Securing Storage in Cloud Computing

Synopsis submitted in partial fulfillment of the requirement for the degree of



B. Tech.

In

###### Computer Science and Engineering

Under the Supervision of

##### Mr. Pankaj Singh Yadav

By:

**Ansh Sachdeva (41325502716)**

Nitin Kumar (41725502716)



**JIMS Engineering Management Technical Campus, Greater Noida**

**Guru Gobind Singh Indraprastha University, Delhi**

Session: 2019-2020

**DECLARATION**

This is to certify that Synopsis Report Entitled “Secure Authentication in cloud computing” which is submitted in partial fulfillment of the requirement for the award of degree B.Tech. in Computer Science and Engineering to JIMSEMTC, Greater Noida affiliated to Guru Gobind Singh Indraprastha University, Delhi comprises only original work and studies carried out by students himself. The matter embodied in this synopsis has not been submitted for the award of any other degree.

#### Date: 11/9/19 Ansh Sachdeva (41325502716)

*Nitin Kumar (41725502716)*

#### 

## 

## 

## 

## **Approved By:**

## 

## 

|  |  |  |
| --- | --- | --- |
| **Signature** **Mr. Pankaj Singh Yadav****Project Guide****CSE Department JIMSEMTC, Gr.Noida** |  | **Signature** **Dr. Vikas Chaudhary****HOD****CSE Department****JIMSEMTC, Gr.Noida** |

**ACKNOWLEDGEMENT**

I express my sincere gratitude to Dr. Vikas Chaudhary (HOD, CSE) and Mr. Pankaj Singh Yadav (A.P, CSE), for his valuable guidance and timely suggestions during the entire duration of my dissertation work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members of school of IT, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally, I would also like to thank my friends for their advice and pointing out my mistakes.

**CONTENTS**

Abstract ........................................................................................................... 05

List of Figures ................................................................................................. 06

List of Tables ................................................................................................... 07

1. Introduction ..................................................................................................09

* 1. Background ..........................................................................................09
  2. Problem Statement ...............................................................................11
  3. Purpose and Goals ...............................................................................12

1. Literature Survey…………………………………………………..……...13
   1. Introduction …………………………………………….…….………13
   2. Review ………………………………………..………………………13
2. Real World Applications ………………………………..………………...14
   1. Details of application …………………………………….……………14
3. Inferences from literature ………………………………..............................15

4.1. Overview of Cloud Architecture Model ...........................................15

4.2. Security System Design Approach....................................................16

4.3. Overview of the System Entities ......................................................17

4.3.1. Authentication System……………………………………18

4.3.2. Authentication System Security Protection .......................21

4.4 Authorization System .......................................................................21

4.5 Authorization System Security Protection ........................................21

1. Requirements ………………………………………………………………22

5.1 Hardware………………………………………………………………..22

5.2 Software ……………………………………………………………..…22

6. Problem Statement and solution……………………………………………23

6.1 Proposed work …………………...……...………………………..23

6.2 System Design and Methodology…………………………………24

6.3 Implementation …………………………………………………...26

6.4 Sample/Screenshot…………………………………………...……28

6.5 Performance Metrics ………………………………………...........32

7. Proposed Time bound progress ……………………………………………..33

7.1 Progress Sheet ………………………………………………………33

8. Conclusion and Future work ………………………………………………..34

8.1 Conclusion …………………………………………………………34

8.2 Future work ……………………………………………………35

9. References …………………………………………………………………..36

**ABSTRACT**

During the last two decades, the use of internet has been changing every domain of technology. It has also led to the tremendous development and implementation of cloud computing from the last few years. But the shared nature of data in the cloud makes it prone to security attacks. So different security techniques need to be implemented to prevent security breaches. Authentication is one such technique which plays a major role in Cloud Computing security. The various possible security attacks on the Cloud Service Providers (CSP) are prevented by applying different authentication mechanisms, which verifies a user’s identity when a user wishes to request services from cloud servers. There are multiple authentication technologies for verifying the identity of a user before granting access to resources. In this research work, different possible authentication techniques are discussed. It is observed that biometric techniques are proving very helpful in implementing multi-factor authentication and one of the new biometric authentication techniques like palm print is being introduced as well.

The designed security system ensures a secure and reliable environment for cloud-based

application services which is very easy to deploy and exploit on cloud-based platforms.

***Keywords:*** Storage; cloud computing; CSP; SSO; PKI.

**LIST OF FIGURES**

**FIGURE NO. TITLE PAGE NO.**

4.1 Cloud Architecture Model 15

4.2 Authentication Methods in Cloud Computing 16

6.1 Data Flow Diagram 28

6.2 Use Case Diagram 29

6.3 Sample Screenshot 1 30

6.4 Sample Screenshot 2 30

6.5 Sample Screenshot 3 31

6.6 Sample Screenshot 4 31

**LIST OF TABLES**

**TABLE NO. TITLE PAGE NO.**

1 System Security Evaluation 23

**LIST OF ACRONYMS**

ALS Area localization scheme

PKI Public key infrastructure

CSP Cloud Service Providers

SSO Single Sign in

AES Advanced Encryption Standard

DES Data Encryption Standard

MD5 Message Digest algorithm 5

RSA Rivest-Shamir-Adleman algorithm

**Chapter 1**

**Introduction**

* 1. Background

Cloud computing, as a new paradigm of information technology, has been developed

very quickly in recent years. The vast spread of Internet resources on the web and fast growth of service providers enabled cloud computing systems to become a large scaled IT service model for distributed network environments. Cloud computing is built on top of already existing Internet technologies and is delivered as a self-service utility. Three service models are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Google, Microsoft Azure Platform, and Amazon Web Services are leading cloud computing vendors in the market of commercial system deployment. Regardless the utilized service model, cloud system can belong to one of the following cloud deployment models: Public, Community, Private or Hybrid. The main characteristics of a cloud environment are abstraction and virtualization which make the technology to be perceived and applied completely in a different manner compared with existing traditional distributed systems. Cloud environment abstracts the implementation details of services and system from users and developers. Besides, resources in cloud computing systems become highly scalable through system virtualization which is achieved by means of resource pooling and sharing.

Integration of IT systems of organizations and businesses inside a cloud computing environment results in many technical and business advantages. However, the adoption of this technology introduces a new concern for businesses about security risks of cloud computing and indirect control over sensitive and private data. Cloud computing has all the security issues associated with distributed applications on the Internet and plus other security issues derived from virtualized and pooled resources. Data storage in a cloud environment is one of the most important concerns from a security point of view. Because multiple cloud customers from the same or different organization can use the same resources or applications, certain security risks should be evaluated and solved before private and sensitive data, applications and system functionality are moved into the cloud. Multi-tenancy requires a policy enforcement mechanism, isolation, service levels, etc. Both cloud deployment model and service model have a high degree of impact on the cloud security solutions and cause different significance on multi-tenancy.

Cloud computing security risks depend highly on the cloud service model. IaaS model delivers computing infrastructure, physical storage, and networking as a service. Customers use those resources in order to build their desired computing platforms through platform virtualization facilities. PaaS model adds another layer on the top of IaaS and delivers platform as a service, together with application development frameworks and tools. SaaS in turn adds another layer on the top of PaaS and delivers application (software) as a service which is consumed by users via a browser or other

client program. There is trade-off between service delivery model and security solutions integrated in it. The higher service level, the more service provider is responsible for the security solutions.

Both cloud service providers and customers are concerned about security issues associated with the cloud environment. Although different cloud domains have different security and policy characteristics corresponding to specific functionality and usage of the system, the important aspects of secure service provisioning are generic among them. All the potential security issues associated with Identity Management, Confidentiality, Authentication, Access Control and Authorization, None-Repudiation are fatal for a cloud environment. Cloud service providers try to overcome security and privacy related issues by offering security solutions to its customers. Security as a Service (SecaaS) is a new instance of cloud service model which delivers security solutions to enterprises by means of cloud-based services from the cloud. These services may be delivered in different forms, what may result in market confusion and complication of the selection process. That is why implementation of SecaaS is still limited, but usage of those cloud-based security services will more than triple in many aspects by 2013, based on the predictions made by Gartner IT research center.

