A

PROJECT REPORT

CLOUD COMPUTING SECURITY: USING DIFFIE-HELLMAN KEY EXCHANGE

ENROLLMENT NO	101294
NAME OF STUDENT	NIKHIL VERMA
NAME OF SUPERVISOR	DR. YASHWANT SINGH

Submitted in partial fulfillment of the Degree of
Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT

Certificate

This is to certify that the work titled **Cloud Computing Security using Diffie-Hellman key Exchange** submitted by **Nikhil Verma** in partial fulfillment for the award of degree of B.Tech in Computer Science and Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature		Signature of Supervisor	
Name	Nikhil Verma	Name of Supervisor	Dr. Yashwant Singh
Roll No	101294	Designation	Asst. Professor (Sr. Grade)
		Date	15-May-2014

ACKNOWLEDGEMENT

I owe a great thanks to many people who have been helping and supporting me during this project. My heartiest thanks to Dr. Yashwant Singh, the Project Guide for guiding and correcting me at every step of our work with attention and care. He has taken pain to go through the project and make necessary correction as and when needed. Thanks and appreciation to the helpful people at college for their support. I would also thank our university and my faculty members without whom this project would have been a distant reality.

Signature of Student	
Name of Student	Nikhil Verma
Date	15- May- 2014

Table of Contents

ABSTRACT

Cloud computing is one of the most significant milestones in recent times in the history of computers. In the last few years, cloud computing has grown from being a promising business concept to one of the fastest growing segments of the IT industry. The basic concept of cloud computing is to provide a platform for sharing of resources which includes software and infrastructure with the help of virtualization. In order to provide quality of service, this environment makes every effort to be dynamic and reliable. As in most other streams of computers, security is a major obstacle for cloud computing. There are various opinions on the security of cloud computing which deal with the positives and negatives of it. This paper is an attempt to investigate the crucial security threats with respect to cloud computing. It further focuses on the available security measures which can be used for the effective implementation of cloud computing.

CHAPTER 1 :INTRODUCTION

1.1Introduction

Several trends are opening up the era of Cloud Computing, which is basically an Internet-based development and also the use of computer technology. The processors that are getting powerful and cheaper, together with the software as a service (SaaS) computing architecture, are helping us transform data centers into pools of computing services on a huge scale. The better network bandwidth and reliable yet flexible network connections make it even possible that users can now get high quality services from data and software that reside solely on data centers that are remote to us. Moving data into the cloud offers great convenience to users since they don't have to think about the difficulties of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) are both well known examples. While these internet-based online services provide large amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are dependent on their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 is such an example . From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons. Firstly, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance.

However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature.

1.2 Problem Statement

1. To study and understand the concepts of Cloud computing.
2. To study and understand how security is implemented in the cloud.
3. To implement the security in cloud using the Diffie-Hellman Key Exchange algorithm.

1.3 Motivation

As data and information is going to be shared with a third party, cloud computing users will want to avoid an untrusted cloud service provider. Protecting private and important sensitive information, such as credit card details or a patient's medical records from unwanted attackers or malicious insiders is of critical importance. And in order to ensure that the privacy of the users and their critical details do remain safe proper algorithms will be needed that can provided the users with the safety they need.

1.4 Objectives

1.3.1 DATA INTEGRITY

One of the most important issues related to cloud security risks is data integrity. The data stored in the cloud may suffer from damage during transition operations from or to the cloud storage provider. There are the risk of attacks from both inside and outside the cloud provider, such as the attacked Red Hat Linux's distribution servers. It becomes more difficult due to the fact that the servers belonging to cloud providers use the same system installations and are physically located in the same place.

1.3.2 DATA INTRUSION

Another security risk that may occur with a cloud provider, such as the Amazon cloud service, is a hacked password or data intrusion. If someone gains access to an Amazon account password, they will be able to access all of the account's instances and resources. Thus the stolen password allows the hacker to erase all the information inside any virtual machine instance for the stolen user account, modify it, or even disable its services. Furthermore, there is a possibility for the user's email(Amazon user name) to be hacked (see for a discussion of the potential risks of email), and since Amazon allows a lost password to be reset by email, the hacker may still be able to log in to the account after receiving the new reset password.

1.3.3 SERVICE AVAILABILITY

Another major concern in cloud services is service availability. Amazon mentions in its licensing agreement that it is possible that the service might be unavailable from time to time. The user's web service may terminate for any reason at any time if any user's files break the cloud storage policy. In addition, if any damage occurs to any Amazon web service and the service fails, in this case there will be no charge to the Amazon Company for this failure. Companies seeking to protect services from such failure need measures such as backups or use of multiple providers.

1.5 Organization of the Report

Chapter 1 : Introduction

Covers the various project related things such as , introduction, Problem statement , motivation and tells us about the reasons behind the choice of this project.

Chapter 2: Literature Survey

Covers the literature surveyed as well as tells about the concepts that have been studied and understood.

Chapter 3 : Project Design and Implementation

Covers the tools and technologies that are used. It also tells us about the System Design - various design diagrams. Also , it covers the implementation of the project and the Project snapshots.

Chapter 4 : Conclusion

Chapter 2 : Literature Survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration r taken into account for developing the proposed system.

2.1 TYPES OF CLOUDS

Cloud providers typically centre on one type of cloud functionality provisioning: Infrastructure, Platform or Software / Application, though there is potentially no restriction to offer multiple types at the same time, which can often be observed in PaaS (Platform as a Service) providers which offer specific applications too, such as Google App Engine in combination with Google Docs. Due this combinatorial capability, these types are also often referred to as "components". Literature and publications typically differ slightly in the terminologies applied. This is mostly due to the fact that some application areas overlap and are therefore difficult to distinguish. As an example, platforms typically have to provide access to resources indirectly, and thus are sometimes confused with infrastructures. Additionally, more popular terms have been introduced in less technologically centered publications. The following list identifies the main types of clouds (currently in use):

2.1.1 (Cloud) Infrastructure as a Service (IaaS) also referred to as Resource Clouds, provide (managed and scalable) resources as services to the user – in other words, they basically provide enhanced virtualization capabilities. Accordingly, different resources may be provided via a service interface:

Data & Storage Clouds deal with reliable access to data of potentially dynamic size, weighing resource usage with access requirements and / or quality definition. *Examples: Amazon S3, SQL Azure.*

Compute Clouds provide access to computational resources, i.e. CPUs. So far, such low-level resources cannot really be exploited on their own, so that they are typically exposed as part of a "virtualized environment" (not to be mixed with PaaS below), i.e. hypervisors. Compute Cloud Providers therefore typically offer the capability to provide computing resources (i.e. raw access to resources unlike PaaS that offer full software stacks to develop and build applications), typically virtualized, in which to execute cloudified services and applications. IaaS (Infrastructure as a Service) offers additional capabilities over a simple compute service.

Examples: Amazon EC2, Zimory, Elastichosts.

2.1.2(Cloud) Platform as a Service (PaaS), provide computational resources via a platform upon which applications and services can be developed and hosted. PaaS typically makes use of dedicated APIs to control the behavior of a server hosting engine which executes and replicates the execution according to user requests (e.g. access rate). As each provider exposes his / her own API according to the respective key capabilities, applications developed for one specific cloud provider cannot be moved to another cloud host – there are however attempts to extend generic programming models with cloud capabilities (such as MS Azure).

Examples: Force.com, Google App Engine, Windows Azure (Platform).

2.1.3 (Clouds) Software as a Service (SaaS), also sometimes referred to as Service or Application Clouds are offering implementations of specific business functions and business processes that are provided with specific cloud capabilities, i.e. they provide applications / services using a cloud infrastructure or platform, rather than providing cloud features themselves. Often, kind of standard application software functionality is offered within a cloud.

Examples: Google Docs, Salesforce CRM, SAP Business by Design.

2.2 Cloud Platforms

2.2.1 Amazon Web Services - IaaS

Since August 2006, Amazon Web Services (AWS) has provided an infrastructure web services platform in the cloud. By providing a suite of elastic IT infrastructure services, AWS enables access to compute power, storage, and other services. AWS operates on a pay-per-use basis, with no up-front expenses or long-term commitments. This makes AWS a cost-effective way to deliver applications to customers and clients. AWS operates on Amazon.com’s global computing infrastructure. The term "cloud computing" initially has been coined for a few Amazon Web Services, namely the Elastic Compute Cloud (EC2) and the Simple Storage Service (S3).

Amazon Web Services is more than a collection of infrastructure services. It incorporates identity, payment, database, messaging, and other services. All AWS services are priced on a pay as you go model, with no up front expenses or long-term commitments.

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. EC2 can be categorized as an Infrastructure-as-a-Service (IaaS) offering. It allows configuring an Amazon Machine Instance (AMI) and loading it into the Amazon EC2 service. By running multiple AMIs simultaneously, EC2 allows to quickly scale capacity, both up and down, as computing requirements change.

Amazon Simple Storage Service (Amazon S3) is a simple web services interface that can be used to store and retrieve large amounts of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. Data on S3 is unstructured (blobs). There exists no interface to run structured queries or operations on the data on S3 directly from the client. Instead, SimpleDB adds an additional layer on S3 to provide such an interface.

With Amazon CloudFront, content delivery on the Web can be implemented. Even with thin clients, cloud services need a certain portion of user interface running on the client's desktop in order to be usable. CloudFront is the proposed solution for hosting those downloadable user interfaces - in the form of a collection of JavaScript/Ajax code. Cloud Front integrates with other Amazon Web Services.

Amazon SimpleDB is a web service for running queries on structured data in real time. This service works in close conjunction with Amazon Simple Storage Service

(Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process and query data sets in the cloud. Amazon SimpleDB provides the core functionality of a database—real-time lookup and simple querying of structured data.

Amazon Simple Queue Service (Amazon SQS™) is a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available. Configuration rules for EC2 can be based on SQS attributes and dynamically start up or shutdown AMI instances as queue length vary. This allows for implementation of self-adaptive behavior in cloud services.

With EC2, S3, and SQS, Amazon has provided the first implementation for a public cloud. Public clouds are appealing for many use cases, where data security, privacy, and ownership are not of primary concern. This is true for use cases like scientific computing, simulation, image processing, video rendering, and even gaming. However, most enterprises view data as their primary asset. Private clouds - which are very much comparable to traditional hosting solutions (as discusses above), may be a promising approach to keep important data absolutely private.

Amazon Virtual Private Cloud (Amazon VPC) is an extension of the initial, purely public cloud. It provides a secure and seamless bridge between a company's existing IT infrastructure and the AWS cloud. Amazon VPC enables enterprises to connect their existing infrastructure to a set of isolated AWS compute resources via a Virtual Private Network (VPN) connection, and to extend their existing management capabilities such as security services, firewalls, and intrusion detection systems to include their AWS resources. Amazon VPC integrates today (beta status) with Amazon EC2, and will integrate with other AWS services in the future.

