Chapter 5

RESULTS AND DISCUSSION

5.1 INTRODUCTION

This section describes the proof-of-concept of our proposed protocol architecture. The eXtensible Modular Hypervisor Framework has been utilized to build the TrustVisor hypervisor along with the core modules: cryptography operations, TEE and TPM emulator, contains TPM library function to make a secure communication with TPM hardware. The constructed hypervisor has been placed in the cloud server grub entry to make a choice of hypervisor. To ensure the trust worthiness of cloud platform, a remote attestation concept is used along with the most popular and widely used method called Integrity Measurement Architecture (IMA) [93]. Remote attestation uses IMA, it works based on binary attestation concept. After configuring IMA, it calculates and extends the hashes of all components while boot process into their respective PCRs. To ensure the remote attestation with privacy preserving of NC, we used Attestation Identity Key (AIK) for signing hashes of PCRs while performing quote operation. We used TPM emulator for communication with TPM device using TPM driver.

5.2 PERFORMANCE OF TPM COMMANDS

As discussed earlier, we used open source cloud software called eucalyptus to establish an IaaS private cloud for our testing. Eucalyptus [94] consists well defined components such as node controller, cloud controller, walrus, storage controller and cluster controller, those provides efficient communication among resources using web-service. Eucalyptus is an EC2 API-compatible and which is an answer to commercial Amazon EC2
cloud infrastructure. Eucalyptus supports libvirt hypervisor, it consists most popular hypervisors Xen and KVM hypervisors [97]. The eucalyptus components are well defined with web-service based interfaces and those components are developed using high-level and standard packages such as Axis2, Apache, and Rampart [95]. Our proposed architecture utilizes these components to prove that our concept is secure from insider attacks.

The node controller is the central component of our proposed eucalyptus cloud framework, where cloud controller can launch and execute virtual machines. The proposed framework implemented on HP elite Notebook 8540 with configuration of Intel i5 processor, 8GB RAM, 500HDD and Ubuntu 12.04 as a host OS. The Eucalyptus cloud software used for implementing the private cloud that provides an infrastructure for launching virtual machine’s. The experimental results show that proposed framework has greater ability to reduce the TCB minimization and less over heads while communicate with TPM device through the host operating system.

![TPM performance evaluation](image)

**Fig. 5.1: TPM performance evaluation (ms)**
The Figure 5.1 provides the details of each command execution time on TrustVisor hypervisor. The Intel SENTER (AMD SKINIT) instruction takes 20.5ms for the initiation of secure boot along with the TrustVisor hypervisor boot process. The PCR Extend is used to quote respective PCR value and it took 10.68ms. The TPM quote for measuring the PCR values with hash values are calculated and replaced with new hash digest and this operation took 357.68ms. Thus, it shows us that Flicker based environment takes long time to respond for the TPM quote. The seal and unseal operation takes 45.29ms and 537.87ms, when compared to other hypervisor performance in both operations TrustVisor has great ability to reduce the overheads in unseal operation. The remote attestation took 100.3ms for trusting the platform using the PCR values with cryptographic techniques those we discussed earlier sections. The results show us that TrustVisor has great ability to reduce the overheads during the TPM operations.

Table 5.1: HMAC and Basic operations on TrustVisor (ms) [88]

<table>
<thead>
<tr>
<th></th>
<th>Extend</th>
<th>Seal</th>
<th>Unseal</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native Linux</td>
<td>24066</td>
<td>358102</td>
<td>1008654</td>
<td>815654</td>
</tr>
<tr>
<td>TrustVisor</td>
<td>533</td>
<td>11.7</td>
<td>12.6</td>
<td>21000</td>
</tr>
<tr>
<td>HMAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>62.644</td>
<td>0.181</td>
<td>67.461</td>
<td>0.008</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.051</td>
<td>0.003</td>
<td>5.012</td>
<td>0.018</td>
</tr>
</tbody>
</table>
The Table 5.1 illustrates the performance of TrustVisor on cloud environment. HMAC is used to generate the hash digest of software status in PCR registers. Those registers are used for integrity check of client virtual machines contents. The avg and standard deviation is calculated on TrustVisor and Flicker. Flicker takes 62.644ms whereas TrustVisor takes 0.059ms, it indicates the performance and dependent nature of execution of cloud components on node controller. In this way, we can calculate the start deviations of both components are: 0.003ms and 0.181ms respectively. It leverages the potentiality of TrustVisor in cloud environment. Such that this can work with high performance in execution point of view as well as in implementation of cloud components.

