Chapter 1

INTRODUCTION

1.1 Abstract

Most current security solutions are based on perimeter security. However, Cloud computing breaks the organization perimeters. When data resides in the Cloud, they reside outside the organizational bounds. This leads users to a loose of control over their data and raises reasonable security concerns that slow down the adoption of Cloud computing. Is the Cloud service provider accessing the data? Is it legitimately applying the access control policy defined by the user? This project presents a data-centric access control solution with enriched role-based expressiveness in which security is focused on protecting user data regardless the Cloud service provider that holds it. Novel identity-based and proxy re-encryption techniques are used to protect the authorization model.

Data is encrypted and authorization rules are cryptographically protected to preserve user data against the service provider access or misbehavior. The authorization model provides high expressiveness with role hierarchy and resource hierarchy support. The solution takes advantage of the logic formalism provided by Semantic Web technologies, which enables advanced rule management like semantic conflict detection. A proof of concept implementation has been developed and a working prototypical deployment of the proposal has been integrated within Google services.

Background

To the best of our knowledge, there is no data-centric approach providing an RBAC (Role based access control) model for access control in which data is encrypted and self-protected. The proposal in this project supposes a first solution for a data-centric RBAC approach, offering an alternative to the ABAC(Attribute based access control) model. An RBAC approach would be closer to current access control methods, resulting more natural to apply for access control enforcement than ABE-based mechanisms. In terms of expressiveness, it is said that ABAC supersedes RBAC since roles can be represented as attributes. However, when it comes to data-centric approaches in which data is encrypted, ABAC solutions are constrained by the expressiveness of ABE schemes. The cryptographic operations used in ABE usually restrict the level of expressiveness for access control rules.

1.2 Existing System and their drawbacks

Different approaches can be found in the literature to retain control over authorization in Cloud computing.

- A. Lawall, D. Reichelt, and T. Schaller,*"Resource management and authorization for cloud services,"* in Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, ser. S-BPM ONE '15, New York, NY, USA, 2015, pp. 18:1–18:8.

Here, the authors propose to keep the authorization decisions taken by the data owner. The access model is not published to the Cloud but kept secure on the data owner premises.

Drawback:

However, in this approach the CSP becomes a mere storage system and the data owner should be online to process access requests from users.

- D. Y. Chang, M. Benantar, J. Y.-c. Chang, and V. Venkataramappa, *"Authentication and authorization methods for cloud computing platform security,"* Jan. 1 2015, US Patent 20,150,007,274

This approach deals with this issue by enabling a plug-in mechanism in the CSP that allows data owners to deploy their own security modules. This permits to control the authorization mechanisms used within a CSP.

Drawback:

However, it does not establish how the authorization model should be protected, so the CSP could potentially infer information and access the data. Moreover, this approach does not cover Inter-cloud scenarios, since the plug-in module should be deployed to different CSPs. Additionally, these approaches do not protect data with encryption methods.

1.3 Proposed System

This project presents SecRBAC, a data-centric access control solution for self-protected data that can run in untrusted CSPs and provides extended Role-Based Access Control expressiveness. The proposed authorization solution provides a rule-based approach following the RBAC scheme, where roles are used to ease the management of access to the resources. This approach can help to control and manage security and to deal with the complexity of managing access control in Cloud computing. Role and resource hierarchies are supported by the authorization model, providing more expressiveness to the rules by enabling the definition of simple but powerful rules that apply to several users and resources thanks to privilege propagation through roles and hierarchies. Policy rule specifications are based on Semantic Web technologies that enable enriched rule definitions and advanced policy management features like conflict detection.

A data-centric approach is used for data self-protection, where novel cryptograhpic techniques such as Proxy Re-EncryptionEncryption (PRE), IdentityBased Encryption (IBE) and Identity-Based Proxy ReEncryption (IBPRE) are used. They allow to re-encrypt data from one key to another without getting access and to use identities in cryptographic operations.

These techniques are used to protect both the data and the authorization model. Each piece of data is ciphered with its own encryption key linked to the authorization model and rules are cryptographically protected to preserve data against the service provider access or misbehaviour when evaluating the rules. It also combines a user-centric approach for authorization rules, where the data owner can define a unified access control policy for his data. The solution enables a rule based approach for authorization in Cloud systems where rules are under control of the data owner and access control computation is delegated to the CSP, but making it unable to grant access to unauthorized parties.

1.4 Project Architecture Diagram

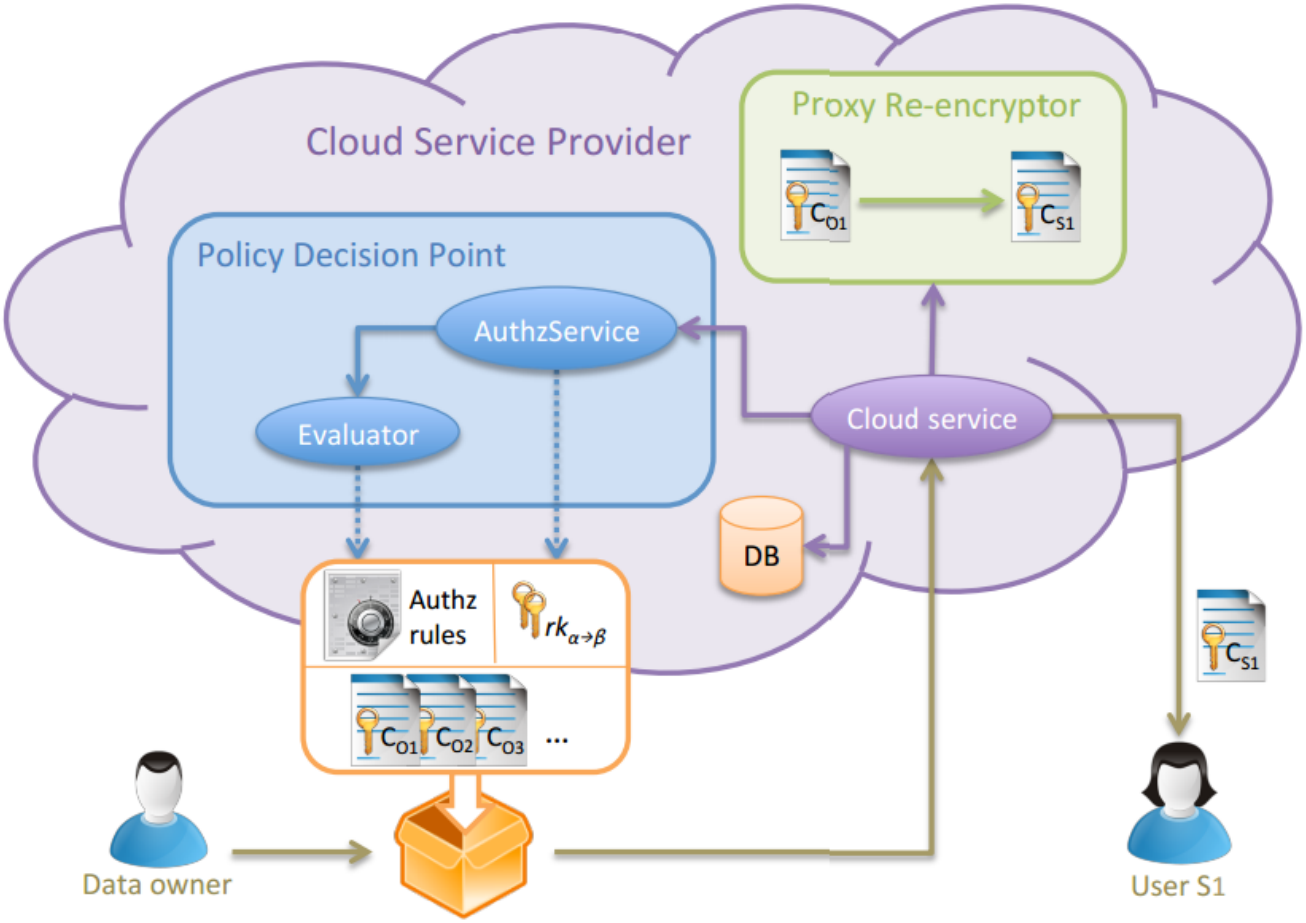


Fig 1. Project Architecture diagram

Advantages of proposed system

- Data owner need not be online all the time to process requests from the users.
- Provides the users end to end control to their data on CSP (Cloud service providers).
- The solution we proposed is applicable to inter-cloud service providers (that means the solution is independent from the particular CSP).
- Both authentication and authorization issues are addressed.

Chapter 2

LITERATURE REVIEW

Different approaches can be found in the literature to retain control over authorization in Cloud computing. In some paper, authors propose to keep the authorization decisions taken by the data owner. The access model is not published to the Cloud but kept secure on the data owner premises. However, in this approach the CSP becomes a mere storage system and the data owner should be online to process access requests from users. Another approach from deals with this issue by enabling a plug-in mechanism in the CSP that allows data owners to deploy their own security modules.

This permits to control the authorization mechanisms used within a CSP. However, it does not establish how the authorization model should be protected, so the CSP could potentially infer information and access the data. Moreover, this approach does not cover Inter-cloud scenarios, since the plug-in module should be deployed to different CSPs. Additionally, these approaches do not protect data with encryption methods. In the proposed SecRBAC solution, data encryption is used to prevent the CSP to access the data or to release it bypassing the authorization mechanism.

However, applying data encryption implies additional challenges related to authorization expressiveness. Following a straightforward approach, one can include data in a package encrypted for the intended users. This is usually done when sending a file or document to a specific receiver and ensures that only the receiver with the appropriate key can decrypt it.

From an authorization point of view, this can be seen as a simple rule where only the user with privilege to access the data will be able to decrypt it (i.e. the one owning the key). However, no access control expressiveness is provided by this approach. Only that simple rule can be enforced and just one single rule can apply to each data package. Thus, multiple encrypted copies should be created in order to deliver the same data to different receivers.

To cope with these issues, SecRBAC follows a data-centric approach that is able to cryptographically protect the data while providing access control capabilities. Several data-centric approaches, mostly based on Attribute- based Encryption (ABE), have arisen for data protection in the Cloud. In ABE, the encrypted ciphertext is labeled with a set of attributes by the data owner. Users also have a set of

attributes defined in their private keys. They would be able to access data (i.e. decrypt it) or not depending on the match between ciphertext and key attributes.

The set of attributes needed by a user to decrypt the data is defined by an access structure, which is specified as a tree with AND and OR nodes. There are two main approaches for ABE depending on where the access structure resides: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In KP-ABE the access structure or policy is defined within the private keys of users. This allows to encrypt data labeled with attributes and then control the access to such data by delivering the appropriate keys to users. However, in this case the policy is really defined by the key issuer instead of the encryptor of data, i.e. the data owner. So, the data owner should trust the key issuer for this to properly generate an adequate access policy. To solve this issue, CP-ABE proposes to include the access structure within the ciperthext, which is under control of the data owner.

Then, the key issuer just asserts the attributes of users by including them in private keys. However, either in KP-ABE or CP-ABE, the expressiveness of the access control policy is limited to combinations of AND-ed and OR-ed attributes. The data-centric solution presented in this project goes a step forward in terms of expressiveness, providing a rule-based approach following the RBAC scheme that is not tied to the limitations of current ABE approaches.

Different proposals have been also developed to try to alleviate ABE expressiveness limitations. Authors in some paper propose a solution based on CP-ABE with support for sets of attributes called Ciphertext Policy Attribute Set Based Encryption (CP-ASBE). Attributes are organized in a recursive set structure and access policies can be defined upon a single set or combining attributes from multiple sets.

This enables the definition of compound attributes and specification of policies that affect the attributes of a set. An approach named Hierarchical Attribute-based Encryption is presented. It uses a hierarchical generation of keys to achieve fine-grain access control, scalability and delegation. However, this approach implies that attributes should be managed by the same root domain authority. In some other paper, authors extend CP-ASBE with a hierarchical structure to users in order to improve scalability and flexibility.

This approach provides a hierarchical solution for users within a domain, which is achieved by a hierarchical key structure. Another approach is Flexible and Efficient Access Control Scheme (FEACS). It is based on KP-ABE and provides an access control structure represented by a formula

involving AND, OR and NOT, enabling more expressiveness for KP-ABE. The aforementioned ABE-based solutions proposed for solving access control in Cloud computing are based on the Attribute-based Access Control (ABAC) model.

Both ABAC and RBAC models have their own advantages and disadvantages. On one hand, RBAC may require the definition of a large number of roles for fine-grain authorization (role explosion problem in RBAC). ABAC is also easier to set up without need to make an effort on role analysis as needed for RBAC. On another hand, ABAC may result in a large number of rules since a system with n attributes would have up to 2n possible rule combinations (rule explosion problem in ABAC).

ABAC separates authorization rules from user attributes, making it difficult to determine permissions available to a particular user, while RBAC is deterministic and user privileges can be easily determined by the data owner. Moreover, the cryptographic operations used in ABE approaches usually restrict the level of expressiveness provided by the access control rules. Concretely, role hierarchy and object hierarchy capabilities provided by SecRBAC cannot be achieved by current ABE schemes. Moreover, private keys in ABE should contain the attributes of the user, which tights the keys to permissions in the access control policy.

In SecRBAC, user keys only identify their holders and they are not tied to the authorization model. That is, user privileges are completely independent of their private key. Finally, no user-centric approach for authorization rules is provided by current ABE solutions. In SecRBAC, a single access policy defined by the data owner is able to protect more than one piece of data, resulting in a user-centric approach for rule management. Additionally, the proposed solution provides support for the ontological representation of the authorization model, providing additional reasoning mechanisms to cope with issues such as detection of conflicts between different authorization rules. Cloud computing is set of resources that are being allocated on demand. Cloud computing proposes new ways to provide services. These new innovative, technical and pricing opportunities bring changes in the way business operated. Cloud computing is the matchless computing technology.

Cloud computing is a new label to an old idea. Cloud computing is a collection of resources and serviced provided by cloud service provider through internet. Cloud services are distributed from data canters sited all over the world. Cloud computing makes possible for its users to use the virtual resources via internet as per requirements. Cloud computing grabbed the spotlight in few years. General example of cloud services are Google Engine, Oracle Cloud, Office 365. As the cloud computing is growing rapidly this also leads to severe security concerns. Lack of security is the only

barrier in wide adoption of cloud computing. The rapid growth of cloud computing has brought many security challenges for users and providers.

2.1 Cloud Service Models

Cloud Software-as-a-Service: Software–as-a-Service is a software distribution scheme which gives right to access the software and its functions remotely as a web-based service. Software-as-a-Service permits organizations to get into business functionality a very low cost normally less than paying for licensed applications since SaaS charges are built on a monthly fee. As so the software is hosted remotely users do not require paying for additional hardware. Software-as-a-Service eliminates the all possibilities for organizations to handle the installation, set-up, daily preservation and maintenance.

Cloud Platform-as-a-service: the capability provided to the users to deploy onto the cloud infrastructure. PaaS model, cloud suppliers brings a computing platform, naturally comprising Operating System, Programming Language execution environment, database and wed servers. Application developers can develop and run their software results on cloud platform with no cost and difficulty of acquiring and handling of the main hardware and software films, for examples Oracle cloud platform-as-a-service, Oracle provides the Database as platform. And other example is windows azure. In other means Platform-as-a-Service is the facility to offer to the users to deploy user-designed or obtained applications on the cloud infrastructure. PaaS can largely be characterized as application development environments proposed as a Service‟ via the cloud supplier. Users use these platforms which is being normally have Integrated Development Environment (IDE), so as it comprises the editor, compiler, build/execute and deploy features to develop their applications and they deploy their applications on the infrastructure provided by the cloud supplier.

Cloud Infrastructure-as-a-Service: The cloud infrastructure such as hardware, servers, routers, storage, and other networking modules all are granted by the IaaS supplier. The end user takes on these offered services based on their requirements and pay for what they have used. The end user is capable of deploy and run any software, which comprise Operation Systems, applications. The end user does not supervise or monitor the core cloud infrastructure, but has hold over the operation systems and deployed application.

At this juncture, the end user needs to experience the resource requirements for the precise application to make use of IaaS properly. Flexibility and scaling are the liabilities of the end user, not

the supplier. Moreover, IaaS is small task performing-it-yourself information hub so as you would require to form the means and make the task completed.

2.2 Cloud Deployment Models

Public Cloud - A cloud is to be entitled as public cloud when the services being provided over network that are available publicly, anyone can access it.

Private Cloud - A private cloud is an infrastructure that provides the services to a single organization, whether managed by internally or by a third party.

Community Cloud - It comprises sharing of computing infrastructure between organizations of identical community.

Hybrid Cloud - A hybrid cloud is a collection of private as well as public cloud options. That remains unique entities but is bound together by standardized or proprietary technology.

2.3 Cloud Computing Security Threats

Cloud computing faces as much security threats as that are existing in the networks, intranets. these threats come in various forms. Cloud computing alliance did research in 2013 on cloud computing security threats and identified these threats.

- Traffic Hijacking
- Insecure Interface and APIs
- Denial of Service
- Malicious Insiders
- Abuse of Cloud Services
- Insufficient Due Diligence
- Shared Technology Vulnerabilities
- Data Breaches
- Unknown Risk Profile
- Perimeter Security Model Broken

2.4 Cloud Security Issues

While cost and ease of use are the two main strong benefits of the cloud computing, there are some major alarming issues that need to be referenced when allowing moving critical application and sensitive data to public and shared cloud environment. The main aspect describing the achievement of any new computing technology is the height of security it provides whether the data located in the cloud is protected at that level that it can avoid any sort of security issue. So we must say that Security and privacy are the key challenges in the cloud computing. Here are some security issues, that we have presented

2.4.1 Data confidentiality issue:

Confidentiality is a set of rules or an agreement that bounds access or location restriction on certain types of information so in cloud data reside publicly so Confidentiality refers to, customer's data and computation task are to be kept confidential from both cloud provider and other customers who is using the service. We must make sure that user's private or confidential information should not be accessed by anyone in the cloud computing system, including application, platform, CPU and physical memory. It is clear that user's confidential data is disclosed to service provider on the following situation only.