Identity and Access Control Service should provide identity management and access control to cloud resources for registered entities. Such entities can be people, software processes or other systems. In order to give a proper level of access to a resource, the identity of an entity should be verified first, which is the authentication process that precedes the authorization process. Besides authentication and authorization processes, audit logging mechanism should be used to keep track of all successful and failed operations regarding authentication and access attempts by the application.

Confidentiality is achieved by different encryption mechanisms, which is the procedure of encoding data by means of cryptographic algorithms. Providing such a service will guarantee privacy of sensitive and private data and the intended entity can only decode it. Cryptographic algorithms, which are computationally hard to crack together with encryption and decryption procedures, digital signatures, hashing, certificates, key exchange and management form an encryption system which can be delivered as a service and assure confidentiality and non-repudiation in a cloud environment.

As such, the centralization of security services and implementation of those services through standardized security frameworks under the model of SecaaS can be viewed as an innovative and beneficial utility for a cloud environment. This approach promotes the delivery of security services to customers in a professional and standardized manner. Many motives can be pointed for such kind of solution for a cloud environment: 1- aggregation of security skills and security experts, 2- effective centralized solution, 3- standardization of security practices, 4- competitive advantage in the market over the

competitors. The effective management of security in cloud-based applications is one

of the core factors for the successful cloud computing platform.

Identity as a Service (IDaaS) is one area of SecaaS and it aims to provide security

services within the scope of “identity eco-system” of a cloud environment. Existing

cloud-based identity service mechanisms require constant improvements and

enhancements as identity associated security risks have become one of the most

significant issues for a cloud environment. Privacy protection for identity information is

critical factor for a successful identity system. The contributions of this research will

be within the area of identity services for cloud environments and will be focused on

designing a cloud security system which addresses current identity-related security

issues.

**1.2. Problem Statement**

Cloud computing offers on demand services to customers with the properties of distributed systems, such as unlimited virtual resources, dynamic scalability, as well as cost advantages for business organizations. Security issues that arise within this computing environment result in various obstacles from both business and technological perspectives. There is a continuous development of security solutions with lots of challenges for a cloud environment. Security as a Service is a rather new approach to provide security solutions for a cloud environment in a professional and centralized way. Because SecaaS delivery model is very broad and not a concrete implementation and currently still in its improvement stage, few cloud providers have a system that contains centralized security infrastructure, which can provide all the needs of customers from the security perspective. Cloud-based IDaaS is not a well-established practice and there is a big need of transparent and simplified cloud security infrastructure that will provide identity management services to cloud-based software services. As a solution to this problem, this master thesis project will investigate how to manage authentication and authorization systems in cloud environments and offer an approach of cloud security system for providing authentication and authorization services to cloud based software services through IDaaS model. At the same time, the project will focus on how to deliver those services in an interoperable and secure manner.

**Objective**: Creating a secure system of authentication using RSA,MD5 and secure of file by AES algorithm for user of any organization.

**1.3. Purpose and Goals**

The main purpose of this thesis research is to achieve a solution that provides secure and

interoperable authentication and authorization systems in a cloud environment. The goals

of this master thesis are the following:

Design security system architecture for a cloud environment, which aims to deliver two identity services, such as authentication and authorization in a secure and interoperable manner, using

Web Service technology. This solution will assist cloud computing platforms to provide software services to customers in a confidential, authenticated and authorized environment.

Develop and deploy a prototype of designed authorization service that will contain the main important features and findings of this investigation.

Provide an approach of how to build cloud security system for ensuring identity management and access control solutions for cloud-based application service providers through open and platform-independent architecture

**Chapter 2**

**Litrature Survey**

1. AES and DES based secure storage in cloud computing

Publisher: IEEE, Author: Shefali Ojha; Vikram Rajput

**Published in:**[2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)](https://ieeexplore.ieee.org/xpl/conhome/8039353/proceeding)

**Abstract:**

Cloud computing is one of the most growing field of research where lots of work done regarding in this field, as much as user increase over cloud security become more concern there are lots of work done regarding in this field and all are using cryptographic technique existing technique have some drawback to overcome these issues we use authentication-based AES and MD5 technique to secure data and login of user over cloud. In this paper we propose present technique of encryption and decryption for our data at the time of login but there is no authentication provided at the given time of login. Due to only basis of trust value security is not provided.

1. Authentication Mechanisms in cloud computing environment

Publisher: ResearchGate, Author: Belbergui Chaimaa, Elkamoun Najib, Rachid Hilal

**Published** **in**: International Journal on Information Technologies & Security 2010

**Abstract**:

Cloud Computing is an emerging and ubiquitous trend. It allows users to enjoy the on- demand services, without the burden of data storage and maintenance costs. However, the outsourcing of resources, raises the security issues. The most critical concerns are access control and authentication. This work presents a survey of the previous researches proposing solutions to authentication issues in Cloud. The main aim is to perform a classification of the authentication mechanisms, related to the cloud ser- vices and deployment models, given that, to our knowledge, there is no syn-thesis work at this level. This classification will be useful to consumers.

1. Storage Techniques in Cloud Computing

Publisher: <http://www.ijarcsse.com/> , Author: Seyed Milad Dejamfar, Sara Najafzadeh **Published in**: International Journal of Advanced Research in Computer Science and Software Engineering

**Abstract:**

Storage is an important topic in cloud computing security. That is why various authentication techniques in cloud environment are presented in this paper. This process serves as a protection against different sorts of attacks where the goal is to confirm the identity of a user and the user requests services from cloud servers. Multiple authentication technologies have been put forward so far that confirm user identity before giving the permit to access resources. Each of these technologies (username and password, multi-factor authentication, mobile trusted module, public key infrastructure, single sign-on, and biometric authentication) is at first described in here. The different techniques presented will then be compared.

**Chapter 3**

**Real World Application**

**Real world examples for PKI**:

Authentication/validation (signing):

- Identity cards - a number of governments are requiring certs on ID cards as the physical card can be replicated - the cert cannot.

- Validating servers - the cert is signed by a trusted CA that you trust, so you validate you are talking to the correct server. If a site was being redirected, then they would not have the cert. This is for web servers and internal use, say for database replication - you want to make sure you are sending your SQL data to a legit box!

- Validating visitors - "client certificate authorization" is used to issue a cert from your PKI to your customers or employees when they access a resource, typically a web site. Passwords can be hacked - certs are much more difficult. Also, can use certs for VPN authentication.

- Workstation logon - here is one of your AD things - use a smartcard cert to logon instead of a password. Much more secure. From a low-level sense, it is also a straighter shot to Kerberos than password-based authentication, but I won't get into that.

- Document signing - proves that the document was unmodified and signed by the person who signed it. When using a certificate, the properly asserts the 'non-repudiation' bit it has legal significance just like a handwritten signature. Note that just having that bit set doesn't mean that your CA is supposed to allow it - if your PKI isn't independently audited it won't hold water in court. Yes, many lawmakers sign bills with their smart card.

- Email signing - shows the recipient that you actually sent that email and that it wasn't intercepted and modified. See above note about non-repudiation - applies here too.

- Machine authentication - domain controllers, workstations, networking devices, etc. can be authenticated to be allowed onto the network.

Use Case of SSO:

SSO is limited to authentication only. For this, all you're saying is that you can use whatever your FB/twitter credentials are for logging into your Liferay system (authentication is delegated to their systems, not handled by yours).

SSO was initially for enterprise users as the proliferation of logins/passwords they'd have in an office was a quick path to a security breach. Implementing an SSO solution like NTLM would have a user authenticate once and access any protected resources.

