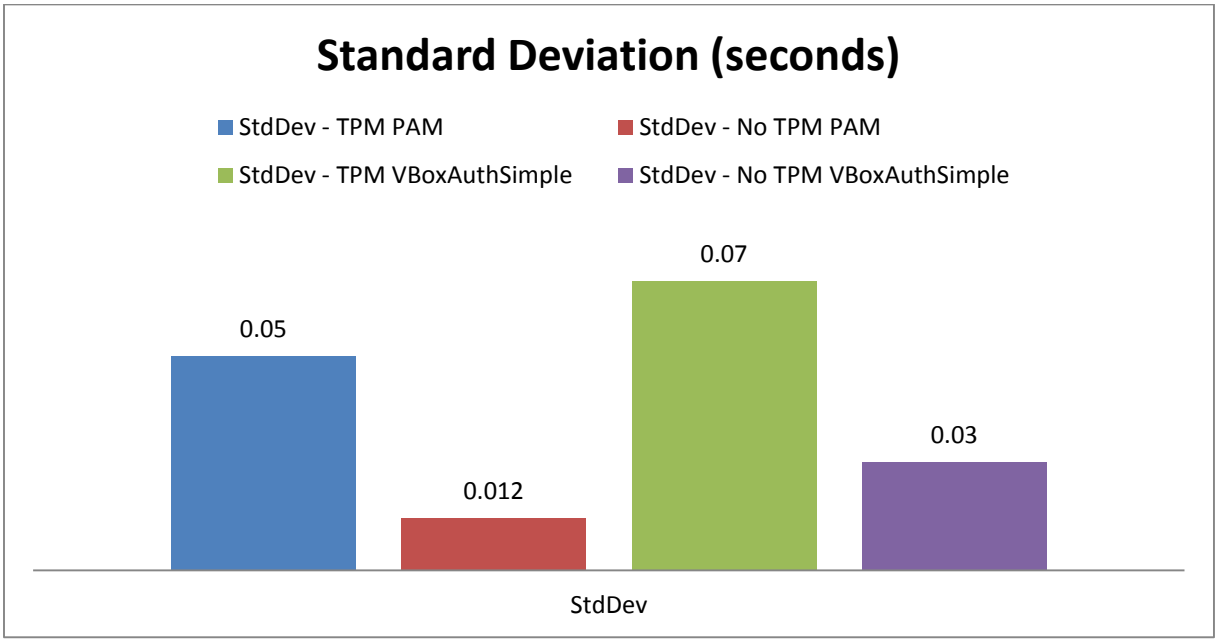


Figure 16 Standard Deviation between Runs

6 DISCUSSION

In this chapter, we discuss the different methods used during the implementation of this proposed model in terms of implementation techniques and alternative approaches available as well as the verification tests used to ensure the proper functionality of this new TPM authentication model.

We also present a security analysis of the different solution components in addition to an analysis of the associated cost of implementation and execution.

6.1 IMPLEMENTATION APPROACH

This work is only targeting the security of Remote Desktop Connections, not command line access as it is utilizing the VRDP capabilities in VirtualBox. In addition, command line access to VM's will not be possible since the VM's are configured to access the network using NAT not Bridged networking so their IP's will not be published and no one will be able to access them through the command line. If specific traffic is needed to be routed to a VM directly, Port Forwarding can be configured within the VirtualBox environment to handle this requirement.

In the following three sub-sections, we present a comparison between the original TPM authentication mechanism using a third-party privacy certificate authority, the new Direct Anonymous Attestation mechanism developed for TPM version 1.2 specifications and the proposed model in the scope of this work where we use TPM EK directly. We also provide a justification of why using any of the current TPM authentication mechanisms will not be possible for the overall proposed solution.

6.1.1 Original TPM Remote Attestation Mechanism

The original remote attestation mechanism for clients using TPM is as illustrated Figure (17) where any TPM enabled client need to follow the below steps to connect to a server [47]:

- TPM enabled client sends its EK and AIK to a third party Privacy Certificate Authority (PCA) so that it verifies that the generated AIK is from a genuine TPM module
- The PCA replies back with a certificate issued only to the originating EK for that specific TPM client
- The TPM client then uses the supplied certificate from the PCA to access the required server
- The server can now verifies that the connection request is from a genuine TPM client (without knowing any information about the client itself) and accordingly grant access to any requested service

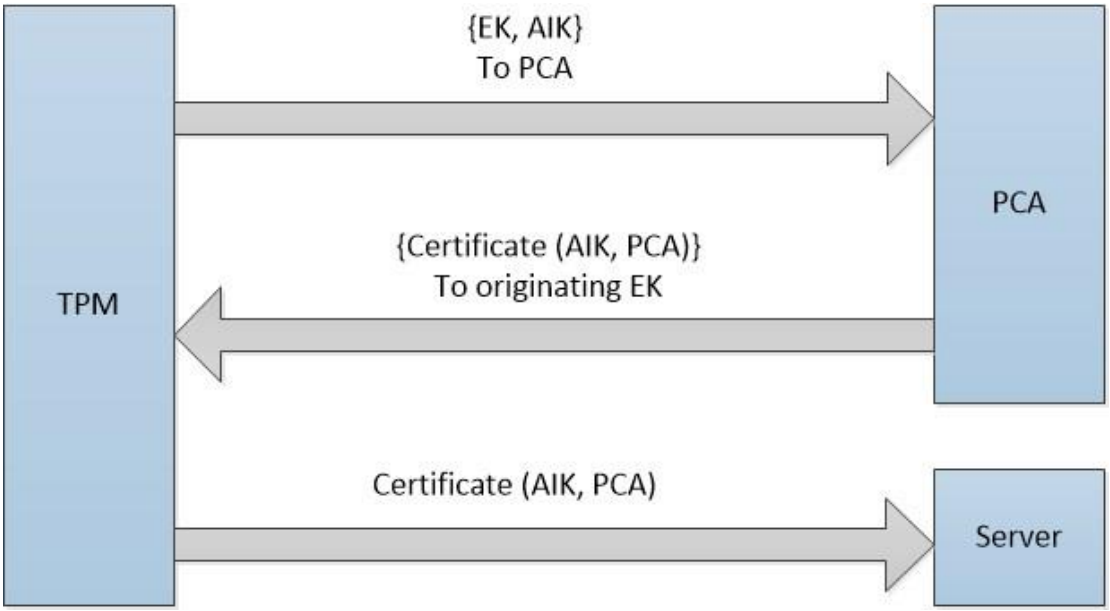


Figure 17 TPM Remote Attestation Model

The original TPM remote attestation mechanism has the following characteristics:

- TPM creates different AIK's for every access request to a server
- The TPM AIK does not have an identifier for the originating client identity, the server will allow access from any client with a certified AIK from a PCA and accordingly the originating client is not known to the server accepting the connection

This original mechanism has some known problems that the PCA must be involved in all the transactions between the client and the server and accordingly it must be always available and at the same time must be well secured which is contradicting. In addition, if there is a privacy requirement then it may be at risk if the PCA and the verifying server exchange information directly which in that case will allow the server to directly identify the TPM client [48], [49].

In addition to that, since the verifying server must be trusting the PCA and is sure that it is issuing a certificate only to a valid TPM, this entails that PCA can't be run by end users or private organizations [50].

6.1.2 Direct Anonymous Attestation Mechanism

With the problems raised against the original TPM remote attestation mechanism, IBM, Intel and HP worked together along with TCG to develop a new direct anonymous attestation mechanism that is available now with TPM version 1.2 specifications. This new mechanism is designed by Ernie Brickell of Intel, Jan Camenisch of IBM and Liqun Chen of HP [51].

DAA is aiming to provide a remote attestation procedure that does not provide a certificate from a trusted authority but instead use cryptographic proof that the connecting client has one [50].

DAA mechanism as illustrated in Figure (18) below works as follows [50]:

- During the JOIN initial step, the DAA issuer will verify the TPM platform and then will issue DAA signature to the platform
- The Platform will then proceed to the SIGN step where it proves to the verifier that it has a DAA signature from the issuer

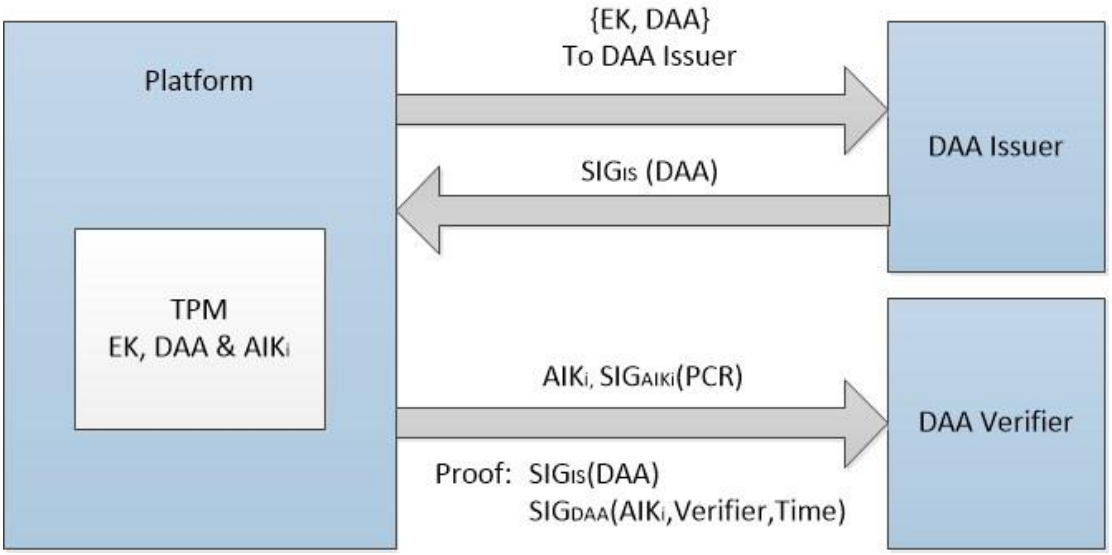


Figure 18 DAA Mechanism

This mechanism possess the below characteristics:

- The DAA issuer and verifier cannot link
- The DAA signature is only needed to be issued once by the issuer which can be a manufacturer or the organization owning the platform
- The mechanism provides a black listing functionality to allow the verifier to block requests from TPM's that are suspected to be fraudulent [52]
- The platform transactions with the verifier are always anonymous but the mechanism allow a level of controlled privacy using Name Based solution between the platform and the verifier to be able to link transactions from the same platform to perform platform profiling and be able to block requests from a platform generating too many transactions
- DAA provides a variety of balances between security and privacy by choosing one of the below models [51]:
    - Random base – for privacy sensitive cases

- Named base – for non privacy-sensitive cases
- A combination of both models

6.1.3 Proposed TPM Machine Authentication Mechanism

The characteristics of the original remote attestation mechanism does not allow for a true identification of the client attempting to access a server since with each new request a new different AIK will be created and verified by the PCA and sent to the server without any information on the requesting client.

In addition, with DAA complete anonymity and the ability to detect fraud TPMs seem to be opposing goals. Some compromise of anonymity must be made to allow fraudulent TPM keys to be detected by verifiers and even though an additional level of flexibility in terms of anonymity is available, still the identity of the client is not shown to the verifying server.

Since the main requirement for the proposed model is to reveal the identity of the client to the server directly to provide a client identification approach that we can use in a machine authentication mechanism, this proposed model will not be dependent on either the use of PCA / TPM AIK or the use of DAA.

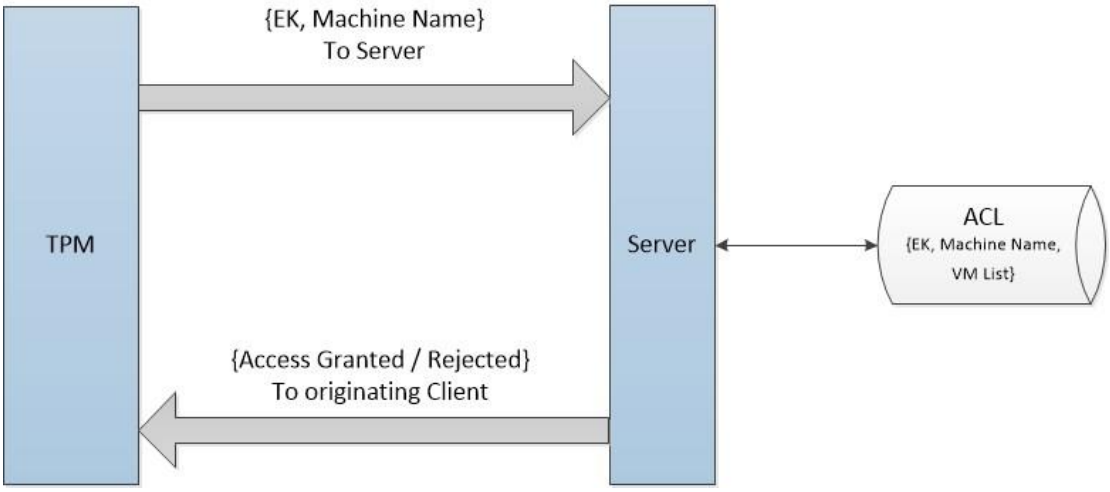


Figure 19 Proposed TPM Authentication Mechanism

As illustrated in Figure (19) above, the proposed mechanism works as follows:

- The client will send its TPM EK - which is a unique identifier of a client - along with the client machine name to the target server receiving the connection request
- The server will then authenticate the client against a pre-defined ACL that already contains a copy of the TPM EK for each authorized client along with the client machine name and a list of all virtual machines that this client has permissions to access

This proposed mechanism will guarantee that the server is aware of the identity of the hardware client attempting to connect and is authenticating it directly and granting the required access based on a well-known ACL.

Since privacy is a problem for security, having a highly secured environment entails a low level of privacy, and vice versa, there has to be somehow a compromise to ensure a level of security exists. Especially in private cloud environments where complete anonymity is not required to strengthen the security measures, the proposed mechanism can provide a good solution to organizations aiming to controlling access to their virtual environments across several teams and at the same time ensure that it is only accessed from authenticated hardware platforms with authenticated user accounts.

