

A PROJECT REPORT ON  
**“PROVIDING USER SECURITY GUARANTEES IN  
PUBLIC INFRASTRUCTURE CLOUD”**

Submitted in partial fulfilment of requirement for the award of the degree of

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

BY

NAME	REGISTER NUMBER
C.ASHAJYOTHI	15091A0519
C.ARUNKUMR	15091A0517
B.GEETHANJLI	15091A0546
U.MADHUSUDHAN	14091A0541

Under the guidance of

**Mr. P. PRATHAP NAIDU** M.Tech.,  
Assistant Professor in Dept. of CSE



ESTD -1995

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING &  
TECHNOLOGY  
(AUTONOMOUS)**

(Affiliated to J.N.T.University Anantapur)

Approved by AICTE, New Delhi; Accredited by NAAC of UGC, New Delhi with 'A+' Grade  
Nandyal-518501

2018-2019

**Dr. SUBBA REDDY KUNAM**  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYAL -518 501, Kurnool (Dist), A.P.

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING &  
TECHNOLOGY  
(AUTONOMOUS)

(Affiliated to J.N.T.University Anantapur)

Approved by AICTE, New Delhi; Accredited by NAAC of UGC, New Delhi with 'A+' Grade  
Nandyal-518501

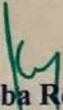


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

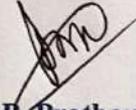
CERTIFICATE

This is to certify that **C.AshaJyothi (15091A0519)**, **C.ArunKumar (15091A0517)**, **B.Geethanjali (15091A0546)**, **U.Madhusudhan (14091A0541)**, of B.Tech., CSE final year has carried out the project work on **"Providing User Security Guarantees in Public Infrastructure Cloud"** under the esteemed guidance of **Mr. P. Prathap Naidu, Assistant Professor** of CSE Department, for the partial fulfilment of the award of degree of **B.Tech** in CSE in **RGM CET, Nandyal** is a bonafide record of work done by them during the year 2018-2019.

Head of Department:

  
**Dr. K. Subba Reddy** M.Tech, Ph.D  
Associate Professor and H.O.D  
Department of CSE

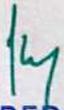
Project Guide:

  
**Mr. P. Prathap Naidu** M.Tech  
Assistant Professor  
Department of CSE

Place: Nandyal

External Examiner:

Date:

  
**Dr. SUBBA REDDY KUNAM**  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYAL-518 501, Kurnool (Dist), A.P.

## CANDIDATE'S DECLARATION

We hereby declare that the work done in the Project titled "**Providing User Security Guarantees in Public Infrastructure Cloud**" submitted towards completion of major project in IV Year II Semester of B. Tech (CSE) at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record of our original work done under the guidance of **Mr. P. Prathap Naidu Assistant Professor**, Dept. of CSE, RGM CET, Nandyal. We have not submitted the matter embodied in this project for the award of any other Degree in any other institutions.

BY

C.Asha Jyothi (15091A0519)  
C.Arun Kumar (15091A0517)  
B.Geethanjali (15091A0546)  
U.Madhusudhan (14091A0541)

  
**Dr. SUBBA REDDY KUNAM**  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYAL-518 501, Kurnool (Dist), A.P.

## ACKNOWLEDGEMENT

At the outset, we express our sincere gratitude to our project guide and supervisor, **Mr. P. Prathap Naidu, Assistant Professor** of Computer Science & Engineering Department, for the guidance and assistance to us, which contribute to successful completion of this project.

We express our gratitude to **Dr. K. Subba Reddy**, Head of the Department of Computer Science & Engineering, Rajeev Gandhi Memorial College of Engineering & Technology, for providing all the facilities and guidelines, required for our academic pursuit.

At the outset we thank our honourable **Chairman, Dr. M. Santhi Ramudu**, for providing us with exceptional faculty and moral support throughout the course.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr.T.Jaya Chandra Prasad** who has been observing us towards our individuality to acknowledge our project work tangentially.

BY

C.Asha Jyothi (15091A0519)  
C.Arun Kumar (15091A0517)  
B.Geethanjali (15091A0546)  
U.Madhusudhan (14091A0541)

**Dr. SUBBA REDDY KUNAM**  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYALUR 501, Kurnool (Dist), A.P.

## ABSTRACT

The infrastructure cloud (IaaS) service model offers improved resource flexibility and availability, where tenants insulated from the minutiae of hardware maintenance rent computing resources to deploy and operate complex systems. Large-scale services running on IaaS platforms demonstrate the viability of this model; nevertheless, many organizations operating on sensitive data avoid migrating operations to IaaS platforms due to security concerns. In this project, we describe a framework for data and operation security in IaaS, consisting of protocols for a trusted launch of virtual machines and domain-based storage protection. We continue with an extensive theoretical analysis with proofs about protocol resistance against attacks in the defined threat model. The protocols allow trust to be established by remotely attesting host platform configuration prior to launching guest virtual machines and ensure confidentiality of data in remote storage, with encryption keys maintained outside of the IaaS domain. Presented experimental results demonstrate the validity and efficiency of the proposed protocols. The framework prototype was implemented on a test bed operating a public electronic health record system, showing that the proposed protocols can be integrated into existing cloud environments.

14  
Dr. SUBBA REDDY KUNAM  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYAL-518 501, Kurnool (Dist), A.P.

<b>CONTENTS</b>		
<b>Chapter No</b>	<b>Title</b>	<b>Page No</b>
	List of figures	i
	Abstract	ii
1	Introduction	1
2	System Analysis	2
	2.1 Existing System	2
	2.2 Proposed System	2
3	Literature Survey	4
	3.1 IASS Services	4
	3.1.1 Hypervisor	5
	3.1.2 Cloud Control Stack	5
	3.1.3 Customers and Users	5
	3.1.4 Attack Model	6
	3.1.5 Confidentiality	6
	3.1.6 Integrity	7
	3.1.7 Availability	7
	3.1.8 Contractual Security	7
	3.1.9 Authentication	8
4	System Specification	9
	4.1 Hardware Requirements	9
	4.2 Software Requirements	9
5	Feasibility Analysis	10
	5.1 Operational Feasibility	10
	5.2 Economic Feasibility	10
	5.3 Technical Feasibility	10
6	System Design	11
	6.1 UML	11
	6.1.1 Introduction to UML	11
	6.1.2 Definition	11
	6.1.3 As a Language	11

	6.2 UML Diagrams	12
	6.2.1 Static	12
	6.2.2 Dynamic	12
	6.3 Modules	13
	6.3.1 Data Owner	13
	6.3.2 Domain Manager	13
	6.3.3 Trusted Third Party	13
	6.3.4 Attacker	13
	6.3.5 End User	13
	6.4 Architecture Diagram	14
	6.5 Class Diagram	15
	6.6 Use case Diagram	16
	6.7 Sequence Diagram	17
7	System Testing	18
	7.1 Test Case Description	18
	7.2 Psychology of Testing	18
	7.3 Testing Objectives	18
	7.4 Levels of Testing	18
	7.4.1 System Testing	18
	7.4.2 Code Testing	19
	7.5 Types of Testing	19
	7.5.1 Installation Testing	19
	7.5.2 Compatibility Testing	19
	7.5.3 Regression Testing	19
8	System Environment	20
	8.1 Client Server	20
	8.1.1 Why Client Server	20
	8.1.2 Front End	21
	8.2 Data Base Connectivity	21
	8.3 Features of the language used	22
	8.3.1 About Java	22

8.3.2 Importance of java to the internet	22
8.3.3 Java can be used to create two types of Programs	22
8.4 Features of java	23
8.4.1 Security	23
8.4.2 Portability	23
8.4.3 The byte code	23
8.4.4 Java virtual machine	24
8.4.5 Overall Description	24
8.5 Java Architecture	24
8.5.1 Compilation Code	25
8.5.2 Simple	25
8.5.3 Object oriented	26
8.5.4 Robust	26
8.6 Java Script	26
8.6.1 Java script versus Java	27
8.7 Hyper Text Markup Language	28
8.7.1 Basic Html Tags	28
8.8 Java data base connectivity	29
8.8.1 What is JDBC	29
8.8.2 What JDBC do	30
8.9 JDBC versus ODBC and other APIS	30
8.9.1 Two tier and three tier modules	31
8.9.2 JDBC driver types	31
8.9.3 JDBC-ODBC bridge	32
8.9.4 What is the JDBC ODBC bridge	32
8.9.5 Java server page	32
8.10 Features of JSP	32
8.10.1 Portability	32
8.10.2 Components	33
8.10.3 Processing	33

	8.11 Access models	33
	8.11.1 Steps in the execution of a JSP application	33
	8.11.2 JDBC connectivity	34
	8.12 Tom cat 6.0 web server	34
9	Cloud OS	35
	9.1 The cloud OS	35
	9.1.1 Infrastructure and cloud drivers	36
	9.1.2 Virtual Machine Manager	36
	9.1.3 Service Manager	36
10	Modules	38
	10.1 Module Description	38
	10.2 Secure Storage	38
	10.3 Trusted Platform Module	39
	10.4 Storage Production Construction	40
	10.5 Test Bed Architecture	41
	10.6 Implementation	41
	10.7 Algorithms	42
	10.7.1 Encryption Algorithm	42
	10.7.2 Decryption Algorithm	43
	10.7.3 Message Authentication code	43
	10.7.4 Session key	43
	10.7.5 Checksum Algorithms	44
11	Output screens	45
12	Conclusion	57
	References	58

14

**Dr. SUBBA REDDY KUNAM**  
 BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
 Professor & Head of the Department of CSE  
 RGM College of Engg. & Tech., (Autonomous)  
 NANDYAL-518 501, Kurnool (Dist), A.P.

## LIST OF FIGURES

Fig 6.4:	Architecture Diagram	14
Fig 6.5:	Class Diagram	15
Fig 6.6:	Use Case Diagram	16
Fig 6.7:	Sequence Diagram	17
Fig 8.4.5:	Development Process of JAVA Program	24
Fig 8.4.5:	Compiling and Interpreting	25
Fig 8.12:	Tomcat 6.0 Web Server	34
Fig 9.0:	Cloud OS	35
Fig 10.3:	Infrastructure Cloud Provider	40
Fig 11.1:	Cloud Server Login page	45
Fig 11.2:	View End Users	46
Fig 11.3:	View File Request Page	47
Fig 11.4:	View Blocked Users	47
Fig 11.5:	View Domains	48
Fig 11.6:	View Block Details	48
Fig 11.7:	View Domain	49
Fig 11.8:	View Verified Blocks Based on Domain	49
Fig 11.9:	View Verification Details	50
Fig 11.10:	Trusted Third Party Login Form	50
Fig 11.11:	View Meta Data	51
Fig 11.12:	Verify Block Data	51
Fig 11.13:	View Safe Details	52
Fig 11.14:	Data Owner Login	53
Fig 11.15:	Data Owner Registration	54
Fig 11.16:	Data owner login	55
Fig 11.17:	View Block Details	55
Fig 11.18:	View Blocked Users	56

## Chapter 1

### INTRODUCTION

Cloud computing has progressed from a bold vision to massive deployments in various application domains. However, the complexity of technology underlying cloud computing introduces novel security risks and challenges. Threats and mitigation techniques for the IaaS model have been under intensive scrutiny in recent years, while the industry has invested in enhanced security solutions and issued best practice recommendations.

From an end user point of view the security of cloud infrastructure implies unquestionable trust in the cloud provider, in some cases corroborated by reports of external auditors. While providers may offer security enhancements such as protection of data at rest, end-users have limited or no control over such mechanisms. There is a clear need for usable and cost-effective cloud platform security mechanisms suitable for organizations that rely on cloud infrastructure.