When developing for the Amazon cloud, one has to distinguish between APIs and tool support for deploying, managing and monitoring instances of Amazon Machine Images (AMIs) in the cloud and between APIs and tools for developing applications running on AMIs in the cloud. Being an IaaS solution, AWS tools mainly focus on managing AMIs for EC2 and setting up SimpleDB configurations for S3. Based on the Eclipse Web Tools Platform, the AWS Toolkit for Eclipse guides Java developers through common workflows and helps to configure Tomcat servers, run applications on Amazon EC2, and debug the software remotely through the Eclipse IDE. There exists also a .NET library that wraps AWS APIs and allows management of AMI instances (preferably running Windows and IIS) from .NET client programs. Similar wrapper libraries exist for Perl, PHP, Ruby and Python.

In the remainder of the section we will take a closer look on EC2, S3, and SQS.

2.2.2 Google AppEngine - PaaS

Google App Engine allows running web applications on Google's infrastructure. This effectively allows clients to build their own Software-as-a-Service applications. App Engine includes the following features:

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on a client computer
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals

Applications can run in one of two runtime environments: the Java environment, and the Python environment. Each environment provides standard protocols and common technologies for web application development. Applications can be made accessible either from a client's domain name (using Google Apps) or through Google's reserved domain called appspot.com. Applications can be shared with the world, or within an organization. App Engine supports integrating an application with Google Accounts for user authentication.

Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that applications run quickly, securely, and without interference from other apps on the system.

Applications run in a secure environment that provides limited access to the underlying operating system. These limitations allow App Engine to distribute web requests for the application across multiple servers, and start and stop servers to meet traffic demands. The sandbox isolates an application in its own secure, reliable environment that is independent of the hardware, operating system and physical location of the web server.

Examples of the limitations of the secure sandbox environment include:

An application can only access other computers on the Internet through the provided URL fetch and email services. Other computers can only connect to the application by making HTTP (or HTTPS) requests on the standard ports.

An application cannot write to the file system. An app can read files, but only files uploaded with the application code. The app must use the App Engine datastore, memcache or other services for all data that persists between requests.

Application code only runs in response to a web request, a queued task, or a scheduled task, and must return response data within 30 seconds.. A request handler cannot spawn a sub-process or execute code after the response has been sent.

App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

2.2.3 Salesforce.com - SaaS / PaaS

Salesforce.com was founded in 1999 as a company specializing in Software-as-a-Service (SaaS) offering for Customer Relationship Management (CRM). It was really one of the pioneers of the SaaS model of distributing software on the Internet. Salesforce.com promises availability of 99.9% for its cloud infrastructure. This figure is comparable with the service levels reached by the competitors. However, the trust.salesforce.com site is truly unique. It shows both recent and historical performance plus uptime data for the Salesforce.com cloud infrastructure.

Salesforce.com's CRM solution is broken down into several modules: Sales, Service & Support, Partner Relationship Management, Marketing, Content, Ideas and Analytics. In order to ease customization and allow for extension of the CRM SaaS solution, Salesforce.com has built up AppExchange. Launched in 2005, AppExchange is a directory of applications built for Salesforce by third-party developers, which users can purchase and add to their Salesforce environment. As of September 2008, there are over 800 applications available from over 450 ISVs. Applications include

services from Google, Constant Contact, Vertical Response, Good Data, and Box.net. The Force.com platform is a further generalization of the initial SaaS solution.
Salesforce users can customize their CRM application on the platform- or the tab-level. In the system, there are tabs such as "Contacts", "Reports", and "Accounts". Each tab contains associated information. For example, "Contacts" has standard fields like First Name, Last Name, and Email. Customization can be done on each tab, by adding user-defined custom fields. Customization can also be done at the "platform" level by adding customized applications to a Salesforce.com instance; that is adding sets of customized / novel tabs for specific vertical- or function-level (Finance, Human Resources, etc) features.
In addition to the web interface, Salesforce offers a Web Services API that enables integration with other systems and has wrapper libraries for programming languages such as Java, VB.NET, C#, and other .NET languages. In April 2009, Salesforce released a slimmed down version of their application for subscribers with Blackberry, iPhone, and Windows mobile devices.

2.3 Comparison of Cloud Platforms

In the previous sections we have discussed eight different approaches to cloud computing, which can be seen as representatives for a growing group of offerings. We have presented Infrastructure-as-a-Service (IaaS) offerings, namely Amazon Web Services (AWS) and VMwaresvCloud. We have discussed three Platform-as-a-Service (PaaS) solutions, namely Google AppEngine, Microsoft's Windows Azure, and Salesforce.com's force.com. Finally, we have given an overview over three Software-as-a-Service (SaaS) solutions, namely Salesforce.com, Microsoft's OfficeLive, and Google Apps.
From a software architect's and developer's standpoint most interesting are the IaaS and PaaS solutions. Those solutions allow for architecting your own applications that may be hosted on the cloud. In contrast, SaaS offerings typically only lend themselves to some limited configuration.

2.4 Common Building Blocks

All cloud computing platforms build on a similar set of building blocks. These are:

- Virtualization: In order to dynamically increase and decrease computing capacity in response to changing workloads or administrative triggers, physical computers have to be abstracted away. Virtualization of compute resources can be achieved on many different layers of abstraction (application server, operating system processes, virtual machines, logical partitions of physical hardware). For cloud computing environments, the logical machine level of abstraction has proven to work well. Virtual machines can easily be replicated when load increases. They can be moved across physical machines for consolidation purposes or to allow for system maintenance.
- WebServices: All cloud-computing platforms allow for access to cloud resources via WebService APIs. These APIs typically are hidden behind wrapper libraries in the common programming languages, such as Java, C#, C++ (and other .NET languages) for the client as well as "web languages" such as Python, Ruby, and PHP.
- Orchestration / Service Bus: All cloud-computing platforms provide some means for integrating services in the cloud. These services include compute services, data stores, and messaging middleware. Platforms differ with respect to their support for integrating external services (hosted on premise or at partner sites) with cloud services.
- Clients-side user interfaces: Cloud computing applications need to expose a client-side user interface. Thus, all cloud-computing platforms provide some means for interacting with the user. This may be as simple as a portal that integrates mashups shown in a web browser. Often, user interfaces can be based on Ajax and JavaScript. However, the most elaborate platforms rely on a fully fledged component model such as JavaBeans/Applets or Silverlight/.NET for programming user interfaces that are downloadable and can be executed on the client's computer.

2.5 Which Cloud to choose ?

Software-as-a-Service (SaaS) solutions have been the initiators of the cloud computing excitement. There are far more SaaS solutions available than the ones we have discussed here. From an end user's perspective, there are a few questions that need to be answered when it comes to using SaaS systems. The most important ones are: "Who owns the data?" and "Can I work offline?" Systems like Salesforce.com CRM and Google Apps ask the user to fully trust the provider for keeping data secure, consistent, and available. Both services are of no use when the computer is not connected to the Internet. On the other hand, they provide unprecedented ease-of-use without any need to install and maintain software on the client's computer.

Microsoft's OfficeLive follows a different route. While still allowing (read) access to the data from anywhere on the Internet, it will download the data to a client's computer in order to implement modify or write operations. This puts a small burden on the user - as he needs to maintain some software on his computer. However, it provides the big advantage of having the data accessible even when the computer is not connected. Also, it is far easier to keep backups of data if local copies are maintained anyway. Another aspect is the richness of the interface: Software on a local computer today still provides for a better user experience than Web 2.0 interfaces.

Amazon's Elastic Compute Cloud (EC2) was the first cloud solution on the market that was appealing not only to the administrator and end-user but also to developers. Infrastructure-as-a-Service (IaaS) historically presented the second step in cloud computing. Amazon Web Services (AWS) define the de facto standard in IaaS. They have defined all the building blocks, like the concept of an Amazon Machine Image and the use of virtual machines, the Simple Storage Service, and the message queuing middleware. AWS also has a long-standing record in providing high levels of service availability. The AMI concept is very flexible - developers can basically pre-configure their virtual machine with all necessary tools, such as application servers, databases, and middleware solutions prior to deploying them on the cloud (via S3). The biggest drawback of the concept is the burden of maintenance - which is still on the developer, as they have to keep the software stack on their AMI in an updated and well-maintained fashion. Amazon is certainly working on this issue and tries to ease this burden by providing certain pre-configured machine images. One may argue that the AWS cloud will slowly move from an IaaS offering to a PaaS offering.

VMware's vCloud, the other IaaS offering discussed here is the most recent addition to the cloud-computing spectrum. In contrast to all other cloud platforms discussed here, VMware envisions a whole cloud-computing ecosystem where independent software vendors and hosters may provide competitive offers for running virtual data centers in the cloud. Having multiple providers may circumvent the risk of vendor lock in and also mitigates the business risk of being dependent on just a single cloud operator. VMware tries to standardize on the interfaces necessary to configure, load, instantiate, move, and replicate virtual machine instances in the cloud. It is suggesting using the Open Virtualization Format (OVF) as the standard for cloud computing instances. VMware has a long-standing record on virtualization. Playing just on the OVF level will allow VMware to build upon its strengths and implement monitoring and system administration interfaces for cloud instances on a generic level. Improved monitoring and self-adaptive management capabilities would be a most desirable feature for all the cloud platforms currently available.

Platform-as-a-Service (PaaS) solutions are - from an architect's and developer's standpoint - the wave of the future. The integration of elaborate development tools on the client side with a standardized and well-maintained execution environments in the cloud provides for ease-of-use and high efficiency. PaaS platforms differ in the set of supported programming languages and the set of additional infrastructure services available with the execution environment in the cloud.

Salesforce.com's Force.com PaaS offering is tightly integrated with a database in the cloud. It is best suited for a specialized set of cloud applications, where customer data is in the center of operation. However, developers have to use the proprietary Apex language in order to code for the cloud. Force.com applications therefore are hardly portable. There is little support for running cloud applications on-premise.

Microsoft's Windows Azure and Google's AppEngine are quite comparable in their basic approach to cloud computing. Both are PaaS offerings that promise ease-of-use and simple maintenance of cloud applications to the architect and developer. Both platforms support integration between powerful client-side integrated development environments - namely Visual Studio for Windows Azure and Eclipse for AppEngine - and the cloud. Both providers have a long-standing trust record for providing services on the Internet.

Microsoft supports the usage of all .NET languages (C#, VB.NET, C++, and many more) with Windows Azure. Google provides a tight integration of Java (and languages supported by the JVM) with AppEngine. Both platforms allow for installation of additional software and there are interoperability stories explaining how to run code written in other languages on Windows Azure and AppEngine.

The biggest difference between AppEngine and Windows Azure is the integration of SQL Azure in Microsoft's cloud computing platform. The availability of a database in the cloud that supports structured queries and complex operations on data may be important for a wide range of applications. Another specialty of Windows Azure is the Service Bus integrated in AppFabric. Not only does the service bus support orchestration of services in the cloud, it also allows reaching out from the cloud onto services running locally in a client's or partner's datacenter. Obviously there is a tradeoff: accessing off-cloud services allows for easier system integration on one hand side, it makes service availability directly dependent on those external services on the other hand side.