6.2 MECHANISM VERIFICATION

To verify the developed TPM authentication model and since it is integrated with the source code of the two main components of the solution VirtualBox and rDesktop, we need to ensure that it is not affecting any of their normal functionalities either being in a standalone mode or when interacting with each other. We conducted a set of tests for the standard use cases representing the normal functionalities performed by both tools without using the new TPM authentication mechanism to detect any impact caused by adding this new module to the normal operational activities.

Tables (8 & 9) below presents the test cases and their corresponding test results for all the normal functionalities of VirtualBox and rDesktop.

Table 8 TPM Model Verification - VirtualBox Use Cases

VirtualBox	
Use Case	Test Result
Invoking VirtualBox GUI (VirtualBox)	Pass
VirtualBox preferences modification through GUI	Pass
Managing VirtualBox through command line (VBoxManag & Vheadless)	Pass
VM Creation	Pass
VM Cloning (Full and Linked)	Pass
VM Deletion	Pass
VM Settings modification through GUI	Pass
VM Settings modification using command line	Pass

VirtualBox	
Use Case	Test Result
Default user Authentication library loaded (VBoxAuth - PAM) through GUI	Pass
Alternate user Authentication library loaded (VBoxAuthSimple) through GUI	Pass
Default user Authentication library loaded (VBoxAuth - PAM) through command line	Pass
Alternate user Authentication library loaded (VBoxAuthSimple) through command line	Pass
VRDP Authentication settings modification through GUI	Pass
VRDP Authentication settings modification through command line	Pass
Remote connections established to virtual machines not using External user Authentication (using either Null or Guest authentication)	Pass
Import and Export of VM images	Pass

Table 9 TPM Model Verification - rDesktop Use Cases

rDekstop	
Use Case	Test Result
rDesktop can connect to remote machines without TPM authentication if the "-w" option is not specified in the connection command	Pass
rDesktop usage info displayed if no arguments are passed	Pass

As shown in Tables (8 & 9) above, the new TPM authentication mechanism is not affecting any functionality for VirtualBox or rDesktop.

In addition, rDesktop can be used with or without TPM authentication using the “-w” command line option as needed. VirtualBox VRDP “Null” and “Guest” authentication are working normally without TPM authentication, only if VRDP authentication is set to “External” will the TPM authentication module be loaded and used with the specified user Authentication library for VirtualBox.

6.3 SECURITY ASSESSMENT

In this section, we discuss the security of the several model components along with future possible enhancements that can be made to increase the security level of the mechanism. The security of VirtualBox and rDesktop are excluded from the scope of this assessment.

6.3.1 Communication

To establish a secure communication channel between the clients and the VirtualBox server, we used OpenSSH [26] and configured authorized key pairs for each client machine so that users can connect seamlessly and transfer the TPM EK to the server without being promoted for a password.

Furthermore, scp being part of OpenSSH is used to securely transfer the TPM EK during the authentication process.

OpenSSH and scp provide an encrypted communication channel over the network using the SSH protocol and like all software tools, need to always be updated to their latest released versions to ensure that they are less vulnerable to security attacks. The latest OpenSSH version is 6.7 released in October 2014 [26].

6.3.2 Database

The Database used to host the ACL is SQLite version 3.8.3, SQLite is one of the most widely used Database engines for application development on multiple platforms and is being maintained by various vendors like Oracle, Adobe and Mozilla [27]. The latest available version is 3.8.6 released in August 2014 [27].

SQLite database is not requiring a server process, it accesses the disk directly and the database itself is a single file. In the scope of this work, the Database file is located on the local hard disk of the VirtualBox server machine however; it can be easily placed on a network storage device with secure access as long as the correct permissions are set.

As a future enhancement to the proposed model, the SQLite database can be encrypted to provide a more secure access using a passphrase and encryption keys. Two encryption tools for SQLite, SQLiteCrypt [53] and SQLCipher [54] can be used to encrypt the Database. While both uses 256-bit AES encryption, SQLCipher is known to have an overhead between 5-15% for the encryption [54] while SQLiteCrypt is known to have a good performance as it only encrypts and decrypts the data block being accessed and not the entire Database [53].

6.3.3 User Accounts

All user accounts used within the scope of this work are local accounts with the same user name and password on the clients and VirtualBox server to enable password - less communication – in addition to the authorized OpenSSH keys for each account - while using scp to transfer the TPM EK. We can achieve the same model using directory services (NIS / LDAP) where we can create centralized accounts that we can use across all machines.

The password-less communication setup has to be done per account per client machine, which ensures that each user must create RSA key pairs for his account on each client he will be using to access the VirtualBox server.

In addition, as an increased security measure, the user must provide the TPM owner password for his client to extract the TPM EK upon initiating the rDesktop connection to the VirtualBox server. This ensures that even if someone is using a client machine with an opened terminal from an authenticated user, he will not be able to connect to VirtualBox without knowing the TPM owner password.

As discussed in section 5.2, we can automate the TPM owner password input using the "expect" [43] script however this will weaken the security of the mechanism as the TPM owner password will be easier to attack since it has to be written in clear text in a local file on the disk.

6.3.4 File System

As discussed in section 6.3.2, the Database file is located on the local hard disk of the VirtualBox server machine but for enhanced security, it would be preferred to host it on a network storage device with secure access permissions.

The client TPM EK is extracted into a file located in the home directory of the user initiating the connection request to VirtualBox and sent with the connection request using scp to /tmp directory on the server. The destination folder on the server can be changed to any other local folder on the server or to a location on a more secure network storage device with the required access permissions to allowed users.

The local TPM EK file is removed from the home directory of the user on the client machine once it is sent to the server with the connection request, also the file on /tmp on the server is removed once the authentication process is completed and access is granted or rejected.

6.3.5 Security Threats

To summarize, Table (10) below lists the possible security threats to the new mechanism with explanation of each:

Table 10 New Mechanism Security Threats

Threat	Description	Mitigation
TPM Public EK of a machine is revealed	If someone can get hold of the TPM public EK of a machine and at the same time is able to alter the source code for rDesktop to use the compromised TPM EK and have knowledge of how VirtualBox is handling the Client machines authentication	As a future enhancement, use Private Key of TPM in a challenge response authentication process

Threat	Description	Mitigation
	Using TPM then he can have access to VM's in VirtualBox	
Database Location and Encryption	Database file is located on a local storage and is not encrypted, if someone can access the database file directly and alter the content then he can have access to VM's in VirtualBox	Move Database to a more secured network storage and encrypt the database file with tools like SQLiteCrypt and SQLCipher
Un-privileged access to the ACL Admin Module	If access to the ACL Admin Module is compromised then someone can have an un-privileged access to alter the contents of the ACL database	Restrict the access to the ACL Admin Module, Ensure that it is located in a secure network storage. As a future enhancement, additional authentication layer can be added before invoking this Admin Module

6.4 COST ANALYSIS

In this section, we review the cost associated with this new TPM authentication mechanism, from implementation as well as execution point of views.

6.4.1 Implementation Cost

For a service provider or an organization to implement this new mechanism, the following activities must be performed, for each of the activities we highlight the frequency at which they need to be performed.

- rDesktop must be built from its source files using the modified file listed in Appendix-B to incorporate the new TPM authentication functionality. This has to be performed on each client machine required to access the VirtualBox server. A normal user without a need for root privileges can perform it and in this case, this will only allow this user to use the modified version of rDesktop. Another alternative is to have a system administrator with root privileges to install it on the client machine for all users or better, to install it on a network shared location where all users on all systems can have access to it by adjusting their $PATH variable to point to the required binaries of the modified rDesktop tool

- SQLite package must be installed on the VirtualBox server or network storage as needed to host the ACL. This is a one-time operation that must be performed by a system administrator with root privileges to ensure secure access to the database

- VirtualBox must be built from its source files using the modified files listed in Appendices C, D & E to incorporate the new TPM authentication functionality. VirtualBox Extension pack must also be installed to allow for the "external"

authentication for virtual machine. This is a one-time operation to be performed on the server hosting the VirtualBox environment and must be performed by a system administrator with root privileges to ensure secure access on the VirtualBox server and also to ensure that the ACL hosted in the SQLite database is only accessed by the system administrator

- For each virtual machine created inside VirtualBox, the system administrator must define the VRDP authentication mechanism required; in this case, it should be set to "external" to load the TPM authentication module. In addition, for every virtual machine the network settings must be configured to use "NAT" to ensure that all connection requests are routed only through the host. This is a one-time operation per virtual machine whether the virtual machine is newly created in VirtualBox or is imported from a previous installation

- The system administrator of VirtualBox will also need to define the authentication library used with "external" authentication methods for virtual machines. This is a one-time operation performed on the server level to configure either using "VBoxAuth" or "VBoxAuthSimple" libraries

- The system administrator must create the user accounts for users accessing the VirtualBox environment; this can be done using accounts already created on a directory service like NIS or LDAP. If "VBoxAuthSimple" authentication library is used, the system administrator will also need to add the allowed accounts per virtual machine to the XML settings file per VM. This is an on-going operation as needed to add or modify users of the system

- The initial creation of the ACL database as well as the on-going addition or removal of VirtualBox virtual machines to the ACL is done automatically with the creation or removal of virtual machines in VirtualBox. No implementation cost is associated with this activity

- To add clients to the ACL of the TPM authentication module, the system administrator will need to use the developed ACL administration module script listed in Appendix-A. It can be placed locally on the VirtualBox server or on a network storage location with access permissions to the SQLite database. This is an on-going operation to add, remove or modify client machines for each virtual machine in VirtualBox

- User requiring access to the VirtualBox server will need to create RSA key pairs for SSH and add them to the "authorized_Keys" file to allow for password-less connection from each client being used to the VirtualBox server. This is a one-time operation per user on each client machine

- For each client machine, TPM must be enabled and initialized and ownership set with a password that will be used during establishing the connection to the VirtualBox server. This is a one-time operation that is performed by the system

administrator through the installation of TPM management tools on the client machine and setting the owner password

6.4.2 Execution Cost

The execution cost associated with the use of this new TPM authentication mechanism can be classified as follows:

- Network overhead introduced with the sending of the TPM EK from the client machine to the VirtualBox server

The original TPM EK file is of size 4 KB and contains some un-needed lines so we truncated the file to limit the data we send form the client to the server to include only the public key signature lines as illustrated in Figures (20) and (21) below. The new truncated file is almost 4 KB in size as well so the network overhead introduced by this new mechanism is only an additional 4 KB over the connection request.

- Overall cost overhead in terms of connection time introduced with the use of the new authentication mechanism

As discussed in chapter 5, Using TPM Authentication with PAM (VBoxAuth) introduced a delay of 0.74 seconds while the original average runtime using PAM alone is 0.2 seconds. The cost overhead accordingly is around 370%.

In addition, Using TPM Authentication with VBoxAuthSimple introduced a delay of 0.7 seconds while the original average runtime using VBoxAuthSimple is 0.2 seconds. The cost overhead accordingly is around 350%.

The introduced overall delay in establishing the connection is due to the fact of introducing a dependency on the TPM EK file transfer between the client and the server machines over the network and the time needed to read it and compare its content with the clients' entries in the ACL. Since the mechanism is network dependent, the connection processing time might vary from one time to another based on the network utilization however, the calculated standard deviation over multiple runs shows a maximum of 0.07 seconds variation when using VBoxAuthSimple and only 0.05 seconds when using PAM (VBoxAuth) as illustrated in Figure (16) in section 5.4.6.

This introduced delay is only required once with the initial connection establishment between the client machine and the VirtualBox server and no overhead is added for any further requests once the remote desktop connection is established. Even though the overhead is ranging from 350-370%, the actual delay in terms of time is a maximum of 0.74 seconds.

- The use of network storage devices instead of local hard disks on clients and server machines to host the SQLite database or the temporary TPM EK file might also introduce some cost during the execution depending on the I/O throughput of

the storage device. This is outside the scope of this work but the assumption would be that the introduced overhead at that time would also be very low.

```
Public Endorsement Key:
  Version:  01010000
  Usage:    0x0002 (Unknown)
  Flags:    0x00000000 (!VOLATILE, !MIGRATABLE, !REDIRECTION)
  AuthUsage: 0x00 (Never)
  Algorithm:        0x00000020 (Unknown)
  Encryption Scheme: 0x00000012 (Unknown)
  Signature Scheme:  0x00000010 (Unknown)
  Public Key:
        8daeb10b 1735639b cb47b440 6a8abbc4 7364084c 103c05a2 c9cf8a9e cb7f6242
        2346e257 efe46735 edbee3dd 44a1d899 4d5032ae 4b7829d0 595adaa6 29bad9f7
        5f560574 a21a163b 8aeae94b 54563f07 1db34c39 3813a159 9386e533 c62cc451
        c36107eb 87deb0c9 de3d4a79 c0038cd8 0ff3340b 7f5fb33f 5ed41de2 ce56f242
        2b0dd5a2 6f29d0df 3aefac1d ef20c023 3ed3a9a3 f7892733 7f543157 74c05f04
        3717db0d d44e001a 6ed154fe d75f5e09 325a6ee5 f960975a e70e35bc f4e5b093
        f51e46bc 5a28295e bf4366b8 43ecf70c fb42fa60 fc3383cd 015cf308 41a8930a
        18eb3ec7 52798583 08067eae 13686186 b6a09d26 5066cb1b bac686e9 05392a47
```

Figure 20 Original TPM EK

```
8daeb10b 1735639b cb47b440 6a8abbc4 7364084c 103c05a2 c9cf8a9e cb7f6242
2346e257 efe46735 edbee3dd 44a1d899 4d5032ae 4b7829d0 595adaa6 29bad9f7
5f560574 a21a163b 8aeae94b 54563f07 1db34c39 3813a159 9386e533 c62cc451
c36107eb 87deb0c9 de3d4a79 c0038cd8 0ff3340b 7f5fb33f 5ed41de2 ce56f242
2b0dd5a2 6f29d0df 3aefac1d ef20c023 3ed3a9a3 f7892733 7f543157 74c05f04
3717db0d d44e001a 6ed154fe d75f5e09 325a6ee5 f960975a e70e35bc f4e5b093
f51e46bc 5a28295e bf4366b8 43ecf70c fb42fa60 fc3383cd 015cf308 41a8930a
18eb3ec7 52798583 08067eae 13686186 b6a09d26 5066cb1b bac686e9 05392a47
```

Figure 21 Truncated TPM EK

7 CONCLUSION AND FUTURE DIRECTIONS

This work discussed a new machine authentication mechanism using TPM that allows organizations and cloud service providers the possibility of identifying physical hardware clients attempting to establish a remote desktop connection to the graphical user interface of virtual machines. This mechanism in conjunction with the normal user authentication processes can provide a better way of controlling and restricting the access to virtual machines based on a pre-defined access control list. Accordingly, we can use it to ensure proper setup of multi-tenancy environments.