One such mechanism is platform integrity verification for compute hosts that support the virtualized cloud infrastructure. Several large cloud vendors have signaled practical implementations of this mechanism, primarily to protect the cloud infrastructure from insider threats and advanced persistent threats. We see two major improvement vectors regarding these implementations. First, details of such proprietary solutions are not disclosed and can thus not be implemented and improved by other cloud platforms. Second, to the best of our knowledge, none of the solutions provides cloud tenants a proof regarding the integrity of compute hosts supporting their slice of the cloud infrastructure. To address this, we propose a set of protocols for trusted launch of virtual machines (VM) in IaaS, which provide tenants with a proof that the requested VM instances were launched on a host with an expected software stack.

ly  
Dr. SUBBA REDDY KUNAM  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIEV  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
NANDYAL 518 501, Kurnool (Dist), A.P.

## Chapter 2

### SYSTEM ANALYSIS

#### 2.1 Existing System

The protocols allow trust to be established by remotely attesting host platform configuration prior to launching guest virtual machines and ensure confidentiality of data in remote storage, with encryption keys maintained outside of the IaaS domain. Presented experimental results demonstrate the validity and efficiency of the proposed protocols. The framework prototype was implemented on a test bed operating a public electronic health record system, showing that the proposed protocols can be integrated into existing cloud environments.

#### LIMITATIONS:

- The underlying compute host will still have access encryption keys whenever the VM performs cryptographic operations.
- This shifts towards the tenant the burden of maintaining the encryption software in all their VM instances and increase the attack surface.

#### 2.2 Proposed System

Presented an IaaS storage protection scheme addressing access control. The authors analyze access rights management of shared versioned encrypted data on cloud infrastructure for a restricted group and propose a scalable and flexible key management scheme. Access rights are represented as a graph, making a distinction between data encryption keys and encrypted updates on the keys and enabling flexible join/leave client operations, similar to properties presented by the protocols in this project. Despite its advantages, the requirement for client-side encryption limits the applicability of the scheme in and introduces important functional limitations on indexing and search. In our model, all cryptographic operations are performed on trusted IaaS compute hosts, which are able to allocate more computational resources than client devices. Abundant works have been proposed under different threat models to achieve various search functionality.

  
**DR. SUBBA REDDY KUNAM**  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, IEEE  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
Narasimha Rao Road, 501, Kurnool (Dist), A.P.

### Advantages:

- Domain based storage protection protocol allow domain managers store encrypted data volumes partitioned according to administrative domains.
- We describe a trusted VM launch (TL) protocol which allows tenants referred to as domain managers to launch VM instances exclusively on hosts with an attested platform configuration and reliably verify this.
- We introduce a domain-based storage protection protocol to allow domain managers store encrypted data volumes partitioned according to administrative domains.
- We introduce a list of attacks applicable to IaaS environments and use them to develop protocols with desired security properties, perform their security analysis and prove their resistance against the attacks.
- IaaS provides more flexibility.

14  
Dr. SUBBA REDDY KUNAM  
BE, M.Tech, Ph.D, MIEEE, MISTE, MCSI, FIETE, FIE(I)  
Professor & Head of the Department of CSE  
RGM College of Engg. & Tech., (Autonomous)  
18 50, (Dist), A.P.

## Chapter 3

### Literature Survey

The public Infrastructure-as-a-Service (IaaS) cloud industry has reached a critical mass in the past few years, with many cloud service providers fielding competing services. Despite the competition, we find some of the security mechanisms offered by the services to be similar, indicating that the cloud industry has established a number of “best-practices,” while other security mechanisms vary widely, indicating that there is also still room for innovation and experimentation. We investigate these differences and possible underlying reasons for it. We also contrast the security mechanisms offered by public IaaS cloud offerings and with security mechanisms proposed by academia over the same period. Finally, we speculate on how industry and academia might work together to solve the pressing security problems in public IaaS clouds going forward.

#### **3.1 IaaS services:**

An IaaS cloud provides compute and storage resources. Compute resources are provided in the form of Virtual Machines (VMs), which are rented by customers on an hourly basis. VMs are sometimes called “instances” in cloud literature, and the two are essentially equivalent. Typically, CSPs offer a menu of preconfigured VMs that customers can select from. VMs with more resources (i.e., faster CPUs or more memory) or specialized software will have a higher hourly rate than more basic VMs. CSPs may also bill for data transferred between VMs and hosts external to the CSP (i.e., external network traffic). Many CSPs also provide services through which customers may rent VMs or machines on a monthly basis (as opposed to hourly), but we exclude these because they are similar to traditional hosting services and outside the scope of this survey.

In general, CSPs provide two types of storage services: block storage, which is used to store virtual block devices for the VMs, and object storage, which provides a key-value store interface that can be used to store arbitrary objects. Block storage generally provides higher performance than object storage but can only be accessed by VMs in the compute service and offers a lower level of durability in the face of failures. In contrast, object storage, although slower, can be accessed by any client with an

Internet connection and provides highly available and durable storage. For example, Amazon estimates the durability of their block store, called the Amazon Elastic Block Service (EBS) to between 99.5% and 99.9%, while they estimate the durability of their object store, called the Amazon Simple Storage Service (S3) to be 99.999999999% [Amazon AWS 2014]. CSPs typically charge fees based on the amount of storage space consumed and the volume of data transferred out of the object storage services.

### **3.1.1 Hypervisor:**

A hypervisor is a low level software component that allows commodity compute hardware to be virtualized and partitioned into VMs, which can then be rented to customers by a CSP. Some examples of well-known commodity hypervisors are Xen, KVM, VMware, and Hyper-V.

### **3.1.2 Cloud control stack:**

A hypervisor on its own cannot be used to implement an IaaS service. Customers typically interact with a web interface that allows customers to remotely create, provision, pause, resume, stop, and destroy VMs over the Internet. In addition, they also need an interface to access and manage data stored in block and object storage services. A cloud control-stack implements these interfaces, as well as the logic that links the customer-facing interface to low-level components such as the hyper-visor, networking components, and storage technologies. Mature cloud control-stacks will offer a variety of web interfaces, including a human usable interface accessible via a web browser as well as programmatic interfaces accessible over hypertext protocols such as SOAP or REST.

### **3.1.3 Customers and users:**

We distinguish a customer, who is an entity that has a business relationship with the CSP, from a user, which is an identity recognized by the CSP and that can be given certain privileges to customer-owned resources on the CSP. For example, an enterprise may establish a paying customer relationship with a CSP and make its employees users of the CSP. Each user will be given CSP privileges commensurate with his or her role within the enterprise. Another example is a customer who deploys an application on a CSP and distributes the application to end users. The customer will then make its

deployed application a user of the CSP, thus giving it access to components and data hosted on the CSP.

#### **3.1.4 Attack Model:**

Security literature from industry sources generally has two purposes. First, it serves to market products and services to potential customers by informing them of the security mechanisms in those products or services. Second, it provides documentation on how to use the security mechanisms for users of the products and services. Although the latter is often much more detailed than the former, there is no explicit requirement to outline the actual type of attack or threat the mechanism is supposed to protect against this is usually left for the customer to infer.

In contrast, academic papers in security generally identify specific threats and attackers that they will analyze or develop solutions to defend against. As a result, when outlining the attacks and threats customers face in the cloud, we draw from the academic literature. Although each paper only deals with a specific threat, by surveying the threats and attack model of each paper, we can serve as an attack model for which to evaluate any security mechanism, whether proposed in academic or implemented by industry.

To construct this attack model, we begin by first defining the cloud security properties of the customer that an attacker may wish to compromise. These properties include traditional security properties such as confidentiality, integrity, and availability, as well as a new one that is specific to the cloud business model.

#### **3.1.5 Confidentiality:**

Customer confidentiality naturally encompasses the content of customer data that are used and stored on the CSP. In addition, we also include information such as data access patterns and existential information under confidentiality. For example, Google's Cloud Platform explicitly tells customers that the names of objects stored on its service exist in a global namespace, so that any customer can tell whether an object of some particular name exists and is being used by some other customer (although not necessarily which one).<sup>3</sup>

### **3.1.6 Integrity:**

Customer integrity also includes the content of customer data. However, since customers often move to the cloud because they would like to access their data and applications in a distributed setting, we also include the consistency of data under the integrity security property. Applications may depend on the consistency guarantees of the CSP to work properly, and subversion of those guarantees by the cloud can result in subtle vulnerabilities, such as those caused by time-of-check to time-of-use errors.

### **3.1.7 Availability:**

Availability normally defines whether CSP resources are available when the customer wants to access them. In addition, we also include durability, which defines whether the data can be recovered by the customer if the data become unavailable for any reason. Another related concern is “lock-in,” where data may be accessible, but customers are either unable to, or face great difficulty, if they want to retrieve their data and move them to a different CSP.

### **3.1.8 Contractual Security:**

Finally, CSPs typically provide a Service Level Agreement(SLA) that governs performance levels and availability. Furthermore, many promise additional security guarantees such as restricting the storage data to a certain region or legal jurisdiction or cloud-side encryption of data at rest. In exchange for these promises, customers agree to pay fees, thus constituting a contract between the CSP and customer. When either the CSP or the customer attempts to subvert the contract so that more or less service is received for the same amount of money, this is an attack on the contractual security between the CSP and the customer. Customers may attempt to attack the CSP to get more service or may attack other customers in an attempt to reduce the service they receive. CSP billing usually operates by multiplying a rate by an measure, such as the number of hours used or number of bytes transferred. However, the rates themselves are often determined by properties, such as performance, location, or the level of data replication. As a result, the security goal is often to ensure that these difficult to verify properties are properly delivered from the CSP to the customer as agreed.

### 3.1.9 Authentication:

All CSPs surveyed use passwords for authenticating humans. In addition, some offer the option of using a second factor, usually a soft or hard security token. All CSPs also support programmatic access to cloud resources via a web-based API, usually REST or SOAP over SSL. However, CSPs differ heavily on how they perform authentication in their API, ranging from basic HTTP authentication, to cookie-based authentication, to certificate-based authentication, to tickets issued by single-sign-on (SSO) or federated ID systems. In a system using SSO, users authenticate to a single authentication service and obtain a ticket that acts as a capability, which can be given to other humans or embedded in programs to grant them access to cloud resources. These tickets are sometimes also referred to as “security tokens” in the literature, but we use the term “ticket” to distinguish them from the tokens used by humans in multifactor authentication. A variety of SSO standards are in use by CSPs: Google uses Google account credentials with the OAuth 2.0 protocol [Sun and Beznosov 2012; Hammer-Lahav et al. 2012], OpenStack CSPs use OpenStack’s Keystone identity service, Microsoft uses a proprietary protocol called Storage Access Key (SAK) [Kaufman and Venkatapathy 2010], and Amazon uses AWS Identity and Access Management (IAM). Because SSOs issue capabilities and not credentials, they simultaneously provide both authentication and access control.

## Chapter 4

### SYSTEM SPECIFICATION

#### 4.1 Hardware Requirements:

- System : Pentium IV 3.5 GHz or Latest Version.
- Hard Disk : 40 GB.
- Monitor : 14' Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 1 GB.

#### 4.2 Software Requirements:

- Operating system : Windows XP or Windows 7, Windows 8.
- Coding Language : Java / J2EE (Jsp,Servlet)
- Data Base : My Sql Server
- Documentation : MS Office
- IDE : Eclipse Galileo
- Development Kit : JDK 1.6
- Server : Tomcat 6.0

## Chapter 5

### **FEASIBILITY ANALYSIS**

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are:

- **Operational Feasibility**
- **Economic Feasibility**
- **Technical Feasibility**

#### **5.1 Operational Feasibility:**

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

#### **5.2 Economic Feasibility:**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

#### **5.3 Technical Feasibility:**

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The system is technically feasible for development and can be developed with the existing facility.