SSO for the web grew out of the same thing - one user trying to manage and maintain accounts across many different systems, which are also a quick path to a security breach. Here the SSO options allow you to use a third party credentialing system for authentication. You still have an account locally, but rely on the third party system for authentication. That way the user doesn't have to have separate accounts all over the place, they use their FB or twitter or one of the other SSO solutions.

**Chapter 4**

**Inferences from Literature**

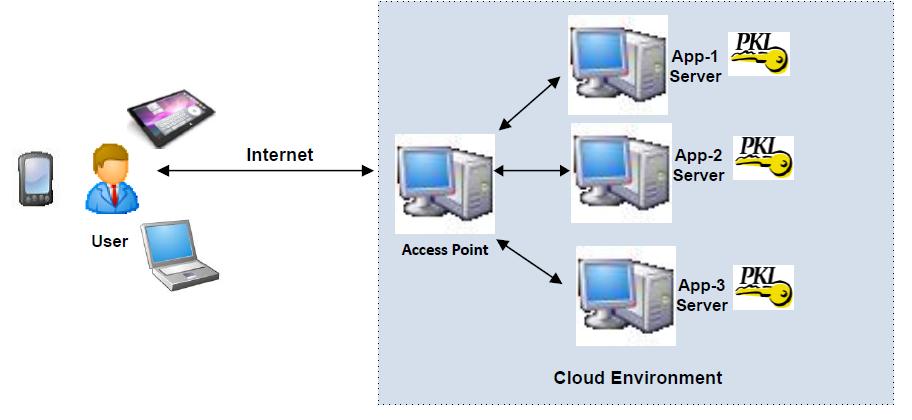
4.1 Overview of Cloud Architecture Model

This chapter describes the architecture of a cloud security system, which is designed for

delivering authentication and authorization services to cloud-based application service

providers. Those services are designed considering the solutions and security

standards outlined in the second chapter.



**Figure 4.1: Cloud Architecture Model**

Figure shows logical representation of a cloud environment: different applications

running on a cloud platform deliver different services to end-users through the Internet

End-users can be people or other business entities; however, regardless of the user type,

communication channel should be secure. As the picture shows, users interact with an access point entity through the Internet. Here, the access point is the logical representation of a cloud access point acting as an entry point to cloud-based services (practically, there is not only one access point, but application servers run behind different types of access points and proxy servers). As the communication is managed through message exchanges, there may be a need for message encryption, decryption, digital signature, etc., according to enterprise requirements and needs. This means that both communicating parties will need public-private key pairs to protect resources; and as different service providers (enterprise entities) may exist and run their applications within the same cloud environment, appropriate PKI system should be adopted for that

environment.

4.2. Security System Design Approach

As mentioned in the previous chapter, SOA implementations, such as Web Services, are

focused on the definition of services and delivery of those services in an interoperable manner between multi-domain environments. Practically, those services are software blocks that are capable to provide some functionality and business logic to service consumers through a service delivery model. In order to provide security solutions as services, there is a need to define an architecture consisting of entities that handle the security system functionality. Those system entities act as security service providers, which constitute secure environment for cloud-based systems. This research focuses on two types of security services: authentication and authorization. Both services are implemented using Web Service technology and interoperability is one of the main

features of the designed security system for delivering those services.

4.3 Authentication Techniques

In every secure communication system particularly in wide spread network such as cloud computing, authentication is a central part. It prevents shared information from unauthorized access. The management module for authentication, authorization and accounting is AAA. If a user tries to access cloud service provider, then the AAA verifies the user's authentication information. Additionally, authentication method assures that the communicating entity is the one claimed.



Figure 4.2 Authentication Methods in Cloud Computing

There are a number of authentication schemes which have been characterized into three types as:

a) Something a user knows − PIN based authentication scheme, username and password and Implicit Password Authentication System (IPAS)

b) Something a user has − smart cards or electronic tokens

c) Something a user is − biometric authentication [1].

These three are further categorized into six types discussed as under:

*A.* ***Username and Password Authentication***

In this technique of authentication, confidentiality and privacy can be maintained up to some level. In order to access the information in the CSP, the username and password are required to be entered by the user to the system. It is observed that this authentication technique fails in providing a higher and reliable security because it is tough to check whether the request is from the authorized user. Furthermore, most of the users choose very easy passwords for a machine to guess. Also, there is a possibility of the best password to get stolen by brute force and dictionary attacks. In cloud computing environment, the input constraints make it difficult for users to use complex passwords which lead to the usage of easy and short passwords. Moreover, users reuse their passwords for recognizing in many different servers which increases the risks to the security of user’s pooled information. Strong passwords help in making brute force attacks impracticable as well as help in avoiding the dictionary attacks. It is believed that the length of the password decides the security it delivers.

Various protocols have been presented that can allow a user to use a single password authentication to recognize in numerous services securely [5]. These protocols prevent users against dictionary attack, cross-site attack, malware and phishing. The main idea of these proposed protocols is that the user’s password remains secure even if the mobile device gets stolen.

*B.* ***Multifactor Authentication***

To make information more secure in cloud computing environment, a combination of authentication techniques needs to be used. This scheme is more secure because it does not just validate the username and password pair but also requires another factor e.g. biometric authentication. It is one of the stronger authentication techniques. Actually, the expectation of authenticity rises exponentially when additional factors are involved in the process of verification.

For cloud computing environment, a multifactor biometric authentication system was proposed that includes finger print and palm vein [6]. The aim is to handle the biometric data in a protected fashion by keeping the data of fingerprint in the central database of the cloud security server and the biometric data of palm vein in multi-component smart cards.

*C.* ***Mobile Trusted Module***

A set of conditions to store, measure, and report software and hardware integrity were introduced by Trusted Computing Group (TCG) through a hardware root-of-trust, that are the Mobile Trusted Module (MTM) and Trusted Platform Module (TPM). Unlike Trusted Platform Module (TPM) which is for PCs, MTM is a security aspect employed in mobile devices, [7]. The integrity and reliability of a mobile platform is ensured by the MTM [8].

Three main issues have been identified in the MTM with their promising solutions as well. The first issue is related to the requirement of balancing more or less distinct goals at the system-level designs. The second issue is the cryptographic algorithms that should be supported by MTM. The third issue is linked to the application of cryptographic primitives.

*D.* ***Single Sign On***

The SSO is a method of accessing the multiple independent software system in such a manner that when a user logs in a system, without being provoked to re-login in each application, gains the access to all the system [9]. This process supports the users to access numerous services and reduce the threat for the administrators to direct users practically. By preventing the user to remember many passwords, it helps to improve user efficiency and decreases the amount of time the user puts in typing numerous passwords to login.

*E.* ***Public Key Infrastructure***

Traditional authentication scheme is based on the secret key and mainly supports the placement of traditional asymmetric cryptographic algorithms, for example, RSA. To prove the identity of user, a private key is used. In the design of security protocols such as Secure Electronic Transaction (SET) and Secure Socket Layer (SSL/TLS), PKI has been used in order to be responsible for authentication. PKI mechanism has to ensure data integrity, data confidentiality, non-repudiation, strong authentication, as well as authorization. The security characteristics of cloud environment have been proposed that uses combination of SSO, Public Key Infrastructure, cryptography techniques, as well as LDAP, to guarantee the integrity, authentication and confidentiality of data and communications [10]. Thus, this model presented benefits of both single technologies and combination of them. PKI plays a key role in security and authentication of users in a distributed environment like that of mobile cloud computing, cloud computing and wireless sensor network.

*F.* ***Biometric Authentication***

It is the process of validating if a user is whom he is demanding to be. There are three important factors of information security that are supported by biometric authentication. These factors include identification, authentication and non-repudiation. It is derived from the Greek word bios meaning "life" and metron meaning “measure". Recognition of an individual’s behavioural and physiological attributes provides the basis for this authentication technique. Besides, it is a strong authentication technique by providing the biological proof of what we are and what we know [11].

Biometric authentication is classified into two types: behavioural and physiological.

