

Implementation of Anonymous Authentication In Cloud

Project Report submitted in partial fulfillment of the requirement for the degree
of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Mrs. Ruchi Verma

By

Shubhanshu Gupta , 111221

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “Implementation of Anonymous Authentication In Cloud”, submitted by Shubhanshu Gupta in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Mrs. Ruchi Verma

Designation

Acknowledgement

There are many people who are associated with this project directly or indirectly whose help and timely suggestions are highly appreciable for completion of this project. First of all, I would like to thank Prof. Dr. SP Ghrrera, Head, Department of Computer Science Engineering for his kind support and constant encouragements, valuable discussions which is highly commendable.

I would like to express my sincere gratitude to my supervisor Mrs. Ruchi Verma, for her supervision, encouragement, and support which has been instrumental for the success of this project. It was an invaluable experience for me to be one of her students. Because of her, I have gained a careful research attitude.

Lastly, I would also like to thank my parents for their love and affection and especially their courage which inspired me and made me to believe in myself.

Date:

Shubhanshu Gupta

Roll No. 111221

Table of Content

S.No.	Topic	Page No.
1	Introduction	9
1.1	Cloud Computing	9
1.2	Characteristics of Cloud Computing	11
1.3	Deployment models	13
1.4	Service models	14
2	Network Security	18
2.1	Basic Concept	18
2.2	Layer Wise Security	18
3	Internet Security	22
3.1	Basic Concept	22
3.2	Common Threats	22
3.3	Technologies for Internet security	23
4	Cloud Security	25
4.1	Security	25
4.2	Security Issues in Cloud Computing	27
4.3	Attackers in Cloud	29
4.4	Security risks	30
4.5	Defence Mechanism	31
5	Authentication	32
5.1	Basic Concept	32
5.2	Threats to Authentication	35
5.3	Anonymous Authentication	36
6	Cryptography Used	37
6.1	Encryption	37
6.2	Kinds Of Encryption	37
6.2.1	Symmetric Encryption	37
6.2.2	Public key Encryption	39
7	System Model	42
7.1	Scope	42
7.2	Objective	42

7.3	Model Of The System	42
7.4	Architecture	43
7.5	Algorithm	44
7.6	UML Diagrams	45
7.6.1	Use Case Diagram	45
7.6.2	Class Diagram	45
7.6.3	Sequence Diagram	46
7.7	Cryptographic Algorithm Used	47
7.7.1	RSA Algorithm	47
7.7.2	AES Algorithm	49
7.7.3	Hybrid Cryptosystem	52
7.8	Java Cryptography Extension (JCE)	53
8	System Requirements	55
8.1	Hardware Requirements	55
8.2	Software Requirements	55
9	Snapshots of Implementation	56
10	Conclusion	64
11	References	65
12	Appendix	66

List of Figures

S.No.	Title	Page No.
1.	Cloud Computing in daily life	10
2.	Cloud Computing model	11
3.	Conceptual view of cloud computing	12
4.	Deployment models	14
5.	Service models	16
6.	Features of Cloud Computing	17
7.	Network and Data security	21
8.	Cloud computing issues	26
9.	Authentication process	33
10.	Symmetric encryption model	38
11.	Encryption with public key	40
12.	Encryption with private key	40
13.	Architecture model of project	43
14.	Use Case Diagram	45
15.	Class Diagram	46
16.	Sequence Diagram	47
17.	RSA algorithm	48
18.	AES Encryption Process	51
19.	AES Data Structures	51
20.	AES Encryption & Decryption	52
21.	Snapshots	56

List of Tables

S.No.	Title	Page No.
1.	Attackers on Cloud	29
2.	Security risks in Cloud	30
3.	Threats and Defense Mechanism	31
4.	Comparison of different symmetric algorithms	38
5.	Comparison of different public key algorithms	41

Abstract

This project deals with the security issue of authentication related to cloud computing. The cloud computing is a new computing model which comes from grid computing, distributed computing, parallel computing, virtualization technology, utility computing and other computer technologies. Despite several cost-effective and flexible characteristics of cloud computing, some clients are reluctant to adopt this paradigm due to emerging security and privacy concerns. Organizations, where confidentiality of information is a vital act, are not assertive to trust the security techniques and privacy policies offered by cloud service providers. Malicious attackers have violated the cloud storages to steal, view, manipulate and tamper client's data. Authentication is an important process in any system to verify whether someone is in fact. In any type of computer network such as private or public, authentication needs username and password. Password is a secret key to verify the person is authentic. When a user wishes to use a system, the first thing is the user has to register with the system, then a unique code is assigned for that person. On each subsequent use, the user must know and use the previously declared password. Authentication and authorization play a major role in accessing the cloud service from vendors. Cloud customers should be authenticating enough to use the cloud services. Cloud authentication could be done in many ways like textual password, biometrics and graphical etc. In this project, I try to present a privacy-preserving security solution for cloud services. It deals with user anonymous access to cloud services and shared storage servers. The solution provides registered users with anonymous access to cloud services. This solution offers anonymous authentication. This means that user's attributes can be proven without revealing user's identity. Thus, users can use services without any threat of profiling their behavior. On the other hand, if users break provider's rules, their access rights can be revoked. This solution offers anonymous access, unlinkability and the confidentiality of transmitted data.

Chapter-1 (Introduction)

1.1 Cloud Computing

Cloud computing refers to applications and services that run on a distributed network using virtualized resources and accessed by common Internet protocols and networking standards. It is distinguished by the notion that resources are virtual and limitless and that details of the physical systems on which software runs are abstracted from the user. Cloud computing is computing in which large groups of remote servers are networked to allow the centralized data storage, and online access to computer services or resources. Cloud computing provides a computer user access to Information Technology (IT) services i.e., applications, servers, data storage, without requiring an understanding of the technology or even ownership of the infrastructure. The “cloud” is defined as the Internet surrounding every part of our daily lives, similar to the clouds in the sky. Cloud computing is an emerging computing paradigm in which resources of the computing infrastructure are provided as services of the internet. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. Cloud computing allows individuals and businesses to use software and hardware that are managed by third parties at remote locations.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a utility over a network. Cloud computing, or in simpler shorthand just "the cloud", also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users. For example, a cloud computer facility that serves European users during European business hours with a specific application (e.g., email) may reallocate the same resources to serve North American users during North America's business hours with a different application (e.g., a web server). With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different application.

Some examples of cloud services are online file storage, social networking sites, webmail, online business applications etc. To understand cloud computing in the most basic terms an example could be when we store our photographs or music albums online rather than on our home computer's hard-disk, we are using cloud computing or when we use social networking sites or mailing services such as Gmail, we are using cloud computing services.

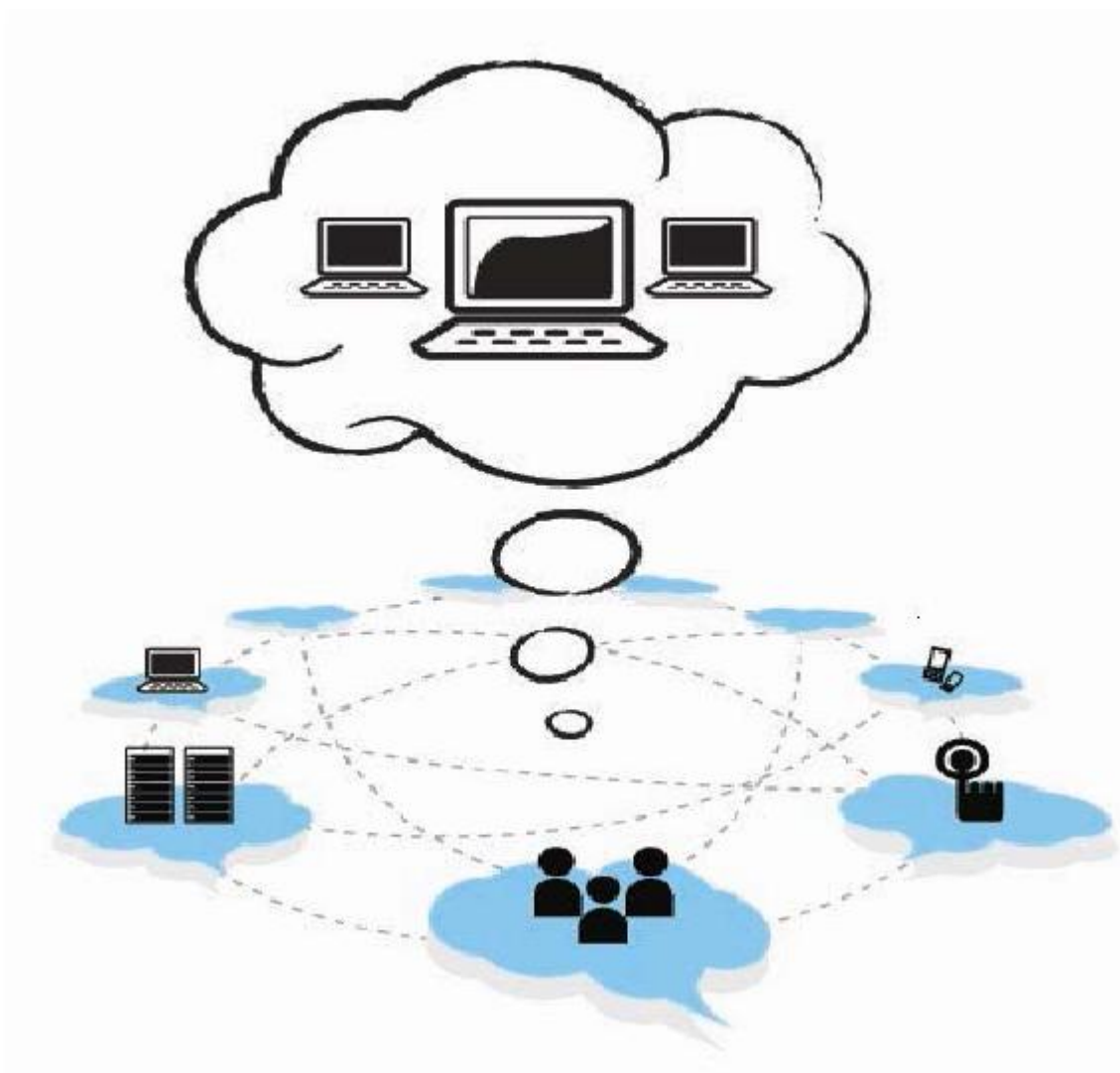


Figure 1: Cloud computing in daily life

Cloud computing takes the technology, services and applications that are similar to those on the Internet and turns them into a self-service utility. The use of the word “cloud” makes reference to two essential concepts:

- **Abstraction-** Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical system that are not specified, data is stored in locations that are unknown, administration of the system is outsourced to others and access by users is ubiquitous.
- **Virtualization-** Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are accessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.

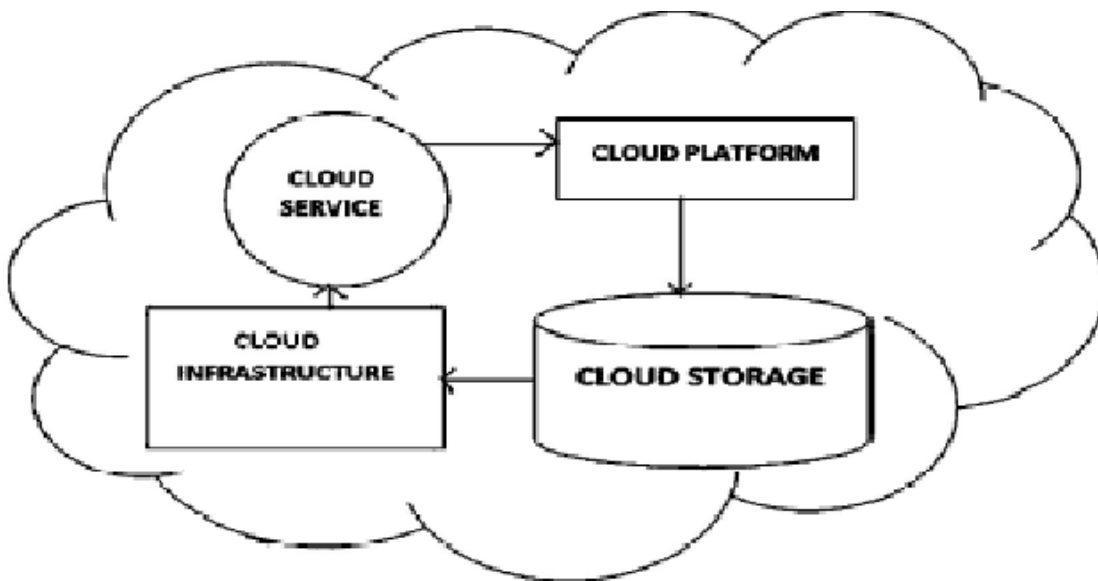


Figure 2: Cloud Computing Model

1.2 Characteristics of Cloud Computing

The key characteristics of cloud computing includes the following:

1.2.1 Multitenancy (shared resources)

Unlike previous computing models, which assumed dedicated resources (i.e., computing facilities dedicated to a single user or owner), cloud computing is based on a business model in which resources are shared (i.e., multiple users use the same resource) at the network level, host level, and application level.

1.2.2 Massive scalability

Although organizations might have hundreds or thousands of systems, cloud computing provides the ability to scale to tens of thousands of systems, as well as the ability to massively scale bandwidth and storage space.

1.2.3 Elasticity

Users can rapidly increase and decrease their computing resources as needed, as well as release resources for other uses when they are no longer required.

1.2.4 Pay as you go

Users pay for only the resources they actually use and for only the time they require them.

1.2.5 Self-provisioning of resources

Users self-provision resources, such as additional systems (processing capability, software storage) and network resources.