2.6 Essential Characteristics of Cloud Computing

Cloud services exhibit five essential characteristics that demonstrate their relation to, and differences from, traditional computing approaches:

- **On-demand self-service**. A consumer can unilaterally provision computing capabilities such as server time and network storage as needed automatically, without requiring human interaction with a service provider.
- **Broad network access.** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs) as well as other traditional or cloud- based software services.
- **Resource pooling**. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a degree of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. Even private clouds tend to pool resources between different parts of the same organization.
- **Rapid elasticity**. Capabilities can be rapidly and elastically provisioned — in some cases automatically — to quickly scale out; and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

- **Measured service.** Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, or active user accounts). Resource usage can be monitored, controlled, and reported — providing transparency for both the provider and consumer of the service.

It is important to recognize that cloud services are often but not always utilized in conjunction with, and enabled by, virtualization technologies. There is no requirement, however, that ties the abstraction of resources to virtualization technologies and in many offerings virtualization by hypervisor or operating system container is not utilized.

2.7 DEPLOYMENT TYPES (CLOUD USAGE)

Similar to P/I/SaaS, clouds may be hosted and employed in different fashions, depending on the use case, respectively the business model of the provider. So far, there has been a tendency of clouds to evolve from private, internal solutions (private clouds) to manage the local infrastructure and the amount of requests e.g. to ensure availability of highly requested data. This is due to the fact that data centre's initiating cloud capabilities made use of these features for internal purposes before considering selling the capabilities publicly (public clouds). Only now that the providers have gained confidence in publication and exposition of cloud features do the first hybrid solutions emerge. This movement from private via public to combined solutions is often considered a "natural" evolution of such systems, though there is no reason for providers to not start up with hybrid solutions, once the necessary technologies have reached a mature enough position.

We can hence distinguish between the following deployment types:

Private Clouds are typically owned by the respective enterprise and / or leased. Functionalities are not directly exposed to the customer, though in some cases services with cloud enhanced features may be offered – this is similar to (Cloud) Software as a Service from the customer point of view.

Example: eBay.

Public Clouds. Enterprises may use cloud functionality from others, respectively offer their own services to users outside of the company. Providing the user with the actual capability to exploit the cloud features for his / her own purposes also allows other enterprises to outsource their services to such cloud providers, thus reducing costs and effort to build up their own infrastructure. As noted in the context of cloud types, the scope of functionalities thereby may differ.

Example: Amazon, Google Apps, Windows Azure.

Hybrid Clouds. Though public clouds allow enterprises to outsource parts of their infrastructure to cloud providers, they at the same time would lose control over the resources and the distribution / management of code and data. In some cases, this is not desired by the respective enterprise. Hybrid clouds consist of a mixed employment of private and public cloud infrastructures achieve a maximum of cost reduction through outsourcing whilst maintaining the desired degree of control over e.g. sensitive data by employing local private clouds.

There are not many hybrid clouds actually in use today, though initial initiatives such as the one by IBM and Juniper already introduce base technologies for their realization.

Community Clouds. Typically cloud systems are restricted to the local infrastructure, i.e. providers of public clouds offer their own infrastructure to customers. Though the provider could actually resell the infrastructure of another provider, clouds do not aggregate infrastructures to build up larger, cross-boundary structures. In particular smaller SMEs could profit from community clouds to which different entities contribute with their respective (smaller) infrastructure. Community clouds can either aggregate public clouds or dedicated resource infrastructures.

We may thereby distinguish between private and public community clouds. For example smaller organizations may come together only to pool their resources for building a private community cloud. As opposed to this, resellers such as Zimory may pool cloud resources from different providers and resell them.

Community Clouds as such are still just a vision, though there are already indicators for such development, e.g. through Zimory and RightScale . Community clouds show some overlap with GRIDs technology.

Special Purpose Clouds. In particular IaaS clouds originating from data centres have a "general purpose" appeal to them, as their according capabilities can be equally used for a wide scope of use cases and customer types. As opposed to this, PaaS clouds tend to provide functionalities more specialized to specific use cases, which should not be confused with “proprietariness” of the platform: specialization implies providing additional, use case specific methods, whilst proprietary data implies that structure of data and interface are specific to the provider.

Specialized functionalities are provided e.g. by the Google App Engine which provides specific capabilities dedicated to distributed document management. Similar to general service provisioning (web based or not), it can be expected that future systems will provide even more specialized capabilities to attract individual user areas, due to competition, customer demand and available expertise.

Special Purpose Clouds are just extensions of “normal” cloud systems to provide additional, dedicated capabilities. The basis of such development is already visible.

2.8 Benefits

The following are some of the possible benefits for those who offer cloud computing-based services and applications:

- **Cost Savings** — Companies can reduce their capital expenditures and use operational expenditures for increasing their computing capabilities. This is a lower barrier to entry and also requires fewer in-house IT resources to provide system support.
- **Scalability/Flexibility** — Companies can start with a small deployment and grow to a large deployment fairly rapidly, and then scale back if necessary. Also, the flexibility of cloud computing allows companies to use extra resources at peak times, enabling them to satisfy consumer demands.
- **Reliability** — Services using multiple redundant sites can support business continuity and disaster recovery.
- **Maintenance** — Cloud service providers do the system maintenance, and access is through APIs that do not require application installations onto PCs, thus further reducing maintenance requirements.
- **Mobile Accessible** — Mobile workers have increased productivity due to systems accessible in an infrastructure available from anywhere.

2.9 Challenges

The following are some of the notable challenges associated with cloud computing, and although some of these may cause a slowdown when delivering more services in the cloud, most also can provide opportunities, if resolved with due care and attention in the planning stages.

• **Security and Privacy** — Perhaps two of the more "hot button" issues surrounding cloud computing relate to storing and securing data, and monitoring the use of the cloud by the service providers. These issues are generally attributed to slowing the deployment of cloud services. These challenges can be addressed, for example, by storing the information internal to the organization, but allowing it to be used in the cloud. For this to occur, though, the security mechanisms between organization and the cloud need to be robust and a Hybrid cloud could support such a deployment.

• **Lack of Standards** — Clouds have documented interfaces; however, no standards are associated with these, and thus it is unlikely that most clouds will be interoperable. The Open Grid Forum is developing an Open Cloud Computing Interface to resolve this issue and the Open Cloud Consortium is working on cloud computing standards and practices. The findings of these groups will need to mature, but it is not known whether they will address the needs of the people deploying the services and the specific interfaces these services need. However, keeping up to date on the latest standards as they evolve will allow them to be leveraged, if applicable.

• **Continuously Evolving** — User requirements are continuously evolving, as are the requirements for interfaces, networking, and storage. This means that a "cloud," especially a public one, does not remain static and is also continuously evolving.

• **Compliance Concerns** — The Sarbanes-Oxley Act (SOX) in the US and Data Protection directives in the EU are just two among many compliance issues affecting cloud computing, based on the type of data and application for which the cloud is being used. The EU has a legislative backing for data protection across all member states, but in the US data protection is different and can vary from state to state. As with security and privacy mentioned previously, these typically result in Hybrid cloud deployment with one cloud storing the data internal to the organization.

.

2.10 CLOUD SECURITY

Cloud Security is:

- The response to a familiar set of security challenges that manifest differently in the cloud. New technologies and fuzzier boundaries surrounding the data center require a different approach.
- A set of policies, technologies, and controls designed to protect data, infrastructure, and clients from attack and enable regulatory compliance.
- Layered technologies that create a durable security net or grid. Security is more effective when layered at each level of the stack and integrated into a common management framework.
- About providing protection whatever delivery model you deploy or use: private, public, or hybrid cloud environments.
- The joint responsibility of your organization and your cloud service provider(s). Depending on the cloud delivery model and services you deploy, security is the responsibility of both parties.

2.10.1 Why is Cloud Security needed?

To manage cloud security in today's world, you need a solution that helps you address threats to enterprise data and infrastructure, including the major trends you are up against.

- Changing attackers and threats: Threats are no longer the purview of isolated hackers looking for personal fame. More and more, organized crime is driving well-resourced, sophisticated, targeted attacks for financial gain.
- Evolving architecture technologies**:** With the growth of virtualization, perimeters and their controls within the data center are in flux, and data is no longer easily constrained or physically isolated and protected.
- Consumerization of IT: As mobile devices and technologies continue to proliferate, employees want to use personally owned devices to access enterprise applications, data, and cloud services.
- Dynamic and challenging regulatory environment**:** Organizations—and their IT departments—face ongoing burdens of legal and regulatory compliance with increasingly prescriptive demands and high penalties for noncompliance or breaches. Examples of regulations include Sarbanes-Oxley(SOX), Payment

Card Industry Data Security Standard (PCI DSS), and the Health Insurance Portability and Accountability Act (HIPAA).

2.11 Security Challenges:

The Cloud Security Alliance, an industry group promoting cloud computing security best practices and standards, has identified seven areas of security risk.
Five of these are:

Abuse and nefarious use of cloud services. Many infrastructure-as a-service (IaaS)providers make it easy to take advantage of their services. With a valid credit card, users can register and start using loud services right away. Cybercriminals actively target cloud services providers, partially because of this relatively weak registration system that helps obscure identities, and because many providers have limited fraud-detection capabilities. Stringent initial registration and validation processes, credit card fraud monitoring, and subsequent authentication are ways to remediate this type of threat.

Multitenancy and shared technology issues. Clouds deliver scalable services that provide computing power for multiple tenants, whether those tenants are business groups from the same company or independent organizations. That means shared infrastructure—CPU caches, graphics processing units (GPUs), disk partitions, memory, and other components—that was never designed for strong compartmentalization. Even with a virtualization hypervisor to mediate access between guest operating systems and physical resources, there is concern that attackers can gain unauthorized access and control of your underlying platform with software-only isolation mechanisms. Potential compromise of the hypervisor layer can in turn lead to a potential compromise of all the shared physical resources of the server that it controls, including memory and data as well as other virtual machines (VMs) on that server.

Data loss or leakage. Protecting data can be a headache because of the number of ways it can be compromised. Some data—customer, employee, or financial data, for example—should be protected from unauthorized users. But data can also be maliciously deleted, altered, or unlinked from its larger context. Loss of data can damage your company's brand and reputation, affect customer and employee trust, and have regulatory compliance or competitive consequences.

Account or service hijacking. Attacks using methods such as phishing and fraud continue to be an ongoing threat. With stolen credentials, hackers can access critical areas of your cloud and potentially eavesdrop on transactions, manipulate or falsify data, and redirect your clients to illegitimate sites. IT organizations can fight back with strong identity and access management, including two-factor authentication where possible, strong password requirements, and proactive monitoring for unauthorized activity.