Situation 1

The first situation where user's information may be disclosed when service provider knows where the user's private information resides in the cloud systems.

Situation 2

The second situation where user's information may be disclosed when service provider has the authority to access and gather user's private information in the cloud systems.

Situation 3

The third situation where user's information may be disclosed when service provider can figure out the meaning of user's information in the cloud systems.

These are the following situations where, service provider can collect or get access user's information or data, if the service provider must know the place of the data in the cloud computing and have the authority to access users data. As we know that the current cloud computing consists of three layers Software layer, Platform layer, Infrastructure layer. Software layer provider the user interface for the user to use the services running on the cloud infrastructure. The platform layer

provides the platform such as operation environment for software to run with the help of provided system resources. And the infrastructure layer provides the hardware resources for computing, storage and network. Although as each service provider has its own software, platform and infrastructure layer with this when user uses the cloud application provided by service provider, it is mandatory for the user to use the platform as well as infrastructure provided by the service provider and therefore service provider is aware of, where the user's data is placed and the full accessibility to the data.

2.4.2 Data availability issue

When keeping data at remote location which is owned by others, data owner may face the problem of system failure of the service provider. And if cloud stops working, data will not be available as the data depends on single service provider. Threats to data availability are flooding attacks causes deny of service and Direct /Indirect (DOS) attack. Cloud computing is to provide on-demand service of different levels. If a certain service is no longer available or the quality of service cannot meet the Service Level Agreement (SLA), customers may lose faith in the cloud system.

2.4.3 Data integrity issue

As the word itself explains the "completeness" and "wholeness" of the data which is the basic and central needs of the information technology, As we know that integrity of data is important in the database equally integrity of data storage is important and necessary requirement in the cloud, it is the key factor that shaken the performance of the cloud. The data integrity proofs the validity, consistency and regularity of the data. It is the perfect method of writing of the data in a secure way the persistent data storage which can be reclaim or retrieved in the same layout as it was stored later. Therefore cloud storage is becoming popular for the outsourcing of day-to-day management of data. So integrity monitoring of the data in the cloud is also very important to escape all possibilities of data corruption and data crash. The cloud provider should provide surety to the user that integrity of their data is maintained in the cloud.

Chapter 3

ANALYSIS

3.1 Introduction

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Analysts in the field of engineering look at requirements, structures, mechanisms, and systems dimensions. Analysis is an exploratory activity.

The Analysis Phase is where the project lifecycle begins. The Analysis Phase is where you break down the deliverables in the high-level Project Charter into the more detailed business requirements. The Analysis Phase is also the part of the project where you identify the overall direction that the project will take through the creation of the project strategy documents.

Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements

3.2 Software Requirement Specification

3.2.1 Introduction

A software requirements specification (SRS) – a requirements specification for a software system – is a complete description of the behaviour of a system to be developed. In addition to a description of the software functions, the SRS also contains non-functional requirements. Software requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

Most current security solutions are based on perimeter security. However, Cloud computing breaks the organization perimeters. When data resides in the Cloud, they reside outside the organizational bounds. This leads users to a loose of control over their data and raises reasonable security concerns that slow down the adoption of Cloud computing. Is the Cloud service provider accessing the data? Is it legitimately applying the access control policy defined by the user? This project presents a data-centric access control solution with enriched role-based expressiveness in which security is

focused on protecting user data regardless the Cloud service provider that holds it. Novel identity-based and proxy re-encryption techniques are used to protect the authorization model.

Data is encrypted and authorization rules are cryptographically protected to preserve user data against the service provider access or misbehavior. The authorization model provides high expressiveness with role hierarchy and resource hierarchy support. The solution takes advantage of the logic formalism provided by Semantic Web technologies, which enables advanced rule management like semantic conflict detection. A proof of concept implementation has been developed and a working prototypical deployment of the proposal has been integrated within Google services.

This project presents SecRBAC, a data-centric access control solution for self-protected data that can run in untrusted CSPs and provides extended Role-Based Access Control expressiveness. The proposed authorization solution provides a rule-based approach following the RBAC scheme, where roles are used to ease the management of access to the resources. This approach can help to control and manage security and to deal with the complexity of managing access control in Cloud computing.

Role and resource hierarchies are supported by the authorization model, providing more expressiveness to the rules by enabling the definition of simple but powerful rules that apply to several users and resources thanks to privilege propagation through roles and hierarchies. Policy rule specifications are based on Semantic Web technologies that enable enriched rule definitions and advanced policy management features like conflict detection.

A data-centric approach is used for data self-protection, where novel cryptograhpic techniques such as Proxy Re-EncryptionEncryption (PRE), IdentityBased Encryption (IBE) and Identity-Based Proxy ReEncryption (IBPRE) are used. They allow to re-encrypt data from one key to another without getting access and to use identities in cryptographic operations.

These techniques are used to protect both the data and the authorization model. Each piece of data is ciphered with its own encryption key linked to the authorization model and rules are cryptographically protected to preserve data against the service provider access or misbehaviour when evaluating the rules.

It also combines a user-centric approach for authorization rules, where the data owner can define a unified access control policy for his data. The solution enables a rule based approach for authorization in Cloud systems where rules are under control of the data owner and access control computation is delegated to the CSP, but making it unable to grant access to unauthorized parties.

3.2.1.1 Definitions, Acronyms, and Abbreviations

- **Authorization Rules**

  Authorization rules is a collection of tuples, where each tuple will have a role name along with the type of access granted to that role. For instance, Role name can be a DOCTOR, TEACHER, STUDENT, AUTHOR, etc, and then the type of access can be READ ONLY ACCESS, READ WRITE ACCESS, etc. An end user of this project can create any number of authorization rule with any number of tuples within it. Th e user can manage all his/her authorization rules at any time by adding a new rule or by removing the existing rule. The authorization rule created in this module will be used in the manage data module for mapping the user data with the appropriate rule.

- **Manage Data**

  Here, the user of this project will be able to perform various operations on his/her data. The operations include data write, data read, data update, and data delete. The user upon writing a new data will be performing the cryptographic operation on their data thus encrypting the data before uploading it to the cloud. The user will also be able to perform other operations like mapping the data with the appropriate authorization rule created in the previous module, and also he/she can perform mapping the user to appropriate role of the defined authorization rule.

- **Privileged Data Access**

  Here, the end user of this project can access the data uploaded by the other users of this project if they have granted the access to this logged in user. The user will be mapped to appropriate role of the authorization rule and they will be able to access the data as per the access policy defined by the rule. For accessing the data, the user will be have provide his identity (which can be his email id, phone number, pan number etc) upon which an email will be sent to him/her after which our project will execute the double encryption algorithm to grant access on this data to that user.

3.2.2 General Description

3.2.2.1 Product Perspective

- Accepting each registration input from the user and creating them in MySQL DB.
- Ability for the user to manage his profile settings like edit profile, change password, retrieve password, etc.

- Providing the authorization module so that the end user can define the roles and the access levels for that roles.
- Providing the data management center for the end user to map the authorization rules and the users for their data.
- Providing the privileged data access module to access other's data shared with them.
- Providing a responsive user interface for all the three portals.

3.2.2.2 Product functions

- The manage my data portal should be accessible only by the data owner who created that data.
- Privileged data should be accessible only if any of the other users have shared the data with them.
- User must be able to add and remove the authorization rule he/she have defined at any point of time.
- All the portals will be implemented as a dynamic web application.

3.2.2.3 User Characteristics

- In this project we will need a small amount of configuration work to be done in the Eclipse box to set up the application.
- User also needs to install Apache Tomcat to host the all the three portals.
- The input provided by the users should be from the web browsers where the data owners will be uploading their data on to the cloud and also map the authorization rule and the users for their data.
- The appropriate portal must be accessed only the appropriate users belonging the correct role.

3.2.2.4 Assumptions and Dependencies

- JDK has to be installed in the machine where all the three subcomponent will be executing.
- The application servers like either the JBOSS or the Apache Tomcat will have to be supported by the host machines.
- There shall not be any firewall or other engines that prevents the remote requests from the portal.
- There shouldn't be any permission related issues on any cluster. The host operating system should take of permitting all the requests to the cluster from the interface layer.

3.2.3 Requirements

3.2.3.1 Functional Requirements

- The portal must allow any of the users to register a new account and get an access to the application themselves.
- Authorization rule module must be dynamic and should allow creation of any number of rules and any number of roles within each role along with the access policy.
- Manage my data module must be provided for the owners of the data to map the authorization rule for their data and to grant the access to other users of this application.
- User must also be able to update/ delete his/her data under the manage my data module.

3.2.3.2 Non Functional Requirements

- Should be easier to access it from the various browsers available.
- Response time of the applications should reflect the real time observations.
- The algorithm should never fail in any of the test cases.
- Each user's activity should be separated from the other user's activities.

3.2.3.3 Software Requirements

- Heroku Cloud Platform
- Operating System: Windows XP or higher, Mac OS X
- JDK 1.6 or above
- Any latest Applications server like Tomcat, JBoss
- Eclipse
- MySQL

3.2.3.4 Hardware Requirements

- Processor: Intel Pentium 4 or higher
- RAM: Min 512MB
- Hard Disk: 40GB

Chapter 4

SYSTEM DESIGN

4.1 Introduction

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development", then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words, the first step in the solution to the problem is the design of the project.

The design of the system is perhaps the most critical factor affecting the quality of the software. The objective of the design phase is to produce overall design of the software. It aims to figure out the modules that should be in the system to fulfill all the system requirements in an efficient manner.

The design will contain the specification of all these modules, their interaction with other modules and the desired output from each module. The output of the design process is a description of the software architecture.

The design phase is followed by two sub phases

- High Level Design
- Detailed Level Design

4.2 System Architecture Diagram

The below figure shows a general block diagram describing the activities performed by this project.

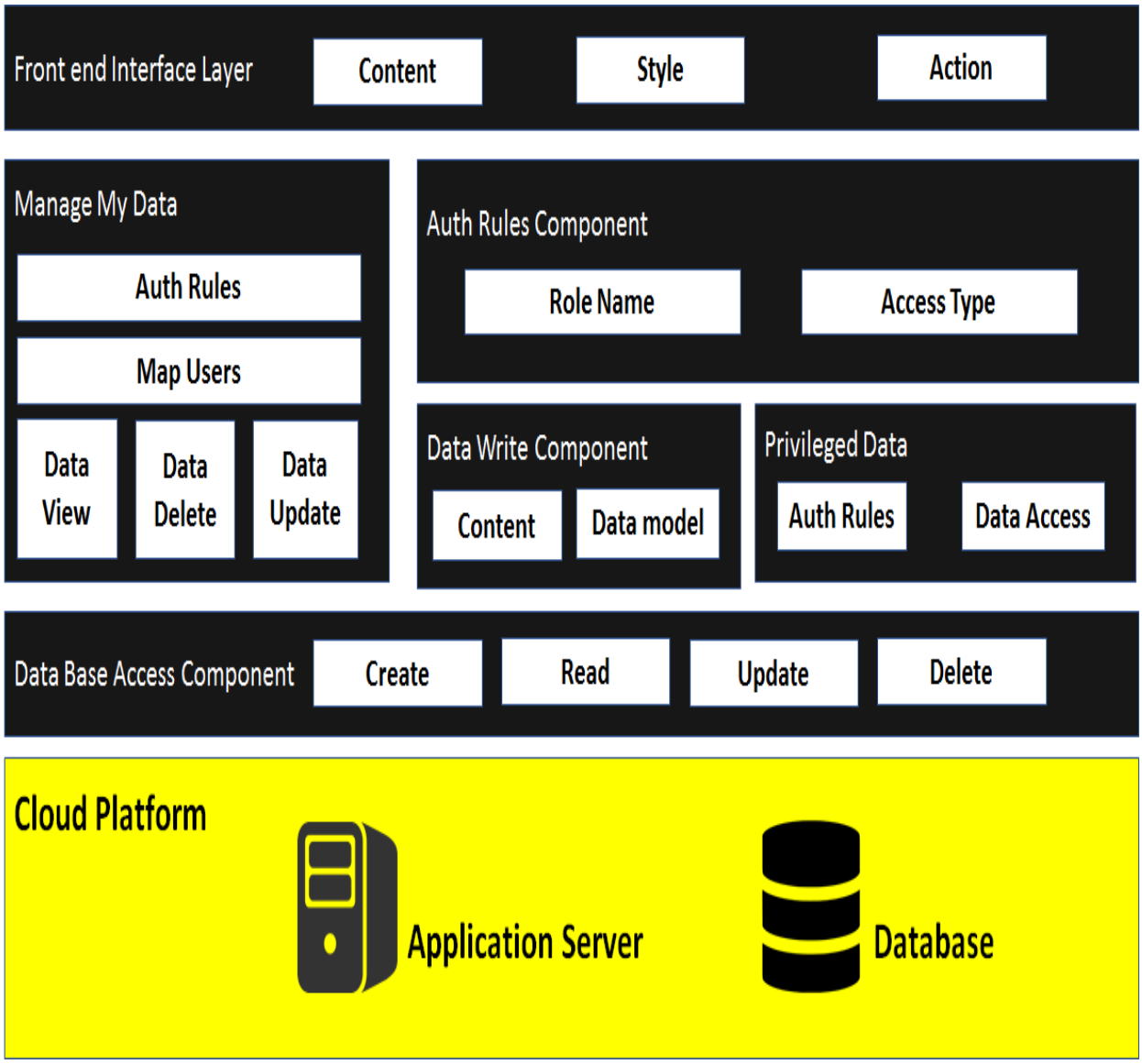


Fig 2. System Architecture diagram

The major divisions in this project are:

Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

Account Operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

Authorization Rules

Authorization rules is a collection of tuples, where each tuple will have a role name along with the type of access granted to that role. For instance, Role name can be a DOCTOR, TEACHER, STUDENT, AUTHOR, etc, and then the type of access can be READ ONLY ACCESS, READ WRITE ACCESS, etc. An end user of this project can create any number of authorization rule with any number of tuples within it. The user can manage all his/her authorization rules at any time by adding a new rule or by removing the existing rule. The authorization rule created in this module will be used in the manage data module for mapping the user data with the appropriate rule.

Manage Data

Here, the user of this project will be able to perform various operations on his/her data. The operations include data write, data read, data update, and data delete. The user upon writing a new data will be performing the cryptographic operation on their data thus encrypting the data before uploading it to the cloud. The user will also be able to perform other operations like mapping the data with the appropriate authorization rule created in the previous module, and also he/she can perform mapping the user to appropriate role of the defined authorization rule.

Privileged Data Access

Here, the end user of this project can access the data uploaded by the other users of this project if they have granted the access to this logged in user. The user will be mapped to appropriate role of the authorization rule and they will be able to access the data as per the access policy defined by the rule. For accessing the data, the user will have to provide his identity (which can be his email id, phone number, pan number etc) upon which an email will be sent to him/her after which our project will execute the double encryption algorithm to grant access on this data to that user.

Chapter 5

HIGH LEVEL DESIGN

5.1 Introduction

The System design phase involves two sub phases

- High Level Design
- Low Level Design

In the high level design, the proposed functional and non functional requirements of the software are depicted. Overall solution to the architecture is developed which can handle those needs. This chapter involves the following consideration

- Design consideration
- Data flow diagram

5.2 Design consideration

There are several design consideration issues that need to be addressed or resolved before getting down designing a complete solution for the system.

5.2.1 Assumptions and dependencies

The main assumptions and dependencies identified are as follows:

- JDK has to be installed in the machine where all the three subcomponent will be executing.
- The application servers like either the JBOSS or the Apache Tomcat will have to be supported by the host machines
- There shall not be any firewall or other engines that prevents the remote requests from the portal.
- There shouldn't be any permission related issues on any cluster. The host operating system should take of permitting all the requests to the cluster from the interface layer.

5.2.2 Mode of operation of a System

The system is capable of operating in two different modes

Table 1. Mode of operation of a system

• **Authorization Rules**	Authorization rules is a collection of tuples, where each tuple will have a role name along with the type of access granted to that role. For instance, Role name can be a DOCTOR, TEACHER, STUDENT, AUTHOR, etc, and then the type of access can be READ ONLY ACCESS, READ WRITE ACCESS, etc. An end user of this project can create any number of authorization rule with any number of tuples within it. Th e user can manage all his/her authorization rules at any time by adding a new rule or by removing the existing rule. The authorization rule created in this module will be used in the manage data module for mapping the user data with the appropriate rule.
• **Manage Data**	Here, the user of this project will be able to perform various operations on his/her data. The operations include data write, data read, data update, and data delete. The user upon writing a new data will be performing the cryptographic operation on their data thus encrypting the data before uploading it to the cloud. The user will also be able to perform other operations like mapping the data with the appropriate authorization rule created in the previous module, and also he/she can perform mapping the user to appropriate role of the defined authorization rule.

• **Privileged Data Access**	Here, the end user of this project can access the data uploaded by the other users of this project if they have granted the access to this logged in user. The user will be mapped to appropriate role of the authorization rule and they will be able to access the data as per the access policy defined by the rule. For accessing the data, the user will be have provide his identity (which can be his email id, phone number, pan number etc) upon which an email will be sent to him/her after which our project will execute the double encryption algorithm to grant access on this data to that user.

5.2.3 User operations

Table 2. User operations

Run the Authorization Rule module	Here the user will be able to create any number of authorization rule and any number of roles within each rule with appropriate level of data access
Run the Manage Data module	Here the user will be able to perform the data read, write, update, and delete operations on his/her data on the cloud. He/she will also be able to map authorization rule and the users to his/her data

Run Privileged Data module	User here will be able to access the privileged data that the other users have shared with them. They can access the data as per the policy defined for them by the data owners.

5.3 Data flow diagram

A data flow diagram is the graphical representation of the flow of data through an information system. DFD is very useful in understanding a system and can be efficiently used during analysis.

A DFD shows the flow of data through a system. It view a system as a function that transforms the inputs into desired outputs. Any complex systems will not perform this transformation in a single step and a data will typically undergo a series of transformations before it becomes the output. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input, ultimately as an effect upon the structure of the whole system.