It also provides a way to prevent malicious insiders from performing un-authorized activities as it combines the hardware and the user accounts into a single authentication process. This authentication process depends mainly on identifying hardware clients and associating it with several user level authentication steps that combines the TPM owner password per machine with the normal user name and password authentication as well as an access control list to determine the allowed access privileges.

This new authentication mechanism proves that it is possible to use TPM Endorsement Key directly to identify a hardware client without using a third party Privacy Certificate Authority or using Direct Anonymous Attestation. The only concern with using this new mechanism would be anonymity and privacy as the identity of the hardware client has to be revealed however, specifically in private clouds or in any cloud environment where security is the main focus, this mechanism can be used without any concerns.

The cost overhead associated with using this new mechanism is relatively small, with a maximum value of 0.74 seconds per connection request that is only added once per session. This small value can be overlooked taken into consideration the security benefits gained from using this mechanism.

As a promising path, we can use TPM Private Keys for a challenge- response key exchange authentication process to secure this new mechanism in case the TPM public Keys are compromised. Something similar to STS Protocol (Station-to-Station) which is a variation of the Diffie-Hellman, first key-exchange protocol but altered to prevent the man in the middle attack by using mutual authentication between the two communicating parties [55], [56].

Also, this mechanism can be ported to other platforms to support different clients other than Linux or even support other Hypervisors than VirtualBox. In addition, a beneficial future direction would be to associate it with other TPM related research activities to enable the verification of the loaded software stack on each client and to perform an identity and integrity check on the cloud server platform.

With the association of this mechanism that is mainly focusing on the client side with other mechanisms that target the server side, cloud service providers as well as organizations hosting private clouds would benefit from a complete secure solution to their cloud environments.

They can ensure access is granted to specific client machines as defined in an ACL that also have a desirable software stack to specific resources, whether to virtual machines or to storage devices or even to networks.

At the same time, clients would benefit from having the assurance that the identity and integrity of the cloud server provider is checked before the connection is established and they start accessing the required cloud service.

REFERENCES

[1] Brian Berger, EVP Marketing & Sales, TCG Director. (2007). *How to Deploy Trusted Systems "A Practical Guide"* [Online].

Available at:

http://www.trustedcomputinggroup.org/files/resource_files/AC0C0AF8-1D09-3519-AD9972448FFE4AFF/RSA_EMEA_2007_TechShowcase_Final.pdf

[Accessed: 24 October 2014]

[2] Dr. Michael Waidner, IBM Distinguished Engineer, CTO Security for IBM. (2010, April 17). *Security and Cloud Computing* [Online].

Available at:

https://www.sit.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_SIT/Presentations/100302a%20Cloud%20Security%20Lecture.pdf

[Accessed: 24 October 2014]

[3] Peter Mell, Timothy Grance. (2011, September). *The NIST Definition of Cloud Computing.* NIST Special Publication 800-145 [Online].

Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[Accessed: 24 October 2014]

[4] Rajkumar Buyya, James Broberg, Andrzej Goscinski."Introduction to Cloud Computing," in *Cloud Computing Principles and Paradigms*. New Jersey: Wiley, 2011, pp. 3-28

[5] Tim Mather, Subra Kumarawamy, Shahed Latif. "What is Cloud Computing?" in *Cloud Security and Privacy, An Enterprise Perspective on Risks and Compliance*. CA: O'Reilly, 2009, pp. 7-25

[6] Cloud Security Alliance. *Security Guidance for critical areas of focus in cloud computing v3.0*

Available at: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

[Accessed: 24 October 2014]

[7] Cloud Computing Wikipedia [Online].

Available at: http://en.wikipedia.org/wiki/Cloud_computing

[Accessed: 24 October 2014]

[8] VirtualBox Manual Pages, Chapter 1. *First Steps* [Online].

Available at: https://www.virtualbox.org/manual/ch01.html

[Accessed: 24 October 2014]

[9] Jacobo Ros, "Security in the Cloud: The threat of coexist with an unknown tenant on a public environment," M.Sc. Thesis in Information Security, Royal Holloway, University of London., United Kingdom, 2012.

[10] Cloud Security Alliance. *Best Practices to secure the cloud with identity management* [Online].

Available at:

https://blog.cloudsecurityalliance.org/2012/08/13/best-practices-to-secure-the-cloud-with-identity-management/

[Accessed: 24 October 2014]

[11] Cloud Security Alliance, (2013, February). *The Notorious Nine Cloud Computing Top Threats in 2013* [Online].

Available at:

https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

[Accessed: 24 October 2014]

[12] Evan R. Sparks. *A Security Assessment of Trusted Platform Modules* [Online].

Available at: http://www.cs.dartmouth.edu/~pkilab/sparks/

[Accessed: 24 October 2014]

[13] Bill Hewitt, "Trusted Computing and the Trusted Platform Module: What All the Fuss Is About," Security Course Projects, Harvey Mudd College, Claremont, CA, USA, 2006.

[14] Trusted Computing Group, *Cloud Computing and Security –A Natural Match* [Online].

Available at:

http://www.trustedcomputinggroup.org/resources/cloud_computing_and_security__a_natural_match

[Accessed: 11 November 2014]

[15] Trusted Computing Group, *How to Use the TPM: A Guide to Hardware-Based Endpoint Security*, [Online].

Available at:

http://www.trustedcomputinggroup.org/resources/how_to_use_the_tpm_a_guide_to_hardwarebased_endpoint_security

[Accessed: 11 November 2014]

[16] Ned Smith, Trusted Computing Group. *Integrating User Authentication with Platform Authentication and Key Management.* CardTech/SecureTech 2007, [Online]

Available at:

http://www.trustedcomputinggroup.org/files/resource_files/ABF954B0-1D09-3519-AD6FB365FA7E21D1/CardTech-SecTech_Ned_Smith_v1.pdf

[Accessed: 11 November 2014]

[17] Microsoft Technet Article. *TPM Base Services* [Online].

Available at: http://technet.microsoft.com/en-us/library/cc734063.aspx

[Accessed: 24 October 2014]

[18] Bryan Parno. *The Trusted Platform Module (TPM) and Sealed Storage*, [Online]

Available at:

http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C634AF8D345B512BE59C33CCBF292AB1?doi=10.1.1.187.2027&rep=rep1&type=pdf

[Accessed: 24 October 2014]

[19] Tien Tuan Anh Dinh, Mark Dermot Ryan, *Trusted Computing: TCG proposals*. Computer Security Lecture Notes, School of Computer Science, University of Birmingham, United Kingdom, 2006.

[20] Hardeep Uppal. *Enabling Trusted Distributed Control with Remote Attestation* [Online].

Available at:

http://courses.cs.washington.edu/courses/cse550/10au/other_for_site/cse551_final_paper_draft5_final.pdf

[Accessed: 24 October 2014]

[21] Zhen Peng Liu et al., 2013, Advanced Materials Research, 756-759, 3209

[22] TPM Chip in Windows 8 Lays Foundation for Widespread Enhancements to Hardware-Based Security [Online].

Available at:

http://threatpost.com/tpm-chip-windows-8-lays-foundation-widespread-enhancements-hardware-based-security-102612/

[Accessed: 24 October 2014]

[23] Microsoft Technet Article. *Deployment Planning for BitLocker Drive Encryption for Windows Vista* [Online].

Available at: http://technet.microsoft.com/en-us/library/dd126731.aspx,

[Accessed: 1 November 2014]

[24] Ubuntu 12.0.4 release download page, [Online]

Available at: http://releases.ubuntu.com/12.04/

[Accessed: 6 September 2013]

[25] Oracle VM VirtualBox download page, [Online].

Available at: https://www.virtualbox.org/wiki/Downloads

[Accessed: 6 September 2013]

[26] OpenSSH download page, [Online].

Available at: http://www.openssh.com/portable.html

[Accessed: 6 September 2013]

[27] SQLite download page, [Online].

Available at: http://www.sqlite.org/download.html

[Accessed: 6 September 2013]

[28] Trusted Computing Group, *TPM Specifications* [Online].

Available at:

https://www.trustedcomputinggroup.org/resources/tpm_main_specification

[Accessed: 6 September 2013]

[29] rDesktop download page [Online].

Available at: http://www.rdesktop.org/#download

[Accessed: 6 September 2013]

[30] VirtualBox 4.3.8 Linux Build Instructions [Online].

Available at: https://www.virtualbox.org/wiki/Linux%20build%20instructions

[Accessed: 10 January 2014]

[31] VirtualBox Manual Chapter 7.1.2. *VBoxHeadless, the remote desktop server*, [Online].

Available at: http://www.virtualbox.org/manual/ch07.html

[Accessed: 18 April 2014]

[32] VirtualBox Manual Chapter 7.1. *Remote Display (VRDP Support)* [Online].

Available at: https://www.virtualbox.org/manual/ch07.html#vrde

[Accessed: 18 April 2014]

[33] VirtualBox Manual Chapter 1.5. *Installing VirtualBox and Extension Packs* [Online].

Available at: https://www.virtualbox.org/manual/ch01.html#intro-installing

[Accessed: 18 April 2014]

[34] VirtualBox Manual Chapter 8.36. *VBoxManage extpack* [Online].

Available at: https://www.virtualbox.org/manual/ch08.html#vboxmanage-extpack

[Accessed: 18 April 2014]

[35] VirtualBox Manual Chapter 7.1.5. *RDP Authentication* [Online].

Available at: http://www.virtualbox.org/manual/ch07.html#vbox-auth

[Accessed: 28 April 2014]

[36] How to configure PAM authentication on Ubuntu for VirtualBox/ RDP, [Online].

Available at: https://help.ubuntu.com/community/VirtualBox/RDP

[Accessed: 9 May 2014]

[37] Adding VirtualBox Simple External Authentication, [Online].

Available at:

http://blog.oracle48.nl/adding-virtualbox-simple-external-authentication/

[Accessed: 8 May 2014]

[38] VirtualBox Manual Chapter 6.2. *Introduction to Networking modes* [Online].

Available at: http://www.virtualbox.org/manual/ch06.html#networkingmodes

[Accessed: 25 April 2014]

[39] TPM Trousers management tool [Online].

Available at: http://trousers.sourceforge.net/

[Accessed: 10 January 2014]

[40] How to use a TPM with Linux [Online].

Available at:

http://www.grounation.org/?post/2008/07/04/8-how-to-use-a-tpm-with-linux

[Accessed: 24 January 2014]

[41] rDesktop 1.7.1 README

(Included with Download package from http://www.rdesktop.org/#download)

[42] Oracle Linux Administrator's Guide for Release 6. Chapter 24. *OpenSSH Configuration*, Section 24.5, *Using the OpenSSH Utilities* [Online].

Available at: http://docs.oracle.com/cd/E37670_01/E41138/html/ch24s05.html

[Accessed: 16 March 2014]

[43] Expect Linux Man Page [Online].

Available at: http://linux.die.net/man/1/expect

[Accessed: 17 April 2014]

[44] Expect Scripts examples [Online].

Available at: http://www.thegeekstuff.com/2010/10/expect-examples/

[Accessed: 4 April 2014]

[45] O. Tange (2011): *GNU Parallel - The Command-Line Power Tool*, ;login: The USENIX Magazine, February 2011:42-47

[46] GNU Parallel Man Page [Online].

Available at: http://www.gnu.org/software/parallel/man.html

[Accessed: 14 April 2014]

[47] Ariel Segall, *TPM Keys Creating, Certifying, and Using Them*, [Online]

Available at:

http://opensecuritytraining.info/IntroToTrustedComputing_files/Day1-7-tpm-keys.pdf

[Accessed: 15 October 2014]

[48] Ben Smyth, Mark Ryan, and Liqun Chen. *Direct Anonymous Attestation (DAA): Ensuring Privacy with Corrupt Administrators*, School of Computer Science, University of Birmingham, UK. ESAS'07 Proceedings of the 4th European conference on Security and privacy in ad-hoc and sensor networks, Pages 218-231.

[49] Jan Camenisch, *Direct Anonymous Attestation Explained*, IBM Research. July 25, 2007, [Online]

Available at:

https://idemix.files.wordpress.com/2009/08/camenisch2007-direct_anonymous_attestation_explained.pdf

[Accessed: 11 November 2014]

[50] Jan Camenisch, *Direct Anonymous Attestation: Achieving Privacy in Remote Authentication*. IBM Zurich Research Laboratory. Joint work with Ernie Brickell, Intel, and Liqun Chen, HP. ZISC Information Security Colloquium, June 15, 2004 [Online]

Available at: http://www.zurich.ibm.com/security/daa/daa-slides-ZISC.pdf

[Accessed: 11 November 2014]

[51] Liqun Chen, *Direct Anonymous Attestation (DAA)*, Trusted Systems Laboratory Hewlett Packard Laboratories, Bristol 12 October 2005 [Online]

Available at:

https://www.trustedcomputinggroup.org/files/resource_files/AC0E3E05-1D09-3519-ADCD93471A61681A/051012_DAA-slides.pdf

[Accessed: 11 November 2014]

[52] Direct Anonymous Attestation Wikipedia [Online].

Available at: http://en.wikipedia.org/wiki/Direct_Anonymous_Attestation

[Accessed: 15 October 2014]

[53] SQLiteCrypt Encryption tool for SQLite [Online].

Available at: http://sqlite-crypt.com/

[Accessed: 15 October 2014]

[54] SQLCipher Encryption tool for SQLite [Online].