Unknown risk. Releasing control of your data to a cloud service provider has important security ramifications. Without clearly understanding the service provider's security practices, your company may be open to hidden vulnerabilities and risks. Also, the complexity of cloud environments may make it tempting for IT managers to cobble together security measures. Unfortunately, that same complexity and the relatively new concept of cloud computing and related technologies make it difficult to consider the full ramifications of any change, and you may be leaving your cloud open to new or still undiscovered vulnerabilities.

2.12 Security Risks in Cloud Computing

2.12.1 Data Integrity

One of the most important issues related to cloud security risks is data integrity. The data stored in the cloud may suffer from damage during transition operations from or to the cloud storage provider.

One example of breached data occurred in 2009 in Google Docs, which triggered the Electronic Privacy Information Centre for the Federal Trade Commission to open an investigation into

Google's Cloud Computing Services . Another example of a risk to data integrity recently occurred in

Amazon S3 where users suffered from data corruption .We can argue that when multiple clients use cloud storage or when multiple devices are synchronized by one user, it is difficult to address the data corruption issue. One of the solutions proposed is to use a Byzantine fault-tolerant replication protocol within the cloud.

Some claim this solution can avoid data corruption caused by some components in the cloud.

2.12.2 Data Intrusion

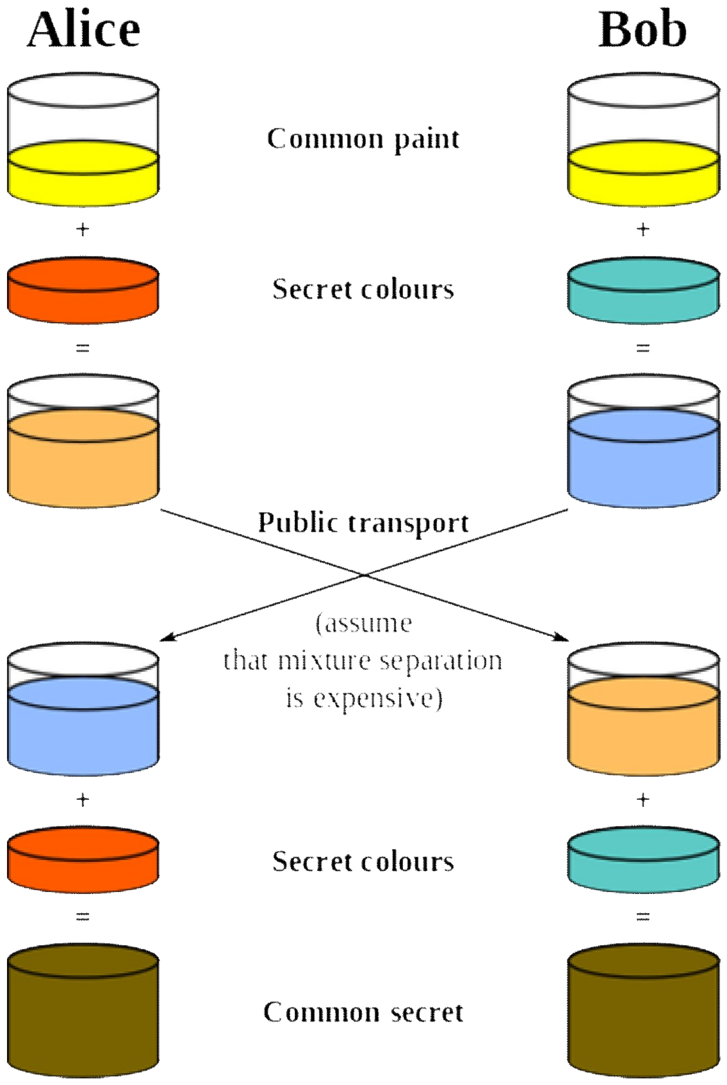
Another security risk that may occur with a cloud provider, such as the Amazon cloud service, is a hacked password or data intrusion. If someone gains access to an Amazon account password, they will be able to access all of the account's instances and resources. Thus the stolen password allows the hacker to erase all the information inside any virtual machine instance for the stolen user account, modify it, or even disable its services. Furthermore, there is a possibility for the user's email (Amazon user name) to be hacked, and since Amazon allows a lost password to be reset by email, the hacker may still be able to log in to the account after receiving the new reset password.

2.12.3 Service Availability

Another major concern in cloud services is service availability. Amazon mentions in its licensing agreement that it is possible that the service might be unavailable from time to time. The user's web service may terminate for any reason at any time if any user's files break the cloud storage policy. In addition, if any damage occurs to any Amazon web service and the service fails, in this case there will be no charge to the Amazon Company for this failure. Companies seeking to protect services from such failure need measures such as backups or use of multiple providers. Both Google Mail and Hotmail experienced service downtime recently. If a delay affects payments from users for cloud storage, the users may not be able to access their data. Due to a system administrator error,
45% of stored client data was lost in LinkUp (MediaMax) as a cloud storage provider.

2.13 Diffie-Hellman Key Exchange Algorithm

Diffie–Hellman key exchange (D–H) is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.In 1976, Whitfield Diffie and Martin Hellman introduced a key exchange protocol using the discrete logarithm problem. In this protocol A and B will set up a random secret key for their private key system, using a public but authenticated channel. They decide on a cyclic group G of order n and a generator g of the group in public. To set up a key A chooses a random integer a 2 [1;n] and sends B g^a, similarly B computes g^b for random b 2 [1;n] and sends it to A. The secret key is g^{ab}, which A computes by computing g^{ba} and B by computing $(g^a)^b$. Notice that an adversary C notices somethings which is now public information. If C can somehow compute or have some non-negligible information about gab from the public information then the scheme is broken.



2.13.1 Pictorial representation of the Algorithm

Cryptographic Explanation

1. A and B agree to use a prime number p = 23 and base g = 5.
2. A chooses a secret integer a = 6, then sends to B , A = g^a mod p
   A = 56 mod 23 = 8
3. B chooses a secret integer b = 15, then sends to A, B = g^b mod p
   B = 515 mod 23 = 19
4. A computes s = B^a mod p
   s = 196 mod 23 = 2
5. B computes s = A^b mod p
   s = 815 mod 23 = 2
6. A and B now share a secret (the number 2).
   Both A and B have arrived at the same value, because $(g^a)^b$ and $(g^b)^a$ are equal mod p. Note that only a, b, and (gab mod p = g^{ba} mod p) are kept secret. All the other values – p, g, g^a mod p, and g^b mod p – are sent in the clear. Once A and B compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

   Of course, much larger values of a, b, and p would be needed to make this example secure, since there are only 23 possible results of n mod 23. However, if p is a prime of at least 300 digits, and a and b are at least 100 digits long, then even the fastest modern computers cannot find a given only g, p, g^b mod p and g^a mod p.

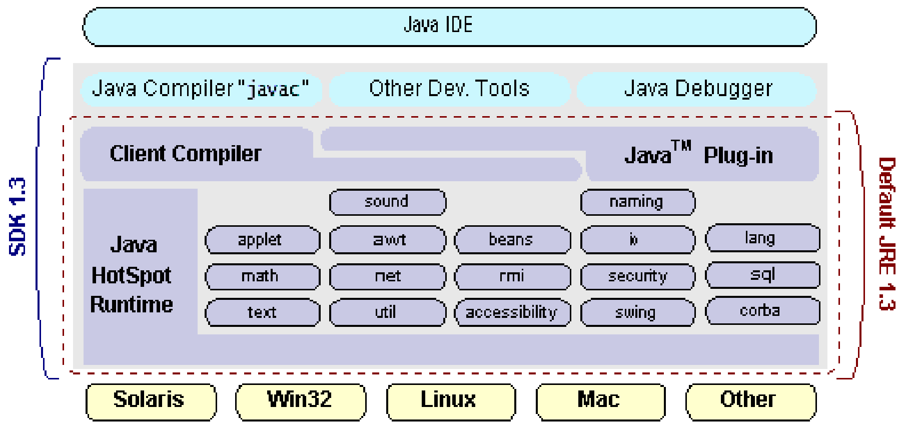
Chapter 3: Project Design and Implementation

3.1 Tools and Technologies Used

1. **Java**

   Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.[10][11] Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

   The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).



Java platform

Main articles: Java (software platform) and Java virtual machine

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time (JIT) compilers were introduced from an early stage that compilebytecodes to machine code during runtime

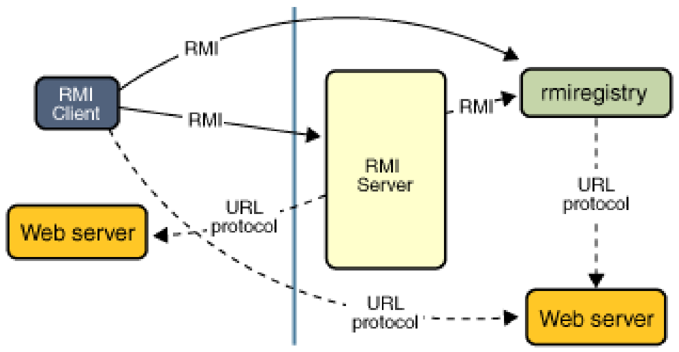
2. RMI

RMI applications often comprise two separate programs, a server and a client. A typical server program creates some remote objects, makes references to these objects accessible, and waits for clients to invoke methods on these objects. A typical client program obtains a remote reference to one or more remote objects on a server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a distributed object application.

Distributed object applications need to do the following:

- Locate remote objects. Applications can use various mechanisms to obtain references to remote objects. For example, an application can register its remote objects with RMI's simple naming facility, the RMI registry. Alternatively, an application can pass and return remote object references as part of other remote invocations.
- Communicate with remote objects. Details of communication between remote objects are handled by RMI. To the programmer, remote communication looks similar to regular Java method invocations.
- Load class definitions for objects that are passed around. Because RMI enables objects to be passed back and forth, it provides mechanisms for loading an object's class definitions as well as for transmitting an object's data.

The following illustration depicts an RMI distributed application that uses the RMI registry to obtain a reference to a remote object. The server calls the registry to associate (or bind) a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method on it. The illustration also shows that the RMI system uses an existing web server to load class definitions, from server to client and from client to server, for objects when needed.



Advantages of RMI

One of the central and unique features of RMI is its ability to download the definition of an object's class if the class is not defined in the receiver's Java virtual machine. All of the types and behavior of an object, previously available only in a single Java virtual machine, can be transmitted to another, possibly remote, Java virtual machine. RMI passes objects by their actual classes, so the behavior of the objects is not changed when they are sent to another Java virtual machine. This capability enables new types and behaviors to be introduced into a remote Java virtual machine, thus dynamically extending the behavior of an application. The compute engine example in this trail uses this capability to introduce new behavior to a distributed program.

Like any other Java application, a distributed application built by using Java RMI is made up of interfaces and classes. The interfaces declare methods. The classes implement the methods declared in the interfaces and, perhaps, declare additional methods as well. In a distributed application, some implementations might reside in some Java virtual machines but not others. Objects with methods that can be invoked across Java virtual machines are called remote objects.

An object becomes remote by implementing a remote interface, which has the following characteristics:

- A remote interface extends the interface java.rmi.Remote.
- Each method of the interface declares java.rmi.RemoteException in its throws clause, in addition to any application-specific exceptions.