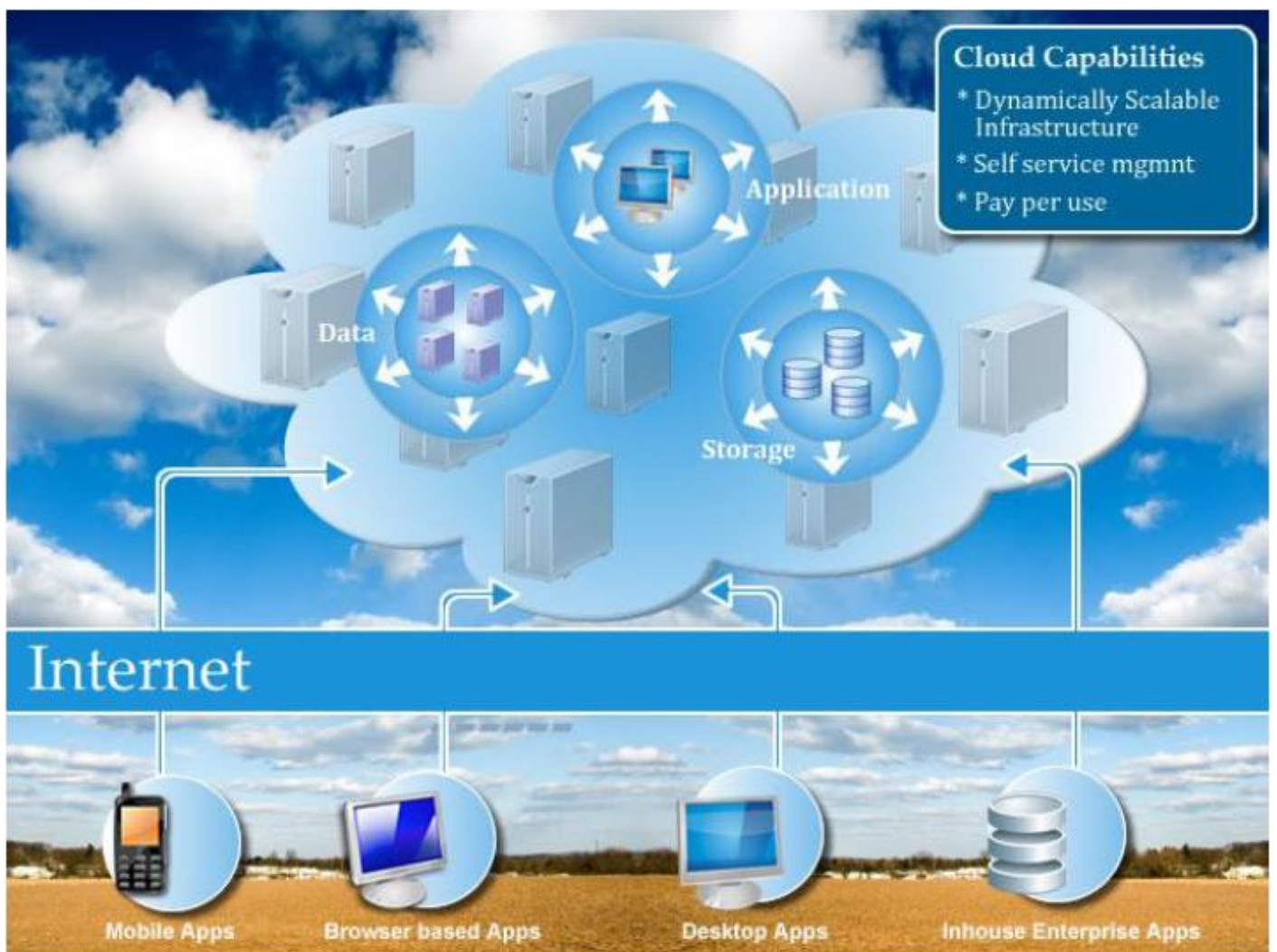


Figure 3: Conceptual view of Cloud Computing

1.3 DEPLOYMENT MODELS

This refers to the location and management of the cloud's infrastructure:

1.3.1 Public Cloud

The cloud infrastructure is made available to the general public or a large industry group and owned by an organization selling cloud services. The organization using public cloud does not control how those cloud services are operated, accessed or secured.

1.3.2 Private Cloud

The cloud infrastructure is operated solely for a single organization. It may be managed by the organization or a third party and may exist on or off-premises. While the organization does not need to physically own or operate all the assets, the key is that a shared pool of computing resources can be rapidly provisioned, dynamically allocated and operated for the benefit of a single organization.

1.3.3 Community Cloud

The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, or compliance considerations). It may be managed by the organizations or a third party and may exist on premises or off-premises.

1.3.4 Hybrid Cloud

The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

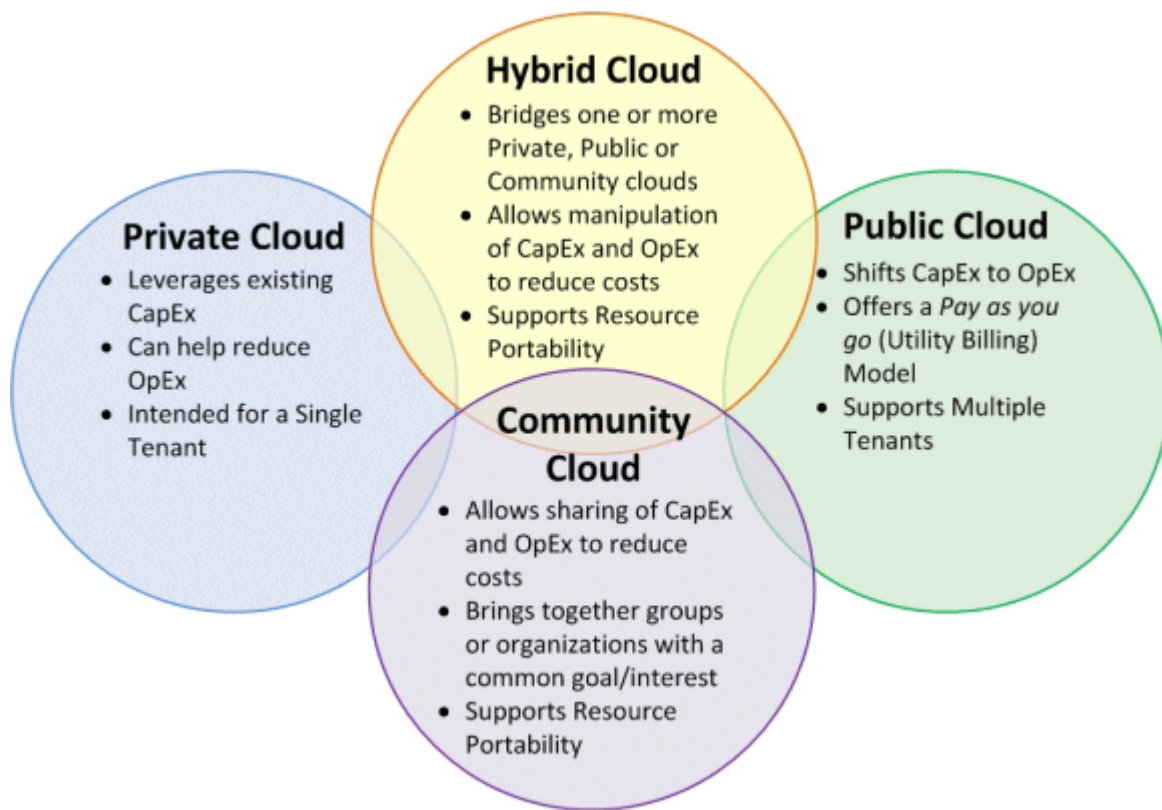


Figure 4: Different deployment models

1.4 Cloud Service Models

This consists of the particular types of services that we can access on a cloud computing platform:

1.4.1 Platform as a Service (PaaS)

PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions and control structures.

The client can deploy its applications on the cloud infrastructure or use applications that were programmed using languages and tools that are supported by the PaaS service provider. The service provider manages the cloud infrastructure, the operating systems and the enabling software. The client is responsible for installing and managing the application that it is deploying.

Some examples of PaaS services are:

- Force.com
- Google AppEngine

- Windows Azure Platform

1.4.2 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) provides virtual machines, virtual storage, virtual infrastructure and other hardware assets as resources that clients can provision.

The IaaS service provider manages all the infrastructure, while the client is responsible for all other aspects of the deployment. This can include the operating system, applications and user interactions with the system.

Some examples of IaaS are:

- Amazon Elastic Compute Cloud (EC2)
- Eucalyptus
- GoGrid

1.4.3 Software as a Service (SaaS)

Software as a service (or SaaS) is a way of delivering applications over the Internet as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management. SaaS is a complete operating environment with applications, management and the user interface.

In the SaaS model, the application is provided to the client through a thin client interface (usually a browser) and the customer's responsibility begins and ends with entering and managing its data and user interaction.

Everything from the application down to the infrastructure is the vendor's responsibility.

SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Key characteristics of SaaS are Scalability, Multi-Tenant efficiency and Configurability.

Some examples of SaaS are:

- GoogleApps
- SalesForce.com
- SQL Azure
- Oracle On Demand

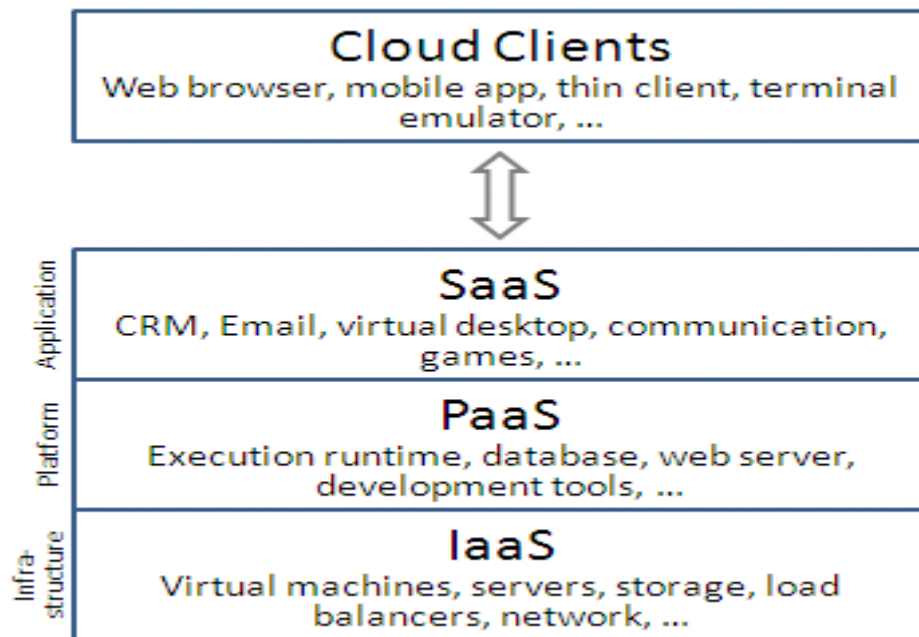


Figure 5: Common Service Models

The three different service models taken together are known as SPI model of cloud computing. Many other service models are there such as:

- Storage as a Service (StaaS)
- Identity as a Service (IdaaS)
- Database as a Service (DaaS)

The different service models takes the common form of :

XaaS or "<Something> as a Service"

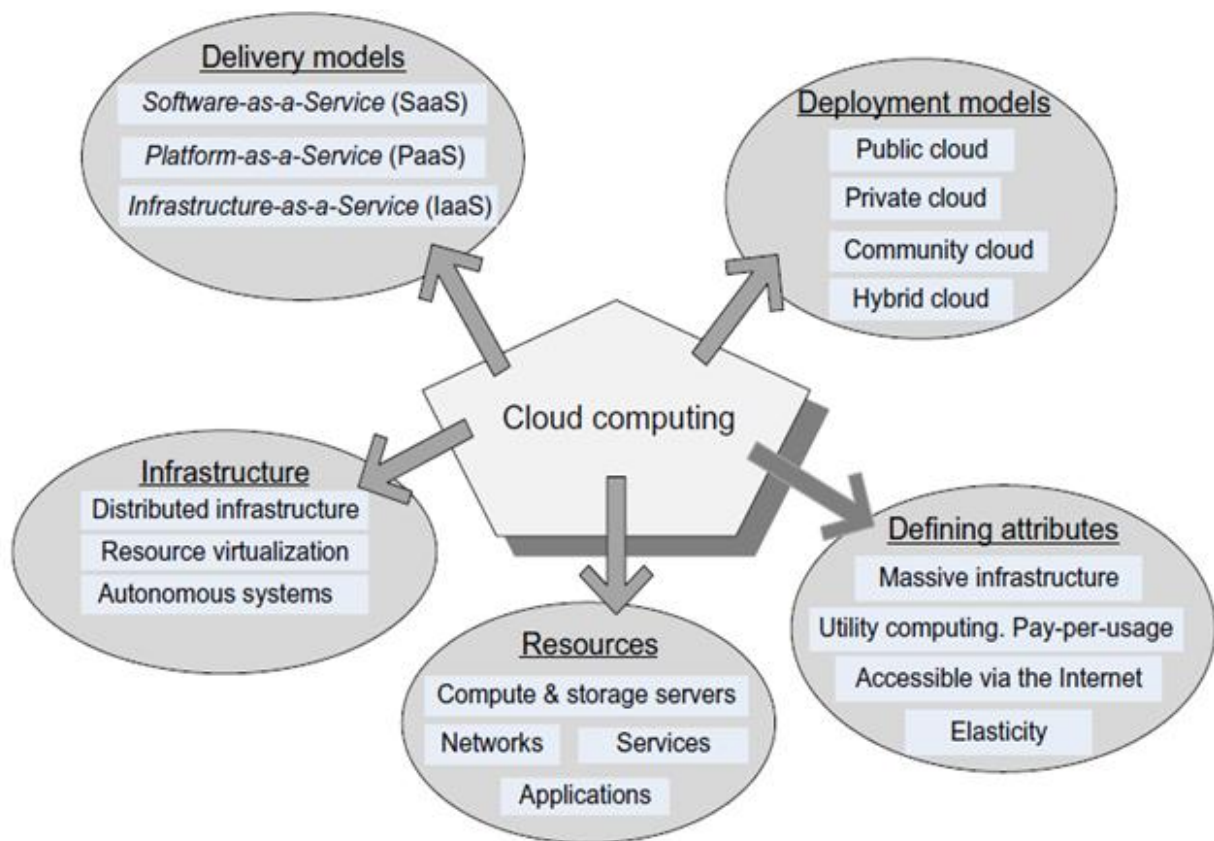


Figure 6: Cloud Computing Features

Chapter-2 (Network Security)

2.1. Basic Concept

Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority. Network security covers a variety of computer networks, both public and private, that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals. The internet structure has allowed for many security threats to occur. The architecture of the internet, when modified can reduce the possible attacks that can be sent across the network. Knowing the attack methods, allows for the appropriate security to emerge. Many businesses secure themselves from the internet by means of firewalls and encryption mechanisms.

2.2. Layer Wise Security

Network design is a well-developed process that is based on the Open Systems Interface (OSI) model. The OSI model has several advantages when designing networks. It offers modularity, flexibility, ease-of-use, and standardization of protocols. The protocols of different layers can be easily combined to create stacks which allow modular development. The implementation of individual layers can be changed later without making other adjustments, allowing flexibility in development. Security has become multi-dimensional i.e. it spans through different layers. The security on different layers i.e. physical, hardware, application, operating system and network layers.

2.2.1 Physical Layer Security

Physical security is often viewed as the first line of defense of a system. It forbids the intruder to access the system physically (to sit and access information on an already logged in computer).

2.2.2 Hardware Layer Security

There are two aspects of hardware security, the first one consists of security at the hardware level in CPUs, and the second one consists of hardware security at the level of the enterprise and the users. An example of a

security issue at the CPU level is the interrupt handling. The interrupt vector table is a target for hackers that are able to exploit the system vulnerabilities at the lower level.

2.2.3 Application Layer Security

Application layer security is very important since it involves entrance of data. The application layer consists of software and database development mainly. The threats at this layer are: buffer overflows, backdoors, incompleteness of data and viruses. There are specific guidelines we will be following to achieve security at this layer.

2.2.4 Operating System Layer Security

As will be seen in the next section of this report, the computers we were provided with have Pentium2 processors. We therefore decided to install Windows2000 on them, since WindowsXP would be too slow, as well as Fedora. This section aims to present the vulnerabilities and protection schemas for Windows2000 Operating System and Linux OS. Hardening the operating system involves many things that are not only operating system specific, but may often vary from one "flavor" of an operating system to another.

2.2.5 Network Layer Security

Network Layer Security among mutually trusting hosts is a relatively straightforward problem to solve. The standard protocol technique, employed in IPSEC, involves "encapsulating" an encrypted Network Layer packet inside a standard Network packet, making the encryption transparent to intermediate nodes that must process packet headers for routing, etc. Outgoing packets are authenticated, encrypted, and encapsulated just before being sent to the network, and incoming packets are decapsulated, verified, and decrypted immediately upon receipt. Key management in such a protocol is similarly straightforward in the simplest case. Two hosts can use any key-agreement protocol to negotiate keys with one another, and simply use those keys as part of the encapsulating and decapsulating packet transforms.