Available at: https://www.zetetic.net/sqlcipher/

[Accessed: 15 October 2014]

[55] Pranav J Vyas and Bhushan H Trivedi. *Analysis of Key Exchange Protocols using Session Keys*, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868. Foundation of Computer Science FCS, New York, USA. Volume 1– No.4, February 2012 – www.ijais.org

[56] Razieh Mokhtarnameh et al. *A Comparison Study on Key Exchange-Authentication protocol*, International Journal of Computer Applications (0975 – 8887) Volume 7– No.5, September 2010

APPENDIX A – ACL ADMINISTRATION MODULE

```
#! /bin/bash

MENU="
1   Display Clients Database
2   Export Clients Database to CSV File
3   Add a new client
4   Modify an existing client name (All Entries)
5   Modify an existing client TPM EK (All Entries)
6   Delete an existing client (Single Entry)
7   Delete an existing client (All Entries)
8   Clear all Database Entries
9   Quit
"

# Define SQLite DB

SQLite_DB="/home/assers/VirtualBox_Clients.db"

while true; do
  clear
  echo -e "VirtualBox ACL Administration Module"
  echo -e
"================================================="
  echo "$MENU"
  echo -n "Please choose one of the above options: "
  read INPUT # Read user input and assign it to variable
INPUT

  case $INPUT in
    1)
    echo -e
    sqlite3 $SQLite_DB "select * from Clients"
    echo -e
    echo press ENTER to continue
    read
    ;;
    2)
    echo -n "Please enter full path and file name to
export to :"
    read CSVPath    #Read csv file path from user and
assign it to variable CSVPath
    sqlite3 -header -csv $SQLite_DB "select * from
Clients;" > "$CSVPath"
    echo -e "Clients Database exported to '$CSVPath'"
    echo press ENTER to continue
    read
    ;;
```

```
    3)
    echo -n "Please choose an existing VM Name from
Database to assign Client to :"
    read VMName #Read existing VM Name from user and add
it to variable VMName
    VMExist=$(sqlite3 $SQLite_DB "SELECT EXISTS(SELECT 1
FROM Clients WHERE VM_Name='$VMName' LIMIT 1)")
    if [[ $VMExist > 0 ]] ; then
        echo -n "Please enter New Client Name :"
        read ClientName    #Read new Client Name from
user and add it to DB
        Client_newExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE VM_Name='$VMName' AND
Client_Name='$ClientName' LIMIT 1)")
        if [[ $Client_newExist > 0 ]] ; then
           echo -e "Client '$ClientName' already exists
for VM '$VMName'...Aborting!!"
        else
          echo -n "Please enter full path to the TPM EK
file for new Client :"
          read EKPath #Read Full path for TPM EK file
from user and add it to variable EKPath
       if [ -f $EKPath ] ; then
           EKData=$(sed -n '10,17p' "$EKPath") #Read the
full EK File for the client, extract the public key and
assign it to variable EKData
           UUID=$(sqlite3 $SQLite_DB "select UUID from
Clients where VM_Name='$VMName' LIMIT 1 OFFSET 0")
           sqlite3 $SQLite_DB "insert into
Clients(VM_Name,UUID,Client_Name,EK)
values('$VMName','$UUID','$ClientName','$EKData')"
           echo -e "New Client '$ClientName' with TPM EK
file '$EKPath' added to Access list for VM '$VMName'"
       else
        echo -e "TPM EK file '$EKPath'does not exist,
please enter a valid path!"
       fi
     fi
    else
        echo -e "The VM '$VMName' does not exist in the
Clients Database, please make sure that it's created
first in VirtualBox before adding clients to its Access
List....Aborting!!"
    fi
    echo press ENTER to continue
    read
    ;;
    4)
    echo -n "Please enter Client Name to Modify :"
```

```
    read ClientName_old #Read existing Client Name from
user and add it to variable ClientName_old
    ClientExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE
Client_Name='$ClientName_old' LIMIT 1)")
    if [[ $ClientExist > 0 ]] ; then
       echo -n "Please enter the new Client Name :"
       read ClientName_new #Read new Client Name from
user and add it to variable ClientName_new
       NewClientExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE
Client_Name='$ClientName_new' LIMIT 1)")
       if [[ $NewClientExist > 0 ]] ; then
          echo -e "The Client '$ClientName_new' already
exist in the Clients Database....Aborting!!"
       else
          sqlite3 $SQLite_DB "update Clients SET
Client_Name=('$ClientName_new') WHERE
Client_Name=('$ClientName_old')"
          echo -e "All entries for Client
'$ClientName_old' changed successfully to
'$ClientName_new'"
       fi
    else
        echo -e "The Client '$ClientName_old' does not
exist in the Clients Database....Aborting!!"
    fi
    echo press ENTER to continue
    read
    ;;
    5)
    echo -n "Please enter Client Name to Modify :"
    read ClientName_old #Read existing Client Name from
user and add it to variable ClientName_old
    ClientExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE
Client_Name='$ClientName_old' LIMIT 1)")
    if [[ $ClientExist > 0 ]] ; then
       echo -n "Please enter full path to the TPM EK file
:"
       read EKPath_new #Read Full path for TPM EK file
from user and add it to variable EKPath_new
       if [ -f $EKPath_new ] ; then
        EKData_new=$(sed -n '10,17p' "$EKPath_new") #Read
TPM EK File and assign it to variable EKData_new
        sqlite3 $SQLite_DB "update Clients set
EK=('$EKData_new') Where Client_Name='$ClientName_old'"
        echo -e "New TPM EK for '$ClientName_old' added
to all Entries in the Database"
       else
```

```
     echo -e "TPM EK file '$EKPath_new'does not exist,
please enter a valid path!"
       fi
    else
        echo -e "The Client '$ClientName_old' does not
exist in the Clients Database....Aborting!!"
    fi
    echo press ENTER to continue
    read
    ;;
    6)
    echo -n "Please enter the Client Name to Delete :"
    read ClientNameDel #Read new Client Name from user
and add it to variable ClientName
    DelClientExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE
Client_Name='$ClientNameDel' LIMIT 1)")
    if [[ $DelClientExist > 0 ]] ; then
       echo -n "Please choose an existing VM Name from
Database to Modify :"
       read VMName #Read existing VM Name from user and
add it to variable VMName
       VMExist=$(sqlite3 $SQLite_DB "SELECT EXISTS(SELECT
1 FROM Clients WHERE VM_Name='$VMName' LIMIT 1)")
       if [[ $VMExist > 0 ]] ; then
          echo -n "This will delete Client
'$ClientNameDel' from ACL of VM '$VMName', Are you sure
you want to continue? [Y/N]:"
          read -n 1 REPLY
          if [[ $REPLY =~ ^[Yy]$ ]]
             then
             sqlite3 $SQLite_DB "Delete FROM Clients
WHERE Client_Name=('$ClientNameDel') AND
VM_Name='$VMName'"
             echo -e
             echo -e "Entry for '$ClientNameDel' deleted
successfully"
             echo -e
          else
             echo -e
             echo -e "Aborting..!!"
          fi
       else
          echo -e "The VM '$VMName' does not exist in the
Clients Database....Aborting!!"
       fi
    else
       echo -e "The Client '$ClientNameDel' does not
exist in the Clients Database....Aborting!!"
    fi
```

```
    echo press ENTER to continue
    read
    ;;
    7)
    echo -n "Please enter the Client Name to Delete :"
    read ClientNameDel #Read new Client Name from user
and add it to variable ClientName
    DelClientExist=$(sqlite3 $SQLite_DB "SELECT
EXISTS(SELECT 1 FROM Clients WHERE
Client_Name='$ClientNameDel' LIMIT 1)")
    if [[ $DelClientExist > 0 ]] ; then
       echo -n "This will delete all entries for this
Client in the Database, Are you sure you want to
continue? [Y/N]:"
       read -n 1 REPLY
       if [[ $REPLY =~ ^[Yy]$ ]]
          then
          sqlite3 $SQLite_DB "Delete FROM Clients WHERE
Client_Name=('$ClientNameDel')"
          echo -e
          echo -e "All entries for Client
'$ClientNameDel' deleted successfully"
          echo -e
       else
          echo -e
          echo -e "Aborting..!!"
       fi
    else
       echo -e "The Client '$ClientNameDel' does not
exist in the Clients Database....Aborting!!"
    fi
    echo press ENTER to continue
    read
    ;;
    8)
    ClearDB=$(sqlite3 $SQLite_DB "SELECT EXISTS(SELECT 1
FROM Clients LIMIT 1)")
    if [[ $ClearDB > 0 ]]; then
       echo -n "This will delete all entries from Clients
Database, Are you sure you want to continue? [Y/N]:"
       read -n 1 REPLY
       if [[ $REPLY =~ ^[Yy]$ ]]
       then
           sqlite3 $SQLite_DB "DELETE FROM Clients"
           echo -e
           echo -e "Clients Database cleared, please add
VM's from VirtualBox !!"
           echo -e
           break
       else
```

```
            echo -e
            echo "Aborting..!!"
        fi
    else
        echo -e "Clients Database is already empty, please
add VM's from VirtualBox.....Aborting!!"
        break
    fi
    echo press ENTER to continue
    read
    ;;
    9|q|Q) # If user presses 7, q or Q we terminate
    exit 0
    ;;
    *) # All other user input results in an usage message
    #clear
    echo -e
    echo Please choose alternatves 1-7
    sleep 2
    ;;
  esac

done
```

APPENDIX B – rDesktop.c

```
--- ./orig_files/rdesktop.c   2014-06-08
19:57:59.585251791 +0200
+++ ./modified_files/rdesktop.c    2014-06-08
20:03:58.021242809 +0200
@@ -50,10 +50,13 @@

 #include "ssl.h"

+//Asser Include Lines
+#include <stdlib.h>
+
 #define RDESKTOP_LICENSE_STORE
"/.local/share/rdesktop/licenses"

 uint8 g_static_rdesktop_salt_16[16] = {
-    0xb8, 0x82, 0x29, 0x31, 0xc5, 0x39, 0xd9, 0x44,
+    0xb8, 0x82, 0x29, 0x31, 0xc5, 0x39, 0xd9, 0x44,
     0x54, 0x15, 0x5e, 0x14, 0x71, 0x38, 0xd5, 0x4d
 };

@@ -147,7 +150,9 @@
          "Version " PACKAGE_VERSION ". Copyright (C)
1999-2011 Matthew Chapman et al.\n");
     fprintf(stderr, "See http://www.rdesktop.org/ for
more information.\n\n");

-    fprintf(stderr, "Usage: %s [options]
server[:port]\n", program);
+    //Asser - Modified rDesktop Usage Notes
+    fprintf(stderr, "rDesktop with TPM Support Usage: %s
-u USERNAME -p - [Any other options] -w
server[:VRDP_Port]\n", program);
+    fprintf(stderr, "rDesktop Usage: %s [options]
server[:port]\n", program);
 #ifdef RDP2VNC
     fprintf(stderr, "   -V: vnc port\n");
     fprintf(stderr, "   -Q: defer time (ms)\n");
@@ -460,6 +465,16 @@
 main(int argc, char *argv[])
 {
     char server[64];
+
+    //Asser - Variables
+    char server_copy[64];
+    char scp_command[512];
+    char tpm_command[512];
+    char rm_command[512];
```

```
+    char *absServer;
+    int tpm_return_val;
+    int scp_return_val;
+
     char fullhostname[64];
     char domain[256];
     char password[64];
@@ -509,7 +524,7 @@
 #endif

     while ((c = getopt(argc, argv,
-                  VNCOPT
"Au:L:d:s:c:p:n:k:g:fbBeEmzCDKS:T:NX:a:x:Pr:045h?")) != -
1)
+                  VNCOPT
"Au:L:d:s:c:p:n:k:g:fbBeEmzCDKS:T:NX:a:x:Pr:045wh?")) !=
-1)
     {
          switch (c)
          {
@@ -845,6 +860,50 @@
                    g_use_rdp5 = True;
                    break;

+//Asser - Main lines
+              case 'w':
+
sprintf(tpm_command,"/usr/sbin/tpm_getpubek >
~/Long_TPM_PubEK.`hostname`.%s;echo $?",g_username);
+                FILE* pipe = popen(tpm_command, "r");
+                if(pipe)
+                {
+                    fscanf(pipe,"%d",&tpm_return_val);
+                }
+                pclose(pipe);
+                if (tpm_return_val == 0)
+                {
+                    fprintf(stderr, "\nTPM Public EK
generated and will be sent to remote server!\n\n");
+                    STRNCPY(server, argv[optind],
sizeof(server));
+                    STRNCPY(server_copy, server,
sizeof(server));
+                    absServer=strtok(server_copy,":");
+                    sprintf(scp_command,"sed -n '10,17p'
~/Long_TPM_PubEK.`hostname`.%s >
~/TPM_PubEK.`hostname`.%s;scp ~/TPM_PubEK.`hostname`.%s
%s@%s:/tmp;echo
$?",g_username,g_username,g_username,g_username,absServer
);
```

```
+                   FILE* pipe = popen(scp_command,
"r");
+                   if(pipe)
+                   {
+
fscanf(pipe,"%d",&scp_return_val);
+                   }
+                   pclose(pipe);
+                   if (scp_return_val == 0)
+                   {
+                       fprintf(stderr, "\nTPM Public EK
sent to remote server and will be deleted from local
machine!\n\n");
+                       sprintf(rm_command,"rm
~/*TPM_PubEK.`hostname`.%s",g_username);
+                       system((char*)rm_command);
+                   }
+                   else
+                   {
+                       fprintf(stderr, "\nTPM Public EK
was not sent to remote server, please make sure that your
client and the server host are both running openSSH and
that your account have the required access to perform
secure file copy to the server host!\n\n");
+                       fprintf(stderr,"\nDeleting local
copy of the generated TPM Public EK & exiting!\n\n");
+                       sprintf(rm_command,"rm
~/*TPM_PubEK.`hostname`.%s",g_username);
+                       system((char*)rm_command);
+                       exit(1);
+                   }
+               }
+               else
+               {
+                   fprintf(stderr, "\nTPM Public EK
generation failed, please make sure that you are running
from a machine with a TPM that is activated and you have
the TPM owner password...exiting!\n\n");
+                   exit(2);
+               }
+               break;
+
               case 'h':
               case '?':
               default:
@@ -860,6 +919,7 @@
     }

     STRNCPY(server, argv[optind], sizeof(server));
+
```

```
     parse_server_and_port(server);

     if (g_seamless_rdp)
@@ -1006,7 +1066,7 @@
          if (!rdp_connect(server, flags, domain,
password, shell, directory, g_redirect))
               return EX_PROTOCOL;

-         /* By setting encryption to False here, we have
an encrypted login
+         /* By setting encryption to False here, we have
an encrypted login
             packet but unencrypted transfer of other
packets */
          if (!g_packet_encryption)
               g_encryption = False;
@@ -1501,7 +1561,7 @@
     return ret;
 }

-static int
+static int
 safe_mkdir(const char *path, int mask)
 {
     int res = 0;
@@ -1520,7 +1580,7 @@
     return 0;
 }

-static int
+static int
 mkdir_p(const char *path, int mask)
 {
     int res;
@@ -1538,11 +1598,11 @@
          errno = E2BIG;
          return -1;
     }
-
-    res = 0;
+
+    res = 0;
     pt[0] = bp[0] = '\0';
     strcpy(bp, path);
-
+
     ptok = strtok(bp, "/");
     if (ptok == NULL)
          return safe_mkdir(path,mask);
@@ -1551,14 +1611,14 @@
```

```
     {
          if (ptok != bp)
               strcat(pt, "/");
-
-         strcat(pt, ptok);
+
+         strcat(pt, ptok);
          res = safe_mkdir(pt, mask);
          if (res != 0)
               return res;
-
+
     } while ((ptok = strtok(NULL, "/")) != NULL);
-
+
     return 0;
 }

@@ -1578,7 +1638,7 @@
     sec_hash_sha1_16(ho, hi, g_static_rdesktop_salt_16);
     sec_hash_to_string(hash, 40, ho, 22);

-    snprintf(path, PATH_MAX,
"%s"RDESKTOP_LICENSE_STORE"/%s.cal",
+    snprintf(path, PATH_MAX,
"%s"RDESKTOP_LICENSE_STORE"/%s.cal",
           home, hash);
     path[sizeof(path)-1] = '\0';

@@ -1586,7 +1646,7 @@
     if (fd == -1)
     {
          /* fallback to try reading old license file */
-         snprintf(path, PATH_MAX,
"%s/.rdesktop/license.%s",
+         snprintf(path, PATH_MAX,
"%s/.rdesktop/license.%s",
                home, g_hostname);
          path[sizeof(path)-1] = '\0';
          if ((fd = open(path, O_RDONLY)) == -1)
@@ -1628,9 +1688,9 @@
     sec_hash_sha1_16(ho, hi, g_static_rdesktop_salt_16);
     sec_hash_to_string(hash, 40, ho, 20);

-    /* write licence to {sha1}.cal.new, then atomically
+    /* write licence to {sha1}.cal.new, then atomically
        rename to {sha1}.cal */
-    snprintf(path, PATH_MAX,
"%s"RDESKTOP_LICENSE_STORE"/%s.cal",
```

```
+    snprintf(path, PATH_MAX,
"%s"RDESKTOP_LICENSE_STORE"/%s.cal",
           home, hash);
     path[sizeof(path)-1] = '\0';
```