The TrustVisor design disabled with MMU-based mechanism along with nested page table concept in the AMD specific x86 untrusted legacy operating systems. The nested page table maintains separate page for both host system (hPT) as well as guest operating system (gPT). Hence, the TrustVisor provides an isolated execution environment along with PAL.

**Trusted Computing Base**

The Trusted Computing Base (TCB) of a system is a finite set of software, hardware and/or software components that are essential for security of intact system. The external components of system i.e., outside TCB should not breach security policies and should not get any privileges than granted privileges. Vulnerabilities within TCB affect or compromises security of the entire system.

Trusted computing base of the any cloud server can decide the security perimeter of the cloud environment. The proposed system has less trusted computing base when compared with earlier systems. The Figure 5.2 and Table 5.2 shows the TCB of the algorithms, where the proposed has less lines of code.
Table 5.2: LOC comparison with other hypervisors [52]

<table>
<thead>
<tr>
<th>Hypervisor</th>
<th>Hypapp</th>
<th>XMHF core</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrustVisor</td>
<td>3939</td>
<td>6018</td>
<td>9957</td>
</tr>
<tr>
<td>LockDown</td>
<td>9391</td>
<td>6018</td>
<td>15409</td>
</tr>
<tr>
<td>XTRRec</td>
<td>3500</td>
<td>6018</td>
<td>9518</td>
</tr>
<tr>
<td>SecVisor</td>
<td>2200</td>
<td>6018</td>
<td>8218</td>
</tr>
</tbody>
</table>

The XMHF core for any hypervisor is 6018 and Hypapp will a variable, it depends on the need of the security of cloud server. From the Table 5.2, SecVisor has less lines of code but it is only useful in the commodity operating systems. So, the SecVisor will not be consider for the cloud environments and other hypervisors results in large TCB.
5.3 SECURE VM LAUNCH

In this section, we describe secure VM launch approach in cloud infrastructure using TPM remote attestation on TrustVisor [88] and section 4.3 describes about how application are executed in isolated environment on top of the TrustVisor. This mechanism achieves the data integrity of client applications which are running on trusted hypervisor. An external trusted third party or verifier receives a TPM-generated attestation that includes set of PCR values and following data that have to be extended to convey the following information [88]:

- SKINIT instruction (AMD) used in bootstrap for the execution of TrustVisor using dynamic root trust.
- Next, TrustVisor receives the control of dynamic root of trust.
- One of the PCRs contains cryptographic hash (measurement) of TrustVisor
- TrustVisor generates the identity key using current TPM AIK for μTPM.

A TrustVisor attestation consists set of outputs of HV_quote operation along with other untrusted data to the verifier making sense out of the μPCRs. The trusted party or verifier must trust the TrustVisor using TPM attestation service. Suppose TrustVisor is untrusted, then entire cloud environment can be considering as untrusted [101]. Such that, no trust environment can be constructed with untrusted TrustVisor.

The client VM stored in encrypted form in Cloud Storage such that VM can be executed on NCs, those equipped with TPM device and drivers. The secure VM ensures that decryption key from client is securely taken by using proposed protocol. The proposed secure VM launch protocol obtains decryption key from the client and decrypt the VM on
NC in presence of user. The secure launch is proceeds in two stages: in the first stage, we verify the public key of TrustVisor (pk_t) and client (pk_c). This can be achieved by using client TPM and NC to provide secure channel among client and TrustVisor using Flicker secure channel protocol [87, 102]. The first stage as proceeds: the client sends a VM request to the NC for creation of VM or VM launch. Upon receiving the request from client, NC initiates the TrustVisor and executes the Piece of Application Logic (PAL). The PCR 18 extends with measurement of PAL and its respective input and output. Here, PAL is a tiny application that isolates the execution environment, supports the VM decryption and generates the asymmetric key for TrustVisor. In this section, we explained proposed architecture proof-of-concept with implementation. As we discussed earlier, remote attestation used for verification of VM by external trusted parties. This approach mainly used Intel based processor called Integrity Measurement Architecture (IMA) [93,96]. Our proof of concept proposed algorithm uses TPM secure VM launch protocol for securing the VM in Cloud Storage (CS). The VM kernels are encrypted and stored in cloud storage, so that only trusted or verified NCs can launch the VM.