• Behavioural biometric: It depends on the behaviour of the user. In this type of biometric authentication, signatures, keystrokes and voice prints are used.

• Physiological biometric: It is based on the physical characteristics of human. In this type of biometric authentication, hands, faces, iris, finger prints, palm-print and retina are used.

Palm-print is getting highly popular nowadays because this biometric modality is easy to capture and implement. It involves coarse lines which can be easily detected in the images that have been captured even by using low resolution camera. Moreover, it can be easily integrated into some already existing biometric recognition system because it does not require some special capture device. Thus it proves to be a perfect choice for implementation of multifactor authentication system.

4.4. Authorization System

As already mentioned earlier, different application services may be hosted in a cloud

environment and may use the same physical resources. However, each application service

is logically separated from others. Different types of system entities consume those

services; therefore, application service provider should manage a proper mechanism for

access control decisions. This means that various users, after being successfully

authenticated, should request and access those resources and services for which they are

authorized in a particular enterprise security domain. As the number of the services and

service consumers grow, management of access control mechanism becomes more

complex and expensive: each service provider needs to implement independent access

control mechanism by means of self-governing security policies and policy enforcement

points. Decoupling policies from application services and managing them independently

from application services results in a solution which is more effective for an authorization

system. Applications focus only on system functionality and business value. Having a

single security policy management point makes the entire authorization system more

flexible and secure, meaning that it can be administered, configured and protected

separately from application services. In this way, it is easy to configure and apply

common policies for every application service in a single security domain.

**Chapter 5**

**Requirements**

5.1 Hardware Requirements

* Computer System – Laptop, desktop, mobile device
* Web browser ­– Chrome, Firefox, Edge

5.2 Software Requirements

* Web Technologies – NodeJS, ReactJS, MongoDB, ExpressJS
* Android – Kotlin, Java

Bit About Software Technologies Used:

* Kotlin is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform_software), [statically typed](https://en.wikipedia.org/wiki/Static_typing), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language) with [type inference](https://en.wikipedia.org/wiki/Type_inference). Kotlin is designed to interoperate fully with [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), and the [JVM](https://en.wikipedia.org/wiki/Java_virtual_machine) version of its [standard library](https://en.wikipedia.org/wiki/Standard_library) depends on the [Java Class Library](https://en.wikipedia.org/wiki/Java_Class_Library), but type inference allows its [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)) to be more concise. Kotlin mainly targets the JVM, but also compiles to [JavaScript](https://en.wikipedia.org/wiki/JavaScript) or [native code](https://en.wikipedia.org/wiki/Machine_code) (via [LLVM](https://en.wikipedia.org/wiki/LLVM)). Kotlin is officially supported by Google for [mobile development](https://en.wikipedia.org/wiki/Mobile_app_development) on [Android](https://en.wikipedia.org/wiki/Android_(operating_system)). Since the release of [Android Studio](https://en.wikipedia.org/wiki/Android_Studio) 3.0 in October 2017, Kotlin is included as an alternative to the standard Java compiler. The Android Kotlin compiler lets the user choose between targeting Java 6 or Java 8 compatible bytecode.
* [Node.js](https://nodejs.org/en/) was initially built for Google Chrome, and later open-sourced by Google in 2008. It is built on Chrome’s V8 JavaScript engine. It’s designed to build scalable network applications, and can execute JavaScript code outside of a browser.Node.js works without an enclosing HTML page, instead using its own module system based on CommonJS, to put together multiple JavaScript files.
* [React](https://reactjs.org/) was originally created by a software engineer at Facebook, and was later open-sourced. It is maintained by Facebook, as well as a community of development companies and individual developers.

The React library can be used for creating views rendered in HTML. React views are declarative. This means that developers don’t have to worry about managing the effects of changes in the view’s state (the object that determines how components behave) or changes in the data.

* [Express](https://expressjs.com/) is a web application framework for Node.js, another MERN component. Instead of writing full web server code by hand on Node.js directly, developers use Express to simplify the task of writing server code. There’s no need to repeat the same code over and over, as you would with the Node.js HTTP module.
* [MongoDB](https://www.mongodb.com/) is a NoSQL (non-relational) document-oriented database.

While conventional relational databases have a typical schema design based on columns and tables, MongoDB is schema-less. Data is stored in flexible documents with a JSON (JavaScript Object Notation)-based query language. The content, size, and number of fields in the documents can differ from one to the next. This means that the data structure to be changed over time.

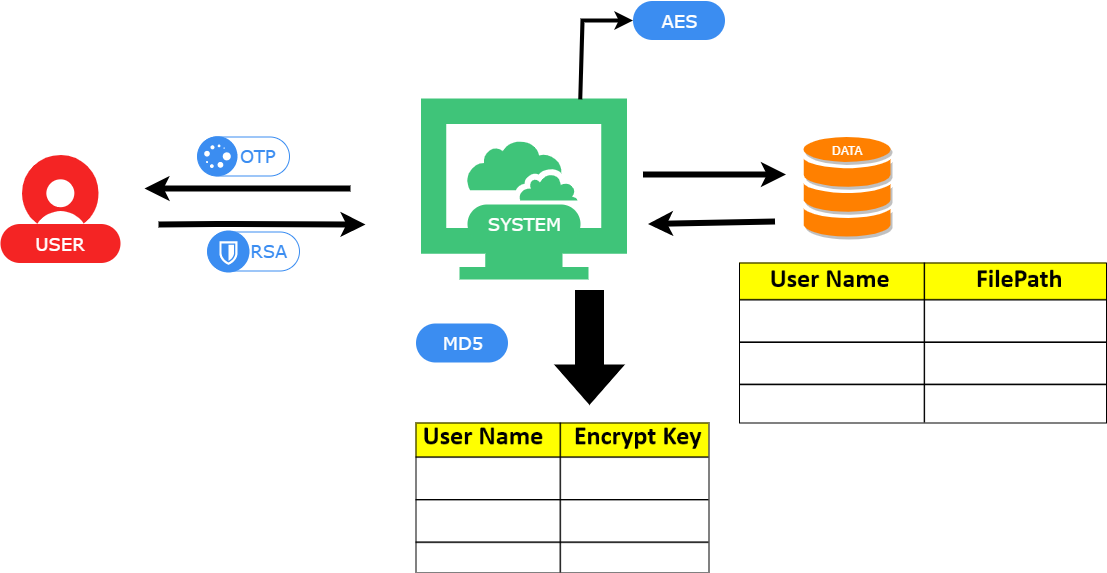
**Chapter 6**

**Problem Statement and solution**

6.1 Proposed work

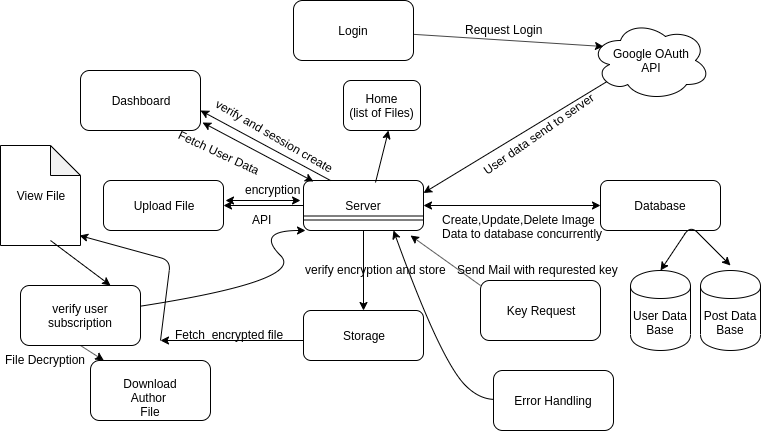
In our proposed model we have worked with the following security algorithms:

* DES algorithm for secured storage
* AES for Secured file encryption
* MD5 hashing for cover the tables from user
* One-time password/google oauth for authentication