In many applications, security at the network layer has a number of advantages over security provided elsewhere in the protocol stack. Network semantics are usually hidden from applications, which therefore automatically and transparently take advantage of whatever network layer security services their environment provides. Especially importantly, the network layer offers a remarkable flexibility not possible at higher- or lower- abstractions: security can be configured end-to-end (protecting traffic between two hosts), route-to-route (protecting traffic passing over a particular set of links), edge-to edge (protecting

traffic as it passes between "trusted" networks via an "untrusted" one), or in any other configuration in which network nodes can be identified as appropriate security endpoints.

2.2.6 Internal Network Security

Although focusing on securing the networks perimeter is important, securing it internally is equally important. If by some way a hacker manages to get in the network, he shouldn't be able to wander around easily without getting caught. Therefore, one should apply the following to make the internal network secure:

- Patch and update all PCs before they are connected to the network, and then on a regular basis.
- System administrators should use one-time passwords only. In this way, in case a hacker cracks the Admin's password, it would only be valid for this one session.
- When an application is installed, some service accounts may be created. They are accounts which do not have a human user associated to them. These accounts are assigned default passwords that will most likely never be changed. Therefore, it is important for an administrator to regularly change these passwords and monitor the logs of these service accounts.
- Monitoring of logs is important: administrators should regularly read the logs to monitor any unusual use of an account. Many freeware tools (such as log-IDS by Adam Richard) can help decipher the logs (which otherwise are almost impossible to read) in something that the administrator can understand. Moreover, by using a centralized syslog server, it will be much more difficult for hackers to access them and edit them.
- Also, available freeware such as EventAlarm are useful when the Administrator want to monitor a user's logging in and out in a fast way. Such a freeware gives pop out screen alarms to the administrator whenever user X or Y logs on or off.

When developing a secure network, the following need to be considered :

- **Access** – Authorized users are provided the means to communicate to and from a particular network.
- **Confidentiality** – Information in the network should remain private.
- **Authentication** – It should be ensured that the users on the network are who they say they are.

- **Integrity** – It should be ensured that the message has not been modified in transit.
- **Non-repudiation** – It should be ensured that the user do not refute that he used the network.

Data security is also a very important part of network security. Data security is the aspect of security that allows a client's data to be transformed into unintelligible data for transmission. Even if this unintelligible data is intercepted, a key is needed to decode the message.

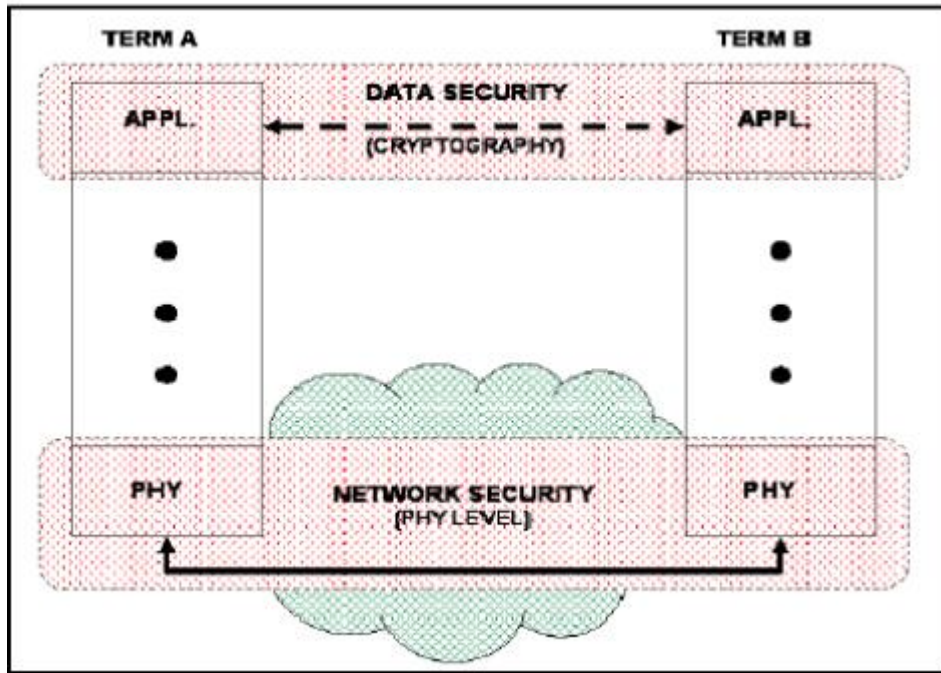


Figure 7: Network security and Data Security

Chapter-3(Internet Security)

3.1 Basic Concept

Internet security is a tree branch of computer security specifically related to the Internet, often involving browser security but also network security on a more general level as it applies to other applications or operating systems on a whole. Its objective is to establish rules and measures to use against attacks over the Internet. The Internet represents an insecure channel for exchanging information leading to a high risk of intrusion or fraud, such as phishing. Different methods have been used to protect the transfer of data, including encryption. Security is a very general and broad concept and could usually point to more specific topics such as network security, data security, computer security, applications security, system security, wireless and mobile device security, Internet Protocol (IP) security, Voice over IP security, web services security, email security, etc. depending on different circumstances. Some are interchangeable and related, and some can result in others.

3.2 Common Threats

3.2.1 Denial-of-Service (DoS)

DoS is designed to interrupt normal system functions and affect legitimate users' access to the system, or unauthorized access in order to execute system commands, obtain confidential information, or to perform destructive attacks. A massive Distributed DoS (DDoS) attack can even paralyze a network system and bring down a website.

3.2.2 Unauthorized Access

Unauthorized access happens when a user gains access to a resource (network, system, application, data, file, etc.) without permission from the owner of the resource. Unauthorized access almost always occurs on a server host which is providing some services on the Internet regardless of the nature of the service.

Unauthorized access might result in a slow network connection, impairment of the host's processing ability, and if happening to a host's files, it could result in modification, and/or destruction of vital data.

3.2.3 Eavesdropping

Interception of communications by an unauthorized party is called eavesdropping. Passive eavesdropping is when the person only secretly listens to the networked messages. On the other hand, active eavesdropping is when the intruder listens and inserts something into the communication stream. This can lead to the messages being distorted. Sensitive information can be stolen this way.

3.2.4 Viruses

Viruses are self-replication programs that use files to infect and propagate. Once a file is opened, the virus will activate within the system.

3.2.5 Worms

A worm is similar to a virus because they both are self-replicating, but the worm does not require a file to allow it to propagate . There are two main types of worms, mass-mailing worms and network aware worms. Mass mailing worms use email as a means to infect other computers. Network-aware worms are a major problem for the Internet. A network-aware worm selects a target and once the worm accesses the target host, it can infect it by means of a Trojan or otherwise.

3.2.6 Trojans

Trojans appear to be benign programs to the user, but will actually have some malicious purpose. Trojans usually carry some payload such as a virus.

3.2.7 Phishing

Phishing is an attempt to obtain confidential information from an individual, group, or organization . Phishers trick users into disclosing personal data, such as credit card numbers, online banking credentials, and other sensitive information.

3.2.8 IP Spoofing Attacks

Spoofing means to have the address of the computer mirror the address of a trusted computer in order to gain access to other computers. The identity of the intruder is hidden by different means making detection and prevention difficult. With the current IP protocol technology, IP spoofed packets cannot be eliminated.

3.3 Technology for Internet Security

3.3.1 Secure Sockets Lay (SSL) Virtual Private Networks (VPNs):

SSL VPNs provide users with secure remote access to an organization's resources. An SSL VPN consists of one or more VPN devices to which users connect using their web browsers. SSL VPNs can provide remote users with access to web applications and client/server applications, as well as connectivity to internal networks.

3.3.2 Cryptography:

Cryptography is encryption and transformation of data to unintelligible cipher text, in order to hide their semantic content, prevent their unauthorized use, or prevent their undetected modification. Federal agencies, industry, and the public rely on cryptography for the protection of information and communications used in electronic commerce, critical infrastructure, and other application areas.

3.3.3 Firewall

A firewall is a typical border control mechanism or perimeter defense. The purpose of a firewall is to block traffic from the outside, but it could also be used to block traffic from the inside. A firewall is the front line defense mechanism against intruders. It is a system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both.

3.3.4 Intrusion Detection Systems

An Intrusion Detection System (IDS) is an additional protection measure that helps ward off computer intrusions. IDS systems can be software and hardware devices used to detect an attack. IDS products are used to monitor connection in determining whether attacks are being launched. Some IDS systems just monitor and alert of an attack, whereas others try to block the attack.

CHAPTER-4(Cloud Security)

4.1 Cloud Security

Cloud computing security is a sub-domain of computer security, network security, and information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing.

Cloud security architecture is effective only if the correct defensive implementations are in place. An efficient cloud security architecture should recognize the issues that will arise with security management. The security management addresses these issues with security controls. These controls are put in place to safeguard any weaknesses in the system and reduce the effect of an attack. While there are many types of controls behind a cloud security architecture, they can usually be found in one of the following categories:

4.1.1 Deterrent controls

These controls are intended to reduce attacks on a cloud system. Much like a warning sign on a fence or a property, deterrent controls typically reduce the threat level by informing potential attackers that there will be adverse consequences for them if they proceed. [Some consider them a subset of preventive controls.

4.1.2 Preventive controls

Preventive controls strengthen the system against incidents, generally by reducing if not actually eliminating vulnerabilities. Strong authentication of cloud users, for instance, makes it less likely that unauthorized users can access cloud systems, and more likely that cloud users are positively identified.

4.1.3 Detective controls

Detective controls are intended to detect and react appropriately to any incidents that occur. In the event of an attack, a detective control will signal the preventative or corrective controls to address the issue. System and network security monitoring, including intrusion detection and prevention arrangements, are typically employed to detect attacks on cloud systems and the supporting communications infrastructure.

4.1.4 Corrective controls

Corrective controls reduce the consequences of an incident, normally by limiting the damage. They come into effect during or after an incident. Restoring system backups in order to rebuild a compromised system is an example of a corrective control.

There are numerous security issues for cloud computing as it encompasses many technologies including networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing, concurrency control and memory management. Therefore, security issues for many of these systems and technologies are applicable to cloud computing. For example, the network that interconnects the systems in a cloud has to be secure. Furthermore, virtualization paradigm in cloud computing leads to several security concerns. For example, mapping the virtual machines to the physical machines has to be carried out securely. Data security involves encrypting the data as well as ensuring that appropriate policies are enforced for data sharing. In addition, resource allocation and memory management algorithms have to be secure.

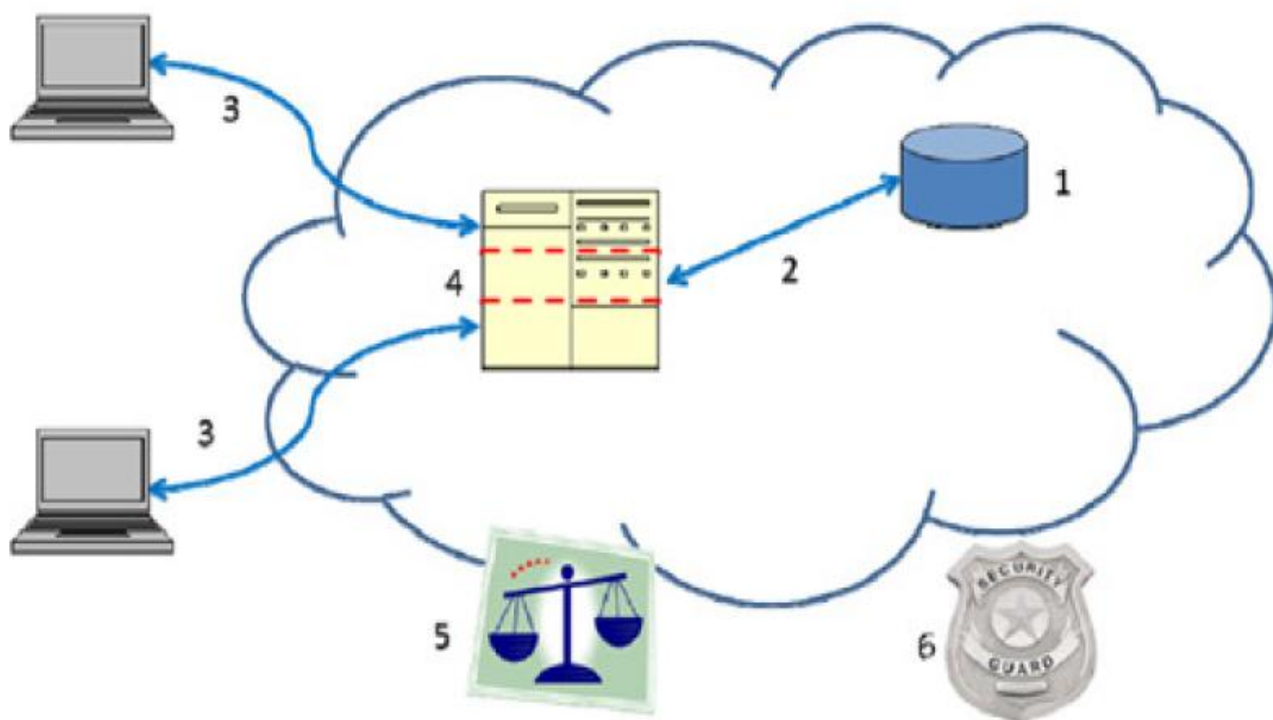


Figure 8: Showing different security issues in cloud computing

The above figure shows six security areas in cloud computing:

- Security of data at rest.
- Security of data in transit
- Authentication of users/processes
- Separation between data of different users
- Cloud legal and regulatory issues
- Incident response

4.2 Security Issues In Cloud Computing

The use of cloud computing introduces new attack vectors that will make attacks either possible or simply easier to carry out. Some of the security issues affecting the cloud model are described below:

4.2.1 Availability

The availability ensures the reliable and timely access to cloud data or cloud computing resources by the appropriate personnel. The availability of cloud service providers is also a big concern, since if the cloud service is disrupted; it affects more customers than in the traditional model.

4.2.2 Data Confidentiality

Confidentiality refers to the prevention of intentional or unintentional unauthorized disclosure of information. Confidentiality in cloud system is related to the areas of intellectual property rights, covert channels, traffic analysis, encryption, and inference. Cloud computing involves the sharing or storage of information on remote servers owned or operated by others, while accessing through the Internet or any other connections. Cloud computing services exist in many variations, including data storage sites, video sites, tax preparation sites, personal health record websites and many more.

4.2.3 Authentication and Authorization

Authentication is the act of confirming the truth of an attribute of a single piece of data or entity.

Authorization or authorisation is the function of specifying access rights to resources related to information security and computer security in general and to access control in particular. More formally, "to authorize" is to define an access policy.

The authentication and authorization applications for enterprise environments may need to be changed, to work with a safe cloud environment.

4.2.4 Network Security

All data flow over the network needs to be secured in order to prevent leakage of sensitive information. This involves the use of strong network traffic encryption techniques such as Secure Socket Layer (SSL) and the Transport Layer Security (TLS) for security.