## Chapter 6

# SYSTEM DESIGN

### 6.1. UML:

#### 6.1.1. Introduction To Uml:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

#### 6.1.2. Definition:

UML is a general purpose visual Unified Modeling Language that is used to specify, visualize, construct and document the artifacts of the software system.

#### 6.1.3 As A Language:

It will provide vocabulary and rules for communication and function on conceptual and physical representation. So it is modeling language.

UML is a graphical language for

1. Visualizing
2. Constructing
3. Documenting
4. Specifying

#### 1. Visualizing:

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

## **2. Constructing**

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

## **3. Documenting**

UML provides variety of documents in addition raw executable codes.

## **4. Specifying**

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

## **6.2 Uml Diagrams:**

It is a language to specifying, visualizing and constructing the artifacts of software system as well as for business models. The UML notation is useful for graphically depicting Object Oriented Analysis and Object Oriented Design (OOA and OOD) modules.

Diagrams are graphical presentation of set of elements. Diagrams project a system, or visualize a system from different angles and perspectives. The UML (Unified Modeling Language) has nine diagrams these diagrams can be classified into the following groups.

### **6.2.1 Static**

- Class diagrams
- Object diagrams
- Component diagrams
- Deployment diagrams

### **6.2.2 Dynamic**

- Use case diagram
- Sequence diagram
- State chart diagram

- Activity diagram
- Collaboration diagram

### **6.3 Modules:**

#### **6.3.1 Data Owner:**

The main function of Data Owner is to select domain and browse file and split file into 4 blocks. Data Owner encrypts and uploads with its MAC with block verification permission. Data Owner updates blocks and delete blocks.

#### **6.3.2 Domain Manager:**

The main function of Domain Manager is to add all domains and view all domains. Domain Manager view all file blocks verification based on specified domain.

#### **6.3.3 Trusted Third Party:**

It is an entity trusted by other components. Trusted Third party view all meta data of all clouds and verify all blocks. Trusted Third Party send block verification details to corresponding data owner.

#### **6.3.4 Attacker:**

Attacker is used to attack blocks in cloud server.

#### **6.3.5 End User:**

End user is to request file and request secret key. End user can download a file.