RMI treats a remote object differently from a non-remote object when the object is passed from one Java virtual machine to another Java virtual machine. Rather than making a copy of the implementation object in the receiving Java virtual machine, RMI passes a remote stub for a remote object. The stub acts as the local representative, or proxy, for the remote object and basically is, to the client, the remote reference. The client invokes a method on the local stub, which is responsible for carrying out the method invocation on the remote object.A stub for a remote object implements the same set of remote interfaces that the remote object implements. This property enables a stub to be cast to any of the interfaces that the remote object implements. However, only those methods defined in a remote interface are available to be called from the receiving Java virtual machine.

3.2 SYSTEM DESIGN

Data Flow Diagram / Use Case Diagram / Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

3.2.1 System Architecture:

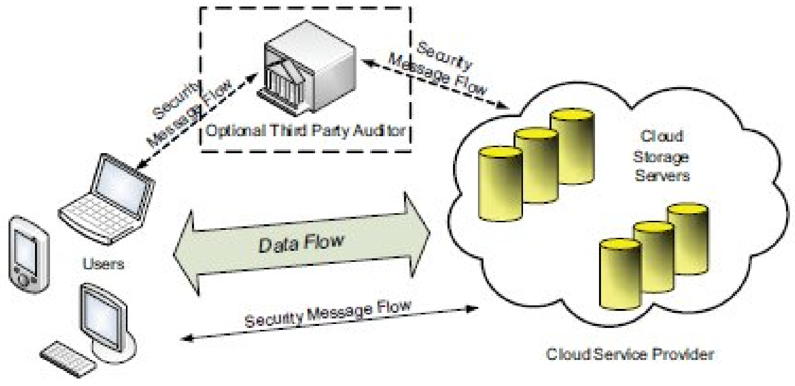


Fig . 3.1: System Architecture

3.2.2 Data Flow Diagram:

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

Cloud Server Data Security Architecture

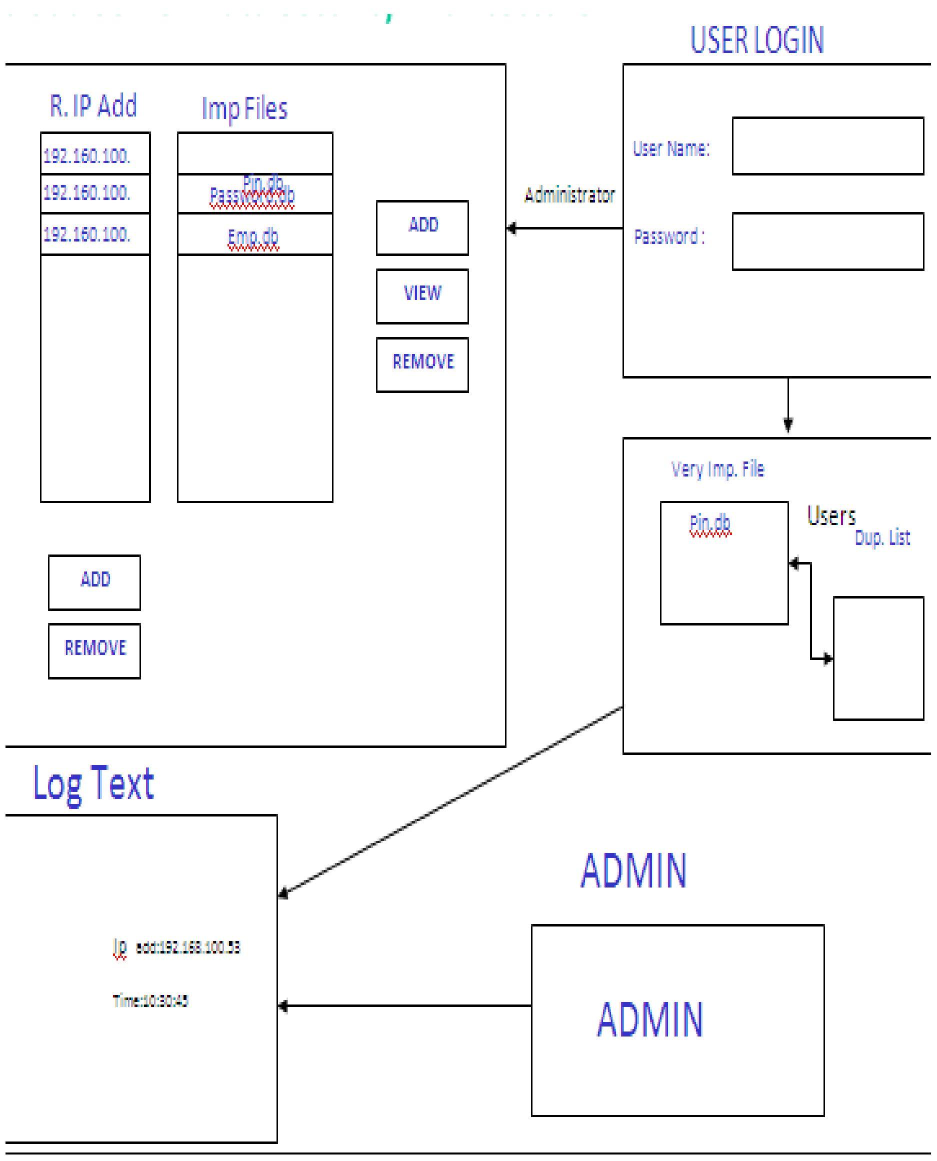


Fig 3.2 Cloud Server Data Security Architecture

3.2.3 DFD:

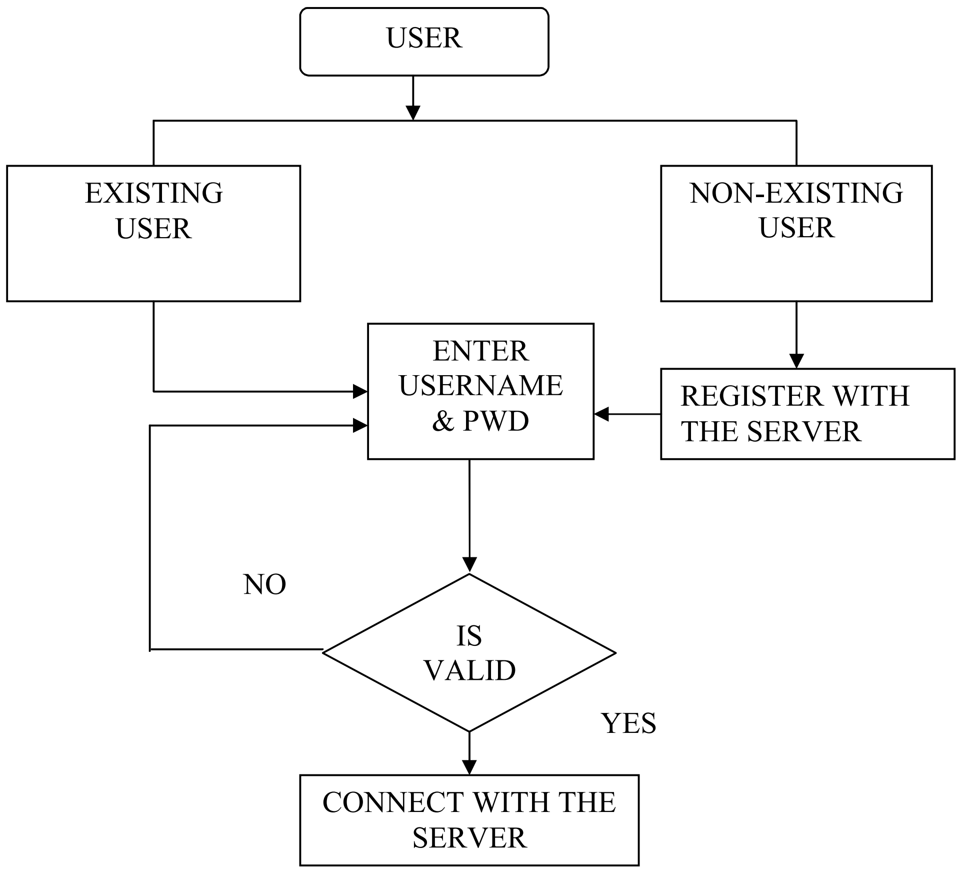


Fig 3.3 Data Flow Diagram

3.2.4 Activity Diagram:

An activity diagram is characterized by states that denote various operations. Transition from one state to the other is triggered by completion of the operation. The purpose of an activity is symbolized by round box, comprising the name of the operation. An operation symbol indicates the execution of that operation. This activity diagram depicts the internal state of an object.

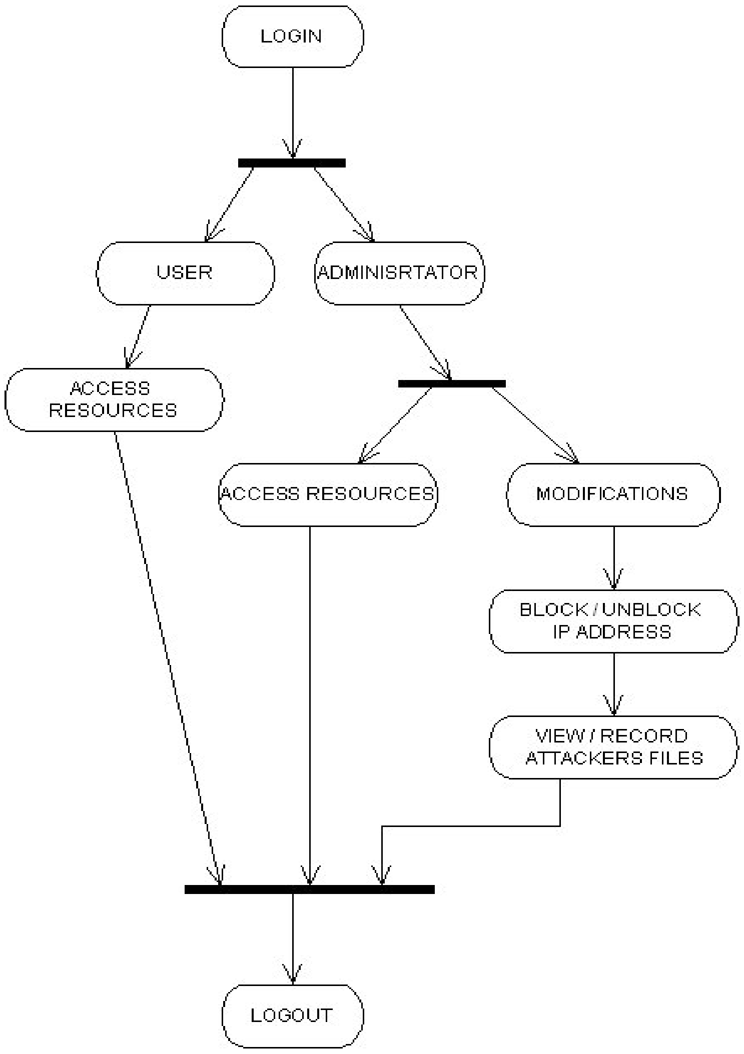


Fig . 3.4 Activity Diagram

3.2.5 UML Sequence Diagram:

The sequence diagrams are an easy and intuitive way of describing the system's behavior, which focuses on the interaction between the system and the environment. This notational diagram shows the interaction arranged in a time sequence. The sequence diagram has two dimensions: the vertical dimension represents the time, the horizontal dimension represents different objects. The vertical line also called the object's *lifeline* represents the object's existence