4.2.5 Data Segregation

Multi-tenancy is one of the major characteristics of cloud computing. As a result of multitenancy, multiple users can store their data using the applications provided by cloud. In such a situation, data of various users will reside at the same location. Intrusion of data of one user by another becomes possible in this environment. This intrusion can be done either by hacking through the loop holes in the application or by injecting client code into the cloud system. A client can write a masked code and inject into the application. If the application executes this code without verification, then there is a high potential of intrusion into other's data. A cloud model should therefore ensure a clear boundary for each user's data.

4.2.6 Backup

The traditional backup methods used with earlier applications and data centers that were primarily designed for web and consumer applications, are not optimally designed for the applications running in the cloud. The cloud vendor needs to ensure that all sensitive enterprise data is regularly backed up to facilitate quick recovery in case of disasters. Also the use of strong encryption schemes to protect the backup data is recommended to prevent accidental leakage of sensitive information.

4.2.7 Identity Management

Identity management or ID management is an area that deals with identifying individuals in a system and controlling the access to the resources in that system by placing restrictions on the established identities. This area is considered as one of the biggest challenges in information security. When a cloud provider has to know how to control who has access to what systems within the enterprise it becomes all the more challenging task. In such scenarios the provisioning and de-provisioning of the users in the cloud becomes very crucial.

4.3 Types of Attackers in Cloud Computing

Cloud computing threats can result from the attackers that can be divided into two categories:

Internal Attackers	<p>An internal attacker has the following characteristics:</p> <ul style="list-style-type: none">• Is employed by the cloud service provider, customer or other third party provider organization supporting the operation of a cloud service.• May have existing authorized access to cloud services, customer data or supporting infrastructure and applications, depending on their organizational role.• Uses existing privileges to gain further access or support third parties in executing attacks against the confidentiality integrity and availability of information within the cloud service.
External Attackers	<p>An external attacker has the following characteristics:</p> <ul style="list-style-type: none">• Is not employed by the cloud service provider, customer or other third party provider organization supporting the operation of a cloud service.• Has no authorized access to cloud services, customer data or supporting infrastructure and applications.• Exploits technical, operational, process and social engineering vulnerabilities to attack a cloud service provider, customer or third party supporting organization to gain further access to propagate attacks against the confidentiality, integrity and availability of information within the cloud service.

Table 1 : Attackers on cloud

4.4 Cloud Security Risks

Risk	Description
Privileged User Access	Cloud providers generally have unlimited access to user data, controls are needed to address the risk of privileged user access leading to compromised customer data.
Data Location and Segregation	Customers may not know where their data is being stored and there may be a risk of data being stored alongside other customers' information.
Data disposal	Cloud data deletion and disposal is a risk, particularly where hardware is dynamically issued to customers based on their needs. The risk of data not being deleted from data stores, backups and physical media during decommissioning is enhanced within the cloud.
Protective Monitoring	The ability for cloud customers to invoke their own electronic investigations procedures within the cloud can be limited by the delivery model in use, and the access and complexity of the cloud architecture. Customers cannot effectively deploy monitoring systems on infrastructure they do not own; they must rely on the systems in use by the cloud service provider to support investigations.
Assuring Cloud Security	Customers cannot easily assure the security of systems that they do not directly control without using SLAs and having the right to audit security controls within their agreements.

Table 2: Security Risks

4.5 Threats and Defense Mechanism

Security Threats	Possible Defence Mechanism
Spoofing Identity	<ul style="list-style-type: none">• Authentication• Protect secrets• Don't store secrets
Tampering with Data	<ul style="list-style-type: none">• Authorization• Hashes• Message authentication codes• Digital Signatures• Tamper – resistant protocol
Repudiation	<ul style="list-style-type: none">• Digital signatures• Time-stamps• Audit trails
Information Disclosure	<ul style="list-style-type: none">• Authorization• Privacy-enhanced protocols• Encryption• Protect secrets• Don't store secrets
Denial of Service (DoS)	<ul style="list-style-type: none">• Authentication• Authorization• Filtering• Throttling• Quality of service (QoS)

Table 3:Threats and defense Mechanism

CHAPTER-5(AUTHENTICATION)

5.1 Basic Concept

Authentication is the act of confirming the truth of an attribute of a single piece of data or entity.

Authentication is a mechanism that establishes the validity of the claimed identity of the individual. There are basically four kind authentication methods:

- Something an individual KNOWS (e.g. password, Personal ID)
- Something an individual POSSESSES (e.g., a token or card).
- Something an individual IS (e.g. fingerprint or voice pattern).
- Something an individual DOES (e.g. history of Internet usage).

Authentication is the process of validating user identity. The fact that the user claims to be represented by a specific abstract object (identified by its user ID) does not necessarily mean that this is true. To ascertain that an actual user can be mapped to a specific abstract user object in the system, and therefore be granted user rights and permissions specific to the abstract user object, the user must provide evidence to prove his identity to the system. Authentication is the process of ascertaining claimed user identity by verifying user-provided evidence.

The evidence provided by a user in the process of user authentication is called a credential. Different systems may require different types of credentials to ascertain user identity, and may even require more than one credential. In computer systems, the credential very often takes the form of a user password, which is a secret known only to the individual and the system. Credentials may take other forms, however, including PIN numbers, certificates, tickets, etc.

There are typically three components involved in the process of user authentication:

- **Supplicant:** the party in the authentication process that will provide its identity, and evidence for it, and as a result will be authenticated. This party may also be referred to as the authenticating user, or the client.
- **Authenticator:** the party in the authentication process that is providing resources to the client (the supplicant) and needs to ascertain user identity to authorize and audit user access to resources. The authenticator can also be referred to as the server.
- **Security authority/database:** storage or mechanism to check user credentials. This can be as simple as a flat file, or a server on the network providing for centralized user authentication, or a set of distributed authentication servers that provide for user authentication within the enterprise or on the Internet.

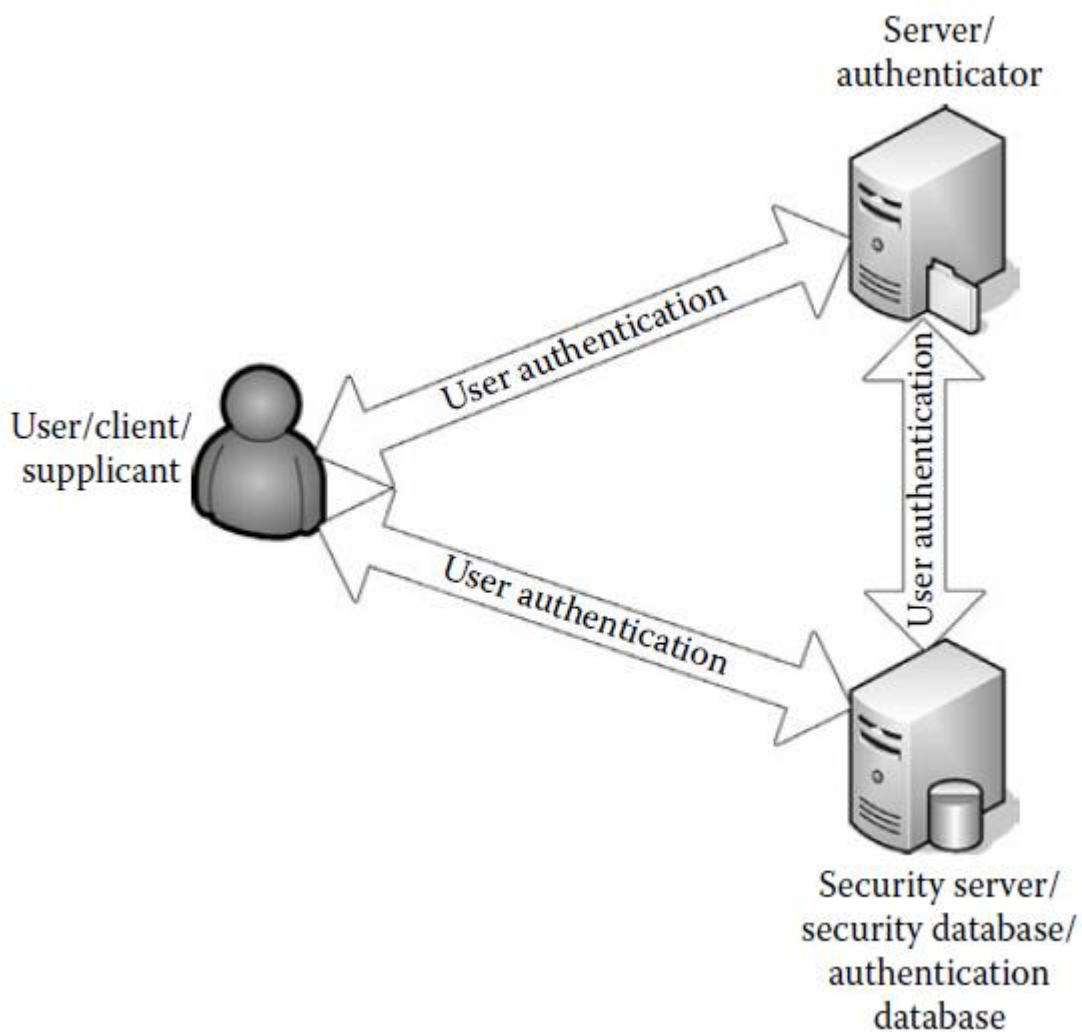


Figure 9: Components of authentication process

In a simple scenario, the supplicant, authenticator, and security database may reside on the same computer. It is also possible and somewhat common for network applications to have the supplicant on one computer and the authenticator and security database collocated on another computer. It is also possible to have the three components geographically distributed on multiple computers.

A complete understanding of authentication can be done by concentrating on the following processes:

Establishing identity

A business partner's identity must be established before it can be trusted in conducting trade. At the most basic level, there must be a process which verifies that an organization or individual exists, has a name, and is entitled to use that name. This process may also establish other identification attributes: for example, organizational affiliation ("Jim Smith works for Philips"); industry segment ("Vivendi is in the entertainment industry"); or occupational certification ("John Smith is a board-certified dentist in England"). Trusted third parties or delegated authorities often play a key role in confirming the identity attributes of participants at the time identification takes place.

Credential management

Once the participant's basic identity and identification attributes are established and verified, it must be issued with a credential that can be used to prove identity. In the "real" world, a credential might be an ID document or a business license. In the digital world, the most robust form of credential is the digital or Server Certificate signed by a trusted Certification Authority.

How authentication works

Authentication allows the receiver of a digital message to be confident of both the identity of the sender and the integrity of the message. When Web visitors connect to Websites, they reach one of two kinds of servers.

If the servers are secure, visitors will get messages indicating that fact; similarly, if they are not secure, there may be warnings to that effect. A secure Web site is one that has been authenticated, a complex process involving public keys/digital certificates, and private keys. The certificate tells users that an independent third party has agreed that the web site belongs to the company it claims to belong to. A valid certificate means that users can be confident that they are sending confidential information to the place they think they are sending it.

5.2 Threats to Authentication

Attackers can take different approaches when they attempt to compromise the user identification and authentication mechanisms of a system. Some of the main threats to authentication are:

- **Bypassing Authentication** : If an attacker does not have a username and a password or other credentials, and is not able to authenticate to a system, he may try to bypass the authentication process. This can be accomplished in a number of ways, depending on the application, and the type of access that attackers have to the computer where the application is running. If an application is running locally on a computer, and an attacker has physical access to this computer, then the attacker can potentially obtain administrative privileges, which may well be already available or may be obtained by privilege escalation. Once the attacker has administrative access, he can typically access all files and processes on the local computer, which allows him to debug running applications or swap files on the file system.
- **Privilege Escalation**: During the logon process, a user authenticates using a set of credentials. When a user tries to access resources, this request is actually performed by processes running on behalf of the user, that use the user access token to be authorized by resource servers. In some cases, users may be able to find a security hole in an application or an operating system. Very often, a service or an application may be running with its own account on a server. This account may have limited or unrestricted access privileges. By providing invalid input, an attacker can change the authentication logic of the service or application and assume the credentials of this application. A popular attack of this type is represented by the stack overflow attack. By providing invalid parameters (typically strings longer than the buffer reserved for user input by the application), an attacker can inject code and an incorrect return address pointing to the injected code into the application stack. This may force the application to execute arbitrary code using its own privileges.
- **Password Guessing and Brute Force**: One of the oldest types of attacks is password guessing. User authentication using a username and password has been available since the dawn of IT, and password guessing attacks have always been relatively easy to implement. They have also had surprising success every so often. If a system requires the user to authenticate with a username and password, an attacker can try to guess the username and the password, and then authenticate as the actual user. Information about usernames can be easily obtained by an attacker. In many cases, the username of a user is the same as the first portion of the user's e-mail address. An attacker may also know the naming convention used by the organization to create user IDs for new users. Alternatively, the attacker may even have access to a directory with all the users in the system — which is very likely for attackers internal to the organization. To launch the attack, the attacker will try one or more passwords that are likely to be used by the actual user as the password.

5.3 Anonymous Authentication

Anonymous authentication is the process of confirming a user's right to access some service. It allows users to log into the system without exposing their actual identity. The user remains anonymous, and the authentication process just confirms that the credential is valid and is in the hands of the owner of the credential. There are two requirements for anonymous authentication:

Secure authentication: Unauthorized users should not be able to grant access to the server.

Anonymity : Server should not know the identity of the user.

These two requirements seem to contradict each other. This can be done in a way that the server realizes that the user belongs to a group of authorized users but it cannot find out which user is it in the set.

CHAPTER-6(CRYPTOGRAPHY USED)

6.1 Encryption

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. In an encryption scheme, the message or information, referred to as plaintext, is encrypted using an encryption algorithm, generating cipher text that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skill are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients, but not to unauthorized interceptors

6.2 Kinds of Encryption

6.2.1 Symmetric key encryption

Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. In symmetric-key schemes the encryption and decryption keys are the same. Thus communicating parties must have the same key before they can achieve secret communication.

A symmetric cipher scheme has five ingredients:

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

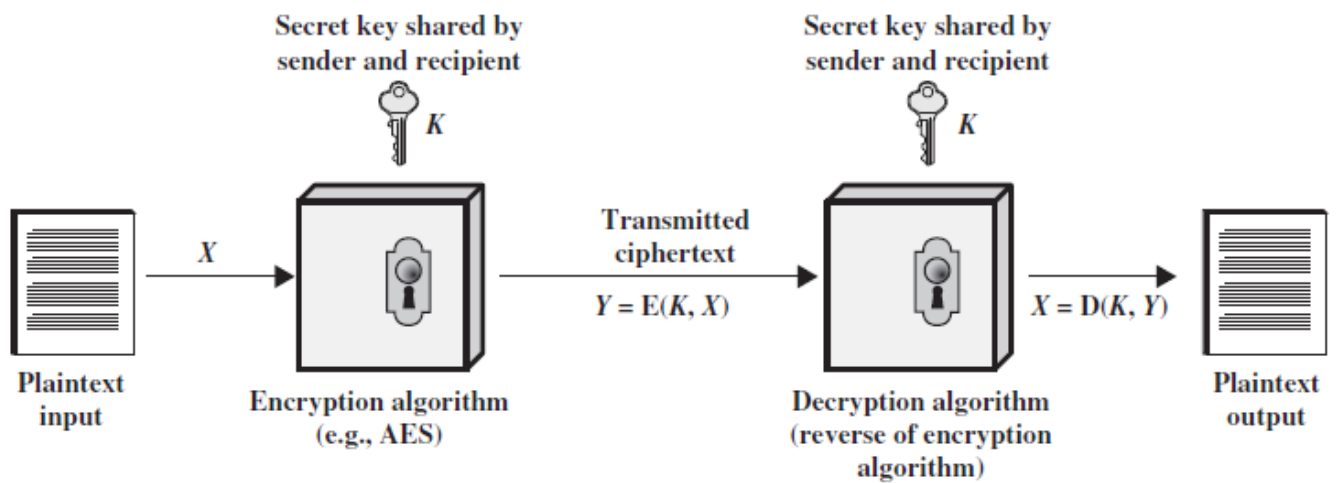


Figure 10 : Model of Symmetric Encryption

Symmetric-key encryption can use either stream ciphers or block ciphers.