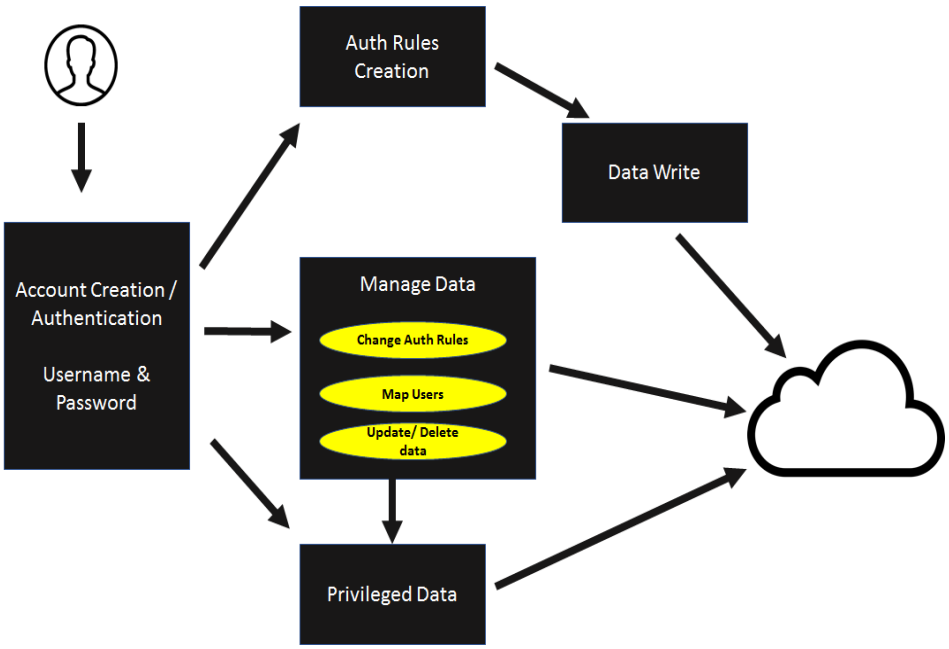


Fig 3. Data flow diagram

Below sections explain the module level flow diagrams of this project.

Module 1: Account Access Layer

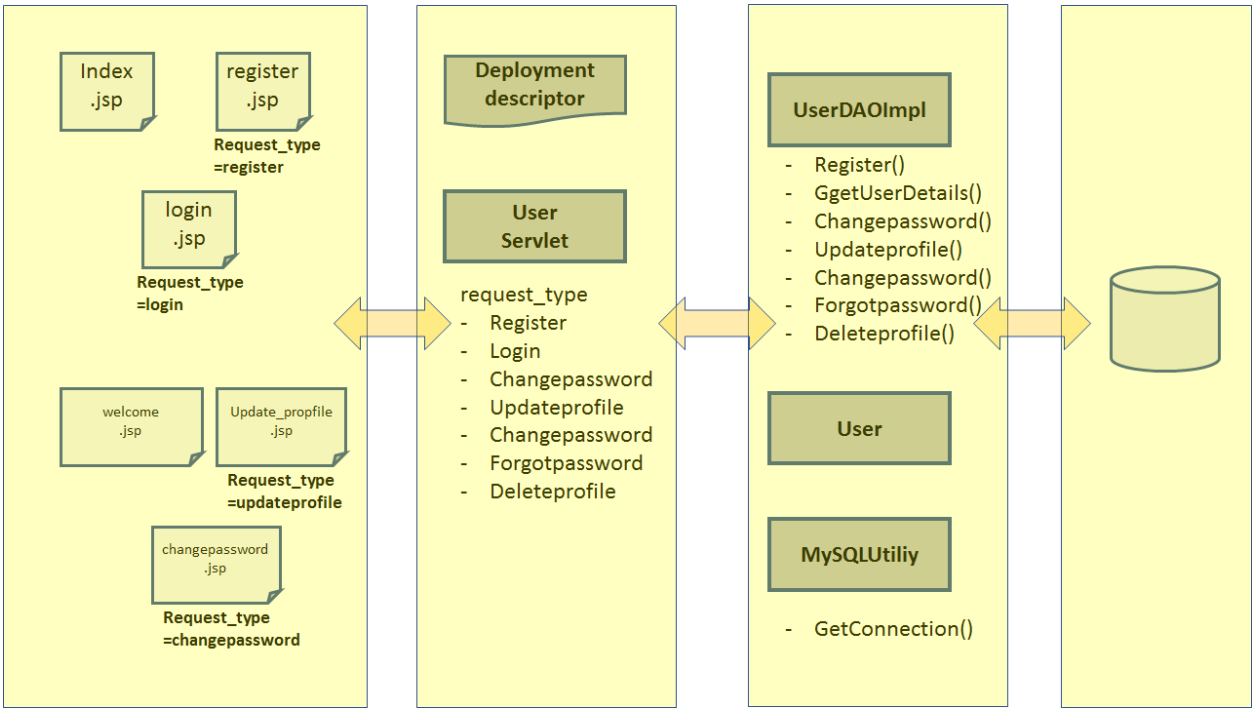


Fig 4. Account Access Layer

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities. The DAO layer is the service layer which provides database CRUD (create, update, read, and delete) services to the other layers.

It will contain the POJO classes to map the database tables into java object. It will also contain the Util classes to manage the database connections.

Module 2: Authorization Rules

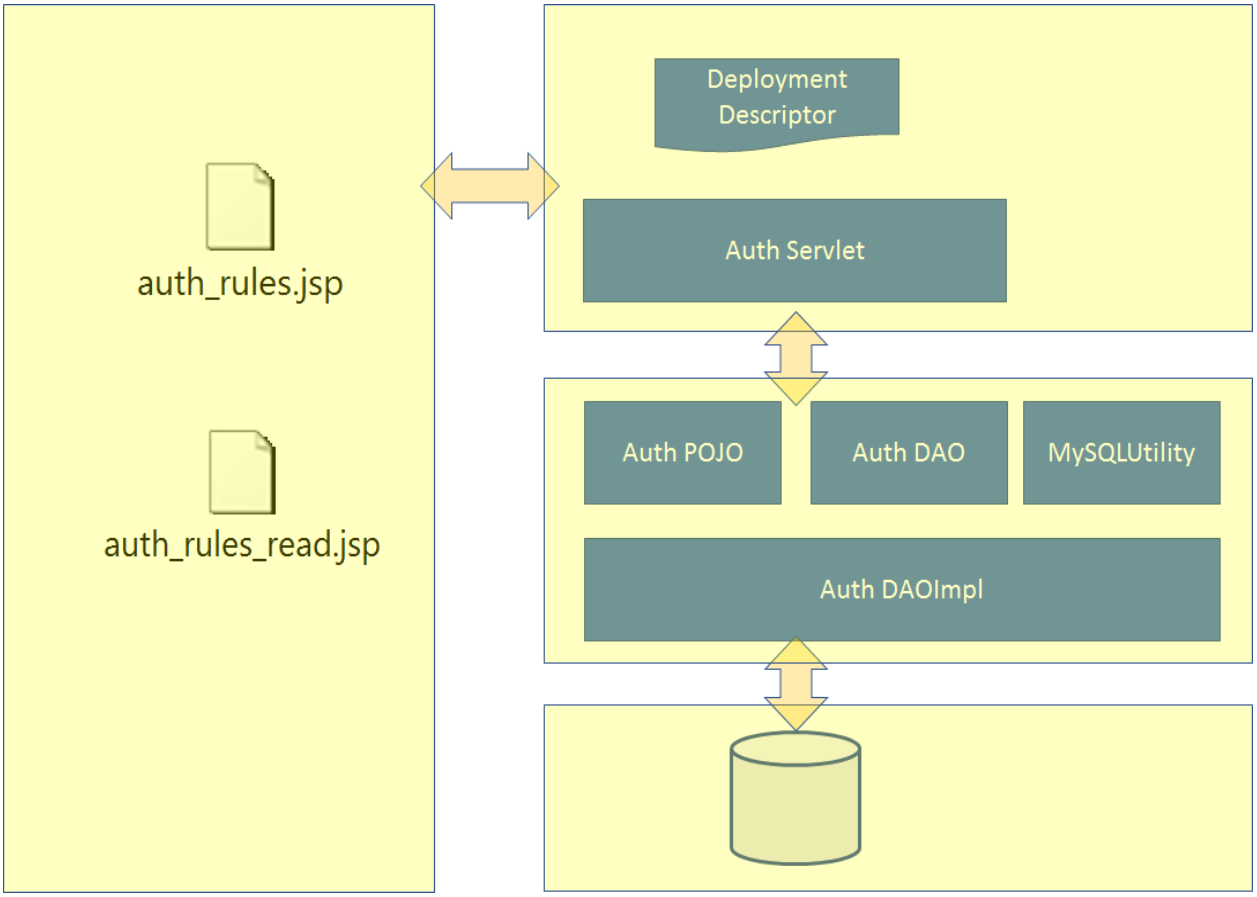


Fig 5. Authorization Rules

Authorization rules is a collection of tuples, where each tuple will have a role name along with the type of access granted to that role. For instance, Role name can be a DOCTOR, TEACHER, STUDENT, AUTHOR, etc, and then the type of access can be READ ONLY ACCESS, READ WRITE ACCESS, etc. An end user of this project can create any number of authorization rule with any number of tuples within it. The user can manage all his/her authorization rules at any time by adding a new rule or by removing the existing rule. The authorization rule created in this module will be used in the manage data module for mapping the user data with the appropriate rule.

Module 3: Manage my Data

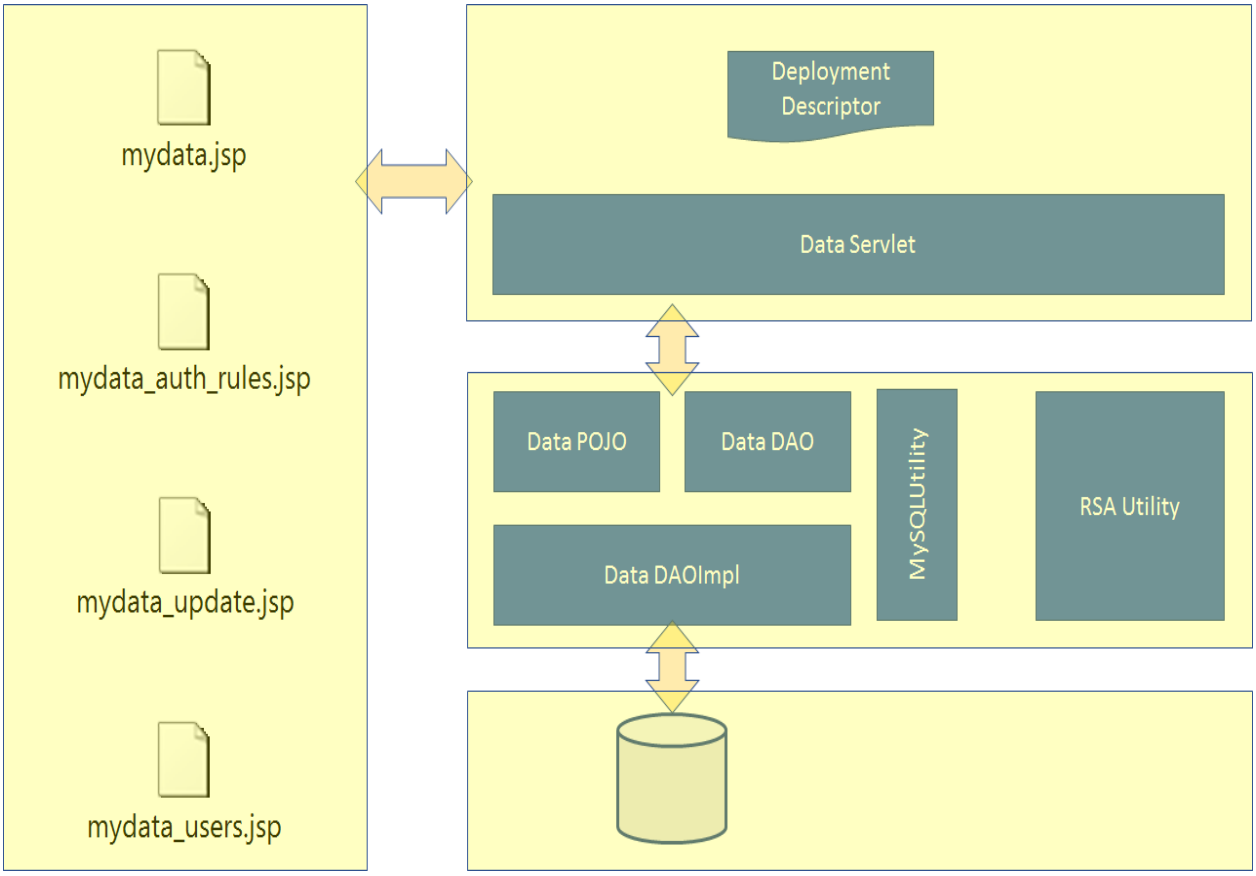


Fig 6. Manage my Data

Here, the user of this project will be able to perform various operations on his/her data. The operations include data write, data read, data update, and data delete. The user upon writing a new data will be performing the cryptographic operation on their data thus encrypting the data before uploading it to the cloud. The user will also be able to perform other operations like mapping the data with the appropriate authorization rule created in the previous module, and also he/she can perform mapping the user to appropriate role of the defined authorization rule.

Module 4: Privileged Data

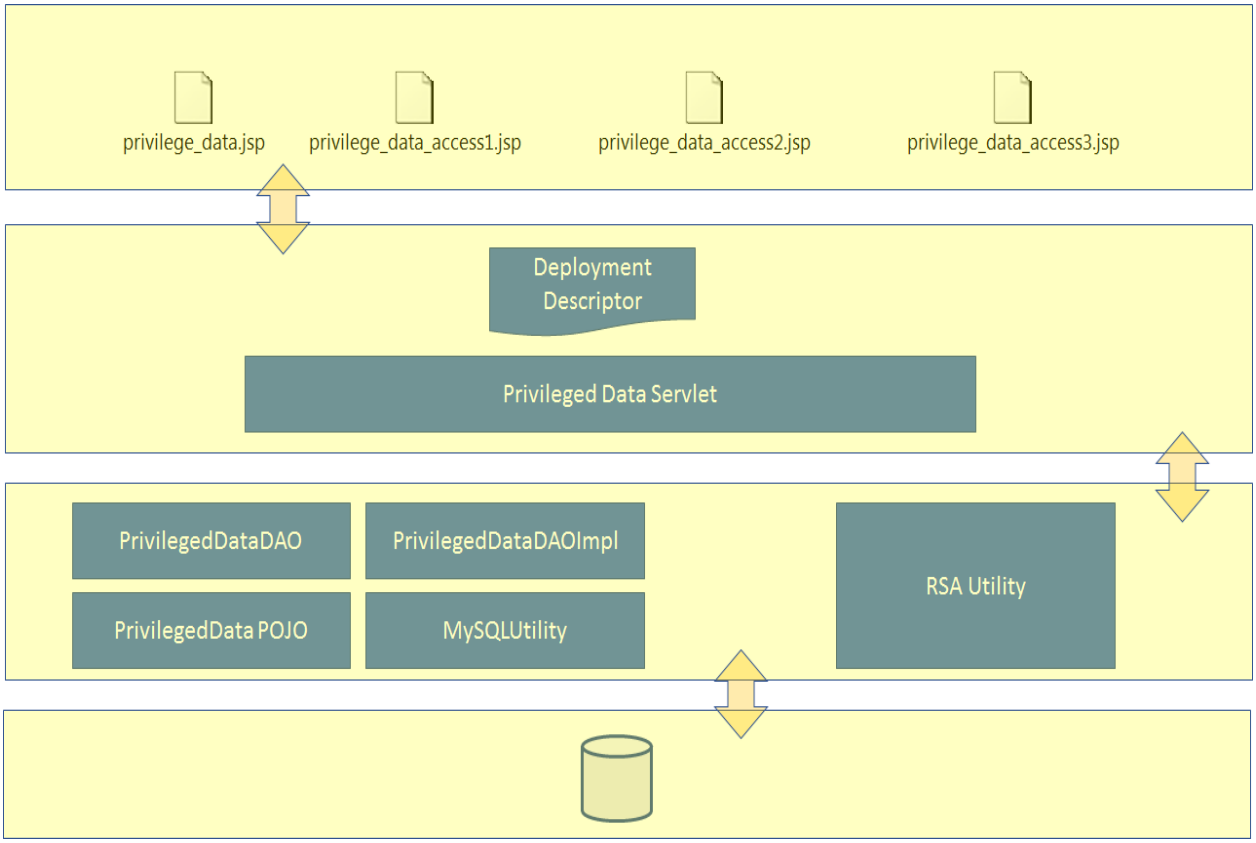


Fig 7. Privileged Data

Here, the end user of this project can access the data uploaded by the other users of this project if they have granted the access to this logged in user. The user will be mapped to appropriate role of the authorization rule and they will be able to access the data as per the access policy defined by the rule. For accessing the data, the user will be have provide his identity (which can be his email id, phone number, pan number etc) upon which an email will be sent to him/her after which our project will execute the double encryption algorithm to grant access on this data to that user.

Chapter 6:

LOW LEVEL DESIGN

6.1 Introduction

During the detailed phase, the view of the application developed during the high level design is broken down into modules and programs. Logic design is done for every program and then documented as program specifications. For every program, a unit test plan is created.

The entry criteria for this will be the HLD document. And the exit criteria will the program specification and unit test plan (LLD). Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. The goal of a low-level design document (LLDD) is to give the internal logical design of the actual program code.

6.2 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.