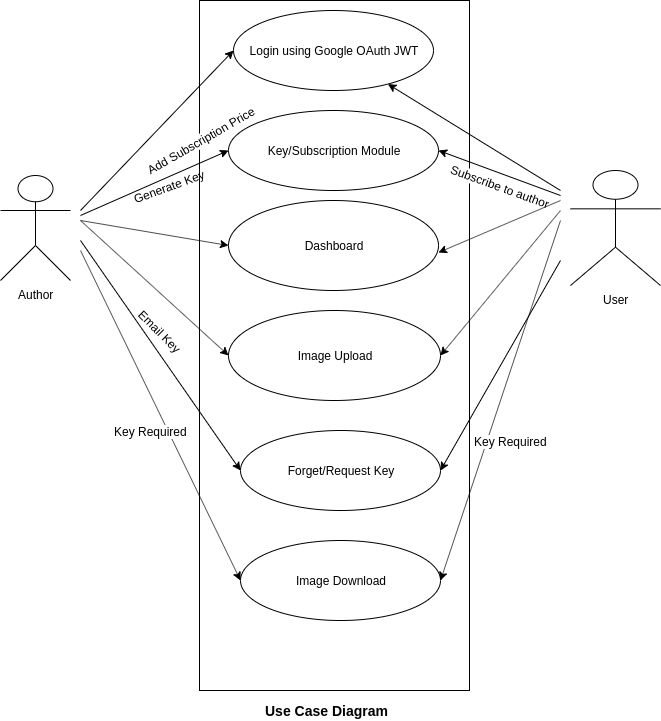


6.2 System Design and use case diagram

**Data Flow Diagram:**



**Use Case Diagram:**

****

Methodology

* DES for Secure Storage

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

DES is a previously dominant algorithm for encryption, and was published as an official Federal Information Processing Standard (FIPS). DES is now considered to be insecure due to the small key size.

Triple DES applies DES three times to each block to increase the key size. The algorithm is believed to be secure in this form.

* AES for Data Security

The Advanced Encryption Standard, or AES, is a symmetric block cipher chosen by the U.S. government to protect classified information and is implemented in software and hardware throughout the world to encrypt sensitive data.

The National Institute of Standards and Technology (NIST) started development of AES in 1997 when it announced the need for a successor algorithm for the Data Encryption Standard (DES), which was starting to become vulnerable to brute-force attacks.

This new, advanced encryption algorithm would be unclassified and had to be "capable of protecting sensitive government information well into the next century," according to the NIST announcement of the process for development of an advanced encryption standard algorithm. It was intended to be easy to implement in hardware and software, as well as in restricted environments (for example, in a smart card) and offer good defences against various attack techniques.

* MD5 for Hash Table

The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message.

The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures. MD5 has been deprecated for uses other than as a non-cryptographic checksum to verify data integrity and detect unintentional data corruption. Although originally designed as a cryptographic message authentication code algorithm for use on the internet, MD5 hashing is no longer considered reliable for use as a cryptographic checksum because researchers have demonstrated techniques capable of easily generating MD5 collisions on commercial off-the-shelf computers.

6.3 Implementation:

Complete Project Code: <https://github.com/MasterKN48/college_mini_project>

AES Algorithm Code:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Aes {

static cipher(input, w) {

const Nb = 4;

const Nr = w.length/Nb - 1;

let state = [ [], [], [], [] ];

for (let i=0; i<4\*Nb; i++) state[i%4][Math.floor(i/4)] = input[i];

state = Aes.addRoundKey(state, w, 0, Nb);

for (let round=1; round<Nr; round++) {

state = Aes.subBytes(state, Nb);

state = Aes.shiftRows(state, Nb);

state = Aes.mixColumns(state, Nb);

state = Aes.addRoundKey(state, w, round, Nb);

}

state = Aes.subBytes(state, Nb);

state = Aes.shiftRows(state, Nb);

state = Aes.addRoundKey(state, w, Nr, Nb);

const output = new Array(4\*Nb);

for (let i=0; i<4\*Nb; i++) output[i] = state[i%4][Math.floor(i/4)];

return output;

}

static keyExpansion(key) {

const Nb = 4;

const Nk = key.length/4;

const Nr = Nk + 6;

const w = new Array(Nb\*(Nr+1));

let temp = new Array(4);

for (let i=0; i<Nk; i++) {

const r = [ key[4\*i], key[4\*i+1], key[4\*i+2], key[4\*i+3] ];

w[i] = r;

}

for (let i=Nk; i<(Nb\*(Nr+1)); i++) {

w[i] = new Array(4);

for (let t=0; t<4; t++) temp[t] = w[i-1][t];

if (i % Nk == 0) {

temp = Aes.subWord(Aes.rotWord(temp));

for (let t=0; t<4; t++) temp[t] ^= Aes.rCon[i/Nk][t];

}

else if (Nk > 6 && i%Nk == 4) {

temp = Aes.subWord(temp);

}

for (let t=0; t<4; t++) w[i][t] = w[i-Nk][t] ^ temp[t];

}

return w;

}

static subBytes(s, Nb) {

for (let r=0; r<4; r++) {

for (let c=0; c<Nb; c++) s[r][c] = Aes.sBox[s[r][c]];

}

return s;

}

static shiftRows(s, Nb) {

const t = new Array(4);

for (let r=1; r<4; r++) {

for (let c=0; c<4; c++) t[c] = s[r][(c+r)%Nb];

for (let c=0; c<4; c++) s[r][c] = t[c];

}

return s;

}

static mixColumns(s, Nb) {

for (let c=0; c<Nb; c++) {

const a = new Array(Nb);

const b = new Array(Nb);

for (let r=0; r<4; r++) {

a[r] = s[r][c];

b[r] = s[r][c]&0x80 ? s[r][c]<<1 ^ 0x011b : s[r][c]<<1;

}

s[0][c] = b[0] ^ a[1] ^ b[1] ^ a[2] ^ a[3]; // {02}•a0 + {03}•a1 + a2 + a3

s[1][c] = a[0] ^ b[1] ^ a[2] ^ b[2] ^ a[3]; // a0 • {02}•a1 + {03}•a2 + a3

s[2][c] = a[0] ^ a[1] ^ b[2] ^ a[3] ^ b[3]; // a0 + a1 + {02}•a2 + {03}•a3

s[3][c] = a[0] ^ b[0] ^ a[1] ^ a[2] ^ b[3]; // {03}•a0 + a1 + a2 + {02}•a3

}

return s;

}

static addRoundKey(state, w, rnd, Nb) {

for (let r=0; r<4; r++) {

for (let c=0; c<Nb; c++) state[r][c] ^= w[rnd\*4+c][r];

}

return state;

}

static subWord(w) {

for (let i=0; i<4; i++) w[i] = Aes.sBox[w[i]];

return w;

}

static rotWord(w) {

const tmp = w[0];

for (let i=0; i<3; i++) w[i] = w[i+1];

w[3] = tmp;

return w;

}

}

Aes.sBox = [

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,

0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,

0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,

0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,

0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,

0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,

0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,

0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,

0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,

0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,

0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,

0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,

0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,

0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,

0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,

0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16,

];

Aes.rCon = [

[ 0x00, 0x00, 0x00, 0x00 ],

[ 0x01, 0x00, 0x00, 0x00 ],

[ 0x02, 0x00, 0x00, 0x00 ],

[ 0x04, 0x00, 0x00, 0x00 ],

[ 0x08, 0x00, 0x00, 0x00 ],

[ 0x10, 0x00, 0x00, 0x00 ],

[ 0x20, 0x00, 0x00, 0x00 ],

[ 0x40, 0x00, 0x00, 0x00 ],

[ 0x80, 0x00, 0x00, 0x00 ],

[ 0x1b, 0x00, 0x00, 0x00 ],

[ 0x36, 0x00, 0x00, 0x00 ],

];

}

export default Aes;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DES Algorithm Code:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const

PC1 = [

57, 49, 41, 33, 25, 17, 9,

1, 58, 50, 42, 34, 26, 18,

10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7, 62, 54, 46, 38, 30, 22,