- Stream ciphers encrypt the digits (typically bytes) of a message one at a time.
- Block ciphers take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size. Blocks of 64 bits have been commonly used.

Algorithm Name	Block or Stream Cipher?	Key Size	Number of Rounds	Block Size
DES	Block	64 bits(effective length 56 bits)	16	64 bits
3DES	Block	56,112,or 168 bits	48	64 bits
AES	Block	128,192 or 256 bits	10, 12 or 14 (depending on block/key size)	128,192 and 256 bits
IDEA	Block	128 bits	8	64 bits
Blowfish	Block	32 to 448 bits	16	64 bits
RC4	Stream	40 to 2048 bits	Up to 256	n/a

RC6	Block	Up to 2048 bits	Up to 255	32,64 or 128 bits
-----	-------	-----------------	-----------	-------------------

Table 4: Comparison of different symmetric algorithms

6.2.2 Public key encryption

Public-key cryptography, also known as asymmetric cryptography, is a class of cryptographic protocols based on algorithms that require two separate keys, one of which is secret (or private) and one of which is public. Although different, the two parts of this key pair are mathematically linked. The public key is used, for example, to encrypt plaintext or to verify a digital signature; whereas the private key is used for the opposite operation, in these examples to decrypt ciphertext or to create a digital signature. The term "asymmetric" stems from the use of different keys to perform these opposite functions, each the inverse of the other – as contrasted with conventional ("symmetric") cryptography which relies on the same key to perform both. Public-key algorithms are based on mathematical problems that currently admit no efficient solution and are inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships. It is computationally easy for a user to generate their own public and private key-pair and to use them for encryption and decryption. The strength lies in it being "impossible" (computationally infeasible) for a properly generated private key to be determined from its corresponding public key. Thus the public key may be published without compromising security, whereas the private key must not be revealed to anyone not authorized to read messages or perform digital signatures. Public key algorithms, unlike symmetric key algorithms, do not require a secure initial exchange of one (or more) secret keys between the parties. In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read. Public-key encryption was first described in a secret document in 1973 before then all encryption schemes were symmetric-key (also called private-key).

A public key cipher scheme has six ingredients:

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

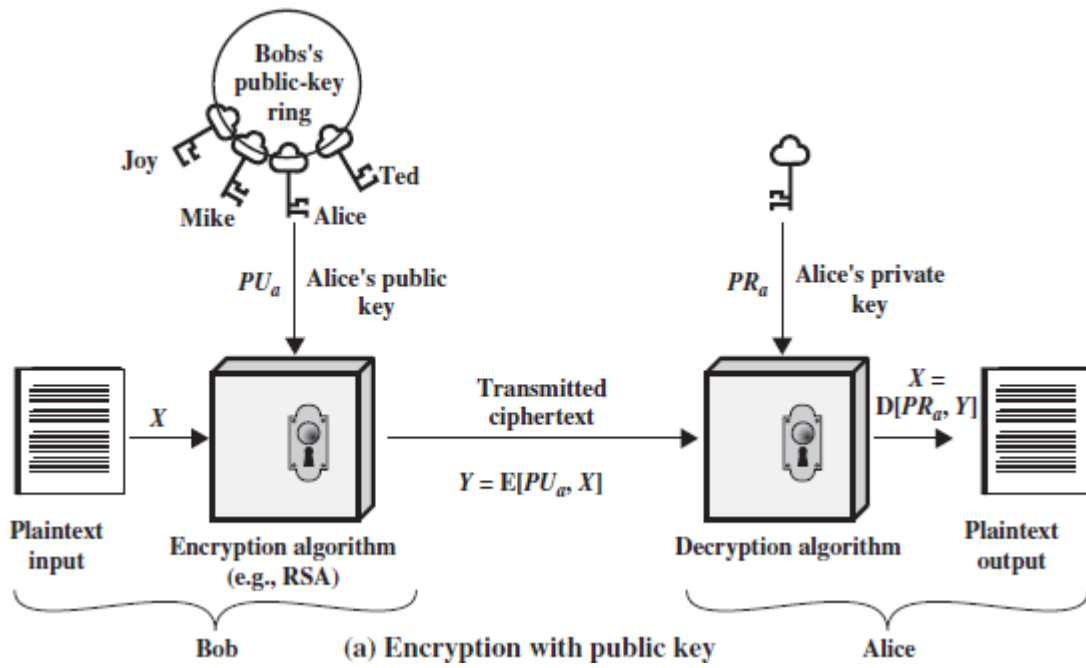


Figure 11 : Encryption with public key

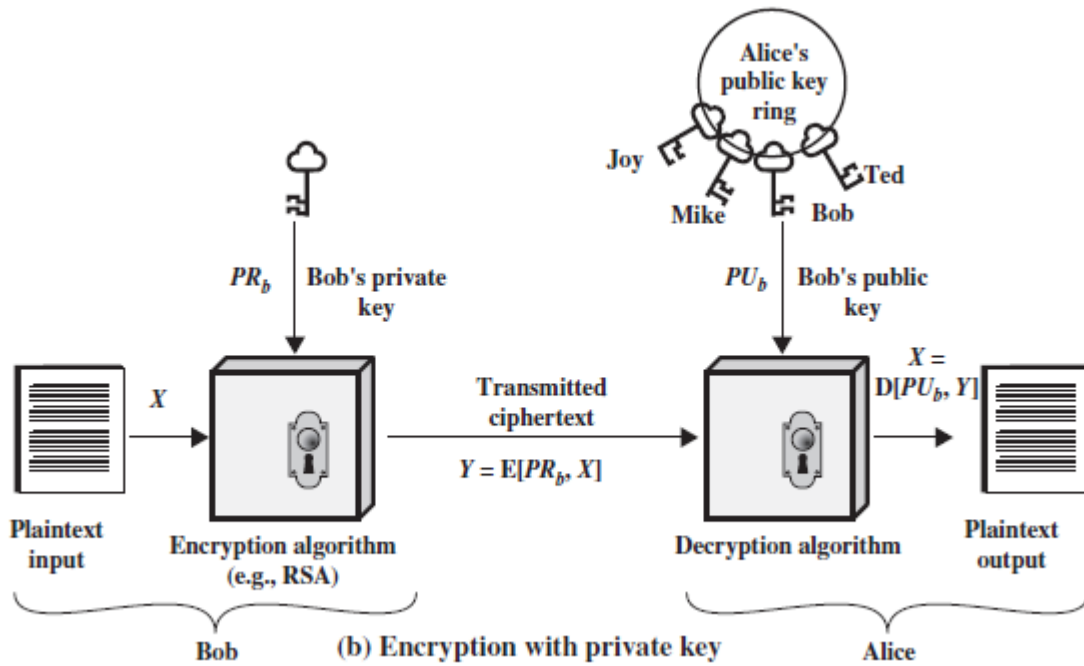


Figure 12: Encryption with private key

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Helman	No	No	Yes
Elliptic Curve	Yes	Yes	Yes
DSS	No	Yes	No

Table 5: Comparison of different public key cryptography algorithm

Chapter-7(System Model)

7.1 Scope

The project is related to the secure authentication in the cloud. This model makes sure that security services such as secrecy, authentication, integrity and unlinkability are provided to the communicating parties.

7.2 Objective

This project has been developed keeping in view the security features that need to be implemented in the authentication process following the fulfillment of these objectives:

- **Anonymity**- Every honest user stays anonymous when uses cloud services. User identities are hidden if users behave honestly and do not break rules.
- **Confidentiality**-Every user's session to CSP is confidential. No one without a secret session key is able to obtain data transmitted between User and CSP.

7.3 Model of System

The solution consists of two fundamental parties:

- **Cloud Service Provider (CSP)**- CSP manages cloud services and shared storages. CSP usually behaves as a trusted party. CSP provides cloud services , authenticates users when they access a cloud service. Nevertheless , when CSP finds that the identity of the user is not correct or that there is a malicious user, then CSP can revoke the rights of the user.
- **User (U)** - U is an ordinary customer who accesses into a cloud and uses cloud services, shared storages, etc. Users are anonymous if they properly follow the rules of CSP.

7.4 Architecture

This proposed architecture is enhanced security model for cloud services like data storage. It consist of CSP i.e. Cloud Service Provider and user's.

- **Cloud Service Provider** - CSP generates a pair of keys i.e. secret key and public key by using cryptography algorithm RSA. CSP stores its own private key generated RSA algorithm and public key is shared by all.
- **User's**- User's must physically register on CSP. CSP check user's id. If user is already registered then there are some messages exchanged between users and CSP for establishing secure communication between them. User's generated a random request and encrypts this random request with the RSA public key of CSP and sends this request message to the server. Now server verifies this request if it is verified then server decrypt this message by using RSA secret key of CSP and generate a random key K_{sym} . Now with the help of this random key server generates a response by applying the X-OR operation to random and K_{sym} . Server send this response to user's. User's use this random key for file uploading and downloading and attain secure communication

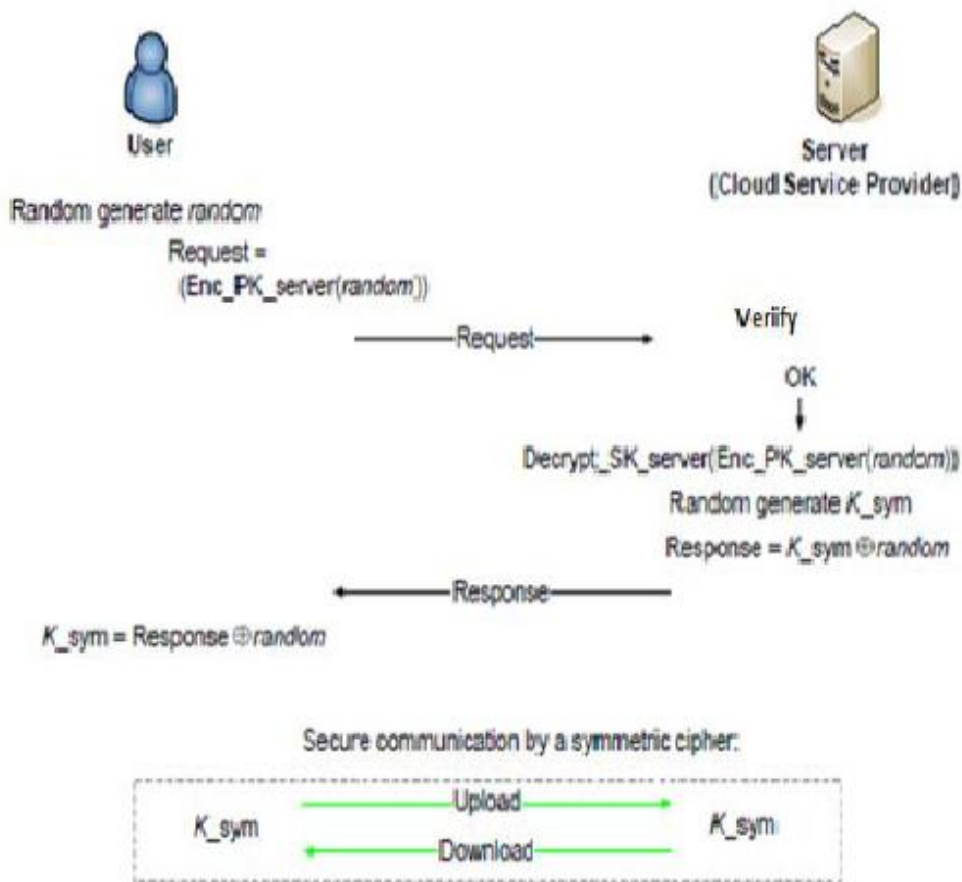


Figure 13: Architecture

7.5 Basic Algorithm

This is the algorithm on which the project is based on:

- **Key Generation** - The CSP generates the random key pair (Pk, Sk). Two large prime numbers P, Q are generated such that $N=PQ$ (N is 1024 bits) and $\phi=(P-1)(Q-1)$. Select a random number $1 < e < \phi$ such that $\gcd(e, \phi) = 1$. Compute the unique integer $d = e^{-1} \pmod{\phi}$. Where Pk (e, N) is the Public Key and Sk (d, N) is the Secret Key.
- **Secure Communication**- RSA provides the secure communication between client and server. Users encrypts random by the RSA public key of CSP. The encrypted Enc PK server (random) is send to the server. Server verifies the encrypted Enc PK server (random) if it verify then the server decrypt this with its secret key and generate random key K_{sym} . It computes the Response = K_{sym} random. CSP sends a response message (random K sym) back to user.
- **File Upload and Download**-The user can upload and download data to CSP. Data privacy are secured by a symmetric cipher. I am using AES which is well known cipher and is supported by many types of software and hardware platforms. To encrypt and decrypt transmitted data, User and CSP use the AES secret key K_{sym} established in the previous phase.
- **Revocation**- Depending on the case of rule breaking, the revocation phase can revoke a user and/or user anonymity. If users misuse a cloud service, they get revoked by CSP.