## 6.4 Architecture Diagram

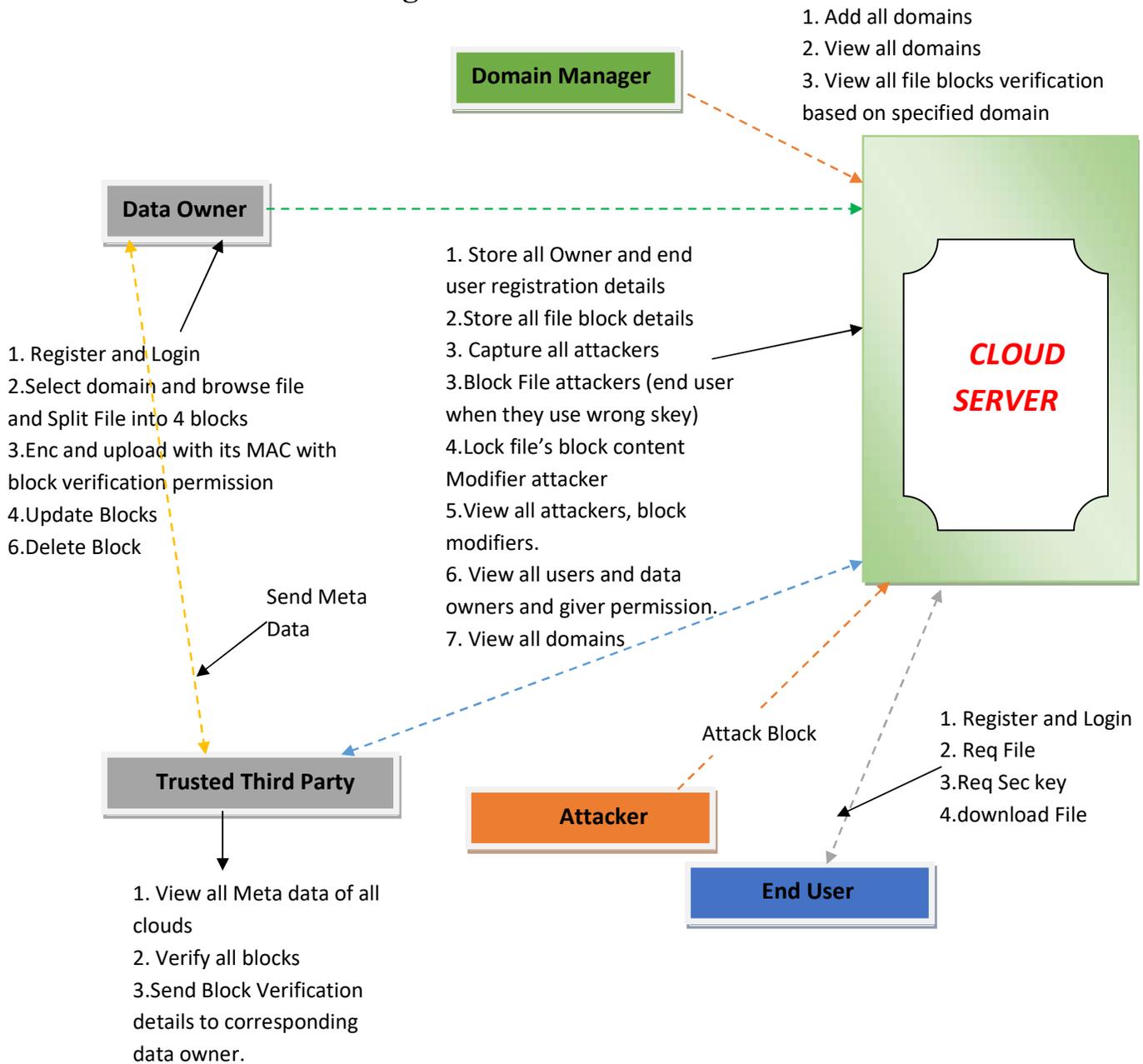


Fig6.4Architecture Diagram

## 6.5 Class Diagram

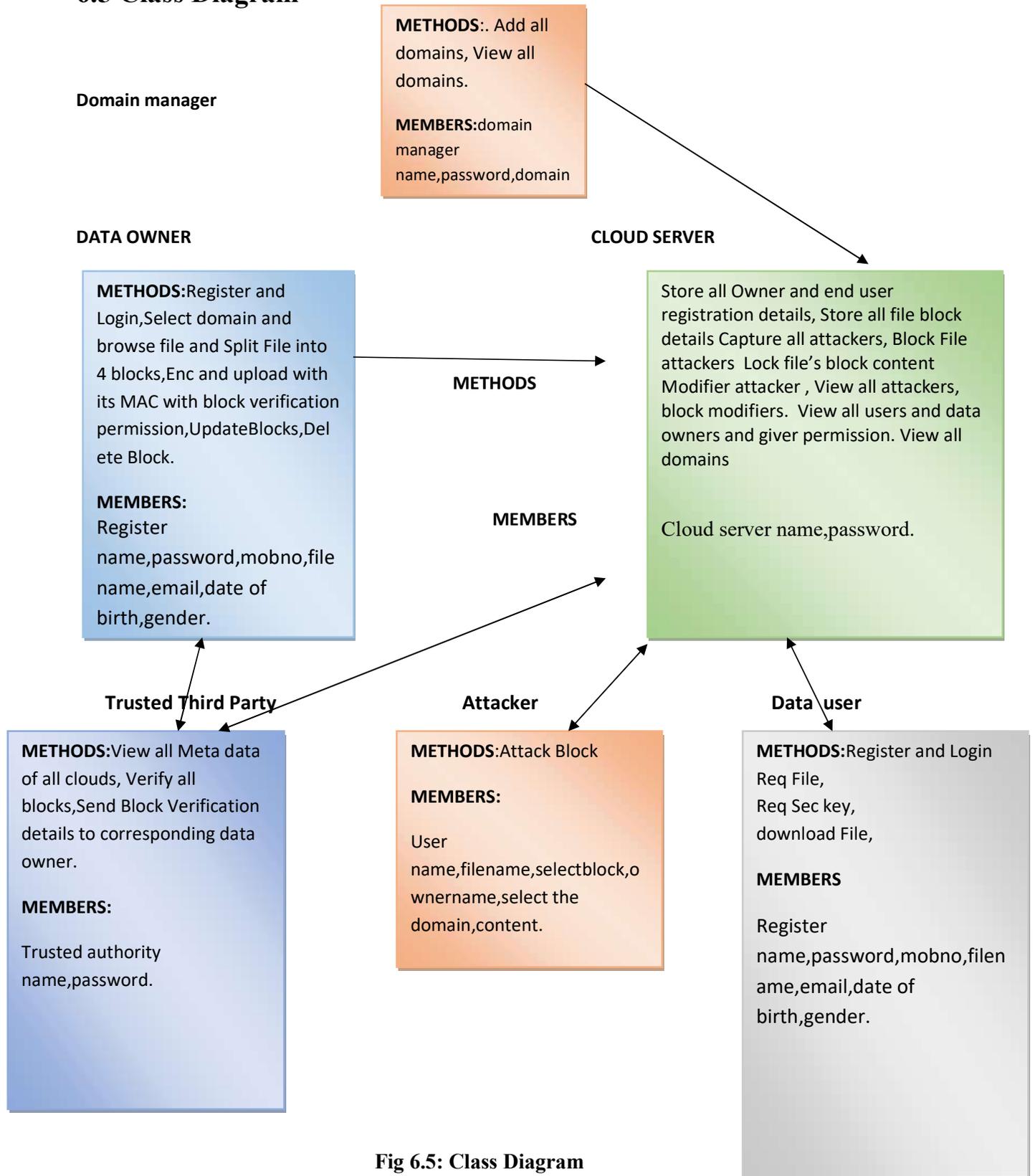


Fig 6.5: Class Diagram

## 6.6 Use Case Diagram

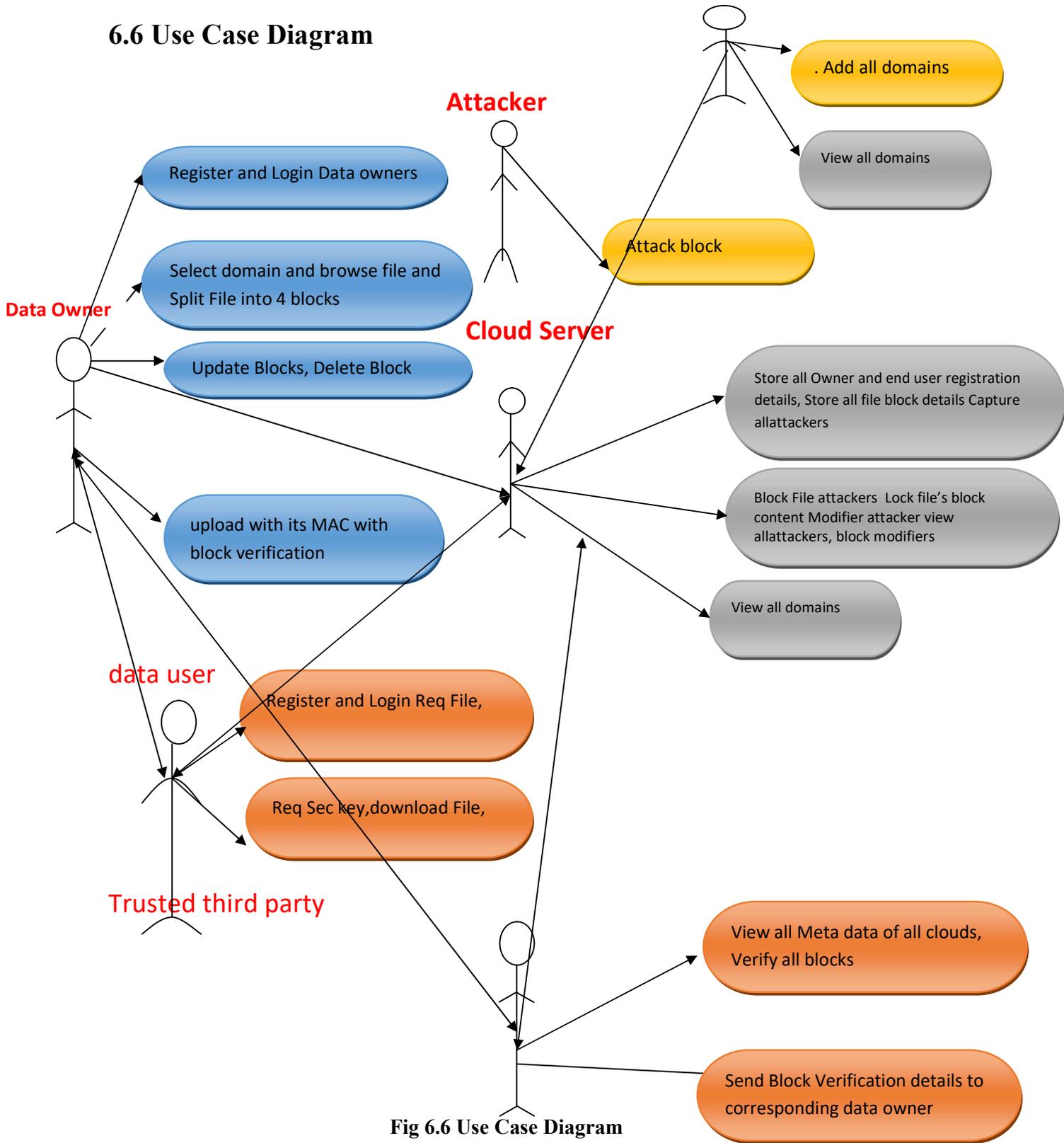


Fig 6.6 Use Case Diagram

## 6.7 Sequence Diagram

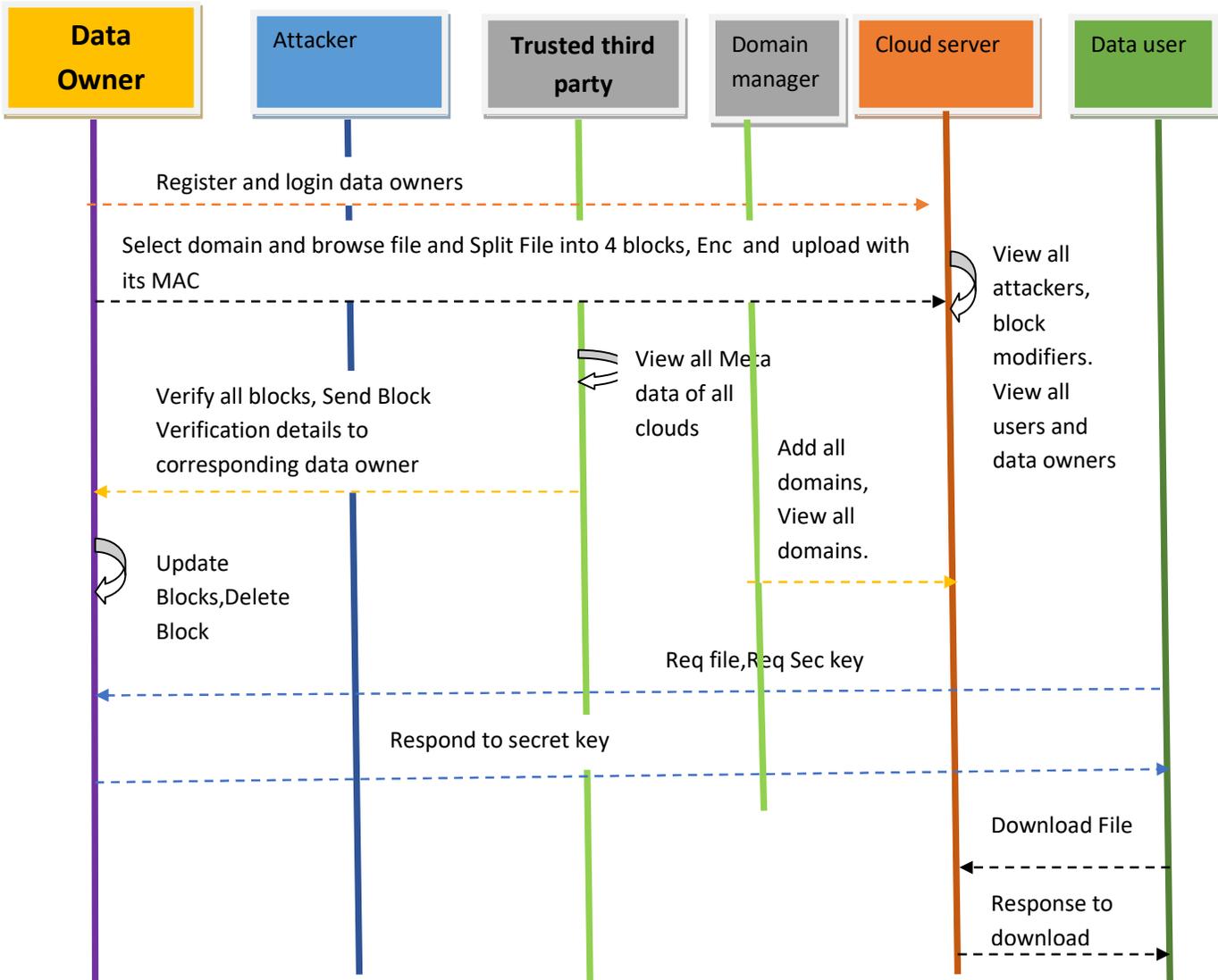


Fig 6.7 Sequence Diagram

## Chapter 7

### SYSTEM TESTING

#### **7.1 Test case Description:**

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

#### **7.2 Psychology of Testing:**

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work. Testing is the process of executing a program with the intent of finding errors.

#### **7.3 Testing Objectives:**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

#### **7.4 Levels of Testing:**

In order to uncover the errors present in different phases we have the concept of levels of testing.

##### **7.4.1 System Testing:**

The philosophy behind testing is to find errors. Test cases are devised with this in mind. A strategy employed for system testing is code testing.

### **7.4.2 Code Testing:**

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

## **7.5 Types of Testing**

### **7.5.1 Installation Testing:**

Most software systems have installation procedures that are needed before they can be used for their main purpose. Testing these procedures to achieve an installed software system that may be used is known as installation testing.

### **7.5.2 Compatibility Testing:**

A common cause of software failure is a lack of its compatibility with other application software, operating systems or target environments that differ greatly from the original (such as a terminal or GUI application intended to be run on the desktop now being required to become a web application, which must render in a Web browser).

### **7.5.3 Regression Testing:**

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly, stops working as intended.

## Chapter 8

### SYSTEM ENVIRONMENT

#### 8.1 Client Server

With the varied topic in existence in the fields of computers, Client Server is one, which has generated more heat than light, and also more hype than reality. This technology has acquired a certain critical mass attention with its dedication conferences and magazines. Major computer vendors such as IBM and DEC, have declared that Client Servers is their main future market. A survey of DBMS magazine revealed that 76% of its readers were actively looking at the client server solution. The growth in the client server development tools from \$200 million in 1992 to more than \$1.2 billion in 1996.

Client server implementations are complex but the underlying concept is simple and powerful. A client is an application running with local resources but able to request the database and relate the services from separate remote server. The software mediating this client server interaction is often referred to as MIDDLEWARE.

The typical client either a PC or a Work Station connected through a network to a more powerful PC, Workstation, Midrange or Main Frames server usually capable of handling request from more than one client. However, with some configuration server may also act as client. A server may need to access other server in order to process the original client request.

The key client server idea is that client as user is essentially insulated from the physical location and formats of the data needs for their application. With the proper middleware, a client input from or report can transparently access and manipulate both local database on the client machine and remote databases on one or more servers. An added bonus is the client server opens the door to multi-vendor database access indulging heterogeneous table joins.

##### 8.1.1 Why Client Server

Client server has evolved to solve a problem that has been around since the earliest days of computing: how best to distribute your computing, data generation and data storage resources in order to obtain efficient, cost effective departmental an enterprise wide data processing. During mainframe era choices were quite limited. A central machine housed both

the CPU and DATA (cards, tapes, drums and later disks). Access to these resources was initially confined to batched runs that produced departmental reports at the appropriate intervals. A strong central information service department ruled the corporation. The role of the rest of the corporation limited to requesting new or more frequent reports and to provide hand written forms from which the central data banks were created and updated. The earliest client server solutions therefore could best be characterized as “SLAVE-MASTER”.

Time-sharing changed the picture. Remote terminal could view and even change the central data, subject to access permissions. And, as the central data banks evolved in to sophisticated relational database with non-programmer query languages, online users could formulate adhoc queries and produce local reports with out adding to the MIS applications software backlog. However remote access was through dumb terminals, and the client server remained subordinate to the Slave Master.

### **8.1.2 Front End**

The entire user interface is planned to be developed in browser specific environment with a touch of Intranet-Based Architecture for achieving the Distributed Concept. The browser specific components are designed by using the HTML standards, and the dynamism of the designed by concentrating on the constructs of the Java Server Pages.

## **8.2 Database Connectivity**

The Communication architecture is designed by concentrating on the Standards of Servlets and Enterprise Java Beans. The database connectivity is established by using the Java Data Base Connectivity.

The standards of three-tier architecture are given major concentration to keep the standards of higher cohesion and limited coupling for effectiveness of the operations.

## 8.3 Features Of The Language Used:

In our project we have used the language called java.

### 8.3.1 About Java:

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

### 8.3.2 Importance of Java to the Internet:

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

### 8.3.3 Java can be used to create two types of programs:

**Applications and Applets:** An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

## **8.4 Features Of Java:**

### **8.4.1 Security:**

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

### **8.4.2 Portability:**

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

### **8.4.3 The Byte code:**

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

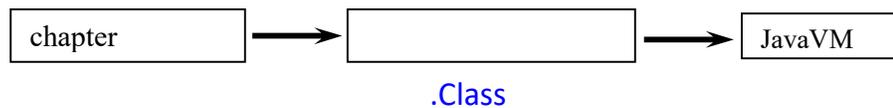
Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

#### 8.4.4 Java Virtual Machine (JVM):

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

#### 8.4.5 Overall Description:



**Fig 8.4.5:**Picture showing the development process of JAVA Program

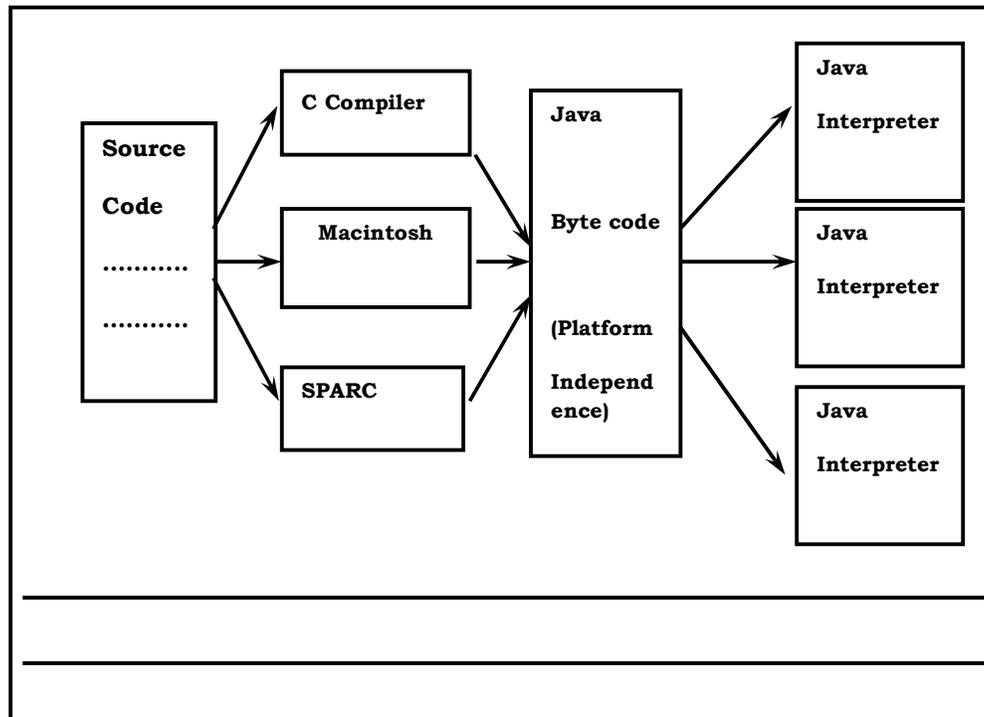
Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a .Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a .class file, which contains the byte code. The .Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

#### 8.5 Java Architecture:

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

### 8.5.1 Compilation of code:

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine



**Fig 8.5.1:**This figure shows how the compiling and interpreting will be done by source code.

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or Sun SARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

### 8.5.2 Simple:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier.

Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

### **8.5.3 Object-Oriented:**

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

### **8.5.4 Robust:**

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.

Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors can and should be managed by your program.

## **8.6 Java Script:**

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser's display accordingly.

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags.

```
<SCRIPTS>..</SCRIPT>.
```

```
<SCRIPT LANGUAGE = "JavaScript">
```

```
JavaScript statements
```

```
</SCRIPT>
```

Here are a few things we can do with JavaScript :

- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser's status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.
- Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application

### 8.6.1 Java Script VS Java:

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

#### Advantages:

- JavaScript can be used for Sever-side and Client-side scripting.
- It is flexible.
- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

## 8.7 Hyper Text Markup Language:

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produce Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized words that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

### 8.7.1 Basic HTML Tags :

<code>&lt;!-- --&gt;</code>	Specifies comments
<code>&lt;A&gt;.....&lt;/A&gt;</code>	Creates hypertext links
<code>&lt;B&gt;.....&lt;/B&gt;</code>	Formats text as bold
<code>&lt;BIG&gt;.....&lt;/BIG&gt;</code>	Formats text in large font.
<code>&lt;BODY&gt;...&lt;/BODY&gt;</code>	Contains all tags and text in the HTML document
<code>&lt;CENTER&gt;...&lt;/CENTER&gt;</code>	Creates text
<code>&lt;DD&gt;...&lt;/DD&gt;</code>	Definition of a term
<code>&lt;DL&gt;...&lt;/DL&gt;</code>	Creates definition list

<b>&lt;FONT&gt;...&lt;/FONT&gt;</b>	Formats text with a particular font
<b>&lt;FORM&gt;...&lt;/FORM&gt;</b>	Encloses a fill-out form
<b>&lt;FRAME&gt;...&lt;/FRAME&gt;</b>	Defines a particular frame in a set of frames
<b>&lt;H#&gt;...&lt;/H#&gt;</b>	Creates headings of different levels
<b>&lt;HEAD&gt;...&lt;/HEAD&gt;</b>	Contains tags that specify information about a document
<b>&lt;HR&gt;...&lt;/HR&gt;</b>	Creates a horizontal rule
<b>&lt;HTML&gt;...&lt;/HTML&gt;</b>	Contains all other HTML tags
<b>&lt;META&gt;...&lt;/META&gt;</b>	Provides meta-information about a document
<b>&lt;SCRIPT&gt;...&lt;/SCRIPT&gt;</b>	Contains client-side or server-side script
<b>&lt;TABLE&gt;...&lt;/TABLE&gt;</b>	Creates a table
<b>&lt;TD&gt;...&lt;/TD&gt;</b>	Indicates table data in a table
<b>&lt;TR&gt;...&lt;/TR&gt;</b>	Designates a table row
<b>&lt;TH&gt;...&lt;/TH&gt;</b>	Creates a heading in a table

#### **ADVANTAGES:**

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent.
- HTML tags are not case-sensitive.

## **8.8 Java Database Connectivity:**

### **8.8.1 What Is JDBC?**

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the

Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

### **8.8.2 What Does JDBC Do?**

Simply put, JDBC makes it possible to do three things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

### **8.9 JDBC versus ODBC and other APIs:**

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms.

So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void \*". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on

every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

### **8.9.1 Two-tier and Three-tier Models:**

The JDBC API supports both two-tier and three-tier models for database access. In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet.

In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages.

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.

### **8.9.2 JDBC Driver Types:**

The JDBC drivers that we are aware of at this time fit into one of four categories:

- JDBC-ODBC bridge plus ODBC driver
- Native-API partly-Java driver

- JDBC-Net pure Java driver
- Native-protocol pure Java driver

### **8.9.3 JDBC-ODBC Bridge:**

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

### **8.9.4 What is the JDBC- ODBC Bridge?**

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of JavaSoft.

### **8.9.5 Java Server Pages (JSP):**

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headers, it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other's work.

## **8.10 Features of JSP:**

### **8.10.1 Portability:**

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

### **8.10.2 Components :**

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components currently supported include Java Beans, and Servlets.

### **8.10.3 Processing :**

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

### **8.11 Access Models:**

A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

#### **8.11.1 Steps in the execution of a JSP Application:**

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.
2. This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can understand the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the

JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

### 8.11.2 JDBC connectivity:

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Component Provider to:

- Perform connection and authentication to a database server
- Manager transactions
- Move SQL statements to a database engine for preprocessing and execution
- Execute stored procedures
- Inspect and modify the results from Select statements.

### 8.12 Tomcat 6.0 web server:

Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Weblogic, is one of the popular application server).

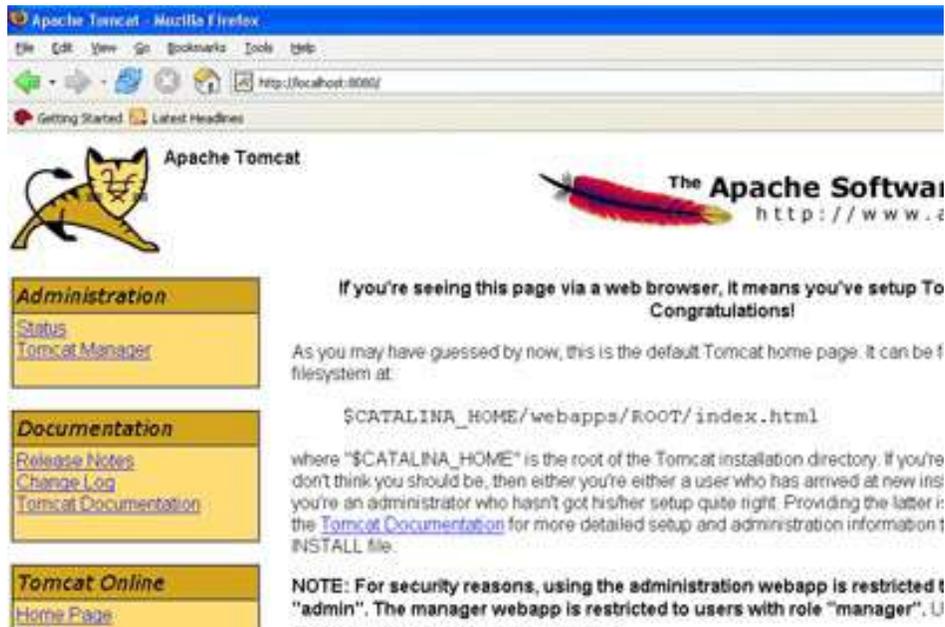


Fig8.11:Tomcat server for running the web services.

## Chapter 9

### Cloud OS

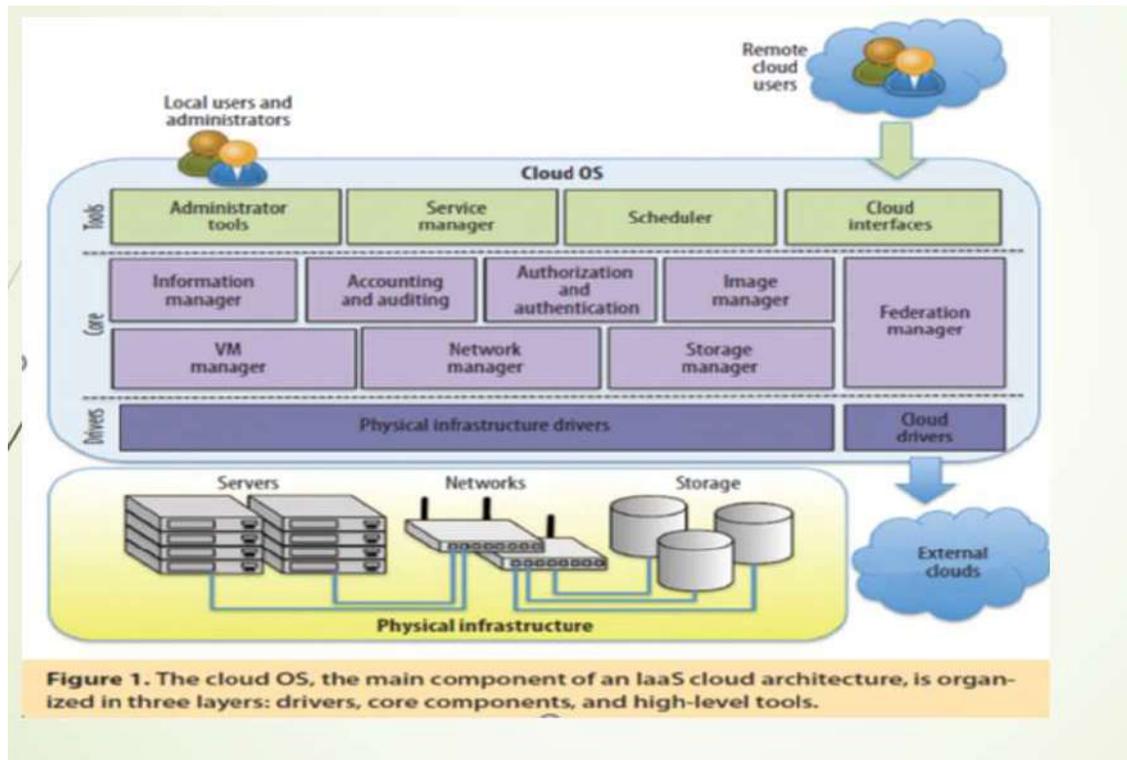


Fig9.0: Cloud OS

### 9.1 The Cloud OS:

As a key component in a modern datacenter, the cloud operating system is responsible for:

1. managing the physical and virtual infrastructure.
2. orchestrating and commanding service provisioning and deployment.
3. providing federation capabilities for accessing and deploying virtual resources in remote cloud infrastructures.

### **9.1.1 Infrastructure and Cloud Drivers:**

- To provide an abstraction of the underlying infrastructure, technology, the cloud OS can use adapters or drivers to interact with a variety of virtualization technologies. These include hypervisor, network, storage, and information drivers.
- The core cloud OS components, including the virtual machine (VM) manager, network manager, storage manager, and information manager, rely on these infrastructure drivers to deploy, manage, and monitor the virtualized infrastructures.
- In addition to the infrastructure drivers, the cloud OS can include different cloud drivers to enable access to remote providers.