The use case diagram has the following components

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

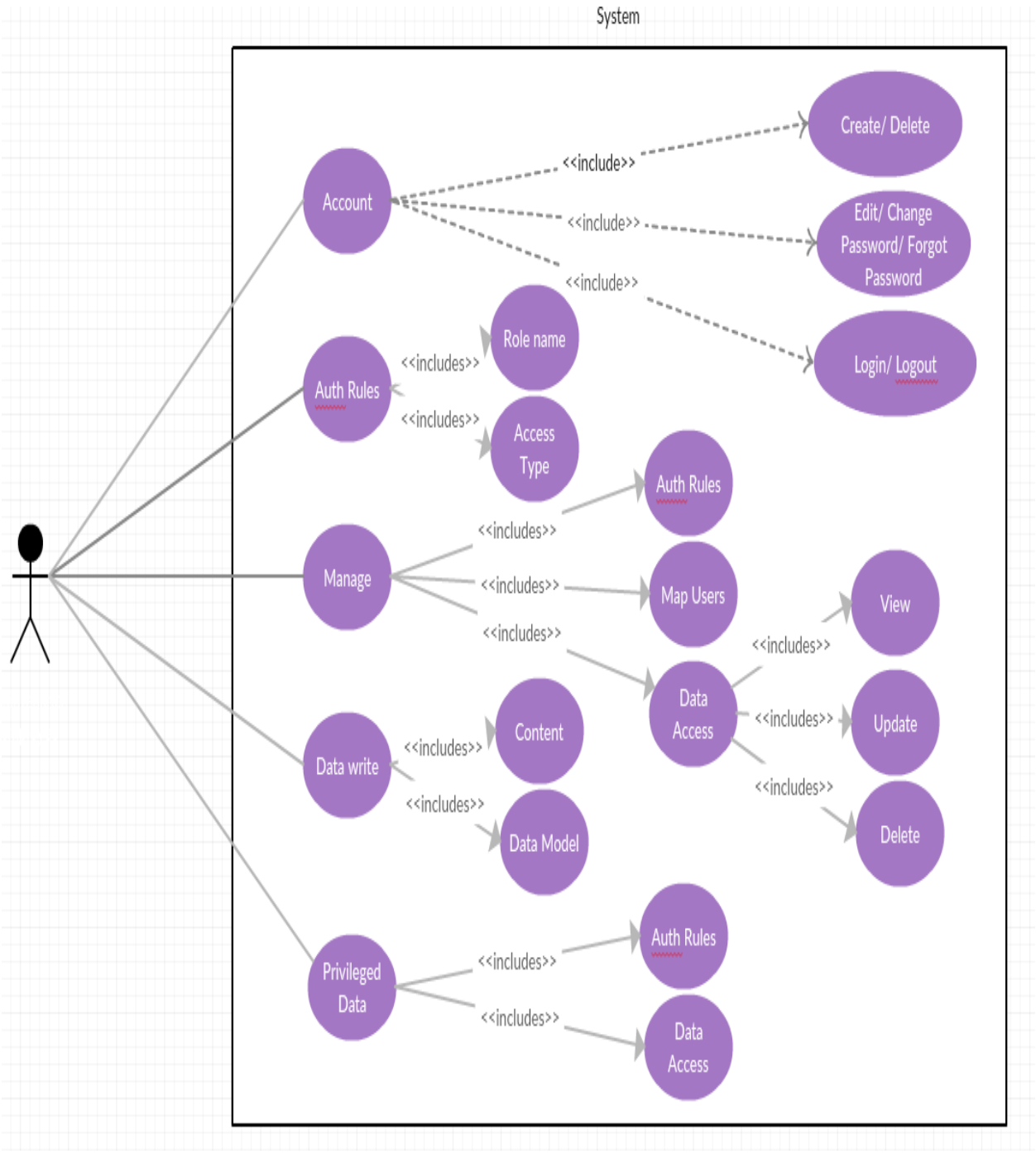


Fig 8. Use case diagram

Here, in this diagram, the actors include the users. The use cases are account, auth rules, manage, data write, privileged data, etc. <<includes>> refer to the relationships between the use cases and the actors.

6.3 Sequence diagram

A sequence diagram in a Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It shows the participants in an interaction and the sequence of messages among them; each participant is assigned a column in a table.

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 realisations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances.

Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response.

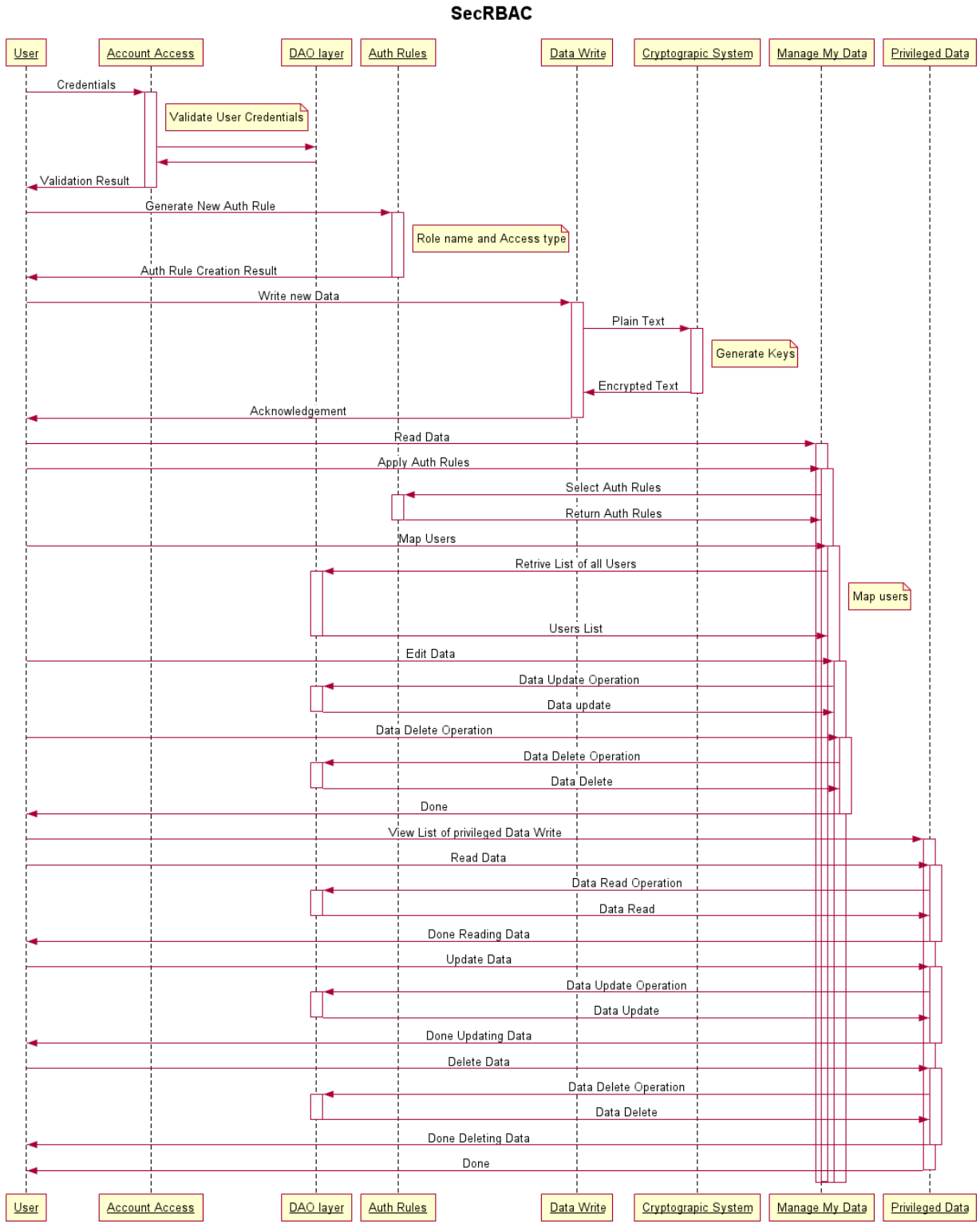


Fig 9. Sequence diagram

6.4 Class Diagrams

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, and the relationships among the classes. The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modelling translating the models into programming code.