14, 6, 61, 53, 45, 37, 29,

21, 13, 5, 28, 20, 12, 4

],

PC2 = [

14, 17, 11, 24, 1, 5,

3, 28, 15, 6, 21, 10,

23, 19, 12, 4, 26, 8,

16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55,

30, 40, 51, 45, 33, 48,

44, 49, 39, 56, 34, 53,

46, 42, 50, 36, 29, 32

],

IP = [

58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7

],

E = [

32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

8, 9, 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1

],

S = [

[

14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,

0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,

4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,

15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13

],

[

15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,

3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,

0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,

13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9

],

[

10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,

13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,

13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,

1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12

],

[

7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,

13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,

10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,

3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14

],

[

2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,

14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,

4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,

11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3

],

[

12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,

10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,

9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,

4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13

],

[

4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,

13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,

1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,

6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12

],

[

13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,

1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,

7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,

2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11

]

],

P = [

16, 7, 20, 21,

29, 12, 28, 17,

1, 15, 23, 26,

5, 18, 31, 10,

2, 8, 24, 14,

32, 27, 3, 9,

19, 13, 30, 6,

22, 11, 4, 25

],

FINAL\_IP = [

40, 8, 48, 16, 56, 24, 64, 32,

39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30,

37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28,

35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26,

33, 1, 41, 9, 49, 17, 57, 25

],

NUM\_OF\_LEFT\_SHIFTS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1];

const chunkString = (str, len) => str.match(new RegExp('.{1,' + len + '}', 'g'));

const hexToBin = (hex) => ("00000000" + (parseInt(hex, 16)).toString(2)).substr(-8);

const decToBin = (dec) => ("0000" + (parseInt(dec, 10)).toString(2)).substr(-4);

const binToHex = (bin) => parseInt(bin, 2).toString(16);

const bin = key => chunkString(key, 2).map(hex => hexToBin(hex)).join("");

const shiftString = (str, shift) => str.slice(shift, str.length) + str.slice(0, shift);

const keySchedule = (key) => {

let subkeys = [];

let perm = PC1.map(index => key[index - 1]).join("");

let C0 = perm.substr(0, perm.length / 2);

let D0 = perm.substr(perm.length / 2);

let prevC0 = C0, prevD0 = D0;

NUM\_OF\_LEFT\_SHIFTS.forEach((shift, i) => {

C0 = shiftString(prevC0, shift);

D0 = shiftString(prevD0, shift);

prevC0 = C0;

prevD0 = D0;

let pair = C0 + D0;

subkeys.push(PC2.map(index => pair[index - 1]).join(""));

});

return subkeys;

};

const expandBlock = block => E.map(index => block[index - 1]).join("");

const stringXOR = (str1, str2, len) => {

let xor = Array(len);

for (let i = 0; i < len; i++) {

xor[i] = (str1[i] === str2[i] ? 0 : 1);

}

return xor.join("");

}

const sBoxOutput = bits => {

return chunkString(bits, 6).map((group, sBox) => {

let row = parseInt(group[0] + group[5], 2);

let col = parseInt(group.slice(1, 5), 2);

return decToBin(S[sBox][16 \* row + col]);

}).join("");

};

const des = (msg, key, subkeys) => {

let perm = IP.map(index => msg[index - 1]).join(""); // init permute (IP)

let L0 = perm.substr(0, perm.length / 2);

let R0 = perm.substr(perm.length / 2);

let prevL0 = L0, prevR0 = R0;

for (let i = 0; i < 16; i++) {

L0 = prevR0;

let sBoxOut = sBoxOutput(stringXOR(subkeys[i], expandBlock(R0), 48));

let finalPerm = P.map(index => sBoxOut[index - 1]).join("");

R0 = stringXOR(prevL0, finalPerm, 32);

prevL0 = L0;

prevR0 = R0;

}

let pair = R0 + L0;

let enc = FINAL\_IP.map(index => pair[index - 1]).join("");

return chunkString(enc, 4).map(binToHex).join("").toUpperCase();

}

// sample example

const encode = (msg, key) => des(msg, key, keySchedule(key));

const decode = (msg, key) => des(msg, key, keySchedule(key).reverse());

let key = bin("133457799BBCDFF1"); // hex

let msg = bin("0123456789ABCDEF"); // hex

let enc = encode(msg, key);

console.log(enc); // => 85E813540F0AB405

console.log(decode(bin(enc), key)); // => 0123456789ABCDEF

let key2 = bin("0DAE3BF4ECCAD161");

let tripleEnc = encode(msg, key);

tripleEnc = decode(bin(tripleEnc), key2);

tripleEnc = encode(bin(tripleEnc), key);

console.log(tripleEnc); // => DDF3FFA6F0FC22DC

let tripleDec = decode(bin(tripleEnc), key);

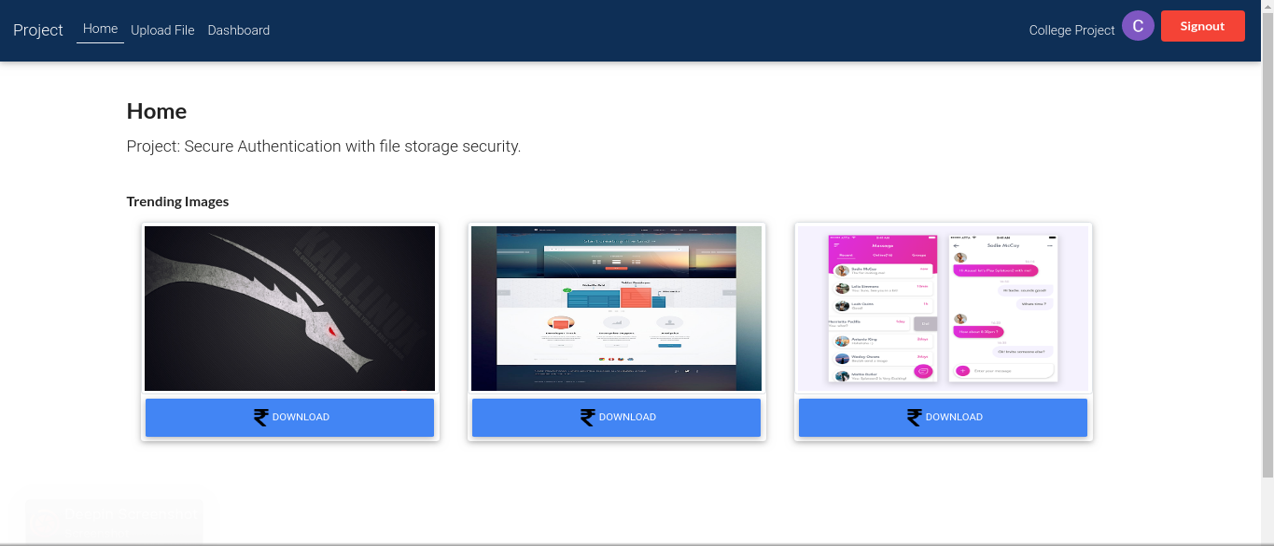
tripleDec = encode(bin(tripleDec), key2);