### **9.1.2 Virtual Machine Manager:**

- A cloud OS defines the VM as the basic execution unit and the virtualized services (group of VMs for executing a multitier service) as the basic management entity.
- This concept helps create scalable applications because the user can either add VMs as needed (horizontal scaling) or resize a VM (if supported by the underlying hypervisor technology) to satisfy a VM workload increase (vertical scaling).
- Individual multitier applications are isolated from each other, but individual VMs in the same applications are not, as they all can share a communication network and services when needed.
- A VM consists of a set of parameters and attributes, including the OS kernel, VM image, memory and CPU capacity, network interfaces etc.

### **9.1.3 Service Manager:**

- The cloud OS should be able to manage and support virtualized multitier services. A multitier service can comprise several component/tiers with some intrinsic dependencies among them. These services can be deployed as a group of interconnected VMs in the cloud with specific deployment dependencies and, optionally, some location, affinity, and elasticity requirements.

- The service manager's admission control function entails deciding whether to accept or reject a service, depending on the service requirements and resource availability in the cloud.
- Once it accepts a service, the service manager is responsible for managing its life cycle, which can involve several actions, including deploying, suspending, resuming, or canceling the service.
- To deploy a new service, the service manager interacts with the scheduler to decide the best placement for the various VMs that comprise the service, according to the selected optimization criteria and service constraints.

## Chapter 10

### MODULES

#### 10.1 MODULE DESCRIPTION

- ❖ Secure Storage
- ❖ Trusted Platform module
- ❖ Storage Protection Construction
- ❖ Test bed Architecture
- ❖ Implementation
- ❖ Algorithm

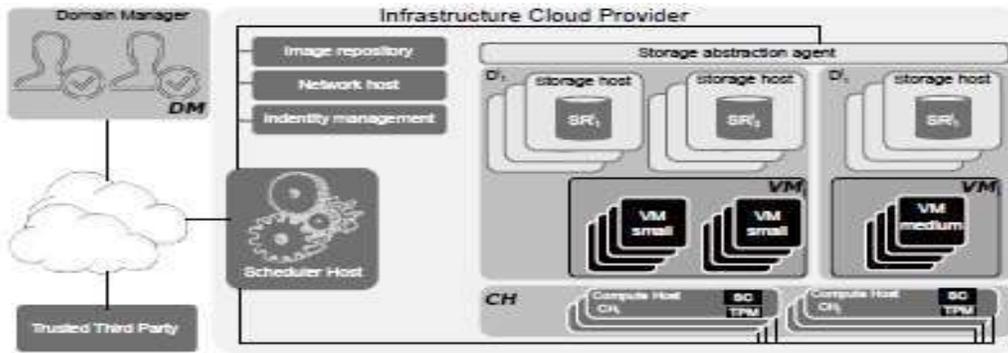
#### 10.2 Secure Storage:

A secure platform architecture based on a secure root of trust for grid environments of cloud computing. Trusted Computing is used as a method for dynamic trust establishment within the grid, allowing clients to verify that their data will be protected against malicious host attacks. The authors address the malicious host problem in grid environments, with three main risk factors: trust establishment, code isolation and grid middleware. The solution established a minimal trusted computing base (TCB) by introducing a security manager isolated by the hypervisor from grid services (which are in turn performed within VM instances). The secure architecture is supported by protocols for data integrity protection, confidentiality protection and grid job attestation. In turn, these rely of client attestation of the host running the respective jobs, followed by interaction with the security manager to fulfill the goals of the respective protocols. We follow a similar approach in terms of interacting with a minimal TCB for protocol purposes following host attestation. However, in order to adapt to the cloud computing model we delegate the task of host attestation to an external TTP as well as use TPM functionality to ensure that sensitive cryptographic material can only be accessed on a particular attested host. In , the authors proposed an approach to protect access to outsourced data in an owner-write-users-read case, assuming an “honest but curious service provider”. Encryption is done over (abstract) blocks of data, with a different key per block. The authors suggest a key derivation hierarchy based on a public hash function, using

the hash function result as the encryption key. The scheme allows to selectively grant data access, uses over-encryption to revoke access rights and supports block deletion, update, insertion and appending. It adopts a lazy revocation model, allowing to indefinitely maintain access to data reachable prior to revocation (regardless of whether it has been accessed before access revocation). While this solution is similar to our model with regard to information blocks and encryption with different symmetric keys, we propose an active revocation model, where the keys are cached for a limited time and cannot be retrieved once the access is revoked.

### **10.3 Trusted Platform Module:**

A Hardware cryptographic co-processor following specifications of the Trusted Computing Group (TCG); we assume CH are equipped with a TPM. The tamper-evident property facilitates monitoring CH integrity and strengthens the assumption of physical security. An active TPM records the platform boot time software state and stores it as a list of hashes in platform configuration registers (PCRs). TPM v1.2 has 16 PCRs reserved for static measurements (PCR0 - PCR15), cleared upon a hard reboot. Additional runtime resettable registers (PCR16-PCR23) are available for dynamic measurements. Endorsement keys are an asymmetric key pair stored inside the TPM in the trusted platform supply chain, used to create an endorsement credential signed by the TPM vendor to certify the TPM specification compliance. A message encrypted (“bound”) using a TPM’s public key is decryptable only with the private key of the same TPM. Sealing is a special case of binding bound messages are only decryptable in the platform state defined by PCR values. Platform attestation allows a remote party to authenticate a target platform and obtain a guarantee that it is up to a certain level in the boot chain – runs software that is identical to the expected one. To do this, an attester requests – accompanied by a nonce – the target platform to produce an attestation quote and the measurement aggregate, or Integrity Measurement List (IML). The TPM generates the attestation quote – a signed structure that includes the IML and the received and returns the quote and the IML itself. The attestation quote is signed with the TPM’s Attestation Identity Key (AIK). The exact IML contents are implementation-specific, but should contain enough data to allow the verifier to establish the target platform integrity. We refer to for a description of the TPM, and to for protocols that use TPM functionality.



**Fig 10.3. Infrastructure Cloud Provider**

#### 10.4 Storage Production construction:

We now describe two protocols that constitute the core of this paper's contribution. These protocols are successively applied to deploy a cloud infrastructure providing additional user guarantees of cloud host integrity and storage security. For protocol purposes, each domain manager, secure component and trusted third party has a public/private key pair ( $pk=sk$ ).

The private key is kept secret, while the public key is shared with the community. We assume that during the initialization phase, each entity obtains a certificate via a trusted certification authority. We first describe the cryptographic primitives used in the proposed protocols, followed by definitions of the main protocol components. Upon reception, SC verifies message integrity and TL:Token freshness by checking respectively the signature  $DM_i$  and nonce  $r$ . When SC first receives a TL:Request message, it uses the local TPM to generate a new pair of TPMbased public/private bind keys,  $(pk_{TPM}; sk_{TPM})$ , which can be reused for future launch requests, to avoid the costly key generation procedure. Keys can be periodically regenerated according to a cloud provider-defined policy. To prove that the bind keys are non-migratable, PCR-locked, public/private TPM keys, SC retrieves the TPM\_CERTIFY\_INFO structure, signed with the TPM attestation identity key  $pk_{AIK}$  using the TPM\_CERTIFY\_KEY TPM command; we denote this signed structure by TCI. TPM\_CERTIFY\_INFO contains the bind key hash and the PCR value required to use the key; PCR values must not necessarily be in a trusted state to create a trusted bind key pair. This mechanism is explained in further detail.

## 10.5 Test bed Architecture:

Along with three of the entities already active in the TL protocol – domain manager, secure component, the trusted third party – DBSP employs a fourth one: the storage resource. In this case, DMi interacts with the other protocol components through a VM instance vml running on CHi. We assume that vml has been launched following the TL protocol. The DBSP protocol includes a public and a private encryption scheme, a pseudorandom function for domain key generation, a signature scheme and a random generator. Figure 3 presents the DBSP protocol message flow.

TL:Token Option 1: A modification can only be achieved by the adversary by either breaking the public key encryption scheme used to produce c1 or trying to make this modification on c1 by direct modification (without first decrypting it) and sign the modified c1 with an own selected private key. The former option fails due to the assumption of public key encryption scheme soundness and the latter due to that modifying a public encrypted structure without knowledge of the private key is infeasible. –

TL:Token Option 2: Direct decryption of c1 fails due to the assumption of soundness of the public key encryption scheme used to produce c1. The only remaining alternative for the adversary is relaying the TL:Token to a platform CH0 2 CHSPi , which is under the full control of the adversary. Further, ADV follows the protocol and issues the command.

TL: Request using the intercepted c1, AttestData and AIK. However, this fails at the TL:Astep since AttestData does not contain a valid AIK certificate unless the adversary has managed to get control of a valid platform in the provider network with a valid certificate or she has managed to break the AIK certification scheme. The former option violates the assumption of physical security of the provider computing resources while the latter option violates the assumption of a sound public key and AIK certification schemes.

## 10.6 Implementation:

The prototype also includes a distributed EHR system deployed over seven VM instances. This system contains one client VM, two front-end VMs, two back-end VMs, a database VM and an auxiliary external database VM. Six of the VM instances operate on Microsoft Windows Server 2012 R2, with one VM running the client application operates on Windows 7. The components of the EHR system communicate using statically defined IP

addresses on the respective VLANs described in Section. Load balancing functionality provided by the underlying IaaS allots the load among front-end and back-end VM pairs. The hosts of the cluster are compatible with the TL protocol, which allows an infrastructure administrator to perform a trusted Another conclusion is that while organizations operating on sensitive data, e.g. public healthcare authorities, consider the risks of migrating data to IaaS clouds as unacceptable, the majority of available providers use commercial of the (COTS) cloud platforms with limited capabilities to enhance the security of their deployments, failing to meet the customer requirements. This demonstrates the need to incorporate integrity verification and data protection mechanisms into popular COTS cloud platforms by default. We hope that these important lessons will inspire new secure, usable and cost-effective solutions for cloud services. On the practical side, specifically regarding the role of the TTP, we envision two scenarios. The TTP could either be managed by the tenant itself (for organizations with enough resources and expertise), or by an external organization (similar to a certificate authority). The first scenario allows the tenant to retain the benefits of cloud services along with additional security guarantees. Similarly, in the second scenario, smaller actors can obtain the same TTP would only operate on a physical slice of the resources (i.e. a subset of compute hosts) that correspond to the respective tenant domains.

### **10.7 Algorithm:**

A message authentication code (MAC) is a cryptographic checksum on data that uses a session key to detect both accidental and intentional modifications of the data.

#### **10.7.1 Encryption Algorithm:**

In cryptography, encryption is the process of encoding messages or information in such a way that only authorized parties can read it. Encryption does not of itself prevent interception, but denies the message content to the interceptor.

We focus on the Infrastructure-as-a-Service model in a simplified form, it exposes to its tenants a coherent platform supported by compute hosts which operate VM guests that communicate through a virtual network.

To address this, we propose a set of protocols for trusted launch of virtual machines (VM) in IaaS, which provide with a proof that the requested VM instances were launched on a host with an expected software stack.

### **10.7.2 Decryption Algorithm:**

The decryption algorithms specify the data and key encryption algorithms that are used to decrypt the SOAP message. The WSS API for decryption (WSSDecryption) specifies the algorithm uniform resource identifier (URI) of the data and key encryption methods. The WSSDecryption interface is the `com.ibm.websphere.wssecurity.wssapi.decryption` package.

Data decryption algorithms specify the algorithm uniform resource identifier (URI) of the data encryption method. WebSphere Application Server supports the following pre-configured data decryption algorithms.

### **10.7.3 Message Authentication Code:**

In cryptography, a message authentication code (MAC), sometimes known as a tag, is a short piece of information used to authenticate a message in other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed. The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

### **10.7.4 Session Key:**

A session key is a single use symmetric key used for encryption all messages in one for communication session. A closely related term is content encryption key (CEK), traffic encryption key (TEK), or multicast key which refers to any key used to encrypt messages, as opposed to other uses, like encrypting other keys (key encryption key (KEK) or key wrapping key).