6.4.1 Class diagram 1: Module 1

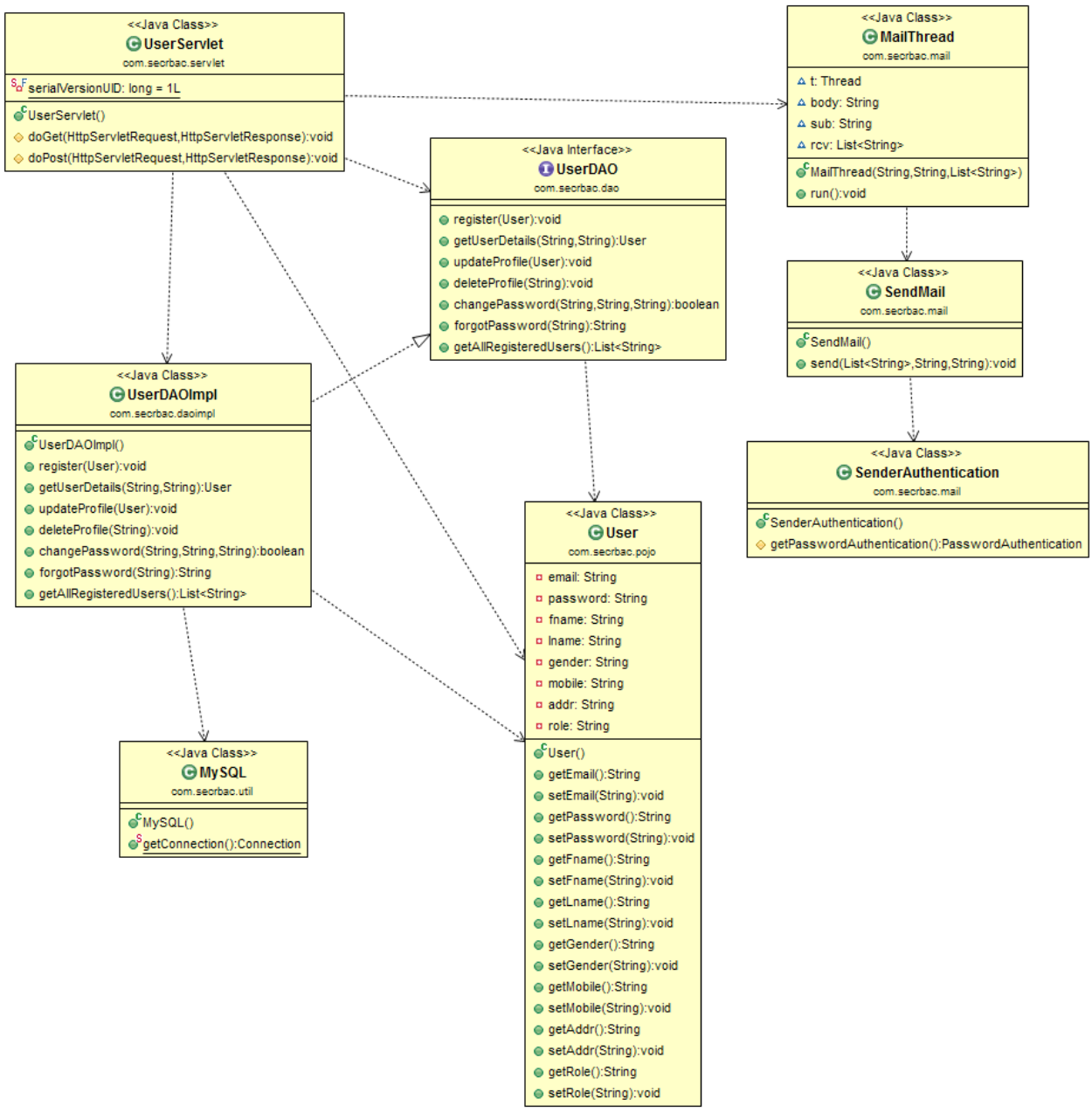


Fig 10. Module 1 class diagram

6.4.2 Class diagram 2: Module 2

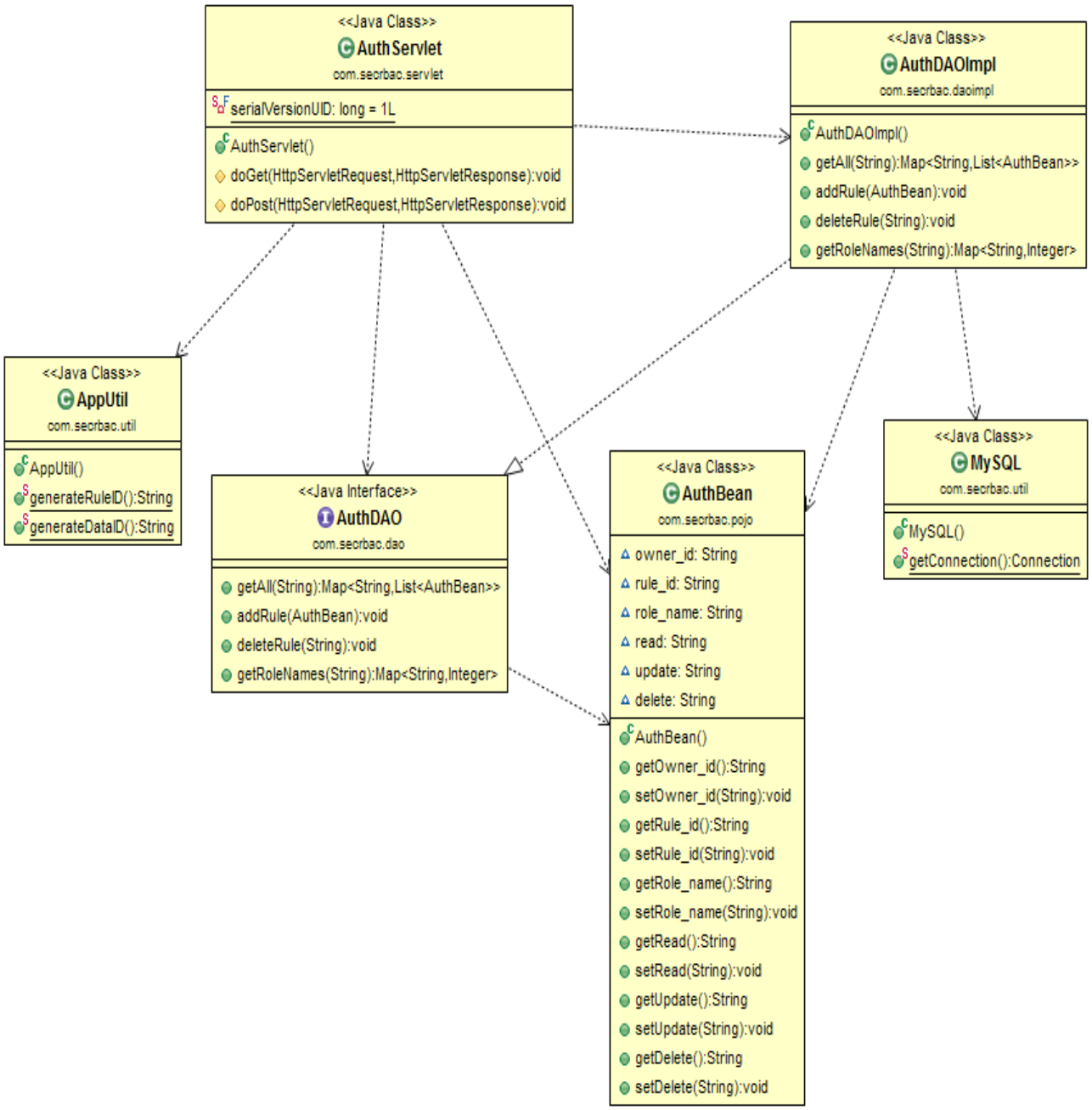


Fig 11. Module 2 Class diagram

6.4.3 Class diagram 3: Module 3

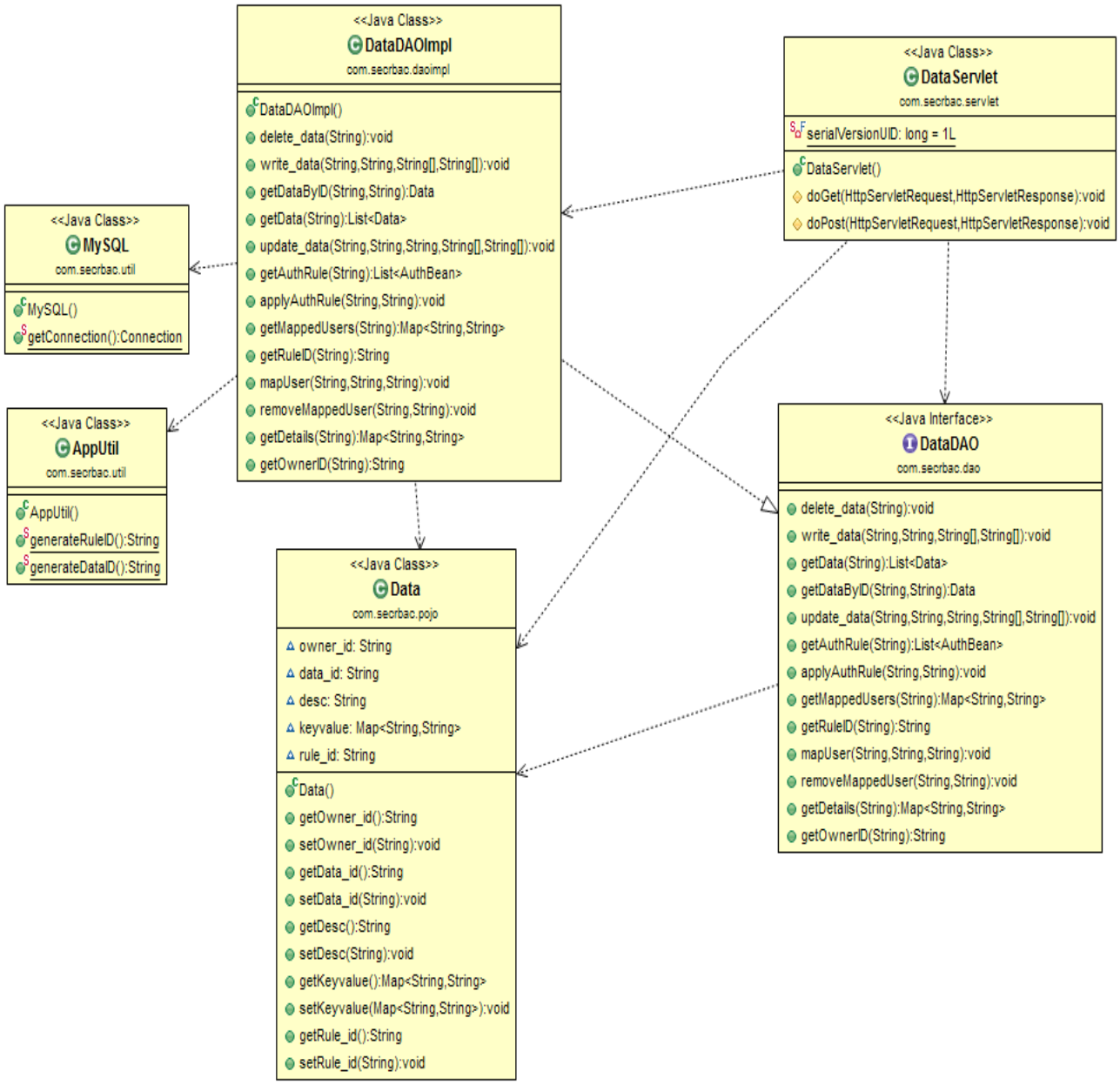


Fig 12. Module 3 class diagram

6.4.4 Class diagram 4: Module 4

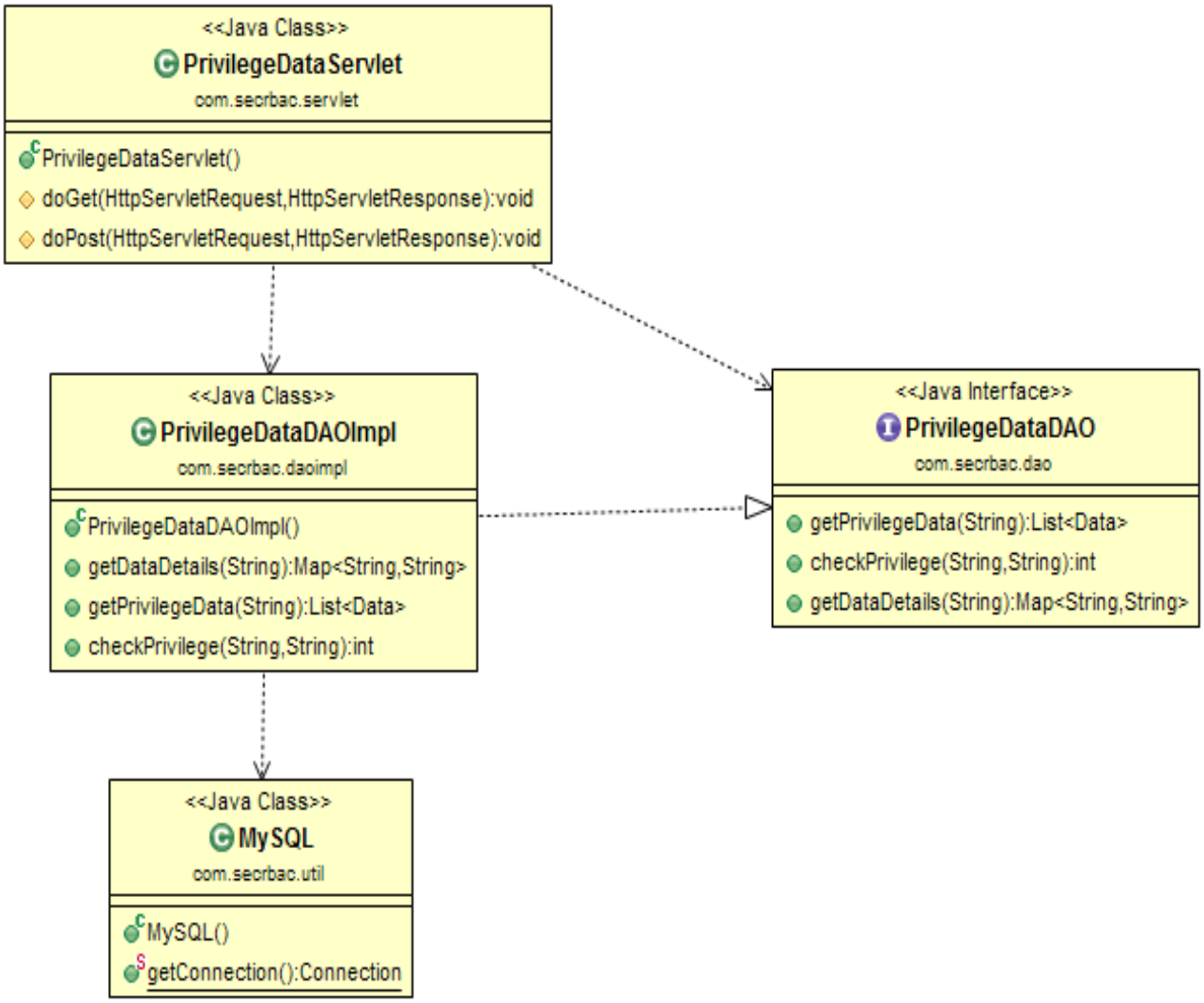


Fig 13. Module 4 class diagram

Chapter 7

IMPLEMENTATION

7.1 Introduction

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In other words, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard.

Implementation is one of the most important phases of the Software Development Life Cycle (SDLC). It encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. Specifically, it involves coding the system using a particular programming language and transferring the design into an actual working system.

This phase of the system is conducted with the idea that whatever is designed should be implemented, keeping in mind that it fulfils user requirements, objective and scope of the system. The implementation phase produces the solution to the user problem.

7.2 Overview of System Implementation

This project is implemented considering the following aspects:

1. Usability Aspect.
2. Technical Aspect.

7.2.1 Usability Aspect

The usability aspect of implementation of the project is realized using two principles:

a. The project is implemented as a Java application

There could be many ways of implementing the this project. We have chosen JAVA to come up with the required reader. The reason being many:

Firstly, Java provides a wonderful libraries which simplifies the implementation part of it.
Secondly, JAVA is platform independent, meaning the project can run on literally any platform which has JVM installed within it.
Thirdly, Oracle Corporation claims more than 70 billion devices run on JAVA which makes the end users used to it.
Lastly, it can be readily portable to any devices like mobile phones, ipads, PDA, and any hand held devices that are capable of running JAVA.

b. Java's view architecture

The interface provided by this application is very user friendly and is developed using Java Swings.

7.2.2 Technical Aspect

The technical aspect of implementation of the project is realized as explained below:

7.2.2.1 Servers

Apache Tomcat to develop the product

Apache Tomcat is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

JBOSS Application server to host the product

WildFly, formerly known as JavaBeans Open Source Software Application Server (JBoss AS, or simply JBoss) is an application server that implements the Java Platform, Enterprise Edition (Java EE). JBoss is written in Java and as such is cross-platform: usable on any operating system that supports Java. JBoss was developed by JBoss, now a division of Red Hat. Licensed under the terms of the GNU Lesser General Public License, JBoss is free and open source software.

7.2.2.2 Database

MySQL officially, but also called /maɪ ˈsiːkwəl/ "My Sequel" is (as of 2008) the world's most widely used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google, Facebook, Twitter, Flickr, Nokia.com, and YouTube.

7.3 Implementation Support

7.3.1 Installation of Eclipse

The following steps should be followed to install eclipse:

- Installation of JVM: Regardless of the operating system, some Java virtual machine (JVM) has to be installed. A Java Runtime Environment (JRE), or a Java Development Kit (JDK) can be installed depending on what is to be done with Eclipse. If Eclipse is intended for Java development, then a JDK has to be installed.
- Download Eclipse from the Eclipse Downloads Page.
- The download will be delivered as a compressed (i.e. a ".zip", or ".tar.gz") file. Decompress this file into the directory of your choice (e.g. "c:\Program Files\Eclipse Indigo" on Windows). You can optionally create a shortcut of the executable file ("eclipse.exe" on Windows, or "eclipse" on Linux).

7.3.2 Installation of Apache Tomcat Server

The following steps should be followed to install Apache Tomcat in Eclipse:

- If Apache Tomcat is not present on the machine, it has to be downloaded and unzipped.
- Start the Eclipse WTP workbench.

- Open Window -> Preferences -> Server -> Installed Runtime to create a Tomcat installed runtime.
- Click on Add to open the New Server Runtime dialog, then select your runtime under Apache (Apache Tomcat v7.0 in this project).
- Ensure the selected JRE is a full JDK and is of a version that will satisfy Apache Tomcat (this scenario was written using SUN JDK 1.6.029). If necessary, you can click on Installed JREs to add JDKs to Eclipse.
- Click Finish. Click Next, and fill in your Tomcat installation directory.

7.3.3 Installation of MySQL Database

- Log into the computer as an administrator
- Download the free MySQL Server Community Edition
- Double-click the downloaded file
- Double-click Setup.exe
- Click Next.
- Click Custom > Next
- Click Install
- Click Skip Sign-Up and then Next
- Configure MySQL.
- Click Next
- Check Standard Configuration and then click Next
- Make sure Install As Windows Service and Launch the MySQL Server Automatically are hecked, then click Next.
- Create a root password. Type in what you want your root password to be and make sure Enable root access from remote machines is checked. Make sure you choose a difficult to guess password and 'write it down so you don't forget it. Click Next.

- Click Execute. This will start the MySQL server. After MySQL has done its thing, click Finish.

- From the Windows task bar, go to Start > All Programs > MySQL > MySQL Server 4.x > MySQL Command line client. This will open a command window asking you for a password.

- Enter your root password and hit Enter. This should initiate the program.

7.3.4 Installing JBOSS server

- JBoss application server can be freely downloaded from the community site: http://www.jboss.org/jbossas/downloads/.

- Installing JBoss AS is a piece of cake: it does not require anything else besides unpacking the archive jboss-as-7.1.1.Final.zip.

- Starting up JBoss AS

  After you have installed JBoss, it is wise to perform a simple startup test to validate that there are no major problems with your Java VM/operating system combination. To test your installation, move to the bin directory of your JBOSS_HOME directory and issue the following command:

  ```
  standalone.bat   # Windows users
  ```

Chapter 8

PSEUDOCODE

8.1 Introduction

Pseudo code is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudo code typically omits details that are not essential for human understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation.

The purpose of using pseudo code is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudo code syntax exists, as a program in pseudo code is not an executable program. Pseudo code resembles, but should not be confused with skeleton programs, including dummy code, which can be compiled without errors. Flowcharts and Unified Modeling Language (UML) charts can be thought of as a graphical alternative to pseudo code, but are more spacious on project.

8.2 Pseudo codes

This contains several folders that make up to an entire plugin. Each folder has a specific function that helps in the functioning of the plugin.

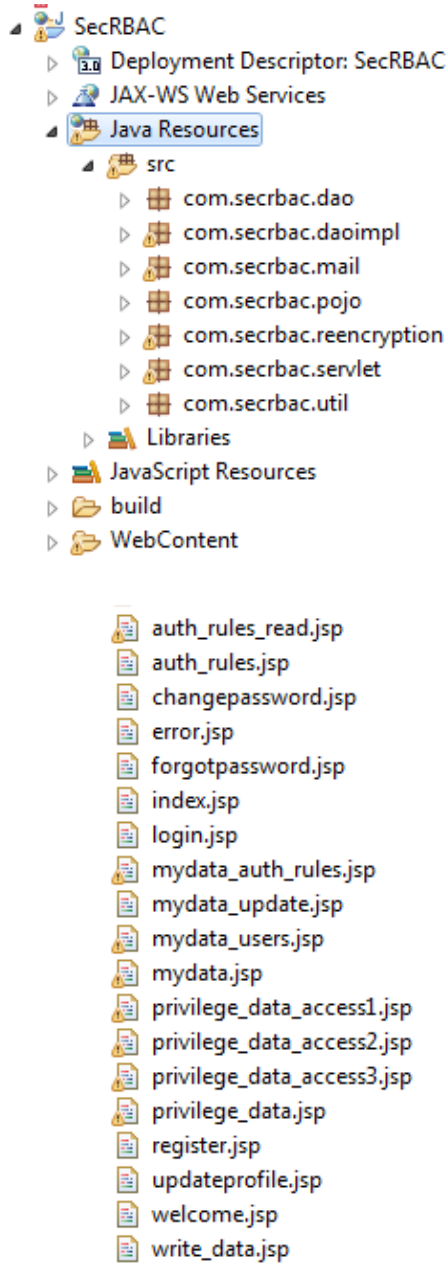


Fig 14. Pseudo codes

The files written under the folder named 'Java Resources' constitutes the back end logic. The files written under the folder named 'WebContent' constitutes the front end logic. The mapping from front end request to the backend servlet will be written inside web.xml

Note: Data access layer is the collection of java programs which exposes all the possible operations on the data base to the outside world.

8.2.1 Package: com.secrbac.dao

This package contains all the DAO interface programs

This is the DAO interface for the Authorization table

This is the DAO interface for the Data table

This is the DAO interface for the Privileged data table

This is the DAO interface for the User table

8.2.2 Package: com.secrbac.daoimpl

This package contains all the DAO implementation programs

This is the DAO implementation for the Authorization table

This is the DAO implementation for the Data table

This is the DAO implementation for the Privileged data table

This is the DAO implementation for the User table

8.2.3 Package: com.secrbac.mail

This package contains the implementation for triggering an email from the system.

This program will create a new thread for triggering an email

This programs provides the sender's email credentials for Google SMTP server for sending out the mails

This programs actually triggers an email through the SMTP protocol

8.2.4 Package: com.secrbac.pojo

This package contains all the POJOs of our project

This POJO represents a role within a Authorization Rule

This program is the POJO to the Data table

This program is the POJO to the User table

8.2.5 Package: com.secrbac.reencryption

This package contains the RSA implementation for generating the Keys, performing encryption, and decryption operations

This program implements the Decryption operation using RSA

```
Public class Decrypt {

	Public static String decrypt(String encMsg, BigInteger n, BigInteger private_key)
throws Exception {
		return new String(decryptMessage(new BigInteger(encMsg).toByteArray(),
private_key, n));

	}

	private static String bytesToString(byte[] encrypted) {

		String test = "";

		for (byte b : encrypted) {
			test += Byte.toString(b);
		}

		returntest;

	}

	public static byte[] decryptMessage(byte[] message, BigInteger private_key,
BigInteger n) { // Actual
		// Message
		// Decryption

		return (new BigInteger(message)).modPow(private_key, n).toByteArray();

	}

}
```

This program implements the Encryption operation using RSA

```
Public class Encrypt {

	// Encrypt message
	public static String encrypt(String message, BigInteger n, BigInteger public_key)
throws Exception {
```

```
            BigInteger bi = new BigInteger(messageEncrypt(message.getBytes(), pub-
lic_key, n));
            returnbi.toString();

//          return new String (messageEncrypt(message.getBytes("UTF-8"), public_key,
n), "US-ASCII");
      }

      privatestaticbyte[] messageEncrypt(byte[] message, BigInteger public_key, BigIn-
teger n) {
            return (new BigInteger(message)).modPow(public_key, n).toByteArray();

      }

      privatestatic String bytesToString(byte[] encrypted) {

            String test = "";

            for (byte b : encrypted) {
                  test += Byte.toString(b);
                  // test +="";
            }

            return test;

      }

}
```

This program implements the Key Generation operation using RSA

```
Public class PKG {

      Public void set_key(String id) {

            RSA rsa = new RSA(id);
            try {
                  Connection con = MySQL.getConnection();


                  ResultSet rs = con.createStatement().executeQuery("select count(*)
            from ENCKEYS where id='" + id + "' ");
                  rs.next();
                  if(rs.getInt(1) == 0) {
                        PreparedStatement ps = con.prepareStatement("insert into
ENCKEYS values (?,?,?,?) ");
                        ps.setString(1, id);
                        ps.setString(2, rsa.get_public_key().toString());
                        ps.setString(3, rsa.get_private_key().toString());
                        ps.setString(4, rsa.getn().toString());
                        ps.execute();
                  }
                  con.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
```

```
	}

	public BigInteger get_public_key(String id) throws Exception {

		BigInteger pk = BigInteger.valueOf(-1);
		set_key(id);

		try {
			Connection con = MySQL.getConnection();
			ResultSet rs = con.createStatement().executeQuery("select publickey
from ENCKEYS where id='" + id + "' ");
			if (rs != null&&rs.next()) {
				pk = new BigInteger(rs.getString(1));
			}
			con.close();
			return pk;

		} catch (Exception e) {
			e.printStackTrace();
			throw e;
		}

	}

	public BigInteger get_private_key(String id) throws Exception { // Private
		// Key
		// generator

		BigInteger pk = BigInteger.valueOf(-1);
		set_key(id);

		try {
			Connection con = MySQL.getConnection();
			ResultSet rs = con.createStatement().executeQuery("select privatkey
from ENCKEYS where id='" + id + "' ");
			if (rs != null&&rs.next()) {
				pk = new BigInteger(rs.getString(1));
			}
			con.close();
			return pk;

		} catch (Exception e) {
			e.printStackTrace();
			throw e;
		}
	}

	public BigInteger get_N(String id) throws Exception {

		BigInteger n = null;
		set_key(id);

		try {
			Connection con = MySQL.getConnection();
			ResultSet rs = con.createStatement().executeQuery("select n from
ENCKEYS where id='" + id + "' ");
			if (rs != null&&rs.next()) {
				n = new BigInteger(rs.getString(1));
			}
			con.close();
			return n;
```

```
            } catch (Exception e) {
                  e.printStackTrace();
                  throw e;
            }
      }

}
```

This program implements the basic Euclidean distance, prime number calculations required by other operations.

```
Public class RSA {

      BigInteger public_key, private_key;

      Private long public_key_temp;

      private BigInteger p;
      private BigInteger q;
      private BigInteger n;
      private BigInteger phi;
      private BigInteger e;
      private BigInteger d;
      privateintbitlength = 1024;
      private Random r;

      String ID;

      RSA(String ID) {

            this.ID = ID;
            public_key_temp = Math.abs(ID.hashCode());

            r = new Random();
            p = BigInteger.probablePrime(bitlength, r);
            q = BigInteger.probablePrime(bitlength, r);
            n = p.multiply(q);

      }

      public BigInteger get_public_key() { // generating public key

            phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
            e = BigInteger.valueOf(public_key_temp);

            while (phi.gcd(e).compareTo(BigInteger.valueOf(1)) != 0) {
                  e = e.divide(phi.gcd(e));

            }

            public_key = e;
```

```
		get_private_key();

		return public_key;

	}

	public BigInteger get_private_key() { // generating private key

		d = public_key.modInverse(phi);

		private_key = extendedEuclid(public_key,
				(this.p.subtract(BigInteger.ONE)).multiply(this.q.sub-
tract(BigInteger.ONE)));

		returnprivate_key;
	}

	public BigInteger getn() {

		return n;

	}

	public BigInteger extendedEuclid(BigInteger a, BigInteger b) {

		BigInteger x = BigInteger.valueOf(1), y = BigInteger.valueOf(0);
		BigInteger xLast = BigInteger.valueOf(0), yLast = BigInteger.valueOf(0);
		BigInteger q, r, m, n;

		while (a.compareTo(BigInteger.valueOf(0)) != 0) {

			q = b.divide(a);
			r = b.remainder(a);
			m = xLast.subtract(q.multiply(x));
			n = yLast.subtract(q.multiply(y));

			xLast = x;
			yLast = y;

			x = m;
			y = n;
			b = a;
			a = r;

		}

		if (xLast.compareTo(BigInteger.valueOf(0)) < 0)
			xLast = xLast.add((this.p.subtract(BigInteger.ONE)).multi-
ply(this.q.subtract(BigInteger.ONE)));

		returnxLast;

	}

	publiclong power(longa, longb, longp) { // power funation a^b%p

		long r = 1;

		while (b!= 0) {

			if ((b& 1) != 0)
				r = r * a % p;
```

```
                a = (a * a) % p;
                b>>= 1;

            }

            return r;
      }

      public BigInteger gcd1(BigInteger x, BigInteger y) { // computing gcd

            if (y.compareTo(BigInteger.valueOf(0)) == 0)
                  return x;

            return gcd1(y, x.remainder(y));

      }

}
```

8.2.6 Package: com.secrbac.servlet

This packages contains all the servlets written within this project. A servlet is the one which sits in the web layer component of the J2EE architecture and processes the client requests.