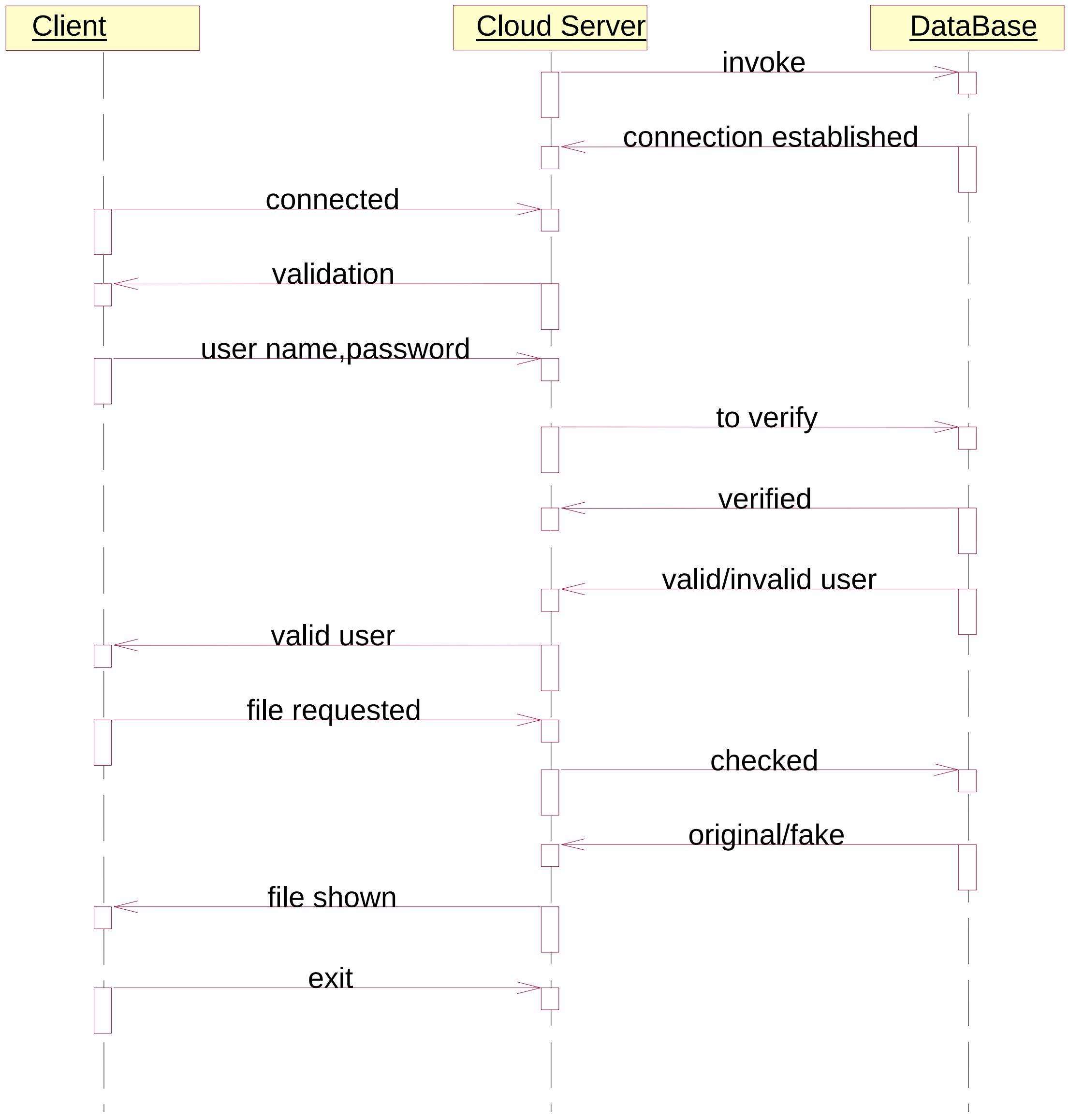


Fig 3.5 Sequence Diagram

3.2.6 CLASS DIAGRAM

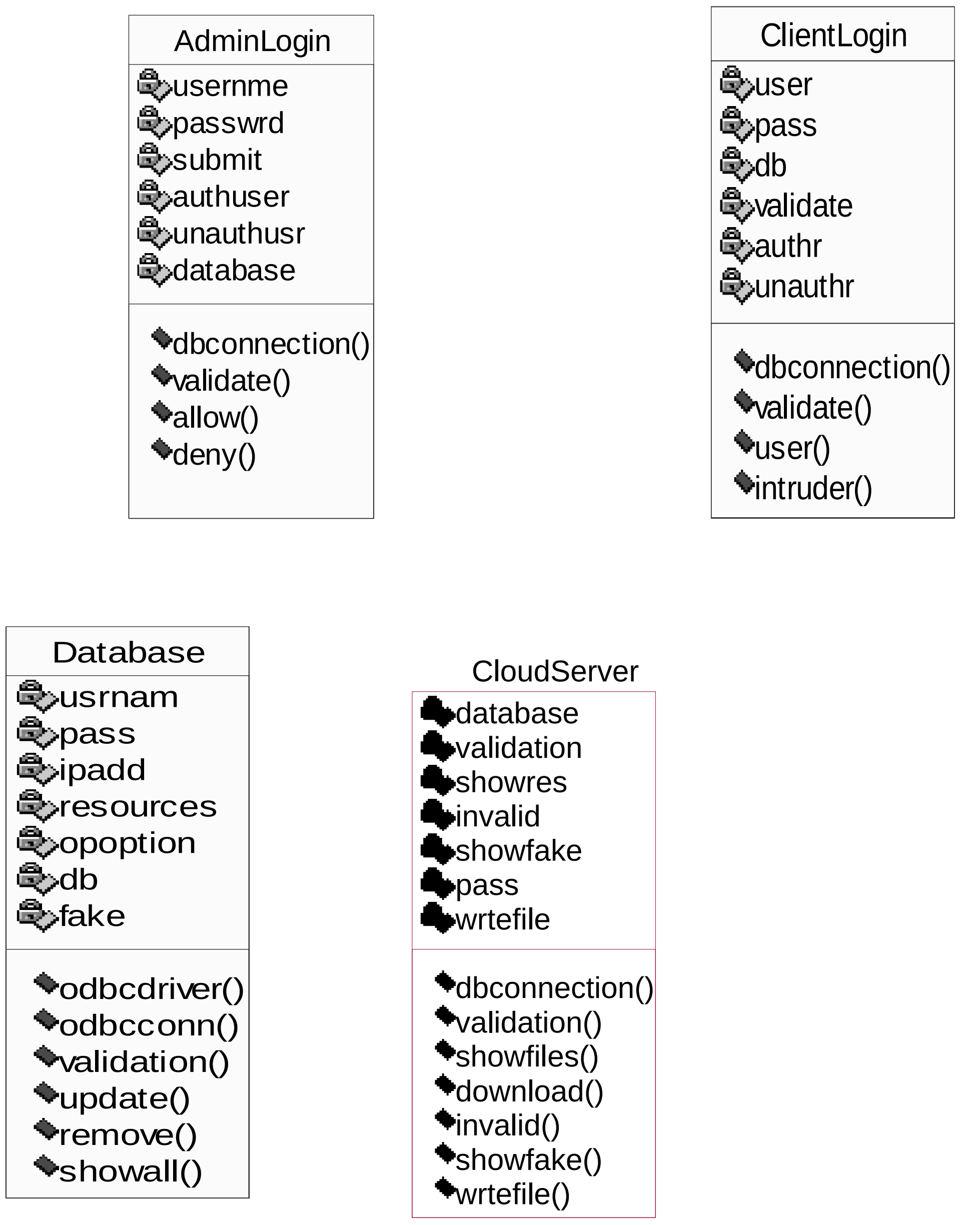


Fig .3.6 Class Diagram

3.2.7 USE CASE DIAGRAM

A use-case diagram is a graph of actors, a set of use cases enclosed by a system boundary, participation associations between the actors and the use-cases, and generalization among the use cases. In general, the *use-case* defines the outside (actors) and inside(use-case) of the system's typical behavior. A use-case is shown as an ellipse containing the name of the use-case and is initiated by actors.An *Actor* is anything that interacts with a use-case. This is symbolized by a stick figure with the name of the actor below the figure.

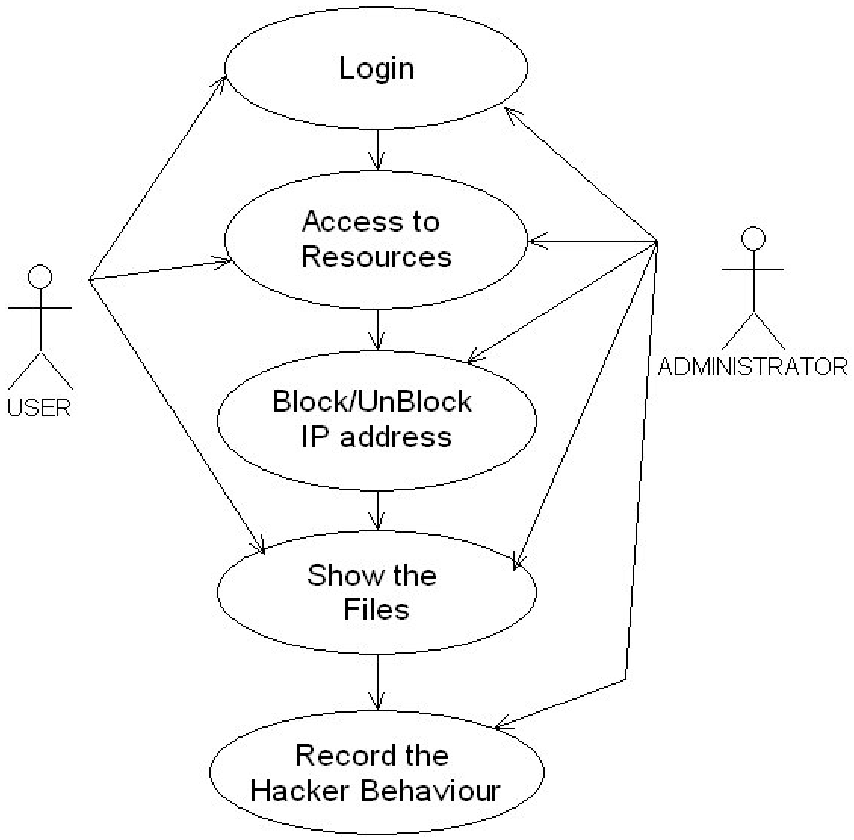


Fig 3.7 Use case Diagram

3.3 IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Main Modules:-

1. Client Module:

In this module, the client sends the query to the server. Based on the query the server sends the corresponding file to the client. Before this process, the client authorization step is involved.In the server side, it checks the client name and its password for security process. If it is satisfied and then received the queries form the client and search the corresponding files in the database. Finally, find that file and send to the client. If the server finds the intruder means, it set the alternative Path to those intruder.