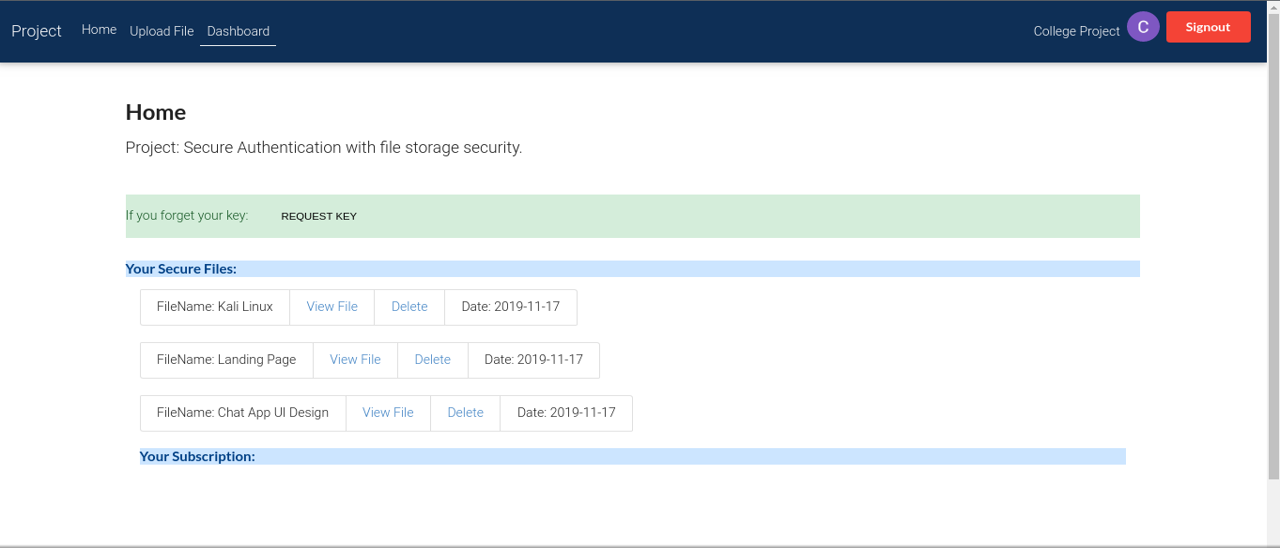
tripleDec = decode(bin(tripleDec), key);

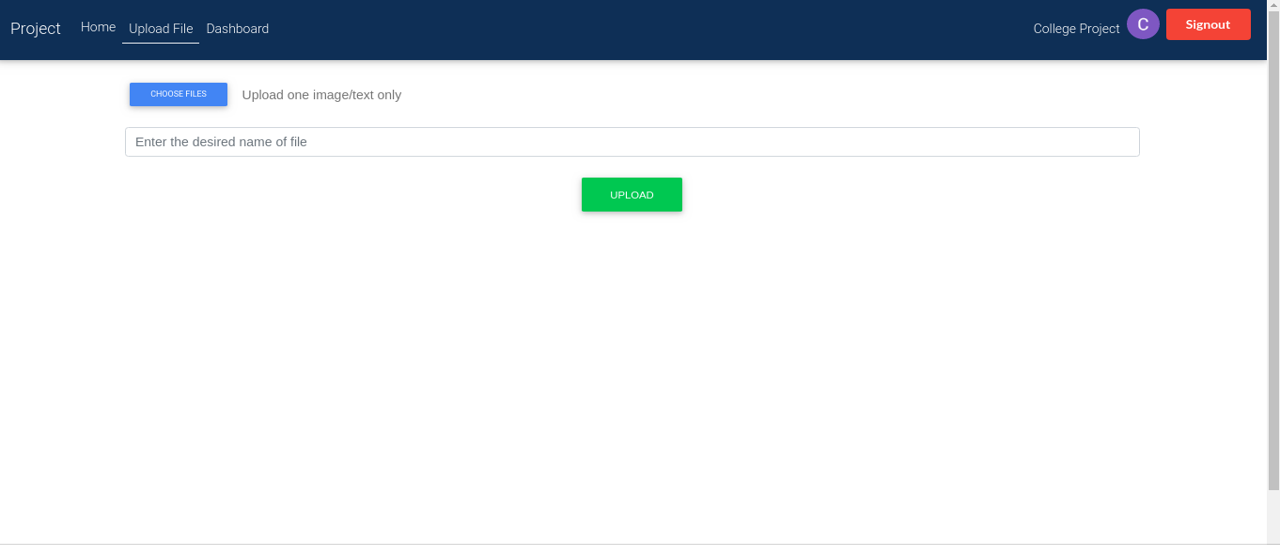
console.log(tripleDec);

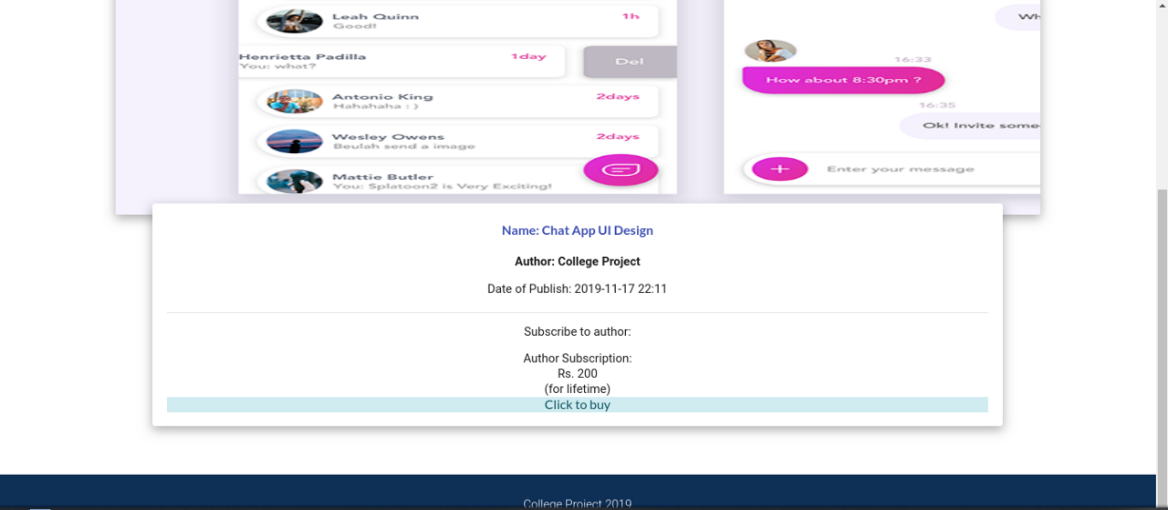
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

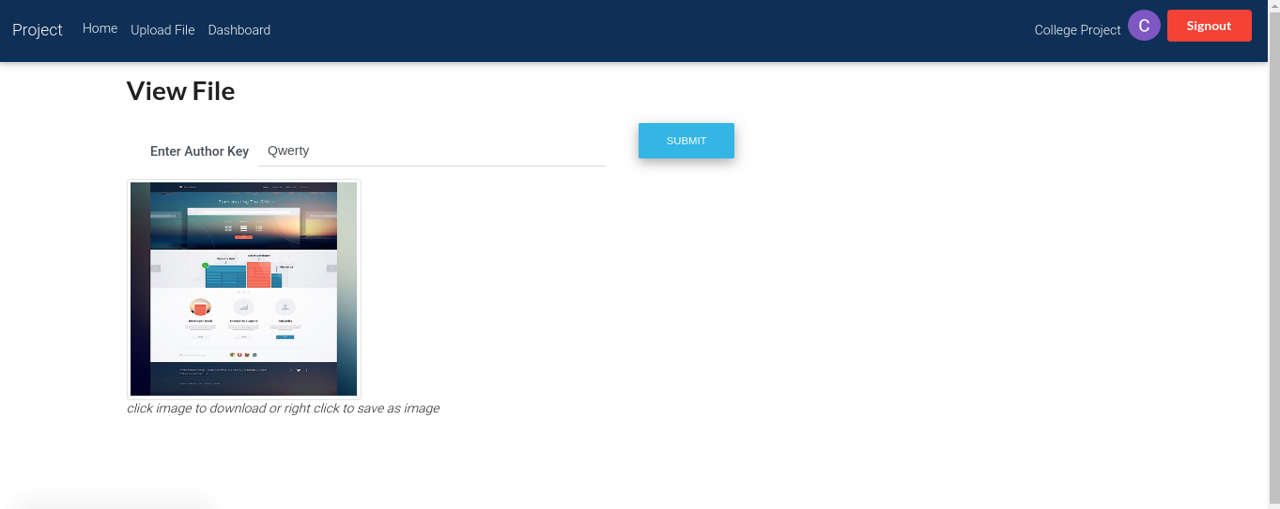
6.4 Sample/Screenshot:











6.5 Performance Metrics

**Web Service Security and Integration Advantages**

Both SSO and authorization services are designed using Web Service technology. As

mentioned in the first chapter, cloud computing platform is completely service-oriented

and is accessed through high level Web API. That is why the integration of these security

services within a cloud environment does not cause technology incompatibility issues.