APPENDIX C – VirtualBoxImpl.cpp

```
--- ./orig_files/VirtualBoxImpl.cpp     2014-06-08
19:55:57.901254840 +0200
+++ ./modified_files/VirtualBoxImpl.cpp 2014-06-08
20:05:08.629241039 +0200
@@ -32,6 +32,8 @@
 #include <iprt/uuid.h>
 #include <iprt/cpp/xml.h>

+#include <stdio.h>
+
 #include <VBox/com/com.h>
 #include <VBox/com/array.h>
 #include "VBox/com/EventQueue.h"
@@ -90,6 +92,7 @@

////////////////////////////////////////////////////////
//////////////////////

 #define VBOX_GLOBAL_SETTINGS_FILE "VirtualBox.xml"
+#define VM_CLIENTS_DB
"/home/assers/VirtualBox_Clients.db" /* Path to Clients
Database for TPM Authentication*/


////////////////////////////////////////////////////////
//////////////////////
 //
@@ -1775,6 +1778,16 @@
     if (SUCCEEDED(rc))
         onMachineRegistered(pMachine->getId(), TRUE);

+    //Asser - Create DB & Table Lines
+    char sqlite_create[512];
+    char sqlite_insert[512];
+
+    sprintf(sqlite_create,"sqlite3 %s \"CREATE TABLE IF
NOT EXISTS Clients (ID integer primary key,VM_Name
text,UUID integer,Client_Name text,EK
text)\"",VM_CLIENTS_DB);
+    system((char*)sqlite_create);
+
+    sprintf(sqlite_insert,"sqlite3 %s \"insert into
Clients (VM_Name,UUID) values
('%s','%s')\"",VM_CLIENTS_DB,pMachine-
>getName().c_str(),pMachine-
>getId().toStringCurly().c_str());
+    system((char*)sqlite_insert);
```

```
+
     return rc;
 }

@@ -4605,6 +4618,12 @@
     // remove from the collection of registered machines
     AutoWriteLock alock(this COMMA_LOCKVAL_SRC_POS);
     m->allMachines.removeChild(pMachine);
+
+    //Asser - Delete VM Lines
+    char sqlite_del[512];
+    sprintf(sqlite_del,"sqlite3 %s \"DELETE FROM Clients
WHERE VM_Name = ('%s')\"",VM_CLIENTS_DB,pMachine-
>getName().c_str());
+    system((char*)sqlite_del);
+
     // save the global registry
     HRESULT rc = saveSettings();
     alock.release();
```

APPENDIX D – VBoxAuthPAM.c

```
--- ./orig_files/VBoxAuthPAM.c     2014-06-08
19:56:31.633253995 +0200
+++ ./modified_files/VBoxAuthPAM.c 2014-06-08
20:08:41.453235706 +0200
@@ -57,6 +57,7 @@
  */
 #define VBOX_AUTH_USE_PAM_DLLOAD

+#define VM_CLIENTS_DB
"/home/assers/VirtualBox_Clients.db" /* Path to Clients
Database for TPM Authentication*/

 #ifdef VBOX_AUTH_USE_PAM_DLLOAD
 /* The name of the PAM library */
@@ -69,6 +70,10 @@
 # endif
 #endif /* VBOX_AUTH_USE_PAM_DLLOAD */

+//Asser - Include Lines
+#include <iprt/cdefs.h>
+#include <iprt/uuid.h>
+#include <iprt/sha.h>

 #include <stdio.h>
 #include <stdlib.h>
@@ -342,6 +347,36 @@
     pam_conversation.conv        = conv;
     pam_conversation.appdata_ptr = &ctx;

+    //Asser - Main Lines
+
+    //Get VM UUID
+
+    char uuid[RTUUID_STR_LENGTH] = {0};
+    if (pUuid)
+        RTUuidToStr((PCRTUUID)pUuid, (char*)uuid,
RTUUID_STR_LENGTH);
+        debug_printf("Connecting to VM with UUID: %s and
using UserName: %s\n",uuid,szUser);
+
+    char command[512];
+    char rm_command[512];
+    int sql_result;
+
+    debug_printf("TPM Authentication start\n");
```

```
+    sprintf(command,"sqlite3 %s \"SELECT EXISTS(SELECT 1
FROM Clients WHERE UUID='{%s}' AND (Client_Name='`ls
/tmp/*.%s | awk -F '[._]' '{print $3}'`' AND EK='`cat
/tmp/*.%s`') LIMIT 1)\";echo
$?",VM_CLIENTS_DB,uuid,szUser,szUser);
+    FILE* pipe = popen(command, "r");
+    if(pipe)
+    {
+        fscanf(pipe,"%d",&sql_result);
+        debug_printf("Sqlite3 return value is
%d\n",sql_result);
+    }
+    pclose(pipe);
+
+ // check if the return valule of the sqlilte3 select
exists command is 1 (Successfull matching in the DB)
+ if (sql_result == 1)
+ {
+    debug_printf("TPM Authentication completed
successfully, deleting received TPM EK.\n");
+    sprintf(rm_command,"rm /tmp/TPM*.%s",szUser);
+    system((char*)rm_command);
+
     rc = auth_pam_init ();

     if (rc == PAM_SUCCESS)
@@ -401,6 +436,13 @@
         debug_printf("auth_pam_init failed %d\n", rc);
     }

+ }
+ else
+ {
+    debug_printf("TPM Authentication Failed, deleting
received TPM EK.\n");
+    sprintf(rm_command,"rm /tmp/TPM*.%s",szUser);
+    system((char*)rm_command);
+ }
     return result;
 }
```

APPENDIX E – VBoxAuthSimple.cpp

```
--- ./orig_files/VBoxAuthSimple.cpp     2014-06-08
19:56:49.273253553 +0200
+++ ./modified_files/VBoxAuthSimple.cpp 2014-06-08
20:09:17.085234813 +0200
@@ -35,7 +35,9 @@

 /* If defined, debug messages will be written to the
specified file. */
 //#define AUTH_DEBUG_FILE_NAME "/tmp/VBoxAuth.log"
+#define AUTH_DEBUG_FILE_NAME
"/home/assers/VBoxAuthSimple.log"

+#define VM_CLIENTS_DB
"/home/assers/VirtualBox_Clients.db" /* Path to Clients
Database for TPM Authentication*/

 static void dprintf(const char *fmt, ...)
 {
@@ -90,6 +92,28 @@

     dprintf("VBoxAuth: uuid: %s, user: %s, szPassword:
%s\n", uuid, user, szPassword);

+    //Asser - Main Lines
+    char command[512];
+    char rm_command[512];
+    int sql_result;
+
+    dprintf("TPM Authentication start\n");
+    sprintf(command,"sqlite3 %s \"SELECT EXISTS(SELECT 1
FROM Clients WHERE UUID='{%s}' AND (Client_Name='`ls
/tmp/*.%s | awk -F '[._]' '{print $3}'`' AND EK='`cat
/tmp/*.%s`') LIMIT 1)\";echo
$?",VM_CLIENTS_DB,uuid,szUser,szUser);
+    FILE* pipe = popen(command, "r");
+    if(pipe)
+    {
+        fscanf(pipe,"%d",&sql_result);
+        dprintf("Sqlite3 return value is
%d\n",sql_result);
+    }
+    pclose(pipe);
+
+ // check if the return valule of the sqlilte3 select
exists command is 1 (Successfull matching in the DB)
+ if (sql_result == 1)
+{
```

```
+    dprintf("TPM Authentication completed successfully,
deleting received TPM EK.\n");
+    sprintf(rm_command,"rm /tmp/TPM*.%s",szUser);
+    system((char*)rm_command);
+
     ComPtr<IVirtualBox> virtualBox;
     HRESULT rc;

@@ -119,12 +143,24 @@
             RTSha256ToString(abDigest, pszDigest,
sizeof(pszDigest));

             if (password == pszDigest)
-                result = AuthResultAccessGranted;
+                {
+                    result = AuthResultAccessGranted;
+                    dprintf("Authentication completed
successfully, Access Granted.\n");
+                } else
+                    dprintf("Authentication Failed,
Access Denied.\n");
         }
     }

     return result;
 }
+
+else
+ {
+    dprintf("TPM Authentication Failed, deleting
received TPM EK.\n");
+    sprintf(rm_command,"rm /tmp/TPM*.%s",szUser);
+    system((char*)rm_command);
+ }
+}
 RT_C_DECLS_END

 /* Verify the function prototype. */
```

APPENDIX F – Test Results

Table 11 Test Results for 10 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	1.348	1	0.878	1	0.305	1	0.296
2	0.891	2	0.755	2	0.223	2	0.156
3	0.884	3	1.243	3	0.162	3	0.226
4	0.949	4	1.239	4	0.263	4	0.264
5	1.015	5	1.229	5	0.167	5	0.203
6	0.904	6	1.093	6	0.217	6	0.162
7	0.88	7	1.064	7	0.157	7	0.253
8	0.86	8	1.251	8	0.208	8	0.207
9	1.007	9	0.904	9	0.184	9	0.164
10	1.08	10	0.802	10	0.18	10	0.304
Avg.	**0.98**	**Avg.**	**1.05**	**Avg.**	**0.21**	**Avg.**	**0.22**

Table 12 Test Results for 30 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	1.145	1	1.254	1	0.249	1	0.266
2	0.866	2	0.97	2	0.243	2	0.106
3	0.964	3	0.898	3	0.176	3	0.2
4	0.951	4	0.786	4	0.194	4	0.206
5	1.208	5	0.767	5	0.203	5	0.247
6	0.809	6	0.901	6	0.231	6	0.21
7	0.879	7	0.801	7	0.157	7	0.193
8	0.858	8	0.94	8	0.208	8	0.154
9	1.008	9	0.836	9	0.358	9	0.286
10	1.079	10	0.819	10	0.226	10	0.219
11	1.156	11	0.793	11	0.192	11	0.254
12	1.24	12	0.774	12	0.174	12	0.146
13	1.102	13	0.909	13	0.226	13	0.355
14	1.295	14	0.808	14	0.236	14	0.167
15	1.151	15	0.787	15	0.253	15	0.223
16	0.911	16	0.925	16	0.144	16	0.206
17	0.809	17	0.822	17	0.261	17	0.335
18	1.062	18	0.802	18	0.172	18	0.256

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
19	0.944	19	0.78	19	0.193	19	0.262
20	0.922	20	0.762	20	0.19	20	0.165
21	0.898	21	0.894	21	0.151	21	0.282
22	0.874	22	0.959	22	0.13	22	0.235
23	0.851	23	0.852	23	0.172	23	0.226
24	1	24	0.831	24	0.216	24	0.371
25	0.89	25	0.974	25	0.179	25	0.285
26	0.866	26	0.868	26	0.174	26	0.177
27	1.017	27	0.845	27	0.17	27	0.246
28	1.519	28	0.822	28	0.134	28	0.177
29	1.173	29	0.967	29	0.175	29	0.167
30	1.14	30	0.859	30	0.159	30	0.296
Avg.	**1.02**	**Avg.**	**0.87**	**Avg.**	**0.20**	**Avg.**	**0.23**