This servlet process all the requests coming from Authorization module

```
Public class AuthServlet extends HttpServlet {


      Private static final long serialVersionUID = 1L;

      @Override
      protectedvoid doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
            doPost(req, resp);
      }
      @Override
      protectedvoid doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

            try {
                  String type = req.getParameter("type");
                  AuthDAO dao = new AuthDAOImpl();
                  User user = (User) req.getSession().getAttribute("user");
                  String email = user.getEmail();
                  if (type.equals("get")) {
                        req.setAttribute("auths", dao.getAll(email));
                        req.getRequestDispatcher("auth_rules_read.jsp").forward(req,
resp);
                  } elseif (type.equals("add_rule")) {
                        String rolenames[] = req.getParameterValues("rolename");
                        String access[] = req.getParameterValues("access");
```

```
                        String rule_id = AppUtil.generateRuleID();
                        for (inti = 0; i<rolenames.length; i++) {
                              AuthBean auth = new AuthBean();
                              auth.setRule_id(rule_id);
                              auth.setOwner_id(email);
                              auth.setRole_name(rolenames[i]);
                              if (access[i].equals("R")) {
                                    auth.setRead("Yes");
                                    auth.setDelete("No");
                                    auth.setUpdate("No");
                              } elseif (access[i].equals("RU")) {
                                    auth.setRead("Yes");
                                    auth.setDelete("No");
                                    auth.setUpdate("Yes");
                              } elseif (access[i].equals("RUD")) {
                                    auth.setRead("Yes");
                                    auth.setDelete("Yes");
                                    auth.setUpdate("Yes");
                              } else {
                                    auth.setRead("No");
                                    auth.setDelete("No");
                                    auth.setUpdate("No");
                              }
                              dao.addRule(auth);
                        }
                        resp.sendRedirect("auth_rules.jsp?msg=Successfully Defined
the New Set of Rules");
                  } elseif (type.equals("delete")) {
                        String rule_id = req.getParameter("rule_id");
                        dao.deleteRule(rule_id);
                        resp.sendRedirect("auth?type=get");
                  }
            } catch (Exception e) {
                  e.printStackTrace();
                  resp.sendRedirect("error.jsp");
            }
      }

}
```

This servlet process all the requests coming from Manage My data module

```
Public class DataServlet extends HttpServlet {

      Private static final long serialVersionUID = 1L;

      @Override
      protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
            doPost(req, resp);
      }

      @Override
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
             try {
                    DataDAO dao = new DataDAOImpl();
                    AuthDAO authDao = new AuthDAOImpl();
                    UserDAO userDao = new UserDAOImpl();
                    User user = (User) req.getSession().getAttribute("user");
                    String type = req.getParameter("type");
                    if (type.equals("write")) {
                           String desc = req.getParameter("desc");
                           String[] key_arr = req.getParameterValues("key");
                           String[] value_arr = req.getParameterValues("value");
                           dao.write_data(user.getEmail(), desc, key_arr, value_arr);
                           resp.sendRedirect("write_data.jsp?msg=Data Written Success-
fully");
                    } elseif (type.equals("getmydata")) {
                           req.setAttribute("data", dao.getData(user.getEmail()));
                           req.getRequestDispatcher("mydata.jsp").forward(req, resp);
                    } elseif (type.equals("delete")) {
                           String data_id = req.getParameter("data_id");
                           dao.delete_data(data_id);
                           resp.sendRedirect("data?type=getmydata");
                    } elseif (type.equals("update_get")) {
                           req.setAttribute("data", dao.getDataByID(req.getParame-
ter("data_id"), user.getEmail()));
                           req.getRequestDispatcher("mydata_update.jsp").forward(req,
resp);
                    } elseif (type.equals("update")) {
                           String desc = req.getParameter("desc");
                           String[] key_arr = req.getParameterValues("key");
                           String[] value_arr = req.getParameterValues("value");
                           String data_id = req.getParameter("data_id");
                           dao.update_data(data_id, user.getEmail(), desc, key_arr,
value_arr);
                           resp.sendRedirect("data?type=getmydata");
                    } elseif (type.equals("authrule_get")) {
                           String data_id = req.getParameter("data_id");
                           req.setAttribute("data_id", data_id);
                           req.setAttribute("desc", req.getParameter("desc"));
                           req.setAttribute("rules", dao.getAuthRule(data_id));
                           req.setAttribute("all_auth_rules", authDao.get-
All(user.getEmail()));
                           req.getRequestDispatcher("mydata_auth_ru-
les.jsp").forward(req, resp);
                    } elseif (type.equals("authrule_apply")) {
                           String data_id = req.getParameter("data_id");
                           String rule_id = req.getParameter("rule_id");
                           String desc = req.getParameter("desc");
                           dao.applyAuthRule(data_id, rule_id);
                           resp.sendRedirect("data?type=authrule_get&data_id=" + data_id
+ "&desc=" + desc);
                    } elseif (type.equals("mapusers_get")) {
                           String data_id = req.getParameter("data_id");
                           String desc = req.getParameter("desc");
                           List<String>availableusers = userDao.getAllRegisteredUsers();
                           Map<String, String>mappedusers = dao.getMappedUsers(data_id);
                           availableusers.removeAll(mappedusers.keySet());
                           availableusers.remove(user.getEmail());
                           req.setAttribute("rolenames", authDao.getRoleNames(dao.getRu-
leID(data_id)));
                           req.setAttribute("availableusers", availableusers);
                           req.setAttribute("users", mappedusers);
```

```
                        req.setAttribute("data_id", data_id);
                        req.setAttribute("desc", desc);
                        req.getRequestDispatcher("mydata_users.jsp").forward(req,
resp);
                } elseif (type.equals("mapuser")) {
                        String data_id = req.getParameter("data_id");
                        String desc = req.getParameter("desc");
                        String email = req.getParameter("email");
                        String role_name = req.getParameter("role");
                        dao.mapUser(data_id, email, role_name);
                        resp.sendRedirect("data?type=mapusers_get&data_id=" + data_id
+ "&desc=" + desc);
                } elseif (type.equals("mapusers_remove")) {
                        String data_id = req.getParameter("data_id");
                        String desc = req.getParameter("desc");
                        String email = req.getParameter("email");
                        dao.removeMappedUser(email, data_id);
                        resp.sendRedirect("data?type=mapusers_get&data_id=" + data_id
+ "&desc=" + desc);
                }
            } catch (Exception e) {
                    e.printStackTrace();
                    resp.sendRedirect("error.jsp");
            }
      }

}
```

This servlet process all the requests coming from Privileged data module

```
Public class PrivilegeDataServlet extends HttpServlet {

      @Override
      protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
            doPost(req, resp);
      }

      @Override
      protectedvoid doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
            try {
                  String type = req.getParameter("type");
                  User user = (User) req.getSession().getAttribute("user");
                  PrivilegeDataDAO dao = new PrivilegeDataDAOImpl();
                  DataDAO dataDAO = new DataDAOImpl();
                  if (type.equals("get")) {
                        req.setAttribute("data", dao.getPrivilegeData(user.ge-
tEmail()));
                        req.getRequestDispatcher("privilege_data.jsp").forward(req,
resp);
                  } elseif (type.equals("access")) {
                        String data_id = req.getParameter("data_id");
                        String desc = req.getParameter("desc");
                        intaccess = dao.checkPrivilege(data_id, user.getEmail());
```

```
                        req.setAttribute("access", access);
                        req.setAttribute("data_id", data_id);
                        req.setAttribute("desc", desc);
                        req.setAttribute("details", dao.getDataDetails(data_id));
                        req.getRequestDispatcher("privilege_data_ac-
cess1.jsp").forward(req, resp);
                  } elseif (type.equals("access2")) {
                        String id = req.getParameter("id");
                        String data_id = req.getParameter("data_id");
                        String desc = req.getParameter("desc");
                        String access = req.getParameter("access");
                        PKG pkg = new PKG();
                        Map<String, String>keyvalue = dataDAO.getDetails(data_id);
                        Map<String, String>keyvalue_enc = new HashMap<>();

                        Iterator<String>it = keyvalue.keySet().iterator();
                        while (it.hasNext()) {
                              String key = it.next();
                              String value = keyvalue.get(key);

                              keyvalue_enc.put(Encrypt.encrypt(key, pkg.get_N(id),
pkg.get_public_key(id)),
                                          Encrypt.encrypt(value, pkg.get_N(id),
pkg.get_public_key(id)));
                        }
                        SendMail mail = new SendMail();
                        List<String>to = new ArrayList<>();
                        to.add(user.getEmail());
                        mail.send(to, "Secret Key for Accessing the Privileged Data",
                                    "Dear " + user.getEmail()
                                                + ", <br/><br/> Use the below key
for accessing the privileged data<br/><br/>N = "
                                                + pkg.get_N(id) +
"<br/><br/><br/>Private key = " + pkg.get_private_key(id));
                        req.setAttribute("keyvalue_enc", keyvalue_enc);
                        req.setAttribute("data_id", data_id);
                        req.setAttribute("access", access);
                        req.setAttribute("desc", desc);
                        req.getRequestDispatcher("privilege_data_ac-
cess2.jsp").forward(req, resp);

                  } elseif (type.equals("access3")) {
                        String n = req.getParameter("n");
                        String secretkey = req.getParameter("secretkey");
                        String data_id = req.getParameter("data_id");
                        String desc = req.getParameter("desc");
                        String access = req.getParameter("access");
                        String key[] = req.getParameterValues("key");
                        String val[] = req.getParameterValues("value");
                        Map<String, String>keyvalue_map = new HashMap<>();
                        for (inti = 0; i<key.length; i++) {
                              keyvalue_map.put(Decrypt.decrypt(key[i], new BigInte-
ger(n), new BigInteger(secretkey)),
                                          Decrypt.decrypt(val[i], new BigInte-
ger(n), new BigInteger(secretkey)));
                        }
                        req.setAttribute("data_id", data_id);
                        req.setAttribute("desc", desc);
                        req.setAttribute("access", access);
                        req.setAttribute("keyvalue_map", keyvalue_map);
                        req.getRequestDispatcher("privilege_data_ac-
cess3.jsp").forward(req, resp);
```

```
                    } elseif (type.equals("delete")) {
                          String data_id = req.getParameter("data_id");
                          dataDAO.delete_data(data_id);
                          resp.sendRedirect("privilegeData?type=get");
                    } elseif (type.equals("update")) {
                          String desc = req.getParameter("desc");
                          String data_id = req.getParameter("data_id");
                          String access = req.getParameter("access");
                          String key[] = req.getParameterValues("key");
                          String val[] = req.getParameterValues("value");
                          String owner_id = dataDAO.getOwnerID(data_id);
                          dataDAO.update_data(data_id, owner_id, desc, key, val);
                          req.setAttribute("data_id", data_id);
                          req.setAttribute("desc", desc);
                          req.setAttribute("access", access);
                          Map<String, String>keyvalue_map = dataDAO.getDe-
tails(data_id);
                          req.setAttribute("keyvalue_map", keyvalue_map);
                          req.getRequestDispatcher("privilege_data_ac-
cess3.jsp").forward(req, resp);
                    }
              } catch (Exception e) {
                    e.printStackTrace();
                    resp.sendRedirect("error.jsp");
              }
       }

}
```

This servlet process all the requests coming from Account Operations module

```
Public class UserServlet extends HttpServlet {
       Private static final long serialVersionUID = 1L;

       @Override
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
              UserDAO dao = new UserDAOImpl();
              try {
                    String request_type = req.getParameter("request_type");
                    if (request_type.equals("register")) {
                          User user = new User();
                          String addr = req.getParameter("addr");
                          user.setAddr(addr);
                          String email = req.getParameter("email");
                          user.setEmail(email);
                          String fname = req.getParameter("fname");
                          user.setFname(fname);
                          String lname = req.getParameter("lname");
                          user.setLname(lname);
                          String gender = req.getParameter("gender");
                          user.setGender(gender);
                          String mobile = req.getParameter("mobile");
                          user.setMobile(mobile);
                          String password = req.getParameter("password");
                          user.setPassword(password);
```

```
                        String role = req.getParameter("role");
                        if (role == null || role.trim().length() == 0)
                              role = "USER";
                        user.setRole(role);

                        if ((addr == null || addr.trim().length() == 0) || (email ==
null || email.trim().length() == 0)
                                    || (fname == null || fname.trim().length() == 0)
                                    || (lname == null || lname.trim().length() == 0)
                                    || (mobile == null || mobile.trim().length() ==
0)
                                    || (role == null || role.trim().length() == 0)
                                    || (gender == null || gender.trim().length() ==
0)
                                    || (password == null || password.trim().length()
== 0)) {
                              resp.sendRedirect(
                                          "register.jsp?msg=Error! All the fields
are mandatory. Please provide the details.");
                        } else {

                              dao.register(user);
                              resp.sendRedirect("register.jsp?msg=Registration Suc-
cessful");
                        }
                  } elseif (request_type.equals("login")) {
                        String email = req.getParameter("email");
                        String password = req.getParameter("password");
                        User user = dao.getUserDetails(email, password);
                        if (email == null || email.trim().length() == 0 || password
== null || password.trim().length() == 0) {
                              resp.sendRedirect("login.jsp?msg=Error! All the fields
are mandatory. Please provide the details");
                        } elseif (user != null) {
                              req.getSession().setAttribute("user", user);
                              resp.sendRedirect("welcome.jsp?msg=Successfully logged
in as " + user.getFname() + ""
                                          + user.getLname() + " (" + user.getRole()
+ ") ");
                        } else {
                              resp.sendRedirect("login.jsp?msg=Invalid Creden-
tials");
                        }
                  } elseif (request_type.equals("updateprofile")) {
                        User user = new User();

                        String addr = req.getParameter("addr");
                        String email = req.getParameter("email");
                        String fname = req.getParameter("fname");
                        String lname = req.getParameter("lname");
                        String gender = req.getParameter("gender");
                        String mobile = req.getParameter("mobile");
                        String role = req.getParameter("role");

                        if (role == null || role.trim().length() == 0)
                              role = "USER";

                        user.setAddr(addr);
                        user.setEmail(email);
                        user.setFname(fname);
                        user.setLname(lname);
                        user.setGender(gender);
```

```
                        user.setMobile(mobile);
                        user.setRole(role);

                        if ((addr == null || addr.trim().length() == 0) || (email ==
null || email.trim().length() == 0)
                                        || (fname == null || fname.trim().length() == 0)
                                        || (lname == null || lname.trim().length() == 0)
                                        || (mobile == null || mobile.trim().length() ==
0)
                                        || (role == null || role.trim().length() == 0)
                                        || (gender == null || gender.trim().length() ==
0)) {
                                resp.sendRedirect(
                                                "updateprofile.jsp?msg=Error! All the
fields are mandatory. Please provide the details");

                        } else {

                                dao.updateProfile(user);
                                req.getSession().setAttribute("user", user);
                                resp.sendRedirect("updateprofile.jsp?msg=Profile Up-
dated Successfully");
                        }

                } elseif (request_type.equals("changepassword")) {
                        String oldpassword = req.getParameter("oldpassword");
                        String newpassword = req.getParameter("newpassword");

                        if (oldpassword == null || oldpassword.trim().length() == 0
|| newpassword == null
                                        || newpassword.trim().length() == 0) {
                                resp.sendRedirect(
                                                "changepassword.jsp?msg=Error! All the
fields are mandatory. Please provide the details");
                        } else {

                                booleanresult = dao.changePassword(((User) req.getSes-
sion().getAttribute("user")).getEmail(),
                                                oldpassword, newpassword);

                                if (result) {
                                        resp.sendRedirect("changepassword.jsp?msg=Suc-
cessfully Updated Your Password");
                                } else {
                                        resp.sendRedirect("changepassword.jsp?msg=Your
Current Password is Wrong");

                                }
                        }
                } elseif (request_type.equals("deleteprofile")) {
                        dao.deleteProfile(((User) req.getSession().getAttri-
bute("user")).getEmail());
                        req.getSession().invalidate();
                        resp.sendRedirect("login.jsp?msg=Profile Deleted Success-
fully");
                } elseif (request_type.equals("forgotpassword")) {
                        String email = req.getParameter("email");
                        if (email == null || email.trim().length() == 0) {
                                resp.sendRedirect("forgotpassword.jsp?msg=Please enter
your email ID");
                        } else {
```

```
                                String password = dao.forgotPassword(email);

                                if (password != null) {
                                       List<String>rcv = new ArrayList<>();
                                       rcv.add(email);
                                       new MailThread(
                                                      "<b>Dear " + email + ",</b><br>
Your Current Password is: <b>" + password + "</b>",
                                                      "Password Recovery", rcv);
                                       resp.sendRedirect("forgotpassword.jsp?msg=Pass-
word Recovery mail sent to your email address");
                                } else {
                                       resp.sendRedirect("forgotpassword.jsp?msg=Email
ID did not match with any account");
                                }
                         }
                  } elseif (request_type.equals("logout")) {
                         req.getSession().invalidate();
                         resp.sendRedirect("login.jsp?msg=Successfully Logged Out");
                  }
            } catch (Exception e) {
                  e.printStackTrace();
                  resp.sendRedirect("error.jsp?msg=OOPS! Something went wrong");
            }
      }

      @Override
      protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
            doGet(req, resp);
      }

}
```

8.2.7 Package: com.secrbac.util

This package contains all the commonly used utility applications

This program generates the Rule Identifier and Data Identifier

This program obtains the connection to the database and returns back to the caller.