Moreover, it can effectively be deployed and exploited through utilizing all the benefits

of service-oriented architecture.

Here are the main Web Service advantages that the proposed cloud security system

obtains from the service-oriented architectural design:

1. **Loosely coupling** – the security services are self-encapsulated software modules

which deliver service functionalities through standard Internet communication

protocols. Their implementation details are hided from service requesters, thus

any internal system modification will not affect the requester side. Besides, the

underlying system complexity does not impede other systems to consume those

services, because they are accessible only through high level interface.

2. **Standardized protocols** – Web Service protocol stack comprises four layers:

service transport layer, XML messaging layer, service description layer, and

service discovery layer. This gives the possibility to choose from broad collection

of well-defined standard protocols for particular system implantation.

3. **Interoperability** – the most significant advantage that the security system

benefits from the Web Service technology is interoperability. The system delivers

its services through public network, such as Internet, in an interoperable manner

independent of its implementation platform. Platform-independent service

provider and requester communicate with each other without any obstacles.

4. **Usability** – Web Service are used by client applications. Regardless what tools

and programming languages are used for its implementation, the client application

connects to the service end-point and makes service calls on that endpoint. The

communication takes place by means of serialized request-response messages

using standard data formats, like XML.

5. **Deploy ability** – security services are deployed over standard Internet

technologies on application servers and all incoming and outgoing messages can

easily pass through firewalls, using SSL over HTTP channel security mechanism.

Web Service technology facilitates managing security solutions for our system through

different built-in security mechanisms provided by Web Services Security protocol. All

these advantages make the system less costly to implement and deploy.

The only disadvantage that the Web Service technology may cause to the security system

is the stateless interaction between the service requester and provider. That is why

additional mechanisms may be required in order to keep track of service requests.

**Evaluation of System Security**

Security evaluation is based on the attack-oriented threat model. Threat model gives a

formal approach to order potential security issues that makes the system security

evaluation easy to understand. The proposed security system is analysed for possible

security threats, considering security considerations for both authentication and

authorization systems, highlighted in Chapter 3. There are five defined possible attacks

for both services: replay attacks, message information disclosure (confidentiality),

message modification, impersonation, and repudiation. Table 1 shows whether both

services are protected against those security threats. Replay attacks can be prevented

using randomly generated session IDs (assertion IDs) in messages for both services.

Message confidentiality is also protected for both services: in case of authentication

service, messages are transmitted over a secure channel, such as SSL/TLS and in case of

authorization service, messages can be encrypted using XML encryption standard.

Message modification and repudiation are prevented using XML digital signature

standard for both services. Impersonation attack is also prevented for both services using

XML digital signature standard, as it can also provide information that the message is

originated from intended entity.

Table 1: System Security Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Security Services** | **Security Threats** | | | | |
| ***Replay Attacks*** | ***Message Information Disclosure*** | ***Message Modification (Tampering)*** | ***Impersonation*** | ***Repudiation*** |
| ***Authentication*** | **+** | **+** | **+** | **+** | **+** |
| ***Authorization*** | **+** | **+** | **+** | **+** | **+** |

**Chapter 7**

**Proposed Time bound progress**

7.1 Progress Sheet

**Proposed Project Activity Chart\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Activity to be Done** | **Date(From)**  **DD-MM-YYYY** | **Date( to)**  **DD-MM-YYYY** |
| 1. | Project title selection | 03-08-2019 | 05-08-2019 |
| 2. | feasibility study | 06-08-2019 | 10-08-2019 |
| 3. | Literature survey | 11-08-2019 | 30-09-2019 |
| 4. | problem statement | 01-10-2019 | 3-10-2019 |
| 5. | requirement elicitation | 03-10-2019 | 07-10-2019 |
| 6. | Solution approach | 08-10-2019 | 15-10-2019 |
| 7. | Designing solution | 15-1000-2019 | 31-10-2019 |
| 8. | Hardware/tool implementation | 01-11-2019 | 08-11-2019 |

**Chapter 8**

**Conclusion and Future Work**

7.1. Conclusions

Implementations of cloud security solutions under the concept of Security as a Service

are in their awaking phase. This research has proposed a cloud security system based on

that concept and made contributions in the area of authentication and authorization

services for a cloud environment. The problem has been solved and the goals have been

achieved.

Through this project, we have proposed a newer security structure for cloud computing environment which includes AES, DES file encryption system, Onetime password to authenticate users and MD5 hashing for hiding information. Centralization and sharing of those identity services in a separate security infrastructure This model ensures security for whole cloud computing structure. The authorization service is implemented using Web Service technology and access control administration is implemented with simplified functionality as a user-friendly web-based application.

In our proposed system, an intruder cannot easily get information and upload the files because he needs to take control over all the servers, which is quite difficult.

7.2. Future Work

In this research a cloud security system has been designed for managing authentication

and authorization services applying quite new cloud service paradigm, such as Security

as a Service. As such, there is a need to do more comprehensive observations and

activities within this area and here are some of them:

* In our proposed model we have used RSA encryption system which is deterministic. For this reason, it becomes fragile in long run process. But the other algorithms make the model highly secured.
* In future we want to work with ensuring secure communication system between users and system, user to user.
* We also want to work with encryption algorithms to find out more light and secure encryption system for secured file information preserving system.
* Centralization of the identity services for a cloud environment represents another

two issues: single point of failure and single target of attack. Therefore, there is a

need to conduct extra work related to data replication and protection for solving

those mentioned problems.

* The proposed system supports delivery of only two identity services. Therefore,

more identity service features can be added, such as single log out, session

refreshment, etc.

System performance should be evaluated in a scalable environment in order to

measure how responsive it is in case of large amount of service requests. This will

also show how resistant the system is against denial of service attacks.

The prototype implementation has some limitations: user can be assigned only

one role at a time, there is no policy set concept applied for this system, and there

is no separately implemented Policy Information Point service. Therefore, more

features can be added to the prototype authorization system. Besides, a prototype

of authentication system can be implemented according to the designed system.

References

[1] National Institute of Standards and Technology (NIST), “The NIST Definition of Cloud

Computing.” Sep-2011.

[2] International Journal of Advanced Research in Computer Science, “ A Review on Cloud Computing Security Using Authentication Techniques”

[3] researchgate.com “Secure Authentication on cloud computing ”,

[4] <https://searchsecurity.techtarget.com/definition/MD5>

[5] <https://searchsecurity.techtarget.com/definition/AES>

[6] <https://searchsecurity.techtarget.com/definition/RSA>

[7] <https://wikipedia.com/>

[8] International Journal of Advanced Research in Computer Science and Software Engineering, “Authentication Techniques in Cloud Computing: A Review”

[9] <https://researchgate.com/>

[10] Network security and cryptograpy by Himanshu sharma

[11] Yashpal Kadam, “Security Issues in Cloud Computing A Transparent View”, International Journal of Computer Science Emerging Technology, Vol-2 No 5 October, 2011 , 316-322

[12] “Cloud Computing: Silver Lining or Storm Ahead?”, Volume 13 Number 2, Spring 2010

[13] Hongwei Li, Yuanshun Dai, Ling Tian and Haomiao Yang, “Identity-Based Authentication for Cloud Computing”, CloudCom 2009, LNCS 5931, pp. 157–166, 2009

[14] Sven Bugiel, Stefan Nurnberger, Ahmad-Reza Sadeghi, Thomas Schneider, “Twin Clouds: Secure Cloud Computing with Low Latency”, CASED, Germany, 2011