7.6 UML Diagrams

7.6.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

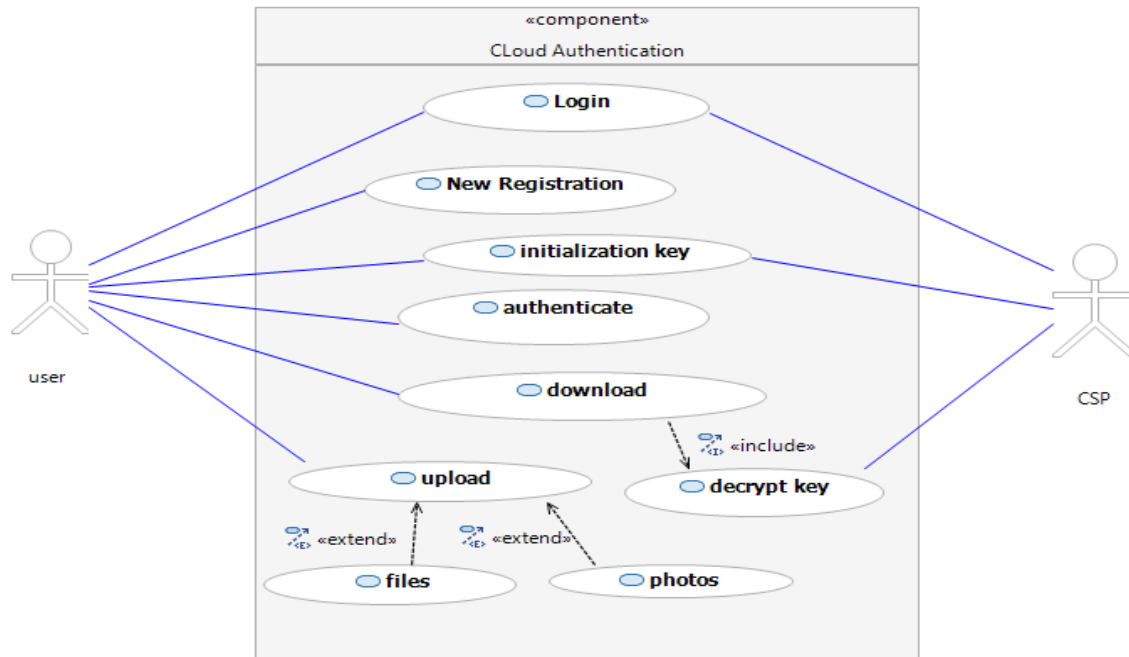


Figure 14: Use Case Diagram

7.6.2 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

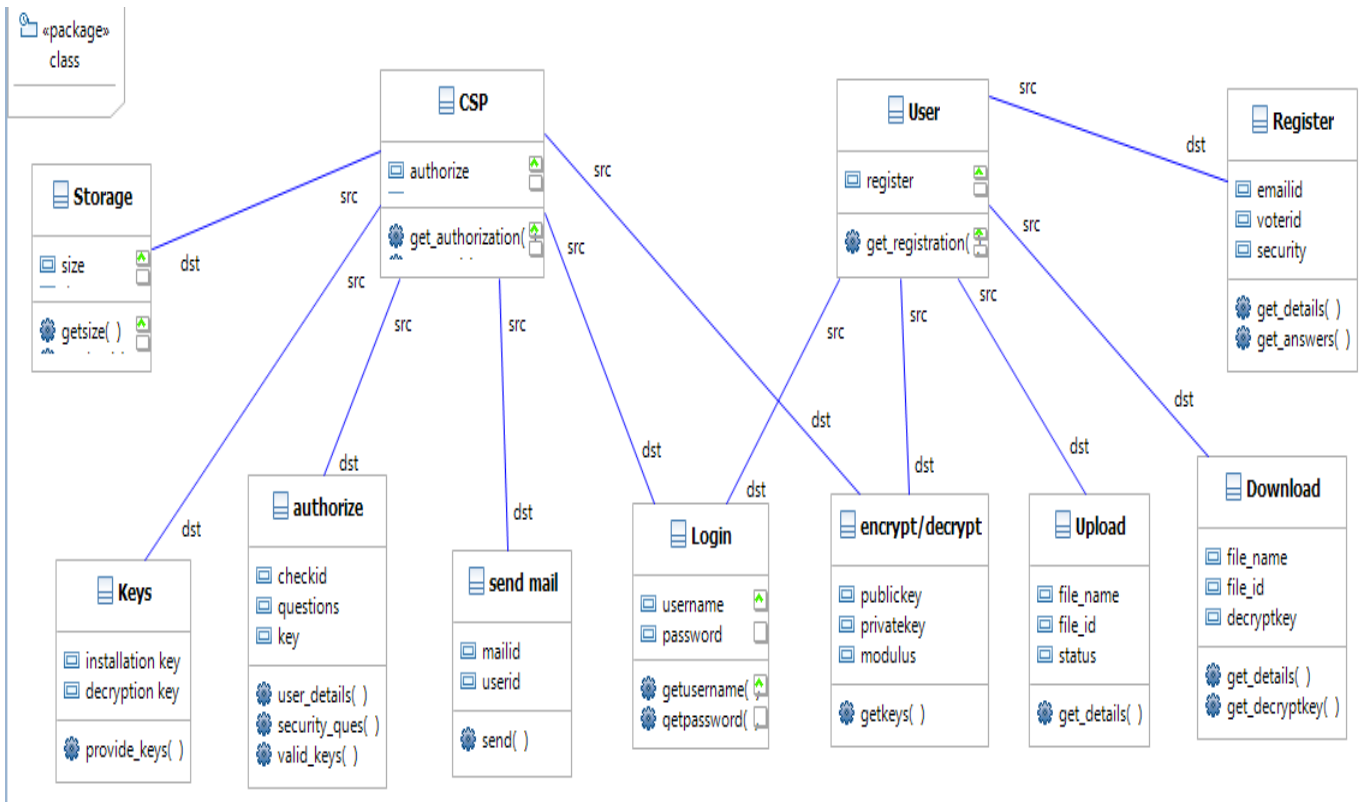


Figure 15: Class Diagram

7.6.3 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

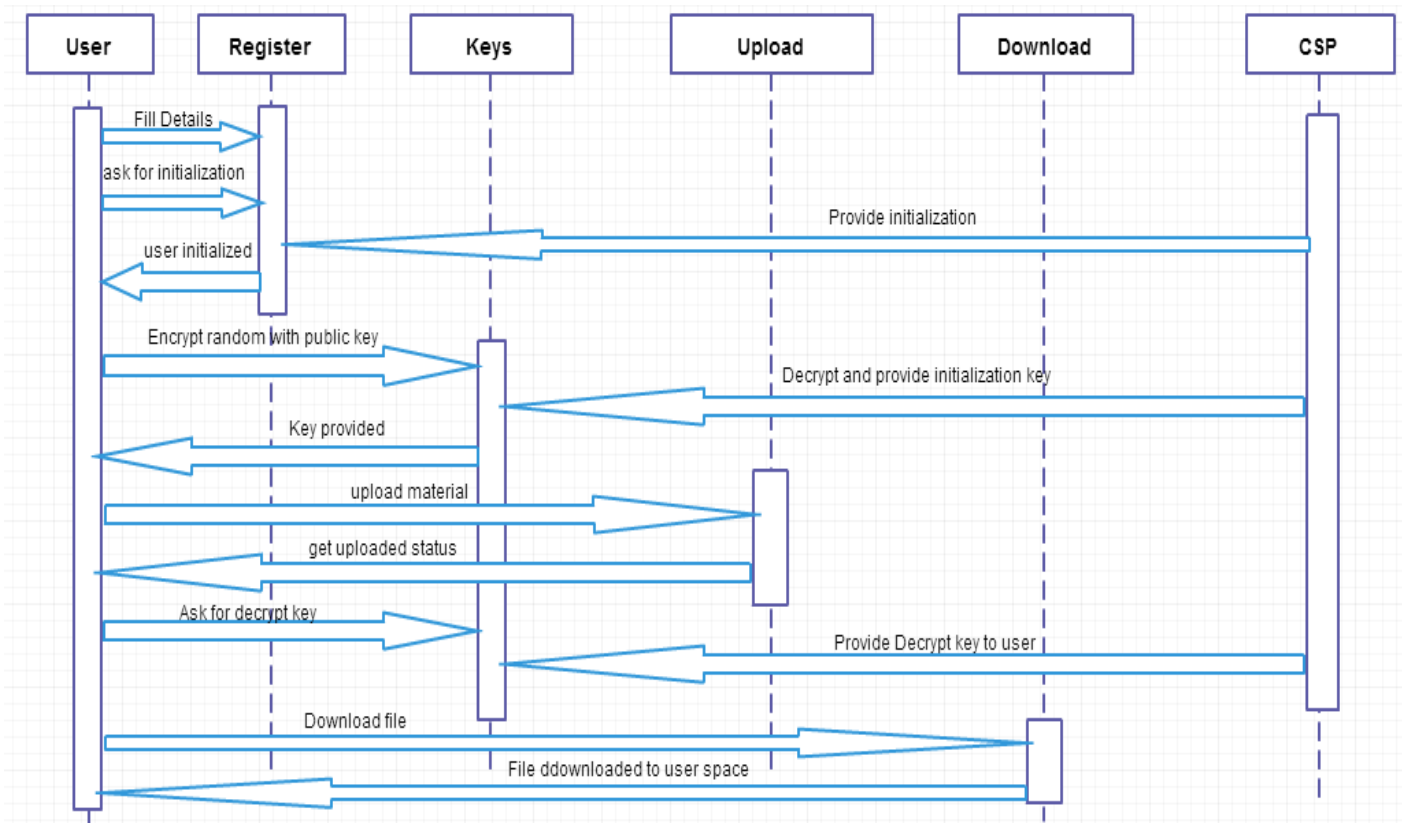


Figure 16: Sequence Diagram

7.7 Cryptographic Algorithms Used

7.7.1 RSA Algorithm

RSA is a public key algorithm based on difficulty of prime factorization. Similar to most public keys password features, RSA password is block encrypted, but different from block encryption used in a secret key algorithm such as DES (Data Encryption Standard) in the sense that a length of a plain text and a key is variable; in other words, a relatively long length of a key can be used in consideration of the secure and reliable system, and a relatively short length of a key can be used for efficient system. In an RSA algorithm, receiver's public key is used to encrypt messages, and the receiver decrypts encrypted messages with its own private key. In an RSA-based signature algorithm, a sender signs messages with its own private key, and a receiver verifies the signed message with sender's public key; this is how an authentication service is guaranteed through an RSA-based signature algorithm. The algorithm is:

7.7.1.1 Key Generation Algorithm

RSA public and private key pair can be generated by the following procedure. Choose two random prime numbers p and q such that the bit length of p is approximately equal to the bit length of q .

The key set is generated by using the following algorithm:

- Select two large prime numbers p and q such that $p \neq q$.
- Compute modulus $n = p * q$
- Compute $\phi(n)$ such that $\phi(n) = (p-1) * (q-1)$.
- Choose a random integer e satisfying $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$
- Compute the integer d , such that $e * d = 1 \pmod{\phi(n)}$. (n, e) is the public key, and (n, d) is the private Key
- n is known as the modulus.
- e is known as the public exponent or encryption exponent or just the exponent.
- d is known as the secret exponent or decryption exponent.

7.7.1.2 Encryption

Encryption refers to algorithmic schemes that encode plain text into non-readable form or cipher text, providing privacy. Encryption is done by using the following steps:

- Obtain the recipient's public key (n, e) .
- Represent the plaintext message as a positive integer m .
- Compute the cipher text $c = m^e \pmod{n}$.
- Send the cipher text c to receiver.

7.7.1.3 Decryption

Decryption refers to algorithmic schemes that decode cipher text or non-readable text into readable form or plain text. Message is decrypted by using the following steps:

- Receiver uses his own private key (n, d) to compute $m = c^d \pmod{n}$.
- Extracts the plaintext from the integer representative m .

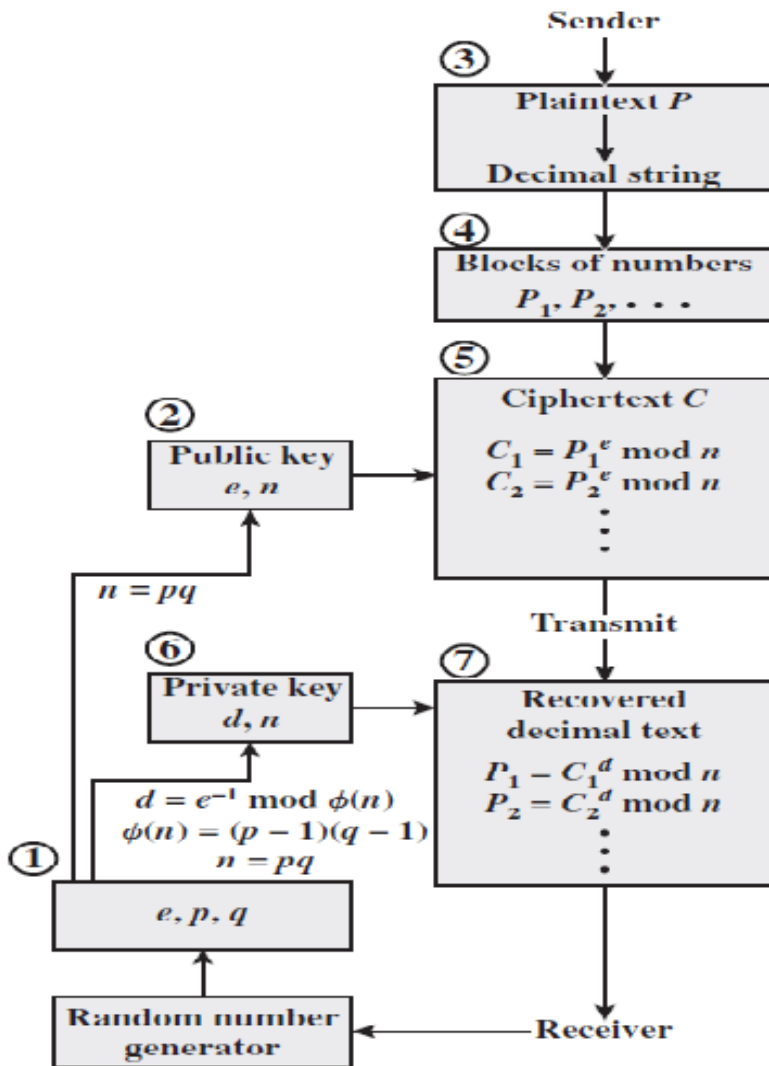


Figure 17: Rsa algorithm

7.7.2 AES Algorithm

AES is based on a design principle known as a substitution permutation network, combination of both substitution and permutation, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field. The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext.

The number of cycles of repetition are as follows: **10 cycles** of repetition for **128-bit keys**, **12 cycles** of repetition for **192-bit keys**, **14 cycles** of repetition for **256-bit keys**.

Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key. The main parts of AES are:

KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule . AES requires a separate 128-bit round key block for each round plus one more.

Different Rounds- These rounds have 4 different processing steps:

SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.

ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.

AddRoundKey- each byte of the state is combined with a block of the round key using bitwise xor.