Chapter 9

TESTING

9.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. The system has been verified and validated by running the test data and live data.

9.2 Levels of Testing

9.2.1 Unit Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method.

For unit testing first we adopted the code testing strategy, which examined the logic of program. During the development process itself all the syntax errors etc. got rooted out. For this developed test case that result in executing every instruction in the program or module i.e. every path through program was tested. Test cases are data chosen at random to check every possible branch after all the loops.

Test cases

Table 3. Test cases for the project

Steps	Test Action	Results
Step 1	Enter the URL http://localhost:8080/SecRBAC/index.jsp	Home page loaded successfully
Step 2	Click on Register	Register page loaded successfully
Step 3	Register as a new account	User registered successfully
Step 4	Click on Login and try logging in for all the users	All users could login successfully
Step 5	Click on Auth Rules and test adding a new Auth Rule and test viewing the created Auth rules	Adding the new Auth rule and Viewing the existing Auth rule, both are working fine
Step 6	Click on Write Data and create a new data by providing the description and entering multiple key value pairs	Writing the data on to the cloud DB is working fine
Step 7	Click on Manage my data	All the data we created are shown successfully
Step 8	Click on Auth rules against data and try setting a auth rule	It worked fine
Step 9	Click on Map users against data and try mapping a user on any of the role	It worked fine
Step 10	Click on Update and try editing the data	It worked fine
Step 11	Click on Delete and try deleting the data	It worked fine

Step 12	Click on My Privileged Data	All the data for which I have privileges shown up successfully
Step 13	Try accessing the privileged data	It worked fine
Step 14	Try executing the proxy re-encryption algorithm by providing a unique identifier	Proxy re-encryption algorithm worked fine
Step 15	Test the basic account operations	It all worked fine

9.2.1.1 User Input

User will be inputting all the data from all both the portal using a web browser.

9.2.1.2 Error Handling

In this system we have tried to handle all the errors that occurred while running the application. the common errors we saw were reading a tuple with an attribute set to null and database connection getting lost.

For Testing we used Top-Down design a decomposition process which focuses as the flow of control, at latter strategies concern itself with code production. The first step is to study the overall aspects of the tasks at hand and break it into a number of independent modules. The second step is to break one of these modules further into independent sub modules. One of the important features is that each level the details at lower levels are hidden. So unit testing was performed first and then system testing.

9.2.2 Integration Testing

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined may not produce the desired functions. Integrated testing is the systematic testing to uncover the errors with an interface. This testing is done with simple data and developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance.

Steps to perform integration testing:

Step 1: Create a Test Plan

Step 2: Create Test Cases and Test Data

Step 3: Once the components have been integrated execute the test cases

Step 4: Fix the bugs if any and re test the code

Step 5: Repeat the test cycle until the components have been successfully integrated

Table 4. Test cases for integration testing

Name of the Test	**Integration testing**
Test plan	To check whether the system works properly when all the modules are integrated.
Test Data	Sample data to be written on cloud db

9.2.3 System testing

Ultimately, software is included with other system components and the set of system validation and integration tests are performed. System testing is a series of different tests whose main aim is to fully exercise the computer-based system. Although each test has a different role all work should verify that all system elements are properly integrated and formed allocated functions.

Table 5. Test cases for Input-Output

Name of the Test	**System Testing**
Item being tested	Over all functioning of GUI with all functions properly linked.
Sample Input	Sample data to be written on cloud db
Expected Output	All the modules like Auth Rules, Privileged data, etc
Actual Output	Application reacts to user inputs in expected manner.

Remarks	Successful

9.2.4 Validation Testing

At the culmination of black box testing, software is completely assembled is as a package. Interfacing errors have been uncovered and the correct and final series of tests, i.e., validation tests begins. Validation test is defined with a simple definition that validation succeeds when the software function in a manner that can be reasonably accepted by the customer.

9.2.5 Output Testing

After performing validation testing, the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system is required tests the output displayed or generated by the system under consideration. The output format is considered in two ways, one is on screen format and the other is printed format. The output format on the screen is found to be corrected as the format was designated in the system has according to the user needs. As for the hard copy the output comes according to the specification requested by the user. The output testing does not result in any correction in the system.

9.2.6 Test data and Output:

Taking various kind soft data plays a vital role in system testing. After preparing the test data system under study is tested using the test data. While testing, errors are again uncovered and corrected by using the above steps and corrections are also noted for future use.

9.2.7 User acceptance Testing:

User acceptance testing of the system is the key factor for the success of the system. A system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system at the time of development and making change whenever required. This is done with regard to the input screen design and output screen design.

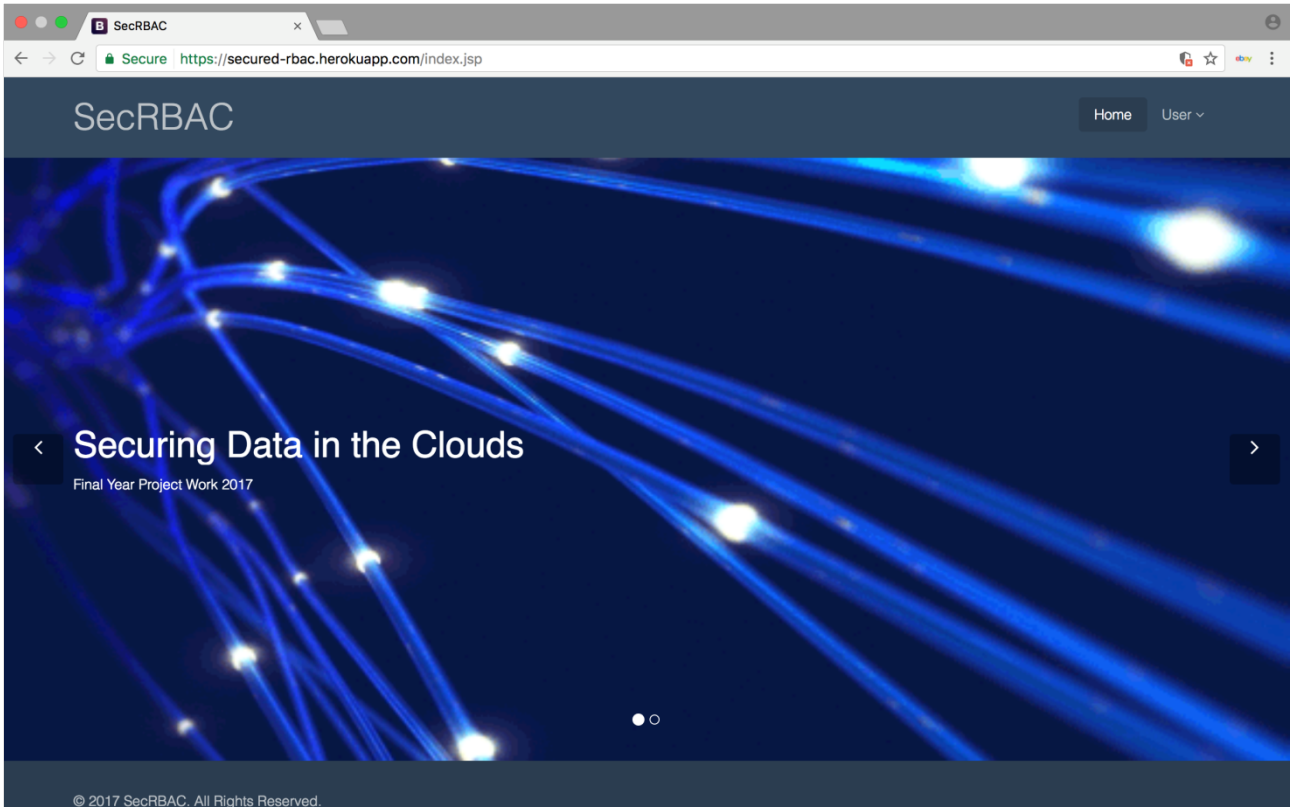
9.2.8 GUI Testing:

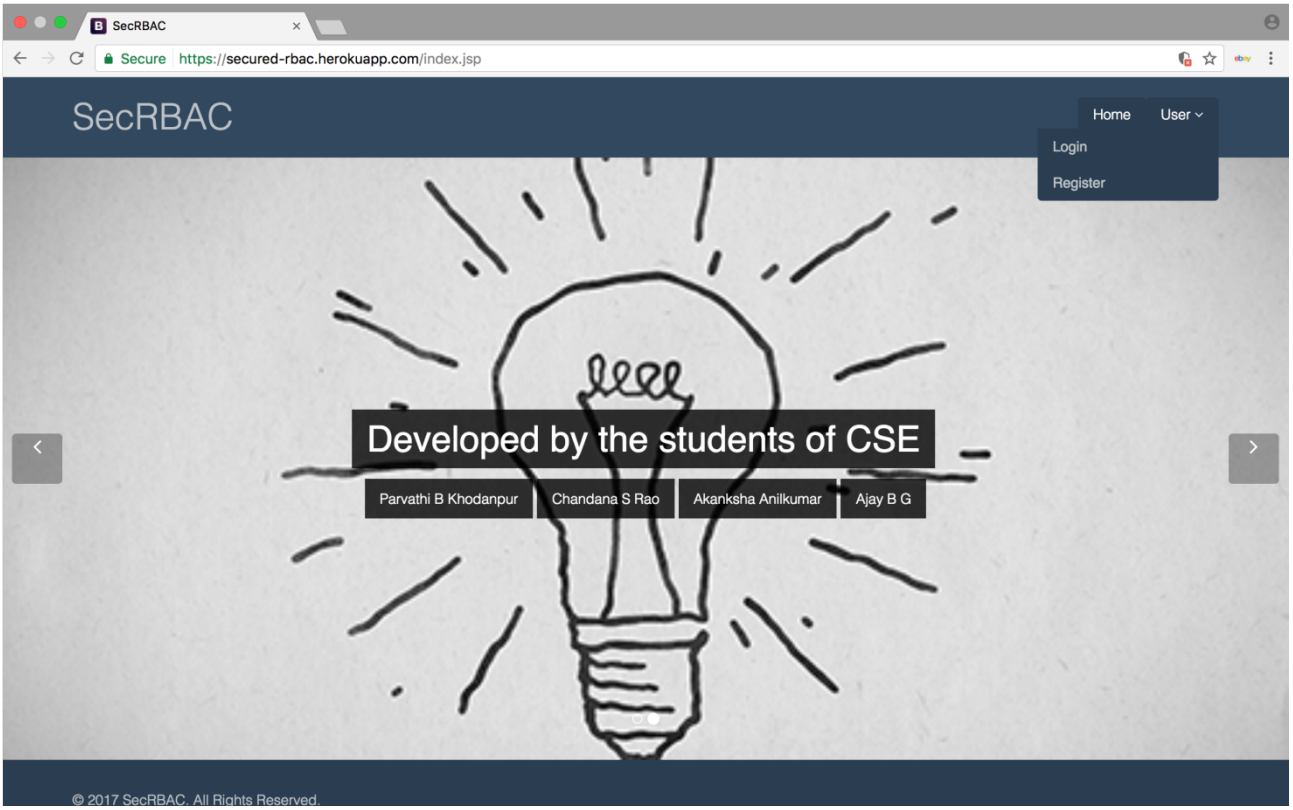
GUI testing is use to ensure the visual clarity of the system, flexibility of the system, user friendliness of the system. The various components which are to be tested are relative layout and, various Links and buttons.