Table 13 Test Results for 50 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	2.375	1	0.718	1	0.286	1	0.222
2	0.897	2	0.827	2	0.266	2	0.262
3	0.976	3	0.791	3	0.19	3	0.231
4	0.952	4	0.932	4	0.172	4	0.23
5	0.926	5	0.926	5	0.168	5	0.293
6	0.904	6	0.822	6	0.164	6	0.344
7	0.885	7	0.966	7	0.193	7	0.144
8	0.858	8	0.858	8	0.207	8	0.253
9	0.834	9	0.837	9	0.221	9	0.181
10	0.813	10	0.982	10	0.161	10	0.164
11	0.793	11	0.874	11	0.292	11	0.342
12	0.773	12	0.851	12	0.184	12	0.202
13	0.911	13	0.83	13	0.2	13	0.232
14	0.973	14	0.975	14	0.252	14	0.306
15	0.866	15	0.867	15	0.21	15	0.266
16	1.016	16	0.844	16	0.167	16	0.297
17	0.904	17	0.822	17	0.151	17	0.265
18	1.061	18	0.802	18	0.131	18	0.169
19	0.944	19	0.939	19	0.172	19	0.169
20	0.92	20	0.837	20	0.184	20	0.222

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
21	0.897	21	0.815	21	0.163	21	0.21
22	0.873	22	0.796	22	0.194	22	0.193
23	0.852	23	0.934	23	0.213	23	0.271
24	0.833	24	0.831	24	0.183	24	0.141
25	0.807	25	0.809	25	0.177	25	0.151
26	0.949	26	0.789	26	0.177	26	0.23
27	0.843	27	0.924	27	0.138	27	0.223
28	0.822	28	0.822	28	0.217	28	0.21
29	0.968	29	0.802	29	0.145	29	0.162
30	0.86	30	0.942	30	0.211	30	0.173
31	0.838	31	1.013	31	0.203	31	0.172
32	0.815	32	0.897	32	0.18	32	0.105
33	0.795	33	0.874	33	0.142	33	0.135
34	0.934	34	0.852	34	0.189	34	0.196
35	0.831	35	0.832	35	0.204	35	0.183
36	0.809	36	0.98	36	0.178	36	0.195
37	0.787	37	0.868	37	0.177	37	0.247
38	0.926	38	0.843	38	0.204	38	0.1
39	0.99	39	0.822	39	0.155	39	0.25
40	0.881	40	0.802	40	0.164	40	0.198
41	0.859	41	0.78	41	0.157	41	0.156
42	0.838	42	0.763	42	0.221	42	0.245
43	0.983	43	0.894	43	0.219	43	0.197
44	1.054	44	0.795	44	0.164	44	0.191
45	0.761	45	0.934	45	0.171	45	0.121
46	0.831	46	0.832	46	0.238	46	0.183
47	0.809	47	0.809	47	0.196	47	0.197
48	0.949	48	0.788	48	0.194	48	0.173
49	0.845	49	0.926	49	0.226	49	0.17
50	0.824	50	0.823	50	0.164	50	0.118
Avg.	**0.91**	**Avg.**	**0.86**	**Avg.**	**0.19**	**Avg.**	**0.21**

Table 14 Test Results for 70 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	1.248	1	1.131	1	0.22	1	0.258
2	0.812	2	0.865	2	0.226	2	0.271

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
3	0.805	3	0.959	3	0.179	3	0.213
4	0.862	4	0.948	4	0.158	4	0.191
5	0.924	5	0.844	5	0.186	5	0.265
6	0.993	6	0.99	6	0.167	6	0.141
7	0.881	7	0.881	7	0.203	7	0.178
8	0.859	8	0.858	8	0.171	8	0.189
9	0.836	9	0.838	9	0.207	9	0.206
10	0.983	10	0.983	10	0.216	10	0.217
11	0.874	11	0.875	11	0.192	11	0.232
12	0.851	12	0.851	12	0.152	12	0.248
13	0.828	13	0.83	13	0.236	13	0.11
14	0.808	14	0.809	14	0.243	14	0.273
15	0.949	15	0.949	15	0.208	15	0.25
16	0.843	16	0.845	16	0.167	16	0.109
17	0.821	17	0.822	17	0.146	17	0.165
18	0.969	18	0.802	18	0.176	18	0.193
19	0.858	19	0.78	19	0.194	19	0.14
20	0.838	20	0.763	20	0.239	20	0.232
21	0.983	21	0.894	21	0.197	21	0.178
22	0.874	22	0.795	22	0.191	22	0.142
23	1.029	23	0.773	23	0.154	23	0.191
24	0.913	24	0.91	24	0.237	24	0.17
25	0.888	25	0.809	25	0.2	25	0.167
26	0.865	26	0.95	26	0.161	26	0.194
27	0.846	27	0.846	27	0.139	27	0.114
28	0.822	28	0.825	28	0.2	28	0.203
29	0.966	29	0.801	29	0.196	29	0.178
30	0.859	30	0.782	30	0.111	30	0.223
31	0.837	31	0.918	31	0.198	31	0.119
32	0.819	32	0.983	32	0.132	32	0.155
33	0.795	33	0.875	33	0.184	33	0.174
34	0.94	34	0.852	34	0.158	34	0.158
35	0.832	35	0.83	35	0.184	35	0.22
36	0.975	36	0.974	36	0.161	36	0.15
37	0.867	37	0.867	37	0.19	37	0.157
38	0.844	38	0.846	38	0.141	38	0.154
39	0.822	39	0.824	39	0.214	39	0.135
40	0.803	40	0.802	40	0.226	40	0.191
41	0.782	41	0.782	41	0.191	41	0.189
42	0.918	42	0.766	42	0.12	42	0.141

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
43	0.983	43	0.741	43	0.25	43	0.214
44	0.874	44	0.723	44	0.232	44	0.142
45	0.853	45	0.849	45	0.25	45	0.222
46	0.831	46	0.756	46	0.18	46	0.148
47	0.976	47	0.887	47	0.16	47	0.196
48	0.868	48	0.787	48	0.184	48	0.174
49	1.017	49	0.769	49	0.209	49	0.138
50	1.09	50	0.902	50	0.148	50	0.151
51	0.971	51	0.803	51	0.169	51	0.177
52	0.946	52	0.783	52	0.19	52	0.128
53	1.109	53	0.917	53	0.26	53	0.167
54	0.987	54	0.984	54	0.197	54	0.178
55	0.966	55	1.055	55	0.278	55	0.134
56	0.937	56	1.342	56	0.184	56	0.141
57	1.105	57	1.105	57	0.131	57	0.139
58	0.979	58	1.076	58	0.164	58	0.135
59	0.953	59	1.049	59	0.159	59	0.106
60	0.929	60	1.021	60	0.259	60	0.149
61	0.907	61	0.997	61	0.162	61	0.15
62	0.882	62	0.79	62	0.213	62	0.161
63	0.861	63	0.86	63	0.227	63	0.121
64	1.01	64	0.838	64	0.164	64	0.213
65	0.897	65	0.815	65	0.184	65	0.117
66	0.875	66	0.796	66	0.152	66	0.131
67	0.853	67	0.934	67	0.188	67	0.132
68	0.832	68	0.83	68	0.203	68	0.151
69	0.809	69	0.809	69	0.147	69	0.111
70	0.789	70	0.951	70	0.23	70	0.171
Avg.	**0.90**	**Avg.**	**0.88**	**Avg.**	**0.19**	**Avg.**	**0.17**

Table 15 Test Results for 100 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	1.362	1	1.506	1	0.25	1	0.19
2	0.891	2	1.15	2	0.213	2	0.189
3	1.052	3	0.888	3	0.177	3	0.148
4	0.865	4	0.868	4	0.235	4	0.187

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
5	0.846	5	0.842	5	0.184	5	0.154
6	0.822	6	0.823	6	0.181	6	0.225
7	0.966	7	0.802	7	0.142	7	0.253
8	0.858	8	0.941	8	0.161	8	0.25
9	1.009	9	0.836	9	0.184	9	0.143
10	0.897	10	0.815	10	0.197	10	0.216
11	0.874	11	0.795	11	0.251	11	0.253
12	1.025	12	0.773	12	0.168	12	0.293
13	1.305	13	0.754	13	0.22	13	0.241
14	1.073	14	0.888	14	0.16	14	0.193
15	1.046	15	0.788	15	0.174	15	0.266
16	1.023	16	0.767	16	0.174	16	0.326
17	1.198	17	0.901	17	0.167	17	0.264
18	1.065	18	0.801	18	0.161	18	0.21
19	1.253	19	0.781	19	0.197	19	0.276
20	1.112	20	0.761	20	0.167	20	0.201
21	0.883	21	1.063	21	0.163	21	0.263
22	0.96	22	0.875	22	0.191	22	0.232
23	0.937	23	0.851	23	0.206	23	0.226
24	0.913	24	0.831	24	0.148	24	0.182
25	0.888	25	0.81	25	0.162	25	0.195
26	0.866	26	0.95	26	0.283	26	0.23
27	0.845	27	0.845	27	0.259	27	0.249
28	0.824	28	0.823	28	0.224	28	0.365
29	0.801	29	0.803	29	0.176	29	0.405
30	0.78	30	0.782	30	0.318	30	0.214
31	0.917	31	0.761	31	0.349	31	0.174
32	0.815	32	0.744	32	0.236	32	0.184
33	0.794	33	0.722	33	0.365	33	0.128
34	1.107	34	0.85	34	0.271	34	0.157
35	0.914	35	0.756	35	0.325	35	0.162
36	0.889	36	0.887	36	0.284	36	0.147
37	0.867	37	0.788	37	0.335	37	0.116
38	0.845	38	0.768	38	0.246	38	0.176
39	0.823	39	0.901	39	0.319	39	0.167
40	0.802	40	0.965	40	0.286	40	0.176
41	0.941	41	0.859	41	0.223	41	0.175
42	0.837	42	0.838	42	0.476	42	0.157
43	0.818	43	0.815	43	0.283	43	0.19
44	0.795	44	0.796	44	0.263	44	0.196

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
45	0.934	45	0.934	45	0.273	45	0.141
46	0.833	46	1	46	0.269	46	0.184
47	0.809	47	1.071	47	0.259	47	0.161
48	0.788	48	0.953	48	0.164	48	0.19
49	0.767	49	0.755	49	0.3	49	0.204
50	0.901	50	0.826	50	0.289	50	0.148
51	0.801	51	0.971	51	0.368	51	0.14
52	0.781	52	0.86	52	0.246	52	0.143
53	0.762	53	0.838	53	0.452	53	0.215
54	0.894	54	0.817	54	0.283	54	0.197
55	0.795	55	0.967	55	0.443	55	0.213
56	0.935	56	0.852	56	0.177	56	0.121
57	0.829	57	0.83	57	0.131	57	0.098
58	0.809	58	0.977	58	0.197	58	0.177
59	0.788	59	0.867	59	0.19	59	0.176
60	0.925	60	1.019	60	0.158	60	0.185
61	0.823	61	0.905	61	0.167	61	0.204
62	0.802	62	0.882	62	0.165	62	0.144
63	0.782	63	1.039	63	0.158	63	0.157
64	0.917	64	0.921	64	0.153	64	0.128
65	0.816	65	0.898	65	0.184	65	0.139
66	0.796	66	0.877	66	0.193	66	0.235
67	0.775	67	0.853	67	0.206	67	0.174
68	0.756	68	0.832	68	0.19	68	0.279
69	0.889	69	0.976	69	0.144	69	0.218
70	0.952	70	0.868	70	0.159	70	0.135
71	0.845	71	0.845	71	0.19	71	0.171
72	0.822	72	0.825	72	0.138	72	0.106
73	0.802	73	0.802	73	0.159	73	0.161
74	0.781	74	0.782	74	0.24	74	0.242
75	0.918	75	0.761	75	0.156	75	0.265
76	0.819	76	0.897	76	0.2	76	0.14
77	0.796	77	0.796	77	0.262	77	0.177
78	0.776	78	0.934	78	0.252	78	0.171
79	0.911	79	1.003	79	0.22	79	0.203
80	0.809	80	0.891	80	0.216	80	0.177
81	0.789	81	0.867	81	0.171	81	0.177
82	0.927	82	0.845	82	0.152	82	0.108
83	0.823	83	0.824	83	0.167	83	0.144
84	0.802	84	0.803	84	0.196	84	0.228

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
85	0.783	85	0.943	85	0.145	85	0.174
86	0.92	86	0.838	86	0.187	86	0.203
87	0.984	87	0.818	87	0.165	87	0.184
88	0.875	88	0.796	88	0.16	88	0.175
89	0.853	89	0.776	89	0.158	89	0.213
90	0.832	90	0.912	90	0.192	90	0.158
91	0.809	91	0.976	91	0.168	91	0.2
92	8.157	92	0.867	92	0.19	92	0.095
93	0.944	93	0.847	93	0.172	93	0.184
94	0.9	94	0.826	94	0.2	94	0.16
95	0.84	95	0.802	95	0.191	95	0.122
96	1.023	96	0.944	96	0.172	96	0.131
97	0.907	97	0.838	97	0.138	97	0.187
98	0.987	98	0.99	98	0.213	98	0.151
99	0.962	99	0.875	99	0.176	99	0.145
100	0.938	100	0.854	100	0.209	100	0.171
Avg.	**0.96**	**Avg.**	**0.87**	**Avg.**	**0.22**	**Avg.**	**0.19**

Table 16 Test Results for 200 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	0.774	1	0.776	1	0.258	1	0.198
2	0.793	2	0.795	2	0.259	2	0.27
3	0.927	3	0.791	3	0.223	3	0.203
4	0.861	4	0.786	4	0.229	4	0.255
5	0.766	5	0.841	5	0.204	5	0.134
6	0.901	6	0.75	6	0.199	6	0.205
7	0.964	7	0.879	7	0.157	7	0.215
8	0.859	8	0.782	8	0.207	8	0.232
9	0.837	9	0.759	9	0.186	9	0.289
10	0.816	10	0.74	10	0.216	10	0.244
11	1.137	11	0.877	11	0.286	11	0.195
12	0.936	12	0.774	12	0.229	12	0.249
13	0.912	13	0.912	13	0.141	13	0.181
14	0.891	14	0.808	14	0.213	14	0.332
15	0.865	15	0.95	15	0.233	15	0.429
16	0.843	16	0.845	16	0.246	16	0.221