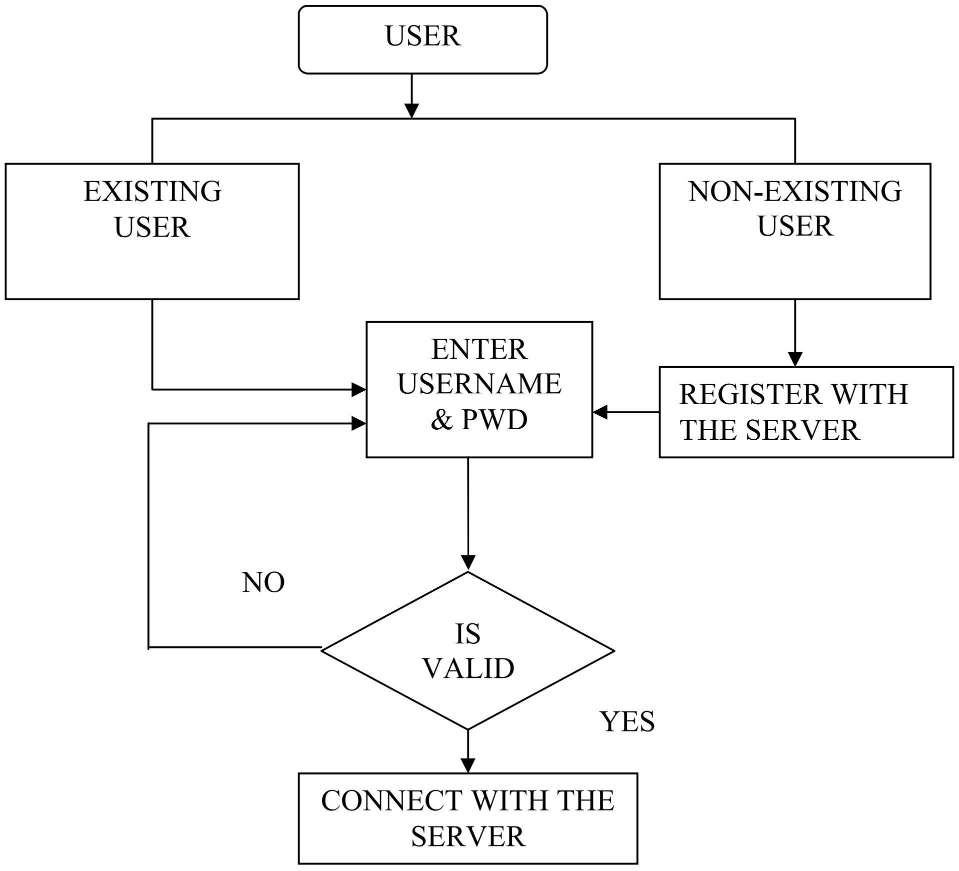


Fig 3.8 Client Module

2. System Module:

Representative network architecture for cloud data storage is illustrated in Figure 1. Three different network entities can be identified as follows:

• User:

Users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.

• Cloud Service Provider (CSP):

A CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems,.

• Third Party Auditor (TPA):

An optional TPA, who has expertise and capabilities that users may not have, is Trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

3. Cloud data storage Module:

Cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data.. users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. In case that users do not necessarily have the
time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. In our model, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead.

4. Cloud Authentication Server:

The Authentication Server (AS) functions as any AS would with a few additional behaviors added to the typical client-authentication protocol. The first addition is the sending of the client authentication information to the masquerading router. The AS in this model also functions as a ticketing authority, controlling permissions on the application network. The other optional function that should be supported by the AS is the updating of client lists, causing a reduction in authentication time or even the removal of the client as a valid client depending upon the request

5. Unauthorized data modification and corruption module:

One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures. Besides, in the distributed case when such inconsistencies are successfully detected, to find which server the data error lies in is also of great significance

6. Adversary Module:

Security threats faced by cloud data storage can come from two different sources. On the one hand, a CSP can be self-interested, untrusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors, Byzantine failures and so on.

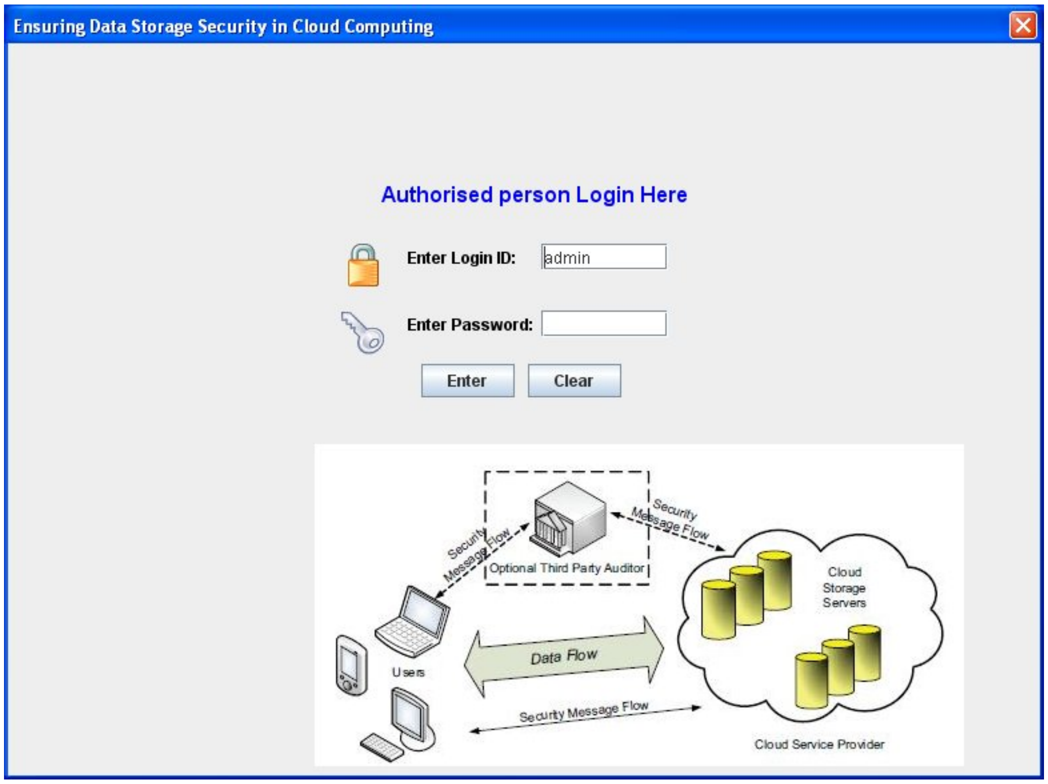
On the other hand, there may also exist an economicallymotivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete users' data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversary with different levels of capability in this paper:

Weak Adversary: The adversary is interested in corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the user.

Strong Adversary: This is the worst case scenario, in which we assume that the adversary can compromise all the storage servers so that he can intentionally modify the data files as long as they are internally consistent. In fact, this is equivalent to the case where all servers are colluding together to hide a data loss or corruption incident.

3.4 Project Snapshots

3.4.1 Cloud Server Login:



Ensuring Data Storage Security in Cloud Computing

Authorised person Login Here

Enter Login ID: admin

Enter Password:

Fig 3.9 Cloud Server Login Page

3.4.2 Client Side Login:

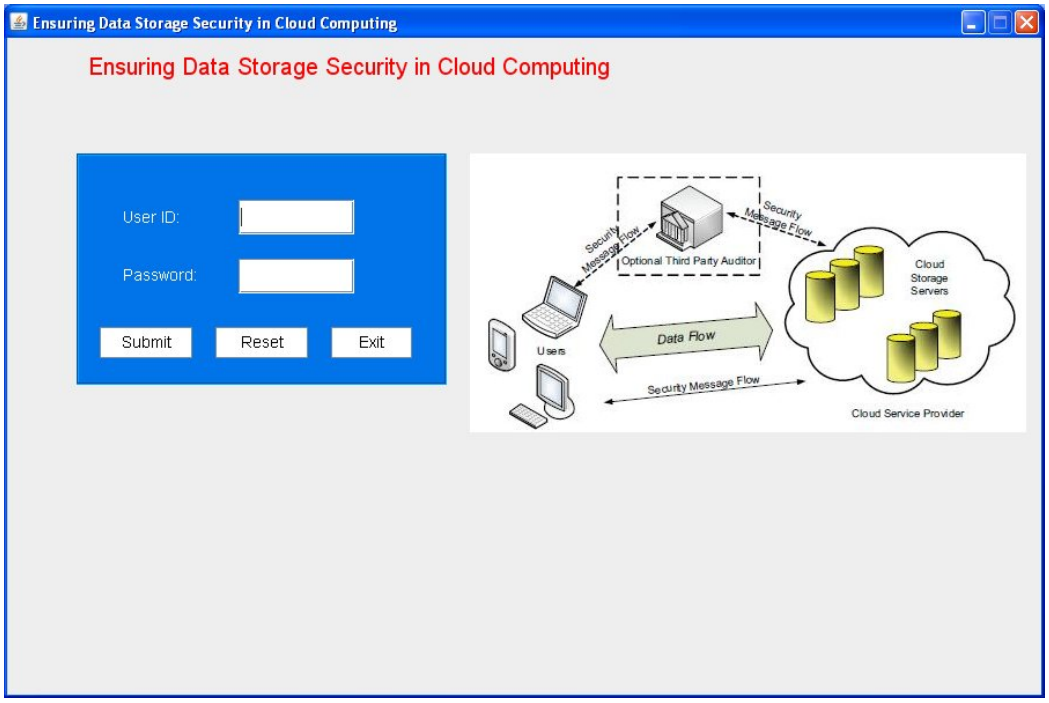
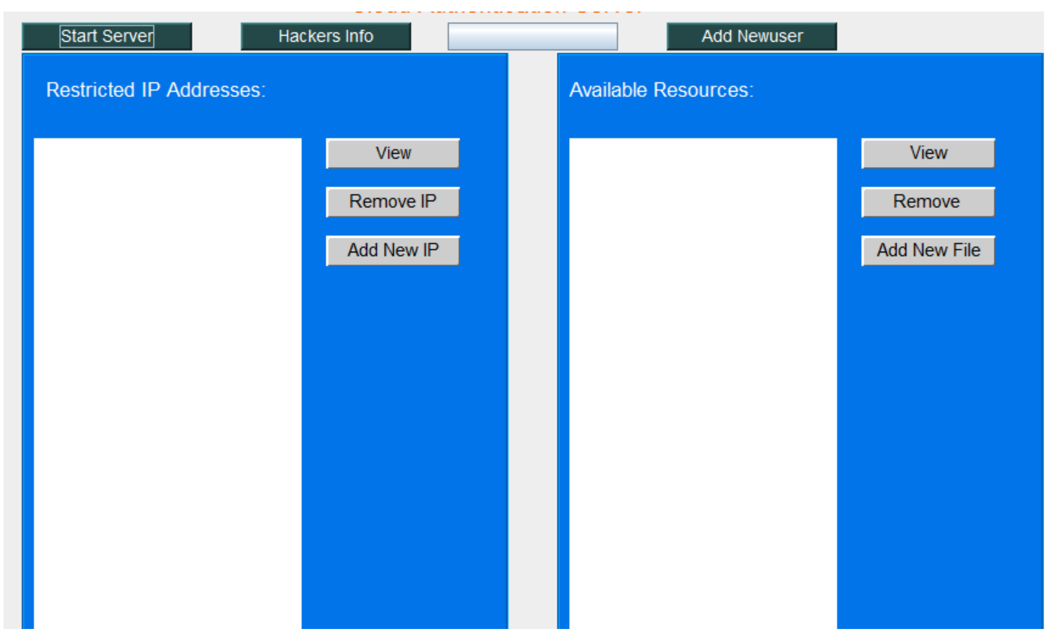
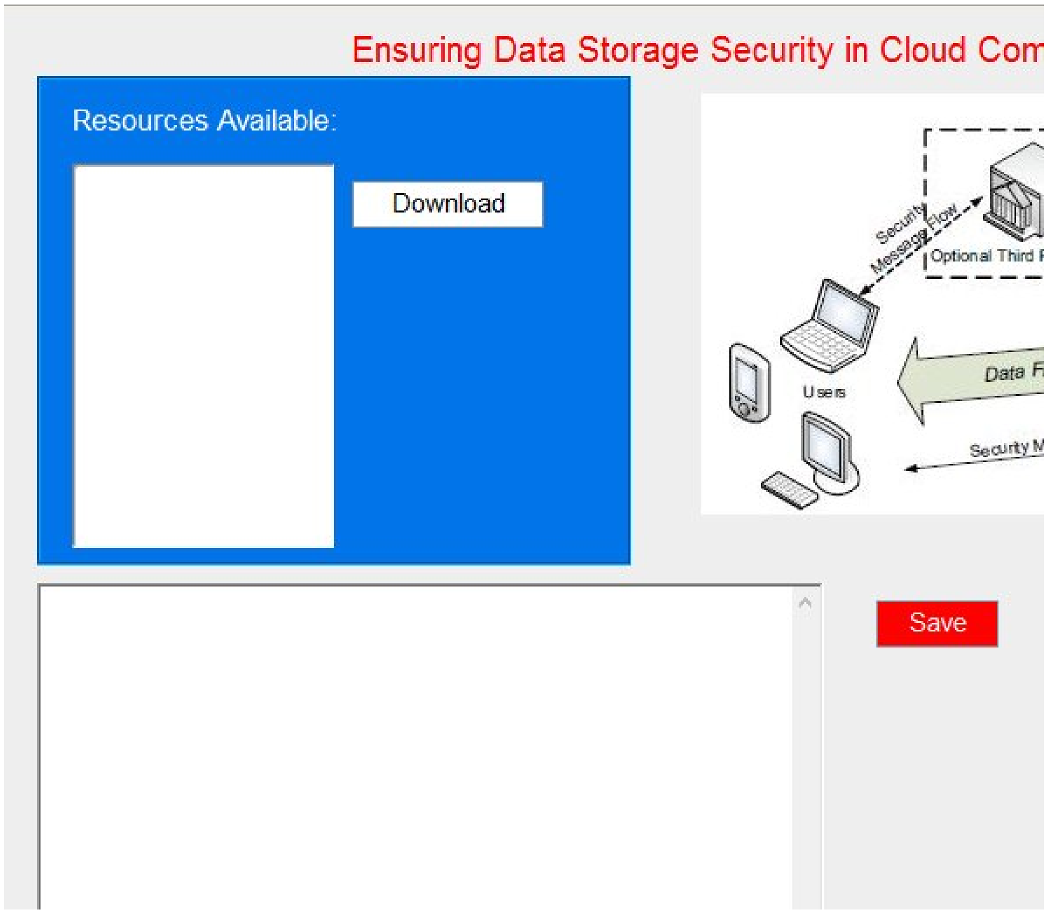


Fig . 3.10 Client Side Login Page

3.4.3 Server Side:



3.4.4 : Client Side



Client Side

```
private void saveActionPerformed(java.awt.event.ActionEvent evt)
                {
                        //GEN-FIRST:event_saveActionPerformed


                try {
                        FileDialog fd=new FileDialog(this,"File Store", FileDialog.SAVE);
                        fd.setVisible(true);
                        String f= fd.getFile();
                        fd.setFile(f); // Filename filter
                        fd.setDirectory("."); // Current directory
                        //fd.show();
                        FileOutputStream out=new FileOutputStream(f);
                        String s=resArea.getText();
                        System.out.println(s);
                        byte b[]=s.getBytes();
                        out.write(b);
                } catch(Exception e)
                        {
                        System.out.println(e);
                        }
  }
```

```
private void submitActionPerformed(java.awt.event.ActionEvent evt)
        {
        try
        {

                ref= (RMISIntf)Naming.lookup("rmi://"+"127.0.0.1"+"/RMIServer");

    if(user.getText().trim().equals(""))
                        {
                         op.showConfirmDialog(this,"Enter The User
Name","Alert",JOptionPane.DEFAULT_OPTION,JOptionPane.ERROR_MESSAGE);
                         user.grabFocus();
                        }
        else
        {
                         if(pass.getText().trim().equals(""))
                                {
                                 op.showConfirmDialog(this,"Enter The
PassWord","Alert",JOptionPane.DEFAULT_OPTION,JOptionPane.ERROR_MESSAGE);
                                 pass.grabFocus();
                                 }
                        else
                        {
                                InetAddress Address = InetAddress.getLocalHost();
                                String c =Address.getHostAddress();
                                ref= (RMISIntf)Naming.lookup("rmi://"+"127.0.0.1"+"/RMIServer");
                                        String ss=ref.CliDet(user.getText(),pass.getText(),c);
                                        if(ss.equals("notok"))
```

3.4.4 Server Side:

```
private void StartSerActionPerformed(ActionEvent actionevent)
  {
      try
      {
          System.setProperty("java.rmi.server.hostname", "127.0.0.1");
          Registry reg = LocateRegistry.createRegistry(1099);
          RMISImpl rmisimpl = new RMISImpl();
          System.out.println("Reached here");
          System.setProperty("java.rmi.server.hostname", "127.0.0.1");
          Naming.rebind("RMIServer", rmisimpl);
          System.out.println("Server Is Runnig...");
      }
      catch(Exception exception)
      {
          System.out.println("Exception in loop 1:" + exception);
      }
  }
```

Fig 3.12 Server Page code

```
public class dbcon
   {                    Connection con;
                        Statement st;
                         ResultSet rs;
dbcon()
{
            try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  con=DriverManager.getConnection("jdbc:odbc:db5");
}catch(Exception e){System.out.println(e);}
}
            public String check(String u,String p)
            { String s=null;
try{
 st=con.createStatement();
 rs=st.executeQuery("select * from login where user='"+u+"' and pass='"+p+"' ");
 if(rs.next())
{
s="ok";
}
else
{
s="notok";
}
}catch(Exception e){System.out.println(e);}
return s;
}
```

```
   private void saveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_saveActionPerformed

        // TODO add your handling code here:

                        try {

                                FileDialog fd=new FileDialog(this,"File Store", FileDialog.SAVE);

                                fd.setVisible(true);

                                String f= fd.getFile();

                                fd.setFile(f); // Filename filter

                                fd.setDirectory("."); // Current directory

                                fd.show();

                                FileOutputStream out=new FileOutputStream(f);

                                String s=resArea.getText();

                                byte b[]=s.getBytes();

                                out.write(b);

                        } catch(Exception e){}

    }


    private void downloadActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_downloadActionPerformed

        // TODO add your handling code here:

            sss=(String)listres.getSelectedValue();

            System.out.println(sss);

          int ch=0;

                        try{


                    File file = new File(sss);

                    FileInputStream in = new FileInputStream(file);

                    int i = in.available();
```

RMSIntf

```
import java.io.*;

import java.rmi.*;


public interface RMISIntf extends Remote{


public String CliDet(String a, String b,  String c) throws Exception;

public String CliDet_key(String user,String key) throws Exception;


public String res(String filename,  String ip,String fileop) throws Exception;

public String checkipp(String ip,String option) throws Exception;

public void store(String aa[],String ip,String username,String fn) throws Exception;


}
```

Chapter4 : Conclusion

This project has covered the most prominent cloud computing solutions available today. The discussion has focused on technical characteristics and tried to differentiate the cloud computing platforms. It has not evaluated business perspectives, potential savings, or economic opportunities opened up by cloud computing. Given the tremendous amount of experience and know how in system optimization and monitoring present in today's datacenters, another important aspect for successful operation of cloud applications will be the question of how to monitor and manage all layers of the cloud stack. Self-adaptive management mechanisms present in the cloud fabric are only in a very limited way exposed to cloud applications and clients. A consistent monitoring model covering all layers of the cloud stack is badly needed. Even more important is the question of how to secure data in the cloud. Current levels of access control available in the cloud platforms are very limited. Identity management and federation as well as management of trust among clients and services in the cloud will be most important.

BIBLIOGRAPHY

1. Amazon.com, “Amazon Web Services (AWS),” Online at http://aws. amazon.com, 2008.

   N. Gohring, “Amazon’s S3 down for several hours,” Online

2. At http://www.pcworld.com/businesscenter/article/142549/amazo s s3 down for several hours.html, 2008.
3. A. Juels and J. Burton S. Kaliski, “PORs: Proofs of Retrievability for Large Files,” Proc. of CCS ’07, pp. 584–597, 2007.

   H. Shacham and B. Waters, “Compact Proofs of Retrievability,” Proc. of Asiacrypt ’08, Dec. 2008.

4. K. D. Bowers, A. Juels, and A. Oprea, “Proofs of Retrievability: Theory and Implementation,” Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.
5. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” Proc. Of CCS ’07, pp. 598–609, 2007.
6. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. of SecureComm ’08, pp. 1– 10, 2008.
7. T. S. J. Schwarz and E. L. Miller, “Store, Forget, and Check: UsingAlgebraic Signatures to Check Remotely Administered Storage,” Proc.

Sites Referred:

http://java.sun.com

http://www.sourcefordgde.com

http://www.networkcomputing.com/

http://www.roseindia.com/

http://www.java2s.com/