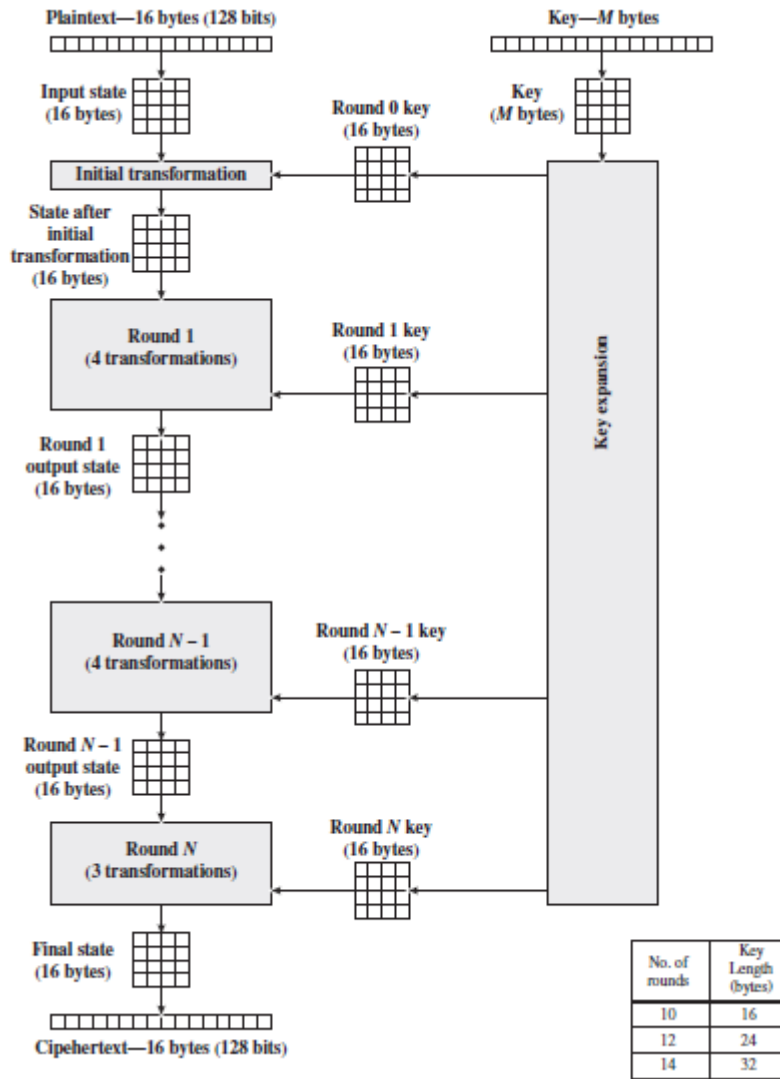


Figure 18: AES Encryption process

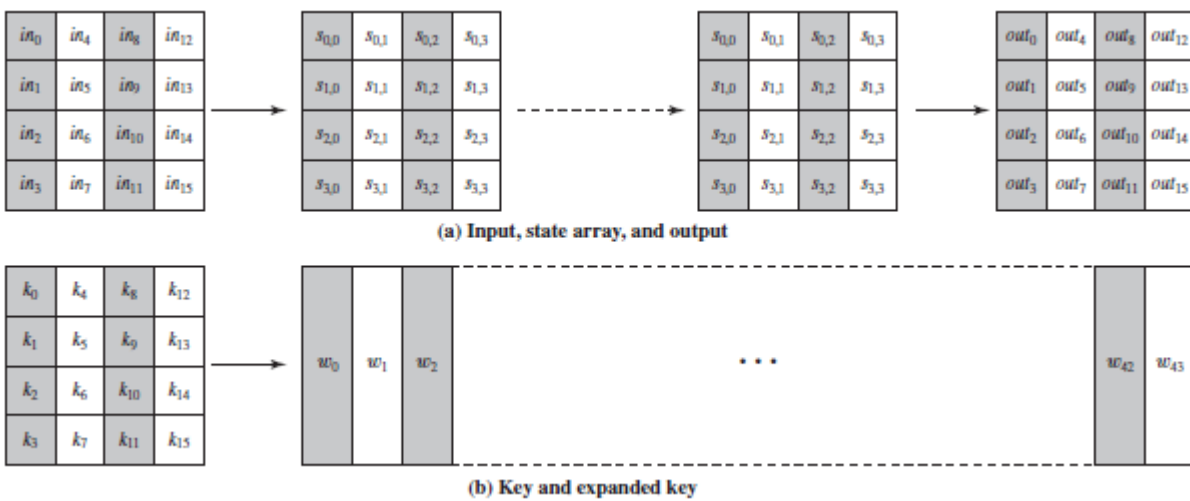


Figure 19: AES Data Structures

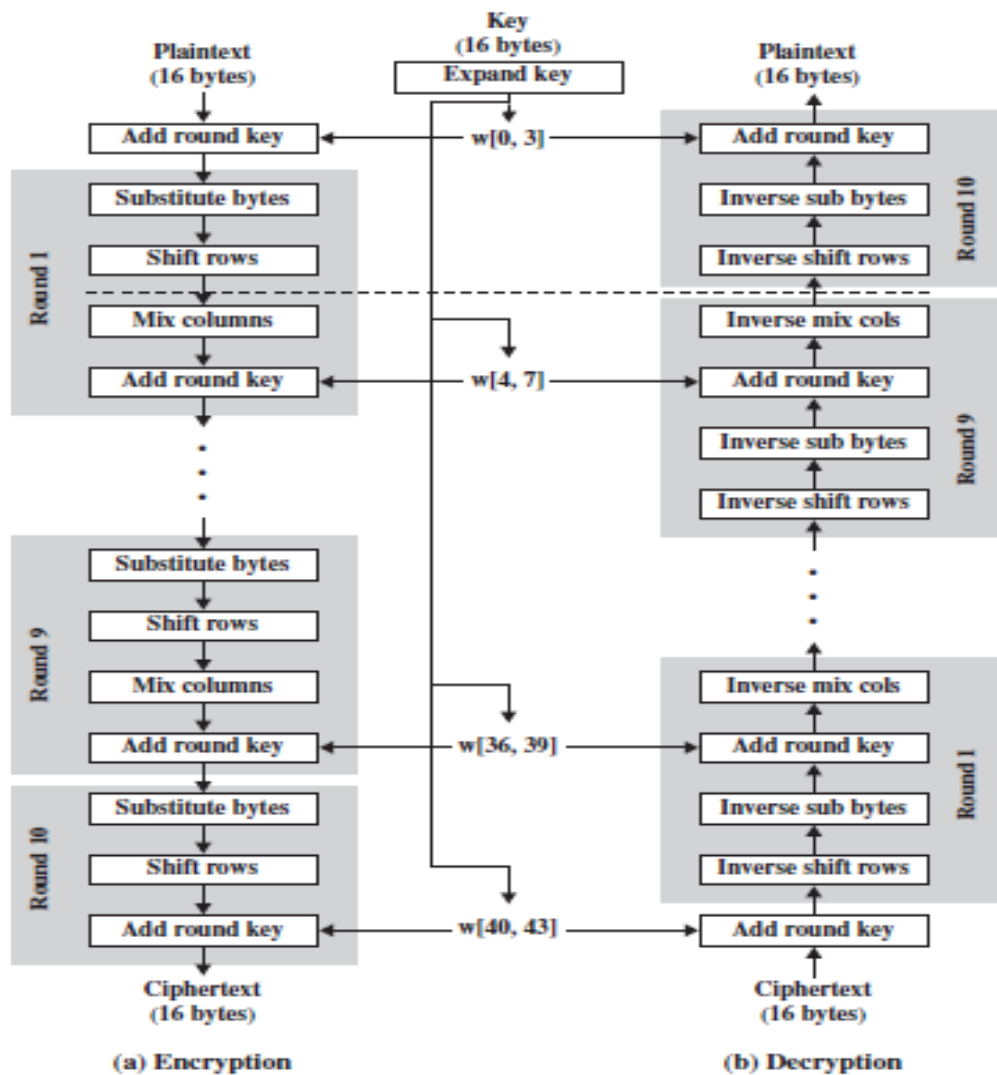


Figure 20: AES Encryption and Decryption

7.7.3 Hybrid Cryptosystem

In cryptography, public-key cryptosystems are convenient in that they do not require the sender and receiver to share a common secret in order to communicate securely (among other useful properties). However, they often rely on complicated mathematical computations and are thus generally much more inefficient than comparable symmetric-key cryptosystems. In many applications, the high cost of encrypting long messages in a public-key cryptosystem can be prohibitive. A hybrid cryptosystem is one which combines the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem.

A hybrid cryptosystem can be constructed using any two separate cryptosystems:

- A key encapsulation scheme, which is a public-key cryptosystem, and
- A data encapsulation scheme, which is a symmetric-key cryptosystem.

The hybrid cryptosystem is itself a public-key system, whose public and private keys are the same as in the key encapsulation scheme. For very long messages the bulk of the work in encryption/decryption is done by the more efficient symmetric-key scheme, while the inefficient public-key scheme is used only to encrypt/decrypt a short key value.

In this project I have used hybrid cryptography by using RSA and AES algorithms.

AES is used as a symmetric cipher which generates a symmetric key and encrypts the data of the files that are uploaded on the cloud. When the user wants to download the files then symmetric key is used to decrypt the data.

RSA algorithm is used as a public key cipher which generates public and private keys and encrypts the symmetric key of AES using its public key and uses the private key to decrypt the encrypted symmetric key of AES.

7.8 Java Cryptography Extension(JCE)

The Java Cryptography Extension (JCE) is an officially released Standard Extension to the Java Platform. JCE provides a framework and implementation for encryption, key generation and key agreement, and Message Authentication Code(MAC) algorithms. JCE supplements the Java platform, which already includes interfaces and implementations of message digests and digital signatures.

The classes that I have used in my project to encrypt and decrypt files are:

- **KeyPairGenerator** : The KeyPairGenerator class is used to generate pairs of public and private keys. Key pair generators are constructed using the getInstance factory methods (static methods that return instances of a given class). A Key pair generator for a particular algorithm creates a public/private key pair that can be used with this algorithm. It also associates algorithm-specific parameters with each of the generated keys.
- **SecureRandom**: This class provides a cryptographically strong random number generator (RNG).
- **KeyPair**: This class is a simple holder for a key pair (a public key and a private key). It does not enforce any security, and, when initialized, should be treated like a PrivateKey.
- **KeyGenerator**: This class provides the functionality of a secret (symmetric) key generator. Key generators are constructed using one of the getInstance class methods of this class. KeyGenerator objects are reusable, i.e., after a key has been generated, the same KeyGenerator object can be re-used to generate further keys.
- **X509EncodedKeySpec** : This class represents the encoding of a public key.

- **KeyFactory** : Key factories are used to convert keys (opaque cryptographic keys of type `Key`) into key specifications (transparent representations of the underlying key material), and vice versa. Key factories are bi-directional. That is, they allow you to build an opaque key object from a given key specification (key material), or to retrieve the underlying key material of a key object in a suitable format.
- **Cipher**: This class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. In order to create a Cipher object, the application calls the Cipher's `getInstance` method, and passes the name of the requested transformation to it. Optionally, the name of a provider may be specified. A transformation is a string that describes the operation (or set of operations) to be performed on the given input, to produce some output. A transformation always includes the name of a cryptographic algorithm (e.g., *DES*), and may be followed by a feedback mode and padding scheme.
- **PKCS8EncodedKeySpec** : This class represents the encoding of a private key.

Chapter-8 (System Requirements)

8.1 Hardware Requirements

- Hard disk: - 40GB
- RAM: - 1 GB
- Processor: - p4
- Multimedia Key Board
- LG Monitor
- Mouse-two or three buttoned mouse

8.2 Software Requirements

- Operating Systems: **WINDOWS**
- Technologies Used: Java , jsp
- Web Technologies Used: HTML, CSS, Javascript
- Database: MySQL
- Application Server: Apache Tomcat
- IDE: Netbeans 8.0.2

Chapter-9 (Snapshots of project)

9.1 New User Login Page



New User

First Name:	<input type="text"/>
Last Name:	<input type="text"/>
Password:	<input type="password"/>
Gender:	<input type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Others
Email-Id:	<input type="text"/>
Voter-Id:	<input type="text"/>
Birthday:	Month <input type="text"/> Day <input type="text"/> Year <input type="text"/>
	<input type="button" value="Submit"/>

Figure 21

9.2 Homepage

cloud computing security: anonymous authentication

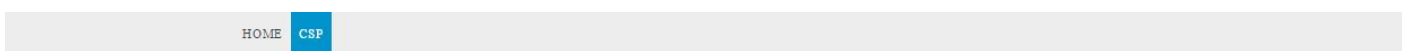


Welcome to Cloud Computing Security

The use of cloud computing has increased rapidly in many organizations. Cloud computing provides many benefits in terms of low cost and accessibility of data. Ensuring the security of cloud computing is a major factor in the cloud computing environment, as users often store sensitive information with cloud storage providers but these providers may be untrusted. Dealing with cloud is predicted to become less popular with customers due to risks of availability failure and the possibility of malicious attackers watching the cloud. A movement towards providing anonymous authentication in cloud has emerged recently. The main aim is to provide safe authentication to reduce security risks that affect the cloud computing user.

Figure 22

9.3 CSP Login Page



cloud computing security: anonymous authentication



CSPLogin

User id:	<input type="text"/>
Password:	<input type="password"/>
	<input type="button" value="Login"/>

Figure 23

9.4 CSP Homepage



cloud computing security: [anonymous authentication](#)



Welcome Admin

Figure 24

9.5 CSP Initialization Page



cloud computing security: [anonymous authentication](#)