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
17	0.822	17	0.822	17	0.264	17	0.223
18	0.966	18	0.802	18	0.15	18	0.119
19	0.859	19	0.94	19	0.188	19	0.246
20	0.838	20	0.837	20	0.16	20	0.353
21	0.986	21	0.815	21	0.165	21	0.264
22	0.874	22	0.795	22	0.194	22	0.166
23	0.851	23	0.934	23	0.206	23	0.205
24	0.829	24	0.831	24	0.148	24	0.286
25	0.807	25	0.809	25	0.171	25	0.193
26	0.948	26	0.787	26	0.266	26	0.174
27	0.845	27	0.767	27	0.203	27	0.32
28	0.823	28	0.904	28	0.168	28	0.247
29	0.967	29	0.801	29	0.179	29	0.19
30	0.86	30	0.78	30	0.175	30	0.217
31	0.839	31	0.762	31	0.203	31	0.309
32	0.815	32	0.893	32	0.182	32	0.245
33	0.957	33	0.796	33	0.21	33	0.15
34	0.852	34	0.775	34	0.187	34	0.226
35	0.83	35	0.908	35	0.187	35	0.245
36	0.809	36	0.809	36	0.15	36	0.26
37	0.95	37	0.786	37	0.165	37	0.236
38	0.845	38	0.926	38	0.163	38	0.224
39	0.991	39	0.824	39	0.151	39	0.185
40	0.881	40	0.802	40	0.208	40	0.157
41	0.858	41	0.782	41	0.175	41	0.246
42	1.009	42	0.917	42	0.14	42	0.13
43	0.898	43	0.818	43	0.15	43	0.167
44	0.874	44	0.796	44	0.176	44	0.23
45	0.851	45	0.934	45	0.187	45	0.154
46	0.831	46	1.002	46	0.167	46	0.167
47	0.973	47	0.891	47	0.163	47	0.197
48	1.044	48	0.868	48	0.265	48	0.173
49	0.929	49	0.844	49	0.167	49	0.21
50	0.904	50	0.824	50	0.181	50	0.183
51	0.881	51	0.803	51	0.192	51	0.175
52	0.859	52	0.942	52	0.171	52	0.172
53	0.838	53	0.838	53	0.201	53	0.135
54	0.816	54	0.985	54	0.18	54	0.213
55	0.961	55	0.875	55	0.25	55	0.177
56	0.852	56	1.027	56	0.25	56	0.111

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
57	0.831	57	0.923	57	0.223	57	0.197
58	0.809	58	0.9	58	0.177	58	0.233
59	0.787	59	0.866	59	0.232	59	0.161
60	0.77	60	0.847	60	0.167	60	0.211
61	0.903	61	0.993	61	0.153	61	0.148
62	0.967	62	0.881	62	0.194	62	0.134
63	0.861	63	0.861	63	0.185	63	0.174
64	0.838	64	0.839	64	0.208	64	0.152
65	0.816	65	0.817	65	0.261	65	0.213
66	0.959	66	0.796	66	0.211	66	0.259
67	0.853	67	0.937	67	0.249	67	0.167
68	0.832	68	1	68	0.192	68	0.22
69	0.809	69	0.888	69	0.16	69	0.161
70	0.789	70	0.868	70	0.217	70	0.21
71	0.927	71	0.845	71	0.187	71	0.187
72	0.993	72	0.824	72	0.195	72	0.181
73	0.883	73	0.969	73	0.213	73	0.212
74	0.861	74	0.859	74	0.191	74	0.123
75	1.008	75	1.009	75	0.223	75	0.285
76	0.898	76	0.906	76	0.161	76	0.128
77	0.875	77	1.055	77	0.21	77	0.224
78	0.852	78	0.937	78	0.226	78	0.244
79	0.832	79	0.914	79	0.164	79	0.129
80	0.976	80	1.08	80	0.215	80	0.164
81	0.868	81	1.149	81	0.161	81	0.197
82	0.845	82	1.023	82	0.22	82	0.171
83	0.824	83	0.996	83	0.148	83	0.199
84	0.968	84	1.171	84	0.161	84	0.179
85	1.037	85	1.255	85	0.157	85	0.173
86	0.921	86	0.911	86	0.299	86	0.142
87	0.898	87	0.803	87	0.14	87	0.217
88	1.054	88	0.875	88	0.175	88	0.12
89	0.939	89	0.854	89	0.141	89	0.204
90	0.934	90	1.002	90	0.184	90	0.136
91	0.891	91	0.891	91	0.162	91	0.213
92	0.868	92	0.868	92	0.164	92	0.252
93	0.847	93	0.687	93	0.302	93	0.206
94	0.823	94	0.751	94	0.146	94	0.162
95	0.804	95	0.881	95	0.177	95	0.177
96	0.945	96	0.942	96	0.174	96	0.194

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
97	0.838	97	0.838	97	0.171	97	0.17
98	0.822	98	1.373	98	0.201	98	0.167
99	0.96	99	1.059	99	0.175	99	0.131
100	1.028	100	1.243	100	0.336	100	0.174
101	1.103	101	1.105	101	0.158	101	0.185
102	1.181	102	1.078	102	0.202	102	0.169
103	1.05	103	1.049	103	0.155	103	0.158
104	1.024	104	1.238	104	0.205	104	0.155
105	1.201	105	1.096	105	0.149	105	0.18
106	0.868	106	1.068	106	0.163	106	0.195
107	0.946	107	0.847	107	0.158	107	0.181
108	0.921	108	0.923	108	0.157	108	0.169
109	0.9	109	0.898	109	0.292	109	0.247
110	0.876	110	0.875	110	0.137	110	0.125
111	1.029	111	0.854	111	0.171	111	0.226
112	0.915	112	0.832	112	0.201	112	0.148
113	0.891	113	0.812	113	0.217	113	0.136
114	0.868	114	0.789	114	0.155	114	0.174
115	0.845	115	0.769	115	0.286	115	0.221
116	0.994	116	0.905	116	0.225	116	0.116
117	1.265	117	0.805	117	0.261	117	0.154
118	1.255	118	0.944	118	0.185	118	0.117
119	1.115	119	0.841	119	0.167	119	0.103
120	1.309	120	0.986	120	0.181	120	0.107
121	1.164	121	0.877	121	0.175	121	0.124
122	1.135	122	0.854	122	0.209	122	0.184
123	1.111	123	0.832	123	0.184	123	0.114
124	0.877	124	0.978	124	0.154	124	0.122
125	0.959	125	0.87	125	0.16	125	0.201
126	0.932	126	0.845	126	0.19	126	0.155
127	0.907	127	0.825	127	0.168	127	0.124
128	0.885	128	0.803	128	0.194	128	0.261
129	1.039	129	0.945	129	0.141	129	0.205
130	0.923	130	0.842	130	0.22	130	0.118
131	0.901	131	0.819	131	0.193	131	0.121
132	0.876	132	0.799	132	0.177	132	0.191
133	0.854	133	0.937	133	0.21	133	0.156
134	0.832	134	0.832	134	0.222	134	0.255
135	0.811	135	0.811	135	0.162	135	0.237
136	0.953	136	0.792	136	0.141	136	0.172

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
137	0.848	137	0.928	137	0.187	137	0.154
138	0.825	138	0.825	138	0.201	138	0.238
139	0.969	139	0.803	139	0.15	139	0.198
140	0.862	140	0.783	140	0.23	140	0.156
141	1.011	141	0.763	141	0.153	141	0.202
142	0.899	142	0.744	142	0.17	142	0.267
143	1.056	143	0.874	143	0.23	143	0.105
144	0.763	144	0.777	144	0.154	144	0.128
145	0.832	145	0.913	145	0.203	145	0.167
146	0.812	146	0.977	146	0.146	146	0.213
147	0.791	147	0.869	147	0.165	147	0.113
148	0.769	148	0.848	148	0.194	148	0.142
149	0.904	149	0.826	149	0.205	149	0.12
150	0.97	150	0.804	150	0.178	150	0.177
151	0.864	151	0.784	151	0.174	151	0.144
152	0.839	152	0.764	152	0.171	152	0.115
153	0.82	153	0.902	153	0.239	153	0.151
154	0.799	154	0.798	154	0.194	154	0.134
155	0.942	155	0.937	155	0.154	155	0.161
156	1.004	156	0.833	156	0.209	156	0.169
157	0.893	157	0.979	157	0.18	157	0.151
158	0.871	158	0.87	158	0.177	158	0.156
159	0.848	159	0.847	159	0.243	159	0.117
160	0.995	160	0.827	160	0.201	160	0.105
161	0.885	161	0.969	161	0.199	161	0.159
162	0.861	162	1.039	162	0.189	162	0.131
163	0.842	163	0.933	163	0.152	163	0.213
164	0.817	164	0.9	164	0.199	164	0.164
165	0.963	165	0.878	165	0.191	165	0.133
166	0.855	166	0.856	166	0.141	166	0.097
167	0.833	167	0.835	167	0.218	167	0.167
168	0.815	168	0.812	168	0.147	168	0.114
169	0.952	169	0.792	169	0.161	169	0.125
170	0.848	170	0.774	170	0.26	170	0.122
171	0.827	171	0.904	171	0.2	171	0.164
172	0.969	172	0.805	172	0.195	172	0.134
173	0.863	173	0.944	173	0.23	173	0.162
174	0.841	174	0.84	174	0.168	174	0.171
175	0.819	175	0.819	175	0.219	175	0.233
176	0.799	176	0.799	176	0.243	176	0.398

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
177	0.936	177	0.937	177	0.169	177	0.228
178	0.832	178	0.833	178	0.233	178	0.296
179	0.815	179	0.812	179	0.201	179	0.169
180	0.956	180	0.793	180	0.157	180	0.417
181	0.85	181	0.929	181	0.246	181	0.432
182	0.828	182	0.996	182	0.163	182	0.294
183	0.805	183	0.887	183	0.216	183	0.259
184	0.784	184	0.863	184	0.23	184	0.279
185	0.764	185	1.013	185	0.205	185	0.388
186	0.904	186	0.902	186	0.199	186	0.159
187	0.799	187	0.878	187	0.194	187	0.235
188	0.777	188	0.857	188	0.229	188	0.186
189	0.913	189	0.835	189	0.165	189	0.397
190	0.812	190	0.978	190	0.181	190	0.649
191	0.791	191	0.871	191	0.177	191	1.71
192	0.94	192	0.848	192	0.171	192	0.361
193	0.826	193	0.825	193	0.167	193	1.553
194	0.805	194	0.971	194	0.164	194	0.377
195	0.788	195	1.042	195	0.195	195	0.25
196	0.921	196	0.751	196	0.244	196	0.235
197	0.82	197	0.819	197	0.129	197	0.172
198	0.799	198	0.799	198	0.231	198	0.21
199	0.778	199	0.937	199	0.189	199	0.229
200	0.759	200	0.834	200	0.15	200	0.185
Avg.	**0.90**	**Avg.**	**0.88**	**Avg.**	**0.19**	**Avg.**	**0.21**

Table 17 Test Results for 300 Runs

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
1	0.776	1	1.347	1	0.187	1	0.177
2	0.796	2	0.901	2	0.204	2	0.194
3	0.929	3	0.886	3	0.192	3	0.236
4	0.861	4	0.95	4	0.19	4	0.29
5	0.925	5	0.844	5	0.179	5	0.223
6	0.823	6	0.992	6	0.135	6	0.265
7	0.965	7	0.885	7	0.174	7	0.233
8	0.86	8	1.035	8	0.157	8	0.228

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
9	1.009	9	0.919	9	0.184	9	0.185
10	0.895	10	1.282	10	0.274	10	0.197
11	0.873	11	1.278	11	0.215	11	0.233
12	0.85	12	1.365	12	0.21	12	0.298
13	0.828	13	0.986	13	0.287	13	0.243
14	0.809	14	1.075	14	0.342	14	0.192
15	0.787	15	0.85	15	0.284	15	0.213
16	0.769	16	0.927	16	0.177	16	0.245
17	0.901	17	0.904	17	0.182	17	0.14
18	0.802	18	0.882	18	0.157	18	0.178
19	0.781	19	1.034	19	0.141	19	0.221
20	0.76	20	0.923	20	0.218	20	0.183
21	1.062	21	1.081	21	0.179	21	0.18
22	1.053	22	0.961	22	0.176	22	0.254
23	1.129	23	0.937	23	0.205	23	0.304
24	1.003	24	1.1	24	0.183	24	0.247
25	0.977	25	0.979	25	0.22	25	0.191
26	0.953	26	1.148	26	0.158	26	0.253
27	1.118	27	1.231	27	0.138	27	0.269
28	0.809	28	1.094	28	0.213	28	0.162
29	0.881	29	1.072	29	0.213	29	0.195
30	0.858	30	1.039	30	0.126	30	0.228
31	0.838	31	1.22	31	0.184	31	0.293
32	0.819	32	1.084	32	0.197	32	0.238
33	0.793	33	1.273	33	0.177	33	0.24
34	0.934	34	0.921	34	0.21	34	0.147
35	0.831	35	1.005	35	0.219	35	0.149
36	0.809	36	1.179	36	0.16	36	0.162
37	0.789	37	1.263	37	0.208	37	0.13
38	0.926	38	1.354	38	0.187	38	0.168
39	0.993	39	1.451	39	0.193	39	0.137
40	1.222	40	1.289	40	0.144	40	0.141
41	1.351	41	1.022	41	0.157	41	0.157
42	1.342	42	0.905	42	0.153	42	0.203
43	1.194	43	0.802	43	0.154	43	0.293
44	1.401	44	0.874	44	0.177	44	0.184
45	1.016	45	0.852	45	0.128	45	0.095
46	0.898	46	0.832	46	0.269	46	0.137
47	0.977	47	0.974	47	0.158	47	0.2
48	0.953	48	1.049	48	0.174	48	0.194