Session keys can introduce complication into a system. However, they solve some real problems. There are two primary reasons to use session keys:

1. Several cryptanalytic attacks become easier as more material encrypted with a specific key is available. By limiting the amount of data processed using a particular key, those attacks are made more difficult.

2. asymmetric encryption is too slow for many purposes, and all secret key algorithms require that the key is securely distributed. By using an asymmetric algorithm to encrypt the secret key for another, faster, symmetric algorithm, it's possible to improve overall performance considerably. This is the process used by PGP and GPG.

Like all cryptographic keys, session keys must be chosen so that they cannot be predicted by an attacker, usually requiring them to be chosen randomly. Failure to choose session keys (or any key) properly is a major (and too common in actual practice) design flaw in any crypto system.

#### **10.7.5 Checksum Algorithm:**

A checksum is a value which is computed which allows you to check the validity of something. Typically, checksums are used in data transmission contexts to detect if the data has been transmitted successfully.

Checksum take on various forms, depending upon the nature of the transmission and the needed reliability. For example, the simplest checksum is to sum up all the bytes of a transmission, computing the sum in an 8-bit counter. This value is appended as the last byte of the transmission. The idea is that upon receipt of  $n$  bytes, you sum up the first  $n-1$  bytes, and see if the answer is the same as the last byte. Since this is a bit awkward, a variant on this theme is to, on transmission, sum up all the bytes, the (treating the byte as a signed, 8-bit value) checksum byte before transmitting it. This means that the sum of all  $n$  bytes should be 0. These techniques are not terribly reliable; for example, if the packet is known to be 64 bits in length, and you receive 64 '\0' bytes, the sum is 0, so the result must be correct. Of course, if there is a hardware failure that simply fails to transmit the data bytes (particularly easy on synchronous transmission, where no "start bit" is involved), then the fact that you receive a packet of 64 0 bytes with a checksum result of 0 is misleading; you think you've received a valid packet and you've received nothing at all. A solution to this is to do something like negate the checksum value computed, subtract 1 from it, and expect that the result of the receiver's checksum of the  $n$  bytes is 0xFF (-1, as a signed 8-bit value). This means that the 0-lossage problem goes away.

## Chapter 11

### OUTPUT SCREENS

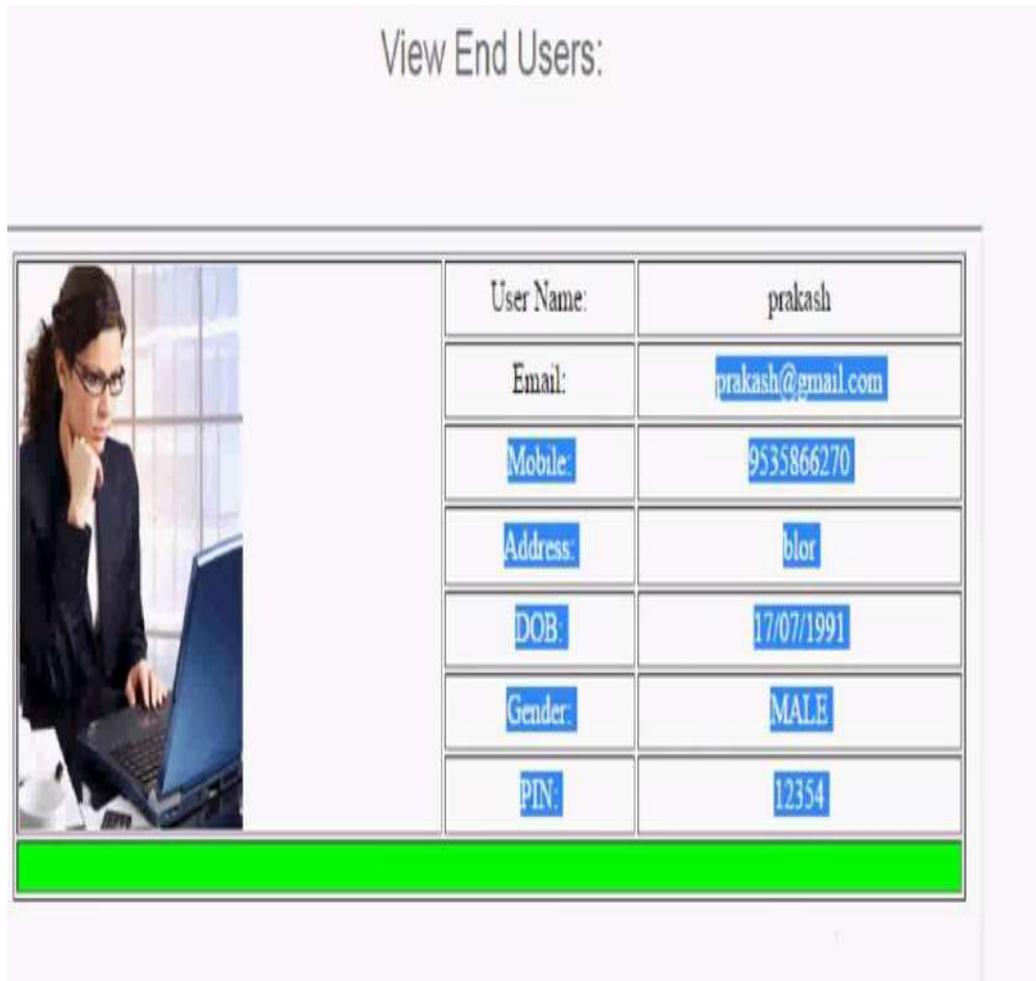


Cloud Server Login!!!!!!

Enter Cloud Server Name:-	cloud
Enter Password:-	*****
	Login

**Fig 11.1: Cloud Server Login Page**

The above describes that how to login into the cloud with cloud server name and password.



**Fig 11.2: View End Users**

The above figure describes that, we can view the end users who are going to be registered into the cloud.

View File Requests:

EndUser Name	File Request	Req Date Time	Processed Status
--------------	--------------	---------------	------------------

**Fig 11.3: View File Request page**

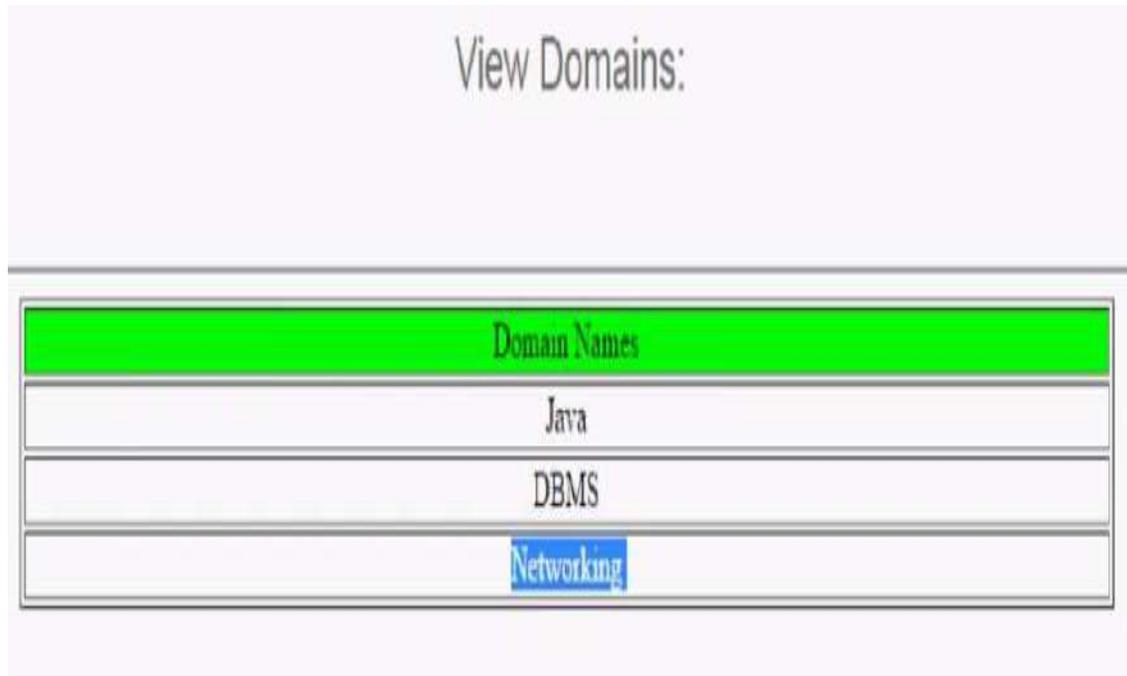
The above figure describes that we can view the file requests from the end users.

View Blocked Users:

User Name	File Name	Owner Name	SK	Date	Un-Block
-----------	-----------	------------	----	------	----------

**Fig11.4:View Blocked Users**

The above figure describes that, We can view the blocked users who are blocked from the trusted third party



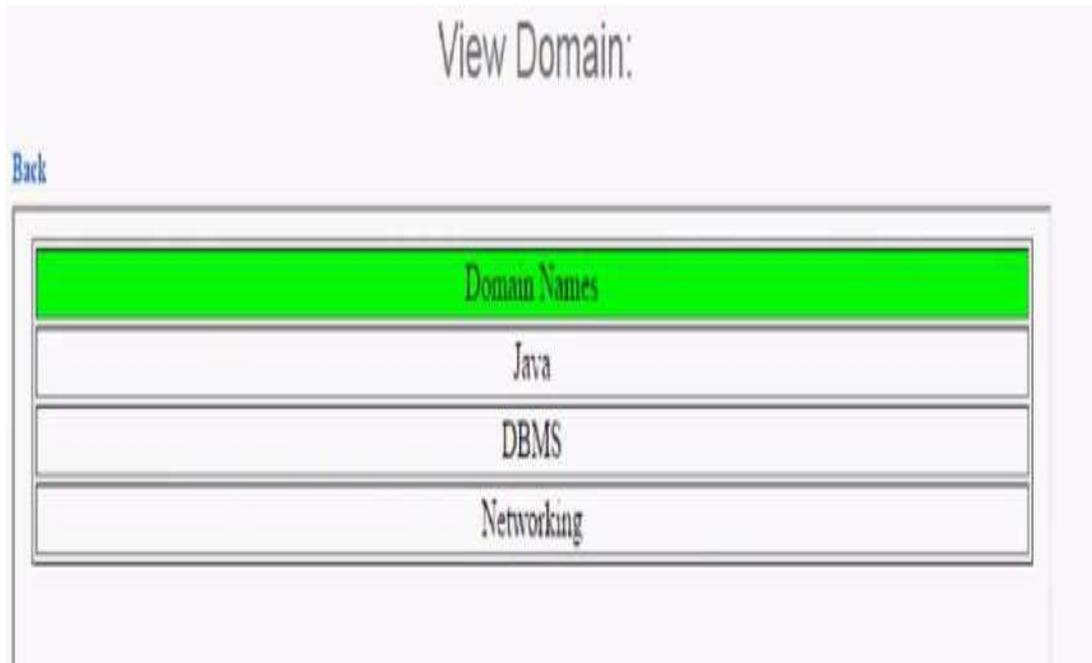
**Fig 11.5: View Domains:**

The above figure describes that, We can view the domain names like what we going to search.