User name emailid Voter-id Request Provide

Figure 25

9.6 User Login Page

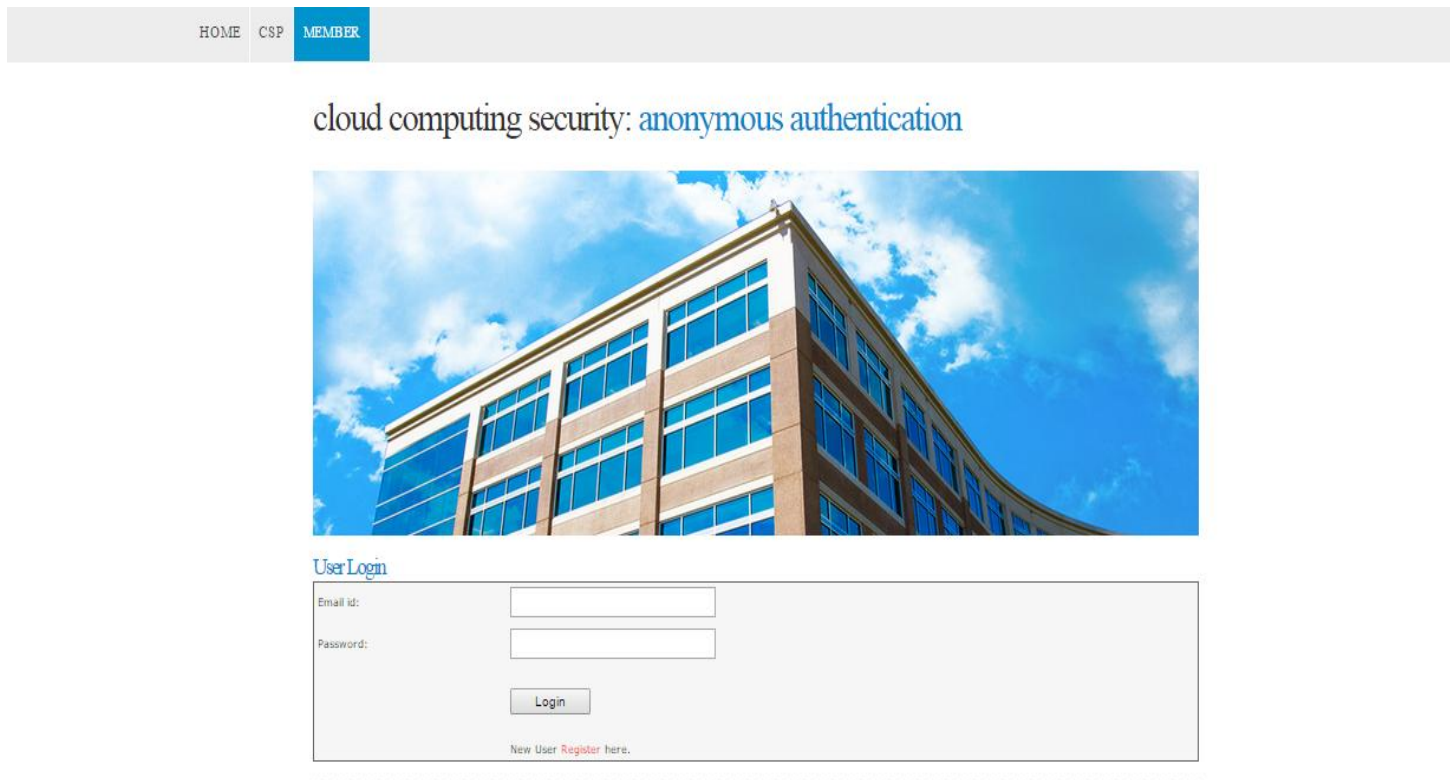


Figure 26

9.7 User Homepage

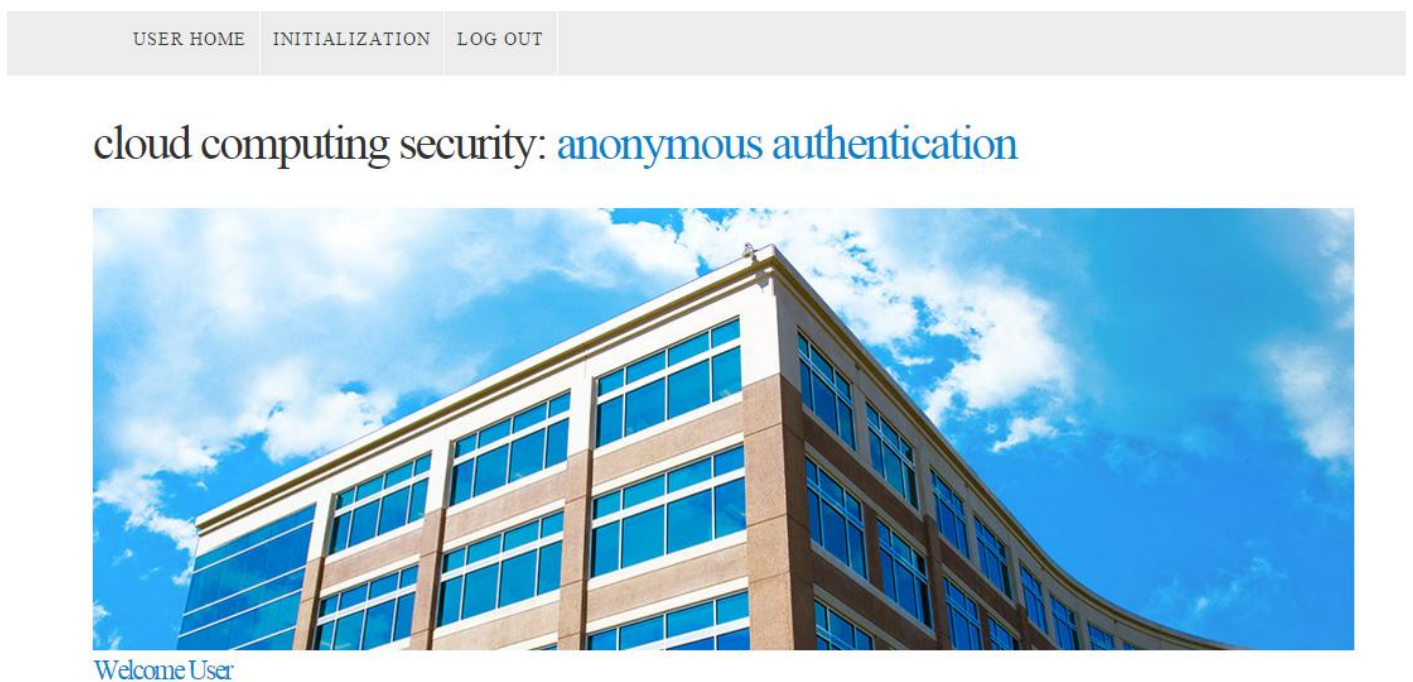


Figure 27

9.8 User Initialization Page

cloud computing security: [anonymous authentication.](#)



Enter the initialization key

Initialization key:	<input type="text"/>
	<input type="submit" value="Submit"/>

Figure 28

9.9 Initialization Steps

cloud computing security: [anonymous authentication.](#)



Please Fill in the initialization steps


Name:	<input type="text"/>
Password:	<input type="text"/>
Email-Id:	<input type="text"/>
Voter-Id:	<input type="text"/>
	<input type="submit" value="Continue"/>

Figure 29

9.10 Security Questions

USER HOME INITIALIZE LOG OUT

cloud computing security: anonymous authentication.



Answer your Security Questions

Enter your first trip?


Enter your birth place?

Figure 30

9.11 Upload a File Page

USER HOME VIEW DETAILS UPLOAD DOWNLOAD LOG OUT

cloud computing security: anonymous authentication



File Upload

First Name:

Last Name:

Choose File: No file chosen

Figure 31

9.12 Download A File

USER HOME VIEW DETAILS UPLOAD DOWNLOAD LOG OUT

cloud computing security: anonymous authentication



Download your file

File id:

Figure 32

9.13 Decrypting the key

cloud computing security: anonymous authentication.



Enter the decryption key

Decrypt key:

9.14 Viewing the files uploaded in cloud

USER HOME	VIEW DETAILS	UPLOAD	DOWNLOAD	LOG OUT
-----------	--------------	--------	----------	---------

cloud computing security: **anonymous authentication**



Files present on the cloud...

File id:	<input type="text" value="1"/>
First Name:	<input type="text" value="matrix"/>
Last Name:	<input type="text" value=".txt"/>
File in encrypted form:	<pre>PK!0?(r?[Content_Types].xml ?(??T?n?0?w??D? V?????[[?0??z??]?I?Q?B? \%?????????1 ?w%?=???^i7+???%?g&??0?A?6? l4??L60#?????S ????x?&??V\$z?3??????%p)0S?????5]??:??"? e?????V?DFK???)_?:?K%?)?l???\!??U??"?v??&j?X??</pre>

Figure 34

Chapter-10

Conclusion

This project presents the solution which offers user anonymity in authentication phase and confidentiality during file uploading and downloading for all users. The project deals with user's anonymous access to cloud services and shared storage servers. The project provides registered users with anonymous access to cloud services. This project provides secure transfer of data between the user and the cloud by encrypting the files that are uploaded by the user in transit and saving the encrypted files in the cloud. This project also provides unlinkability of user's sessions as user is initialized to the cloud every time the user accesses the cloud by sending random initialization keys to the user's email id. Secure downloading of users files is done by decrypting the files of user based on user's random decrypt keys. The authentication phase will be more efficient than related solutions on the client side and also on the server side due to missing expensive bilinear pairing operations and fewer exponentiation operations. Due to this fact, cloud service providers using this solution can authenticate more clients at the same time.

References

1. Published in International general of Cloud Computing:

<http://arxiv.org/ftp/arxiv/papers/1309/1309.2426.pdf>

2. Published in IJCSI international general of Computer Science:

<http://www.ijcsi.org/papers/IJCSI-11-3-1-145-149.pdf>

3. Published in IEEE conference (Nov 2013) www.chennaisunday.com/

4. Published in IEEE conference (Feb 2014) <http://ieeexplore.ieee.org/>

Appendix

A) Sending mail to the user

```
import com.util.DbConnector;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.Random;
import com.sun.mail.smtp.SMTPTransport;
import java.sql.Statement;
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.URLName;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
public class Mail extends HttpServlet {

    public static boolean sendingMail(String msg,String query) {
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
```

```
props.put("mail.smtp.port", "465");
```

```
Session session1 = Session.getDefaultInstance(props,  
new javax.mail.Authenticator() {  
protected PasswordAuthentication getPasswordAuthentication() {  
return new PasswordAuthentication("shubhanshugupta52@gmail.com", "horrendous");  
}  
});
```

```
// System.out.println("Message " + msg);  
try {  
String userid="shubhanshugupta52";  
String to=query;  
Message message = new MimeMessage(session1);  
message.setFrom(new InternetAddress(userid));  
message.setRecipients(Message.RecipientType.TO,  
InternetAddress.parse(to));  
message.setSubject("Encrypted Key");  
message.setText(msg);  
  
Transport.send(message);  
  
System.out.println("Done");  
return true;  
  
} catch (MessagingException e) {  
System.out.println("Mailing Error " +e);  
return false;  
// throw new RuntimeException(e);  
}  
}
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
throws ServletException, IOException {  
response.setContentType("text/html;charset=UTF-8");
```

```

PreparedStatement pst = null;
Connection conn = null;
ResultSet rs=null;
Statement stmt=null;
String query=null;
Random randomGenerator = new Random();
int min=700000;
int randomInt =min+randomGenerator.nextInt(100000);
String message="Your initialization key is "+randomInt;
HttpSession session=request.getSession(false);
    String email=(String)session.getAttribute("mail");
try{
    conn=(Connection) DbConnector.getConnection();
    stmt=conn.createStatement();
    rs=stmt.executeQuery("select emailid from user where emailid="" +email+""");
    while(rs.next())
        query=rs.getString("emailid");
    sendingMail(message,query);

    String sql="insert into initialize values(""+email+"",""+randomInt+"");
    //System.out.println(">>" +sql);
    pst = (PreparedStatement) conn.prepareStatement(sql);
    int a=pst.executeUpdate();
    if(a >0){

        response.sendRedirect("userHome.jsp?msg=successfully logged in");
    }
    else
    {
        response.sendRedirect("userLogin.jsp?msg=check mailid or password");
    }
}
catch(Exception e){
    e.printStackTrace();
}
}

```

B) Uploading a file in encrypted form

```
import com.util.DbConnector;
import com.util.Constant;
import com.util.Utilities;
import com.util.cryptography;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.GeneralSecurityException;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

@WebServlet("/UploadActions4")
@MultipartConfig(fileSizeThreshold=1024*1024*2, // 2MB
                 maxFileSize=1024*1024*10, // 10MB
                 maxRequestSize=1024*1024*50) // 50MB
public class UploadActions4 extends HttpServlet {
    private static final String SAVE_DIR = "uploadsFiles";
```

```

protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException, IOException,
FileNotFoundException {

    String appPath = request.getServletContext().getRealPath("");

    String savePath = appPath + File.separator + SAVE_DIR;
    String firstname = request.getParameter("firstName");
    String lastname = request.getParameter("lastName");
    String last=".enc";
    Part filePart = request.getPart("file"); // Retrieves <input type="file" name="file">
    String fileName = getFileName(filePart);
    InputStream inputStream = null;
    cryptography.generateKey("RSA_private.key", "RSA_public.key");

    File fileSaveDir = new File(savePath);
    if (!fileSaveDir.exists()) {
        fileSaveDir.mkdir();
    }

    for (Part part : request.getParts()) {
        //String filename = getFileName(part);
        part.write(savePath + File.separator + fileName);
    }

    String pathfile=savePath + File.separator + fileName ;
    String path=pathfile+last;
    File file=new File(pathfile);
    String file1=file.getAbsolutePath();
    File files=new File(pathfile+last);
    String file2=files.getAbsolutePath();
    System.out.println(file2);
    String str=new String();
    try {
        str= cryptography.pubkeys();
    } catch (ClassNotFoundException ex) {

```

```

    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
} catch (GeneralSecurityException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
}
try {
    str= cryptography.privkeys();
} catch (ClassNotFoundException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
} catch (GeneralSecurityException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
}
try {
    cryptography.encryptToOutputFile(str, file1, file2);
} catch (ClassNotFoundException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
} catch (GeneralSecurityException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
}

if (filePart != null) {

        System.out.println(filePart.getName());
        System.out.println(filePart.getSize());
        System.out.println(filePart.getContentType());

        //inputStream = filePart.getInputStream();
    }
    inputStream= new FileInputStream(pathfile+last);
    Connection con = null;
String message = null;
HttpSession session=request.getSession(false);
String email=(String)session.getAttribute("mail");
    try {
        con = DbConnector.getConnection();

```

```

String sql = "INSERT INTO upload1 (first_name, last_name, file) values (?, ?, ?)";
    PreparedStatement statement = con.prepareStatement(sql);
    statement.setString(1, firstname);
    statement.setString(2, lastname);
    if (inputStream != null) {
        statement.setBinaryStream(3, inputStream,(int)path.length());
        //statement.setBlob(3, inputStream);
    }
    int row = statement.executeUpdate();
    if (row > 0) {
        message = "File uploaded and saved into database";
    }
} catch (SQLException ex) {
    message = "ERROR: " + ex.getMessage();
    ex.printStackTrace();
} finally {
    if (con != null) {
        try {
            con.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    request.setAttribute("Message", message);
    getServletContext().getRequestDispatcher("/success.jsp").forward(request, response);
}
}

```

```

private static String getFileName(Part part) {
for (String cd : part.getHeader("content-disposition").split(";")) {
    if (cd.trim().startsWith("filename")) {
        String fileName = cd.substring(cd.indexOf('=') + 1).trim().replace("\"", "");
        return fileName.substring(fileName.lastIndexOf('/') + 1).substring(fileName.lastIndexOf('\\') + 1);
    }
}
}

```



```
    return null;
}
}
```

C) Downloading a File

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.File;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import com.util.DbConnector;
import com.util.Constant;
import com.util.cryptography;
import java.io.FileInputStream;
import java.security.GeneralSecurityException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/Download1")
public class Download1 extends HttpServlet {

    private static final int BUFFER_SIZE = 4096;
    private static final String SAVE_DIR = "uploadsFiles";
```

```

protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String appPath = request.getServletContext().getRealPath("");

    String savePath = appPath + java.io.File.separator + SAVE_DIR;
    HttpSession session=request.getSession(false);
    int uploadId = (int)session.getAttribute("Id");
    String key=(String)request.getParameter("_key");
    Connection conn = null;
    PreparedStatement pst = null;
    ResultSet rs=null;
    try {

        conn = DbConnector.getConnection();

        pst = (PreparedStatement) conn.prepareStatement("select * from upload1 where file_id=" + uploadId
+ "");
        rs =(ResultSet) pst.executeQuery();

        if (rs.next()) {
            // gets file name and file blob data
            String fileName = rs.getString("first_name");
            Blob blob = rs.getBlob("file");
            String pathfile=savePath + File.separator + fileName ;
            byte[] data = blob.getBytes(1, (int)blob.length());
            String file1= new String(data);
            InputStream inputStream = null;

            File file= new File(pathfile + ".dec");
            String file2=file.getAbsolutePath();

            ServletContext context = getServletContext();
            String mimeType = context.getMimeType(fileName);
            if (mimeType == null) {
                mimeType = "application/octet-stream";
            }
        }
    }
}

```

```

response.setContentType(mimeType);

String str=new String();
try {
str= cryptography.privkeys();
} catch (ClassNotFoundException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
} catch (GeneralSecurityException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
}
try {
    cryptography.decryptFromOutputFile(str, file1, file2);
} catch (ClassNotFoundException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
} catch (GeneralSecurityException ex) {
    Logger.getLogger(UploadActions4.class.getName()).log(Level.SEVERE, null, ex);
}
inputStream= new FileInputStream(pathfile + ".dec");
int fileLength = inputStream.available();
response.setContentLength(fileLength);
String headerKey = "Content-Disposition";
String headerValue = String.format("attachment; filename=\"%s\"", fileName);
response.setHeader(headerKey, headerValue);

OutputStream outputStream = response.getOutputStream();

byte[] buffer = new byte[BUFFER_SIZE];
int bytesRead = -1;

while ((bytesRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, bytesRead);
}

```

```

        inputStream.close();
        outputStream.close();
    } else {
        // no file found
        response.getWriter().print("File not found for the id: " + uploadId);
    }
} catch (SQLException ex) {
    ex.printStackTrace();
    response.getWriter().print("SQL Error: " + ex.getMessage());
} catch (IOException ex) {
    ex.printStackTrace();
    response.getWriter().print("IO Error: " + ex.getMessage());
} finally {
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
}
}
}
}

```