![Fig 5.3: Secure VM launch](image)

100
The process of secure VM launch ensures that client private key or decryption key is provided to NC to decrypt the VM. The proposed protocol proceeds as follows: the client sent VM request to NC and initialize TrustVisor session and extends the PCR register 18 with measurement of TrustVisor. The secure VM launch protocol execution illustrated in Figure 5.3.

TrustVisor consists of private and public key for session management and secure VM launch operation. TrustVisor sends public key \((t_{pk})\) to node controller and node controller forward a key along with nonce of client. Client responds with nonce, client private key and public key of Trustvisor to the Node controller and the Node Controller forwards the same copy of keys to the TrustVisor. Upon receiving, the TrustVisor decrypt the VM kernel image using private key of client and public key of TrustVisor. Now, a VM execution started in isolated environment apart from the Node controller.

### 5.4 REMOTE ATTESTATION

As described in earlier sections, TPM module stores the measurement in PCR registers and those are utilized for remote verification by extending PCR registers. An actual process starts by invoking nonce to Node controller and the node controller forwards respective PCR value to the third party for verification. The third party matches the PCR hash value with earlier PCR value. Upon matching NC request the client for private key and client send the private key back to node controller for the decryption of VM. After receiving key from client, node controller decrypts the VM kernel and establishes a session in TrustVisor \([88]\) for isolated environment. The Figure 5.4 illustrates the complete process of remote attestation.
The main issue with the earlier mechanism [89] is that client can occasionally or periodically freezing for approximately for very 550ms in the remote attestation phase [22, 98]. This issue hurts the cloud consumer while performing the remote attestation with cloud server. The issue is because of poor TPM operations such as TPM_Quote2 [99,100]. Such that, we replace heavy operations with minimal operations in the software cryptographic operations for attestation protocol.

![Remote Attestation Diagram](image)

**Fig 5.4: Remote Attestation**

To analyze the performance of remote attestation, TrustVisor uses a 1024-bit session key, which is created or generated in boot process. The below table shows, the performance of various hypervisors. The Figure 5.5 and Table 5.3 provide remote attestation performances of various hypervisors (in ms). The Native Linux takes 518.1ms, TGVisor takes 172.1ms, TrustVisor takes 288.3ms, SecVisor takes 240.5ms, Lockdown
takes 298.3ms and proposed framework with TrustVisor takes 169.8ms. Hence, the TrustVisor takes less time for remote attestation for the remote attestation.

![Remote Attestation](image)

**Fig 5.5: Remote attestation evolution (ms)**

Table 5.3: Performance evolution of Remote attestation (ms) [88]

<table>
<thead>
<tr>
<th></th>
<th>Native Linux</th>
<th>TGVisor</th>
<th>CloudVisor</th>
<th>SecVisor</th>
<th>Lockdown</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>518.1</td>
<td>172.1</td>
<td>288.3</td>
<td>240.5</td>
<td>298.3</td>
<td>169.8</td>
</tr>
</tbody>
</table>

As shown in the Figure 5.5, the proposed framework protocol in the remote attestation outperforms the earlier hypervisors by 2.88 times because the proposed framework uses light TPM operations such as TPM_PCR_Extend and TPM_PCR_Extend to check or verify the integrity of the infrastructure where virtual machines to be executed or being executing. The TPM_PCR_Read takes 7.2ms and TPM_PCR_Extend takes 8.2ms to update the PCR register. Such the TPM operation takes less amount overheads while performing the remote attestation with remote party.
5.5 CONCLUSION

In this chapter the results are evaluated with earlier virtualization layer designs. The proposed framework takes less amount of CPU time to execute the commands those are performed when trusted computing techniques are under execution. The Trusted computing base of the proposed framework is reduced by eliminating expensive cryptographic operations. The remote attestation takes less amount of time to prove the trustworthiness of cloud resources. The experimental results clearly show that proposed framework can prevent insider attacks using PAL and it can detect the abnormalities of the insiders by using Integrity Measurement Architecture. Such that the proposed framework can effectively prevent and detect the insider attacks in the cloud infrastructure.