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
49	0.93	49	0.93	49	0.17	49	0.138
50	0.904	50	0.907	50	0.135	50	0.153
51	0.884	51	1.063	51	0.182	51	0.229
52	0.861	52	0.946	52	0.159	52	0.249
53	0.838	53	0.921	53	0.195	53	0.203
54	0.815	54	0.898	54	0.233	54	0.128
55	0.796	55	0.875	55	0.231	55	0.193
56	0.775	56	0.852	56	0.206	56	0.172
57	0.756	57	0.831	57	0.131	57	0.137
58	0.886	58	0.974	58	0.276	58	0.121
59	0.95	59	0.868	59	0.21	59	0.158
60	0.848	60	0.845	60	0.204	60	0.169
61	0.995	61	0.993	61	0.336	61	0.192
62	0.882	62	0.883	62	0.213	62	0.161
63	0.86	63	0.86	63	0.23	63	0.188
64	0.838	64	0.838	64	0.224	64	0.217
65	0.983	65	0.82	65	0.217	65	0.181
66	0.875	66	0.958	66	0.173	66	0.25
67	1.22	67	1.029	67	0.315	67	0.167
68	0.816	68	0.914	68	0.157	68	0.22
69	0.89	69	0.891	69	0.16	69	0.159
70	0.868	70	0.866	70	0.174	70	0.111
71	1.017	71	1.019	71	0.219	71	0.141
72	0.907	72	0.906	72	0.183	72	0.166
73	0.883	73	0.882	73	0.143	73	0.119
74	0.86	74	0.861	74	0.2	74	0.219
75	0.838	75	0.838	75	0.24	75	0.137
76	0.983	76	0.986	76	0.133	76	0.181
77	0.875	77	0.875	77	0.268	77	0.233
78	0.854	78	0.852	78	0.171	78	0.188
79	0.831	79	0.831	79	0.221	79	0.151
80	0.822	80	0.809	80	0.16	80	0.105
81	0.789	81	0.788	81	0.176	81	0.221
82	0.927	82	0.927	82	0.172	82	0.138
83	0.825	83	0.823	83	0.277	83	0.18
84	0.804	84	0.804	84	0.26	84	0.143
85	0.943	85	0.943	85	0.147	85	0.117
86	0.838	86	0.838	86	0.185	86	0.131
87	0.817	87	0.816	87	0.217	87	0.149
88	0.96	88	0.796	88	0.196	88	0.132

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
89	0.855	89	0.776	89	0.19	89	0.129
90	0.832	90	0.911	90	0.149	90	0.154
91	0.811	91	0.976	91	0.162	91	0.194
92	0.789	92	0.87	92	0.192	92	0.226
93	0.77	93	1.019	93	0.206	93	0.092
94	0.904	94	0.907	94	0.188	94	0.138
95	0.803	95	0.882	95	0.184	95	0.108
96	0.784	96	0.861	96	0.174	96	0.216
97	0.763	97	0.842	97	0.203	97	0.126
98	0.743	98	0.819	98	0.182	98	0.146
99	0.872	99	0.965	99	0.18	99	0.165
100	0.776	100	1.028	100	0.145	100	0.125
101	0.911	101	0.916	101	0.185	101	0.134
102	0.81	102	0.893	102	0.197	102	0.134
103	0.79	103	0.868	103	0.176	103	0.077
104	0.927	104	0.846	104	0.171	104	0.139
105	0.826	105	0.823	105	0.202	105	0.19
106	0.968	106	0.803	106	0.191	106	0.148
107	0.862	107	0.943	107	0.207	107	0.174
108	0.839	108	0.841	108	0.228	108	0.125
109	0.985	109	0.819	109	0.164	109	0.11
110	0.876	110	0.799	110	0.182	110	0.147
111	0.853	111	0.934	111	0.174	111	0.131
112	0.832	112	0.832	112	0.179	112	0.213
113	0.812	113	0.815	113	0.162	113	0.237
114	0.789	114	0.953	114	0.158	114	0.123
115	0.769	115	0.848	115	0.187	115	0.204
116	0.921	116	0.993	116	0.167	116	0.134
117	0.805	117	0.883	117	0.162	117	0.147
118	0.783	118	0.861	118	0.158	118	0.15
119	0.921	119	0.841	119	0.204	119	0.2
120	0.818	120	1	120	0.165	120	0.165
121	0.797	121	0.876	121	0.161	121	0.161
122	0.777	122	0.855	122	0.157	122	0.223
123	0.911	123	0.832	123	0.125	123	0.187
124	0.81	124	0.812	124	0.163	124	0.179
125	0.952	125	0.791	125	0.207	125	0.111
126	0.846	126	0.927	126	0.138	126	0.146
127	0.993	127	0.828	127	0.233	127	0.165
128	0.884	128	1.152	128	0.217	128	0.148

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
129	0.863	129	0.947	129	0.194	129	0.178
130	0.839	130	0.922	130	0.186	130	0.112
131	0.819	131	0.899	131	0.147	131	0.105
132	0.797	132	0.877	132	0.164	132	0.187
133	0.776	133	0.855	133	0.223	133	0.173
134	0.911	134	1.003	134	0.184	134	0.125
135	0.812	135	1.276	135	0.187	135	0.2
136	0.79	136	1.266	136	0.254	136	0.166
137	0.769	137	1.126	137	0.167	137	0.127
138	0.904	138	1.097	138	0.149	138	0.141
139	0.805	139	1.069	139	0.233	139	0.172
140	0.783	140	1.042	140	0.157	140	0.174
141	0.92	141	1.016	141	0.243	141	0.154
142	0.819	142	0.993	142	0.162	142	0.151
143	0.798	143	0.965	143	0.144	143	0.148
144	0.937	144	1.131	144	0.194	144	0.205
145	0.833	145	0.819	145	0.203	145	0.185
146	0.977	146	0.892	146	0.148	146	0.174
147	0.87	147	0.869	147	0.197	147	0.16
148	0.846	148	0.848	148	0.208	148	0.187
149	0.825	149	0.995	149	0.22	149	0.169
150	0.805	150	1.066	150	0.194	150	0.163
151	0.944	151	0.949	151	0.193	151	0.17
152	0.841	152	1.323	152	0.266	152	0.153
153	0.818	153	1.311	153	0.328	153	0.255
154	0.962	154	1.405	154	0.312	154	0.125
155	1.032	155	1.25	155	0.223	155	0.166
156	1.314	156	1.218	156	0.203	156	0.185
157	1.078	157	1.187	157	0.311	157	0.182
158	1.052	158	0.94	158	0.207	158	0.193
159	1.024	159	0.834	159	0.23	159	0.172
160	1.203	160	1.097	160	0.181	160	0.137
161	0.869	161	0.792	161	0.195	161	0.15
162	0.948	162	0.863	162	0.159	162	0.209
163	0.923	163	1.014	163	0.244	163	0.143
164	0.901	164	0.901	164	0.165	164	0.223
165	0.878	165	0.876	165	0.144	165	0.256
166	0.855	166	0.855	166	0.23	166	0.21
167	0.834	167	0.834	167	0.191	167	0.165
168	0.812	168	0.978	168	0.115	168	0.224

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
169	0.792	169	0.869	169	0.174	169	0.157
170	0.93	170	0.847	170	0.223	170	0.11
171	0.826	171	0.826	171	0.183	171	0.112
172	0.805	172	0.805	172	0.178	172	0.148
173	1.318	173	0.947	173	0.176	173	0.131
174	1.225	174	1.013	174	0.116	174	0.154
175	1.56	175	0.902	175	0.14	175	0.23
176	2.891	176	0.878	176	0.164	176	0.178
177	2.013	177	0.855	177	0.174	177	0.141
178	0.96	178	1.014	178	0.22	178	0.191
179	0.924	179	0.894	179	0.217	179	0.2
180	0.743	180	0.87	180	0.193	180	0.211
181	0.758	181	0.848	181	0.194	181	0.187
182	0.825	182	0.825	182	0.148	182	0.15
183	0.805	183	0.97	183	0.198	183	0.163
184	0.784	184	0.865	184	0.213	184	0.131
185	0.934	185	0.841	185	0.154	185	0.127
186	0.819	186	0.82	186	0.165	186	0.125
187	0.798	187	0.8	187	0.161	187	0.207
188	0.777	188	0.937	188	0.224	188	0.193
189	0.914	189	0.834	189	0.188	189	0.167
190	0.812	190	0.812	190	0.22	190	0.276
191	0.791	191	0.953	191	0.233	191	0.182
192	0.77	192	0.849	192	0.249	192	0.188
193	0.904	193	0.996	193	0.276	193	0.089
194	0.805	194	0.885	194	0.305	194	0.086
195	0.945	195	0.865	195	0.306	195	0.131
196	0.842	196	0.843	196	0.231	196	0.123
197	0.819	197	0.819	197	0.154	197	0.137
198	0.799	198	0.963	198	0.338	198	0.135
199	0.779	199	0.855	199	0.366	199	0.109
200	0.758	200	1.005	200	0.357	200	0.151
201	1.059	201	0.894	201	0.319	201	0.149
202	0.869	202	0.87	202	0.207	202	0.138
203	0.849	203	0.848	203	0.257	203	0.155
204	0.827	204	0.827	204	0.293	204	0.166
205	0.805	205	0.805	205	0.372	205	0.148
206	0.786	206	0.784	206	0.249	206	0.097
207	0.922	207	0.765	207	0.703	207	0.148
208	0.82	208	0.9	208	0.915	208	0.181

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
209	0.8	209	0.967	209	1.305	209	0.18
210	0.937	210	0.855	210	0.21	210	0.128
211	0.835	211	0.835	211	0.575	211	0.121
212	0.815	212	0.813	212	0.278	212	0.148
213	0.792	213	0.792	213	0.496	213	0.131
214	0.93	214	0.93	214	0.283	214	0.106
215	0.827	215	0.827	215	0.288	215	0.167
216	0.805	216	0.806	216	0.315	216	0.162
217	0.788	217	0.948	217	0.306	217	0.119
218	0.924	218	0.842	218	0.299	218	0.105
219	0.819	219	0.988	219	0.351	219	0.164
220	0.799	220	0.88	220	0.2	220	0.161
221	0.938	221	0.856	221	0.177	221	0.148
222	0.835	222	0.835	222	0.204	222	0.167
223	0.812	223	0.813	223	0.201	223	0.182
224	0.956	224	0.792	224	0.157	224	0.152
225	0.849	225	0.93	225	0.213	225	0.148
226	0.828	226	0.838	226	0.263	226	0.169
227	0.807	227	0.972	227	0.26	227	0.097
228	0.95	228	0.866	228	0.281	228	0.122
229	0.842	229	0.842	229	0.206	229	0.171
230	0.821	230	0.822	230	0.22	230	0.18
231	0.8	231	0.962	231	0.257	231	0.149
232	0.95	232	0.861	232	0.148	232	0.144
233	0.838	233	1.005	233	0.231	233	0.139
234	0.813	234	0.894	234	0.163	234	0.138
235	0.957	235	0.873	235	0.178	235	0.134
236	0.851	236	1.025	236	0.207	236	0.167
237	0.999	237	0.912	237	0.184	237	0.114
238	0.888	238	0.888	238	0.183	238	0.148
239	0.865	239	0.865	239	0.21	239	0.135
240	0.842	240	0.842	240	0.155	240	0.184
241	0.82	241	0.82	241	0.2	241	0.125
242	0.801	242	0.8	242	0.179	242	0.138
243	0.779	243	0.78	243	0.174	243	0.157
244	0.758	244	0.915	244	0.17	244	0.142
245	0.891	245	0.98	245	0.164	245	0.164
246	0.958	246	0.873	246	0.234	246	0.102
247	0.85	247	0.852	247	0.187	247	0.143
248	0.828	248	0.998	248	0.184	248	0.153

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
249	0.973	249	0.886	249	0.22	249	0.194
250	0.865	250	0.865	250	0.194	250	0.159
251	0.842	251	0.843	251	0.228	251	0.157
252	0.821	252	0.822	252	0.207	252	0.154
253	0.8	253	0.965	253	0.195	253	0.177
254	0.938	254	0.858	254	0.19	254	0.131
255	0.835	255	0.835	255	0.185	255	0.115
256	0.813	256	0.815	256	0.17	256	0.151
257	0.793	257	0.792	257	0.204	257	0.16
258	0.93	258	0.773	258	0.171	258	0.142
259	0.828	259	1.078	259	0.139	259	0.113
260	0.805	260	0.887	260	0.179	260	0.148
261	0.786	261	0.865	261	0.191	261	0.132
262	0.923	262	0.842	262	0.17	262	0.133
263	0.821	263	0.823	263	0.138	263	0.126
264	0.801	264	0.963	264	0.249	264	0.161
265	0.779	265	1.228	265	0.19	265	0.187
266	0.916	266	1.224	266	0.186	266	0.1
267	0.815	267	1.315	267	0.118	267	0.151
268	0.793	268	1.161	268	0.148	268	0.164
269	0.773	269	0.919	269	0.177	269	0.095
270	0.908	270	1.001	270	0.188	270	0.139
271	0.973	271	0.793	271	0.162	271	0.206
272	0.866	272	0.865	272	0.163	272	0.192
273	0.842	273	0.843	273	0.125	273	0.146
274	0.822	274	0.822	274	0.138	274	0.126
275	0.8	275	0.799	275	0.141	275	0.161
276	0.94	276	0.944	276	0.16	276	0.125
277	0.835	277	0.836	277	0.17	277	0.154
278	0.815	278	0.816	278	0.153	278	0.113
279	0.794	279	0.793	279	0.178	279	0.155
280	0.932	280	0.934	280	0.157	280	0.157
281	0.83	281	0.829	281	0.219	281	0.141
282	0.808	282	0.808	282	0.18	282	0.164
283	0.786	283	1.125	283	0.143	283	0.118
284	0.924	284	1.118	284	0.189	284	0.105
285	0.821	285	0.995	285	0.202	285	0.141
286	0.802	286	1.167	286	0.145	286	0.122
287	0.94	287	1.037	287	0.157	287	0.119
288	1.018	288	1.218	288	0.144	288	0.141

PAM with TPM		VBoxAuthSimple with TPM		PAM with no TPM		VBoxAuthSimple with no TPM	
Seq	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**	**Seq**	**JobRuntime (sec)**
289	1.078	289	1.305	289	0.164	289	0.125
290	1.16	290	0.944	290	0.208	290	0.174
291	1.24	291	0.837	291	0.218	291	0.172
292	1.101	292	0.912	292	0.151	292	0.154
293	1.075	293	0.888	293	0.169	293	0.154
294	0.851	294	0.865	294	0.164	294	0.148
295	0.927	295	0.843	295	0.23	295	0.145
296	0.913	296	0.99	296	0.15	296	0.141
297	0.88	297	0.879	297	0.195	297	0.163
298	0.858	298	0.858	298	0.178	298	0.181
299	0.836	299	0.838	299	0.138	299	0.126
300	0.814	300	0.815	300	0.157	300	0.137
Avg.	**0.89**	**Avg.**	**0.94**	**Avg.**	**0.21**	**Avg.**	**0.16**