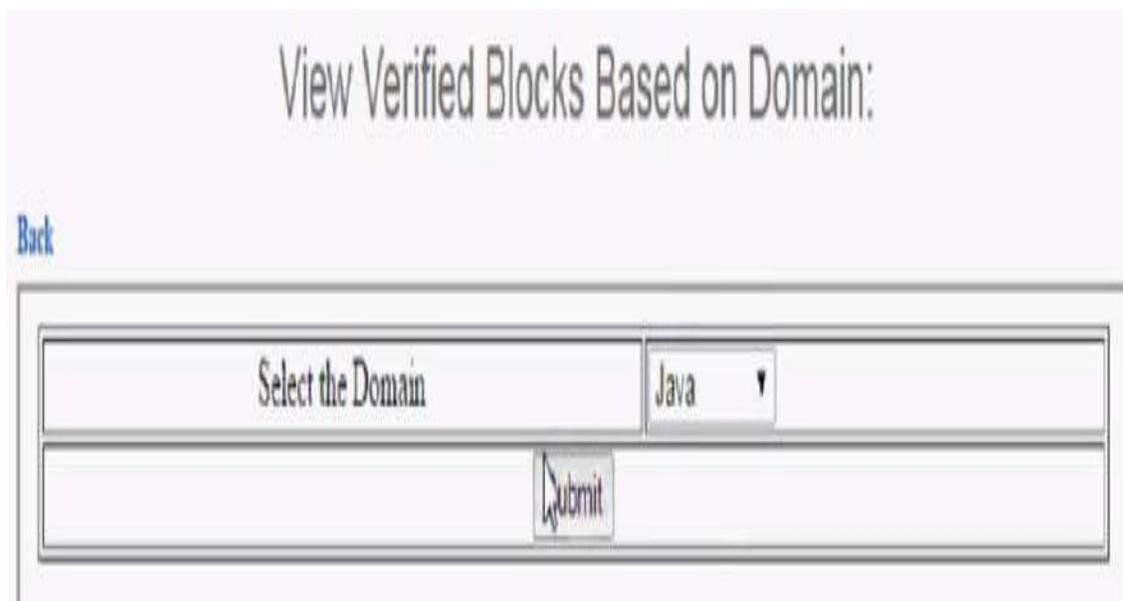
**Fig 11.6: View Block Details**

The above figure describes that, We can view the block details of the domains or files.

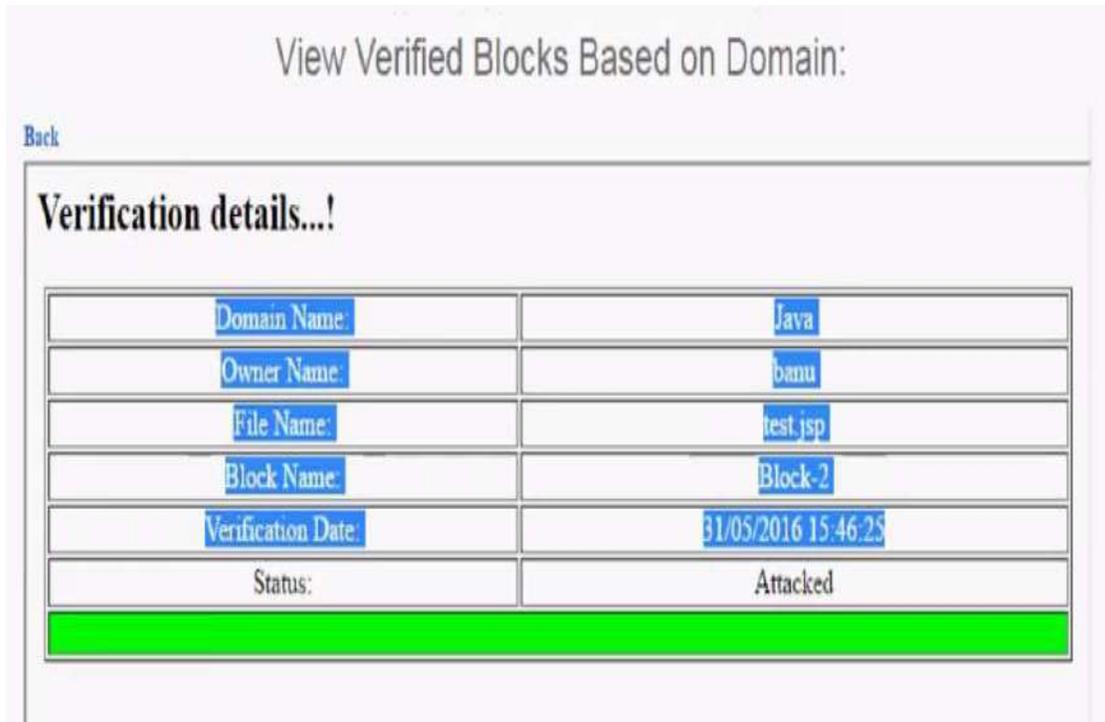


**Fig 11.7: View Domains**

The above figure describes that, We can view the domains that are presented in cloud.



**Fig 11.8: We can verify the blocks based on the domains.**



**Fig 11.9:View Verification Details**

The above figure describes that,We can view the verified blocks based on domain.



**Fig 11.10:Trusted Third Party Login**

By using this page we can login into the trusted third party form based on specified username and password.

View Meta Data:

File Name	test.jsp
Owner Name	banu
MAC-1	-6a668cdaff2ca9d3a4c068341a2924d81470caea
MAC-2	-30b0fa2af8e82a5330944466dd9bbe36d5cb79e
MAC-3	348864444b44bab140a550278527b169fc635745
MAC-4	7b325699a92006f0c07a9dcb214db4a29e3e3022
Sk	[B@105eb6f
Domain	Java

**Fig 11.11: View Meta Data**

The above describes that, We can view the meta data that is all mac addresses.

Verify Block Data:

[Back](#)

Select Domain :-	Java ▼
Owner Name :-	banu
File Name :-	test.jsp
Select the Block	Block-2 ▼
	Verify

**Fig 11.12: Verify Block Data.**

The above figure describes that, using this page we can verify the particular blocks are presented or not.

Verify Block Data:

[Back](#)

**File is safe details Shown Below**

Domain Name :-	Java
File Name :-	test.jsp
Select the Block	Block-2
Cloud Server Mac :-	-30b0fa2af8e82a5330944466dd9bbef36d5cb79e
TTP Mac :-	-30b0fa2af8e82a5330944466dd9bbef36d5cb79e

**Fig 11.13:** View Safe Details

The above figure describes that, We can verify the block data by using the domain name. the domain names such as Java, DBMS, Networking, we can select any one of the domain name. By using this domain name and file name we can verify the block details.

Data Owner Login!!!!!!

Enter DataOwner Name:-	User Name
Enter Password:-	Password
<a href="#">Register</a>	Login

**Fig 11.14:Data Owner Login.**

The above figure describes that, By using this login form we have to enter all the details based on given to login as data owner.

### Data Owner Registration!!!!!!

[click here to Login](#)

User Name (required)	manjunath
Password (required)	*****
Email Address (required)	tmksmanju13@gmail.com
Mobile Number (required)	9535866270
Your Address	Rajaji Nagar,Bnagalore
Date of Birth (required)	05/06/1985
Select Gender (required)	MALE ▾
Enter Pincode (required)	560021
Enter Location (required)	Bangalore
Select Profile Picture (required)	Choose file User.jpg
	Register

**Fig 11.15: Data Owner Registration**

The above figure describes that, we have to fill all the details based on given in the register form to register as data owner.

Data Owner Login!!!!!!

Enter DataOwner Name:-	<input type="text" value="manjunath"/>
Enter Password:-	<input type="password" value="*****"/>
<a href="#">Register</a>	<input type="button" value="Login"/>

**Fig 11.16: Data Owner Login**

The above figure describes that, By using this page we can login as data owner by specifying the username and password.

## View Block Details

File Name	Owner Name	MAC-1	MAC-2
test.jsp	banu	-6a668cdaff2ca9d3a4c068341a2924d81470caea	-30b0fa2af8e82a5330944466dd9bbeef36d5c
test1.jsp	manjunath	-6df43c6fe34cb10abaffd19a28ef472743a05b32	40d97d86681650c293c058aedf619e070b11

**Fig 11.17: View Block Details**

The above describes that, We can view the block details of Mac addresses.

View Blocked Users:

User Name	File Name	Owner Name	SK	Date	Un-Block
-----------	-----------	------------	----	------	----------

**Fig 11.18:** View Blocked Users

The above describes that, We can view the blocked users.

## Chapter 12

### CONCLUSION

From a tenant point of view, the cloud security model does not yet hold against threat models developed for the traditional model where the hosts are operated and used by the same organization. However, there is a steady progress towards strengthening the IaaS security model. In this project we presented a framework for trusted infrastructure cloud deployment, with two focus points: VM deployment on trusted compute hosts and domain-based protection of stored data. We described in detail the design, implementation and security evaluation of protocols for trusted VM launch and domain-based storage protection. The solutions are based on requirements elicited by a public healthcare authority, have been implemented in a popular open-source IaaS platform and tested on a prototype deployment of a distributed EHR system. In the security analysis, we introduced a series of attacks and proved that the protocols hold in the specified threat model. Finally, our performance tests have shown that the protocols introduce a significant performance overhead.

## **REFERENCES**

- [1] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, HotCloud'09, (Berkeley, CA, USA), USENIX Association, 2009.
- [2] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel, "Seeding Clouds With Trust Anchors," in Proceedings of the 2010 ACM Workshop on Cloud Computing Security, CCSW '10, (New York, NY, USA), pp. 43–46, ACM, 2010.
- [3] N. Paladi, A. Michalas, and C. Gehrman, "Domain based storage protection with secure access control for the cloud," in Proceedings of the 2014 International Workshop on Security in Cloud Computing, ASIACCS '14, (New York, NY, USA), ACM, 2014.
- [4] M. Jordon, "Cleaning up dirty disks in the cloud," Network Security, vol. 2012, no. 10, pp. 12–15, 2012.
- [5] Cloud Security Alliance, "The notorious nine cloud computing top threats 2013," February 2013.
- [6] A. Michalas, N. Paladi, and C. Gehrman, "Security aspects of health systems migration to the cloud," in the 16th International Conference on E-health Networking, Application & Services (Healthcom'14), pp. 228–232, IEEE, Oct 2014.
- [7] B. Bertholon, S. Varrette, and P. Bouvry, "Certicloud: a novel team-based approach to ensure cloud IaaS security," in Cloud Computing, 2011 IEEE International Conference on, pp. 121–130, IEEE, 2011.
- [8] M. Aslam, C. Gehrman, L. Rasmusson, and M. Björkman, "Securely launching virtual machines on trustworthy platforms in a public cloud - an enterprise's perspective.,," in CLOSER, pp. 511–521, SciTePress, 2012.

- [9] A. Cooper and A. Martin, "Towards a secure, tamper-proof grid platform," in Cluster Computing and the Grid, 2006.CCGRID 06.Sixth IEEE International Symposium on, vol. 1, pp. 8–pp, IEEE, 2006.
- [10] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in Proceedings of the 2009 ACM workshop on Cloud computing security, pp. 55–66, ACM, 2009.
- [11] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses," IEEE Computer, vol. 45, no. 1, pp. 39–45, 2012.
- [12] S. Graf, P. Lang, S. A. Homemade, and M. Waldvogel, "Versatile key management for secure cloud storage," in Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems, pp. 469–474, IEEE Computer Society, 2012.
- [13] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, "Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services," in Presented as part of the 21st USENIX Security Symposium(USENIX Security 12), (Bellevue, WA), pp. 175–188, USENIX, 2012.
- [14] A.-R. Sadeghi and C. St' unable, "Property-based attestation for computing platforms: Caring about properties, not mechanisms," in Proceedings of the 2004 Workshop on New Security Paradigms, NSPW'04, (New York, NY, USA), pp. 67–77, ACM, 2004.
- [15] A. Sahai, "Ciphertext-policy attribute-based encryption," in In Proceedings of the IEEE Symposium on Security and Privacy, 2007.
- [16] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Financial Cryptography and Data Security, vol. 6054 of Lecture Notes in Computer Science, pp. 136–149, Springer Berlin Heidelberg, 2010.