Chapter 10

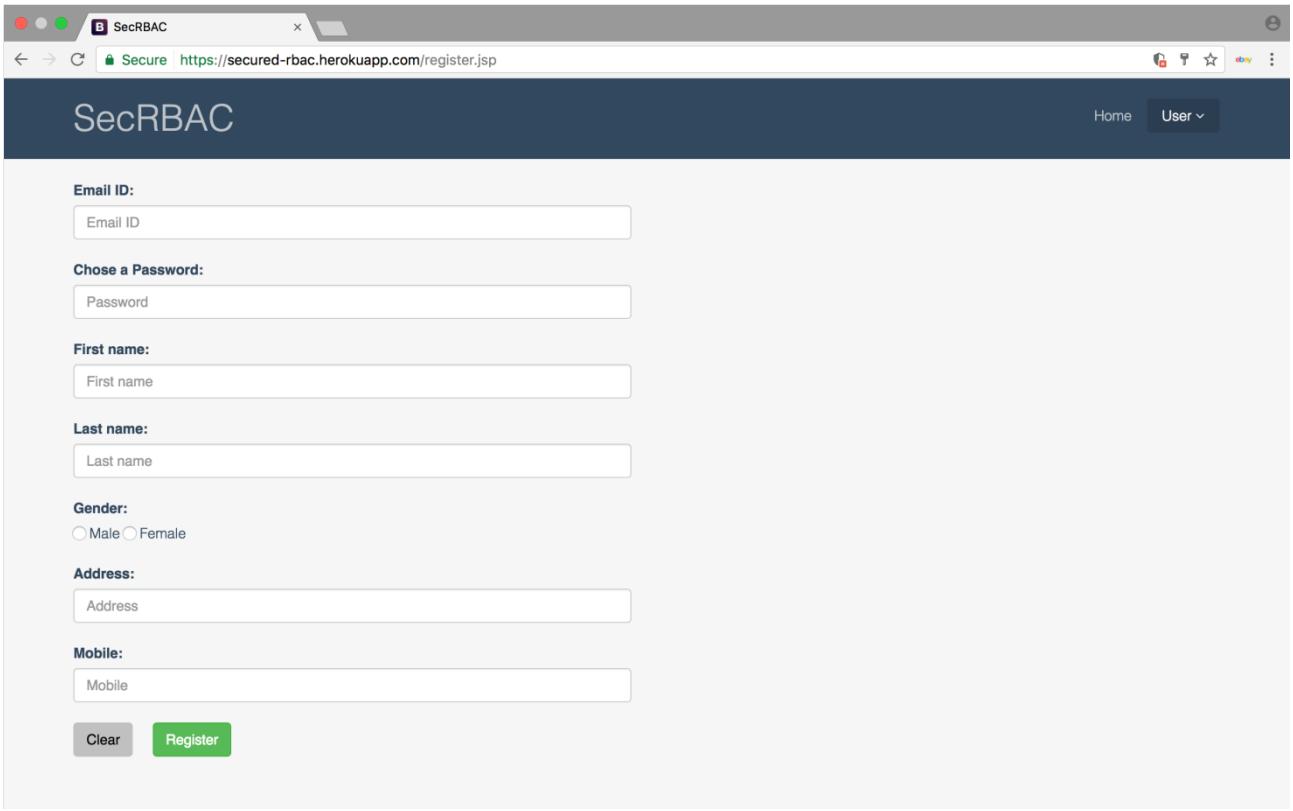
RESULTS

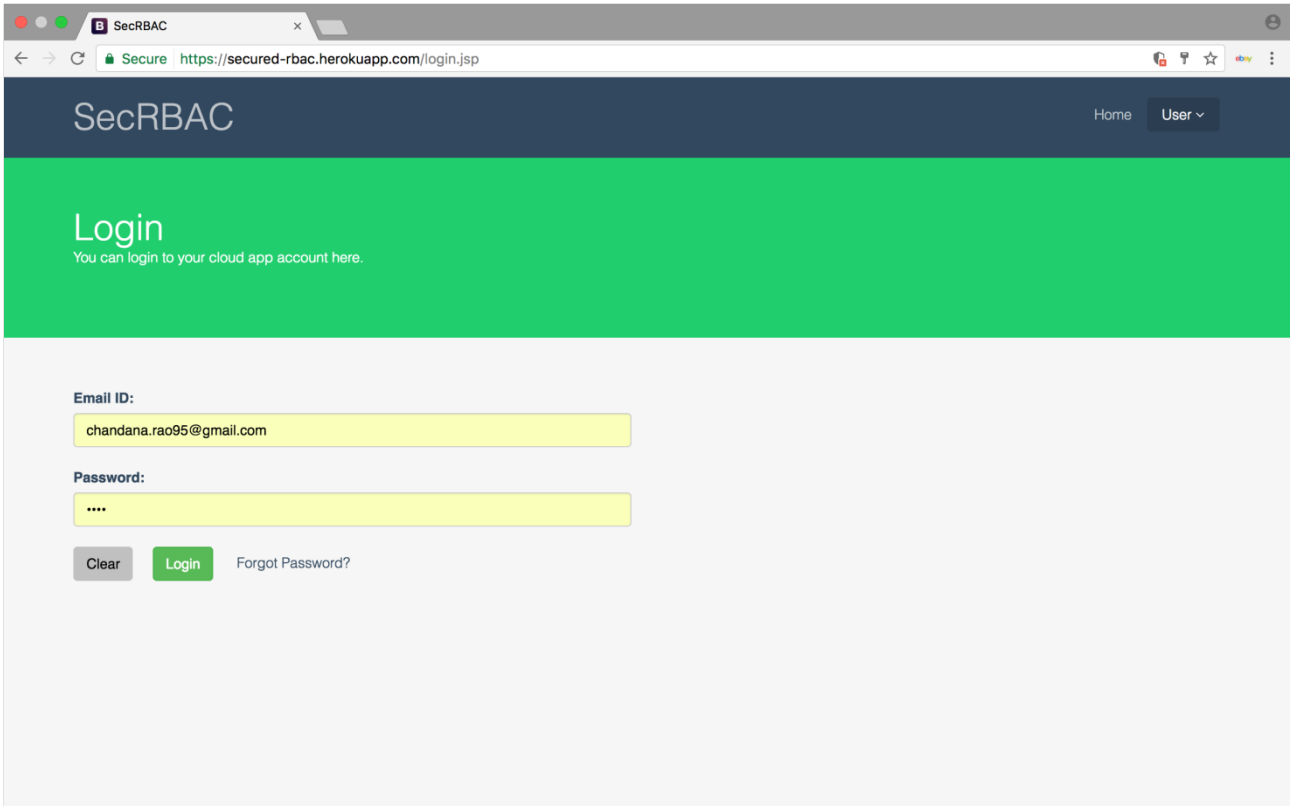
10.1 Homepage



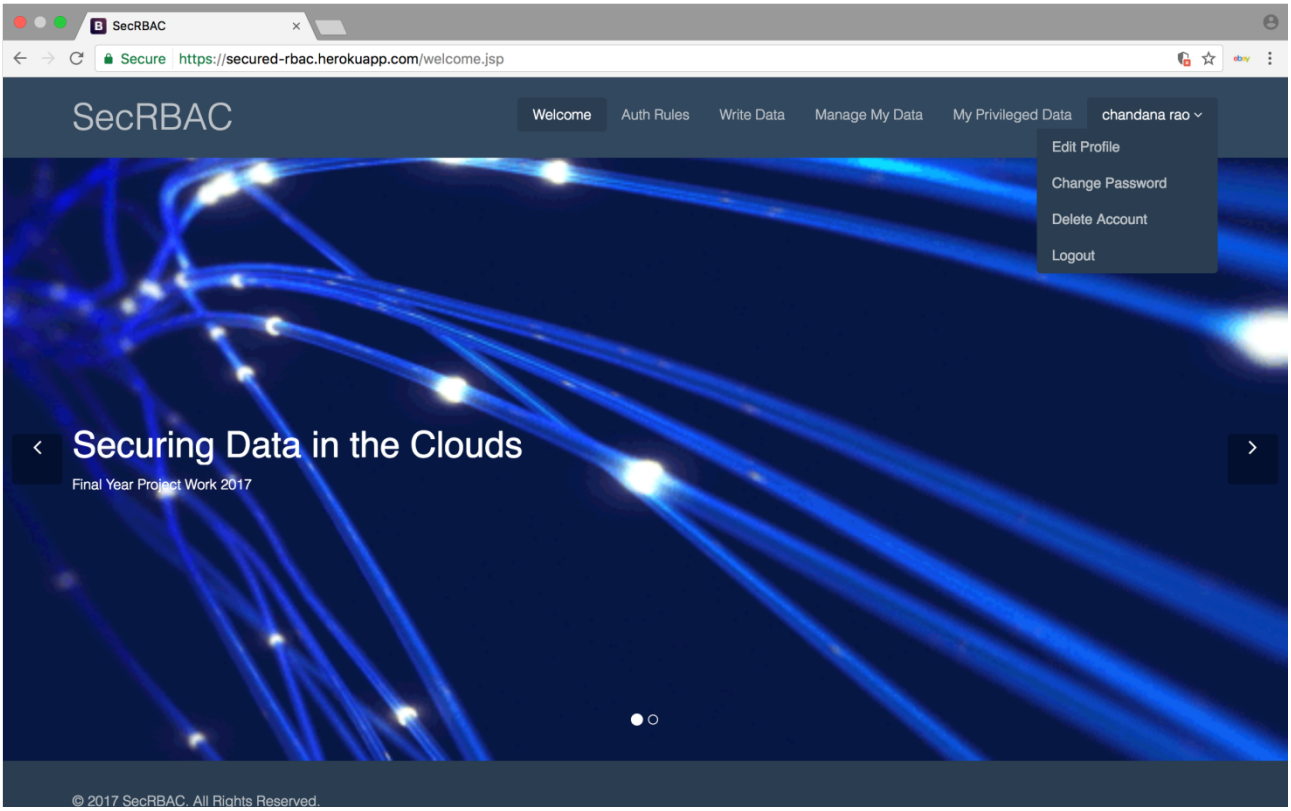


10.2 Register and Login

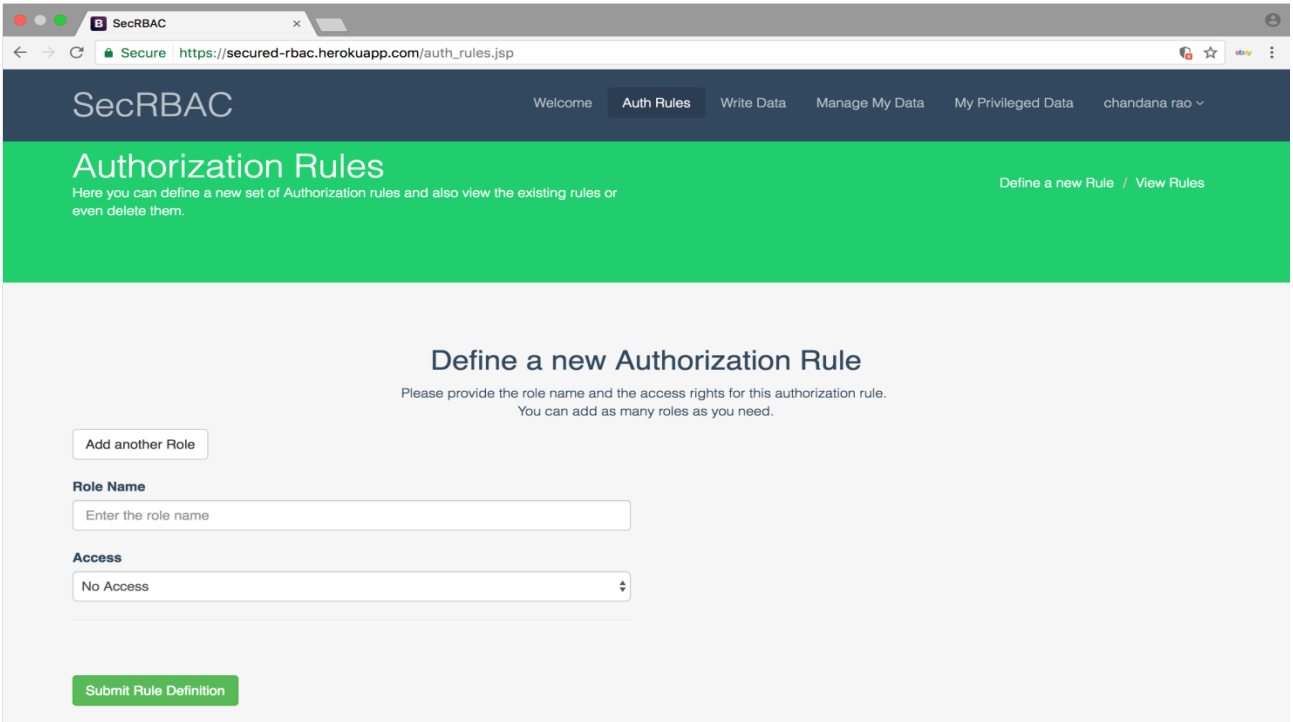




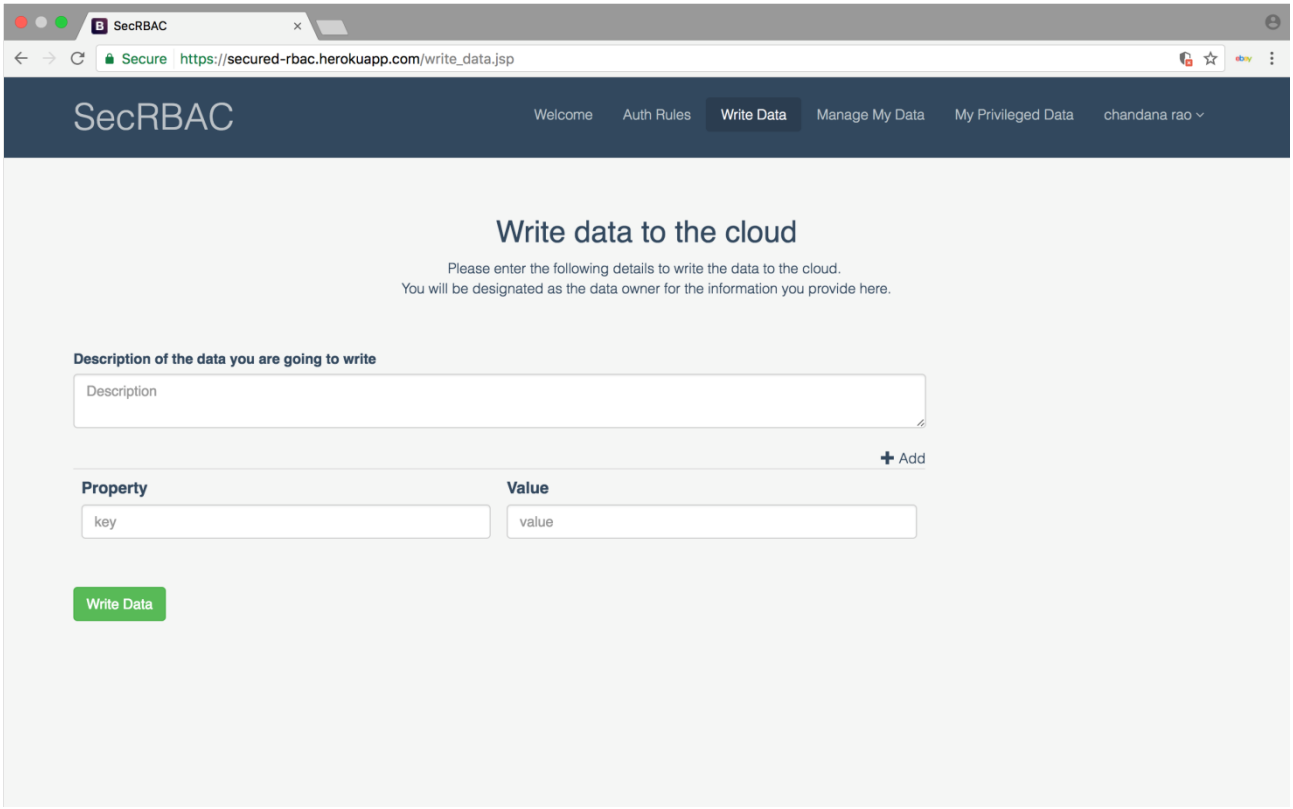
10.3 User Operations



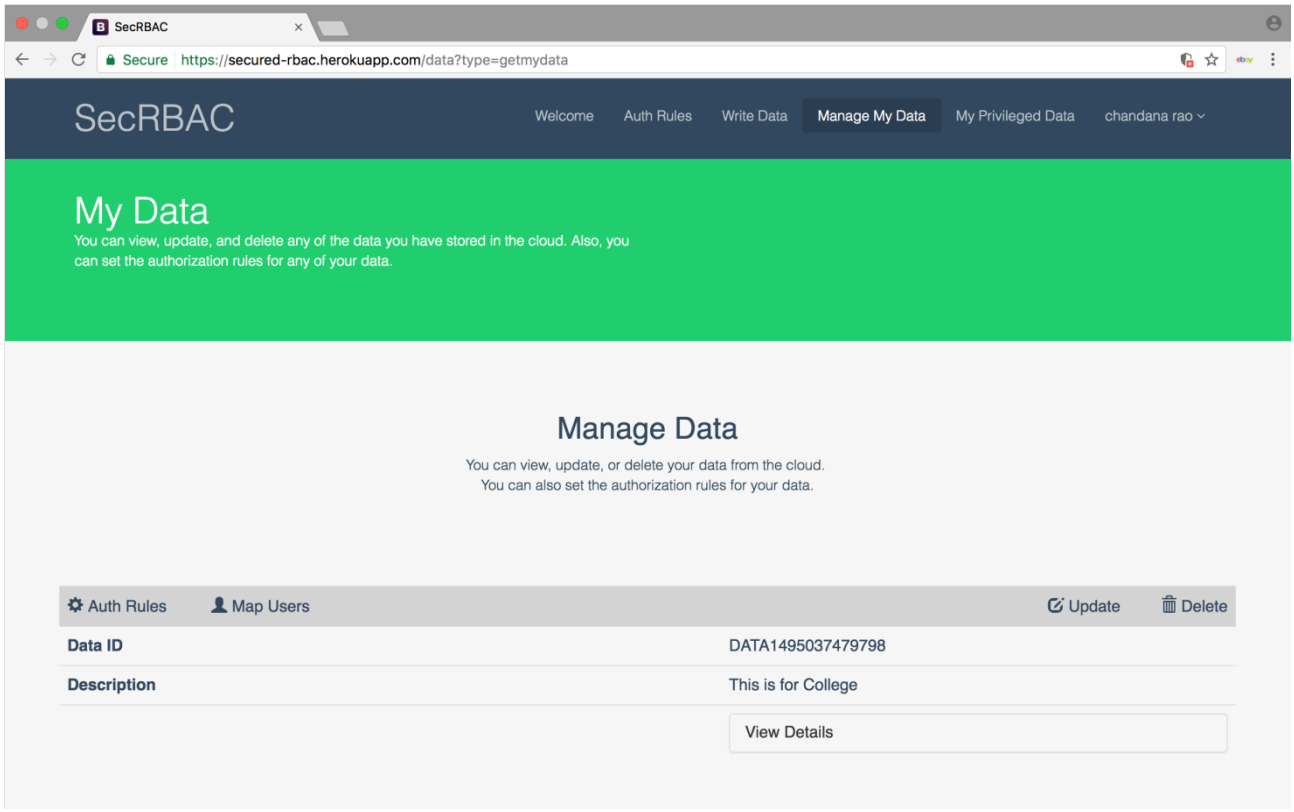
10.4 Auth Rules



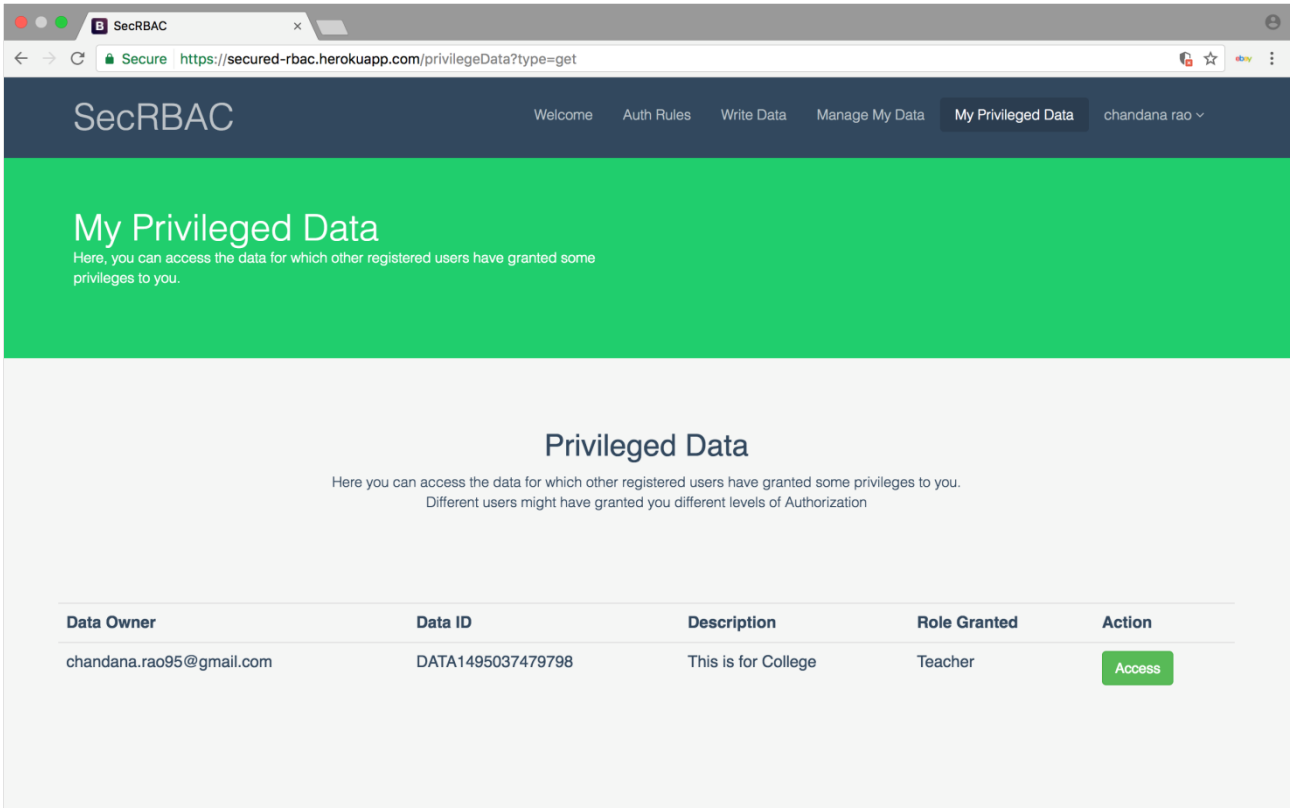
10.5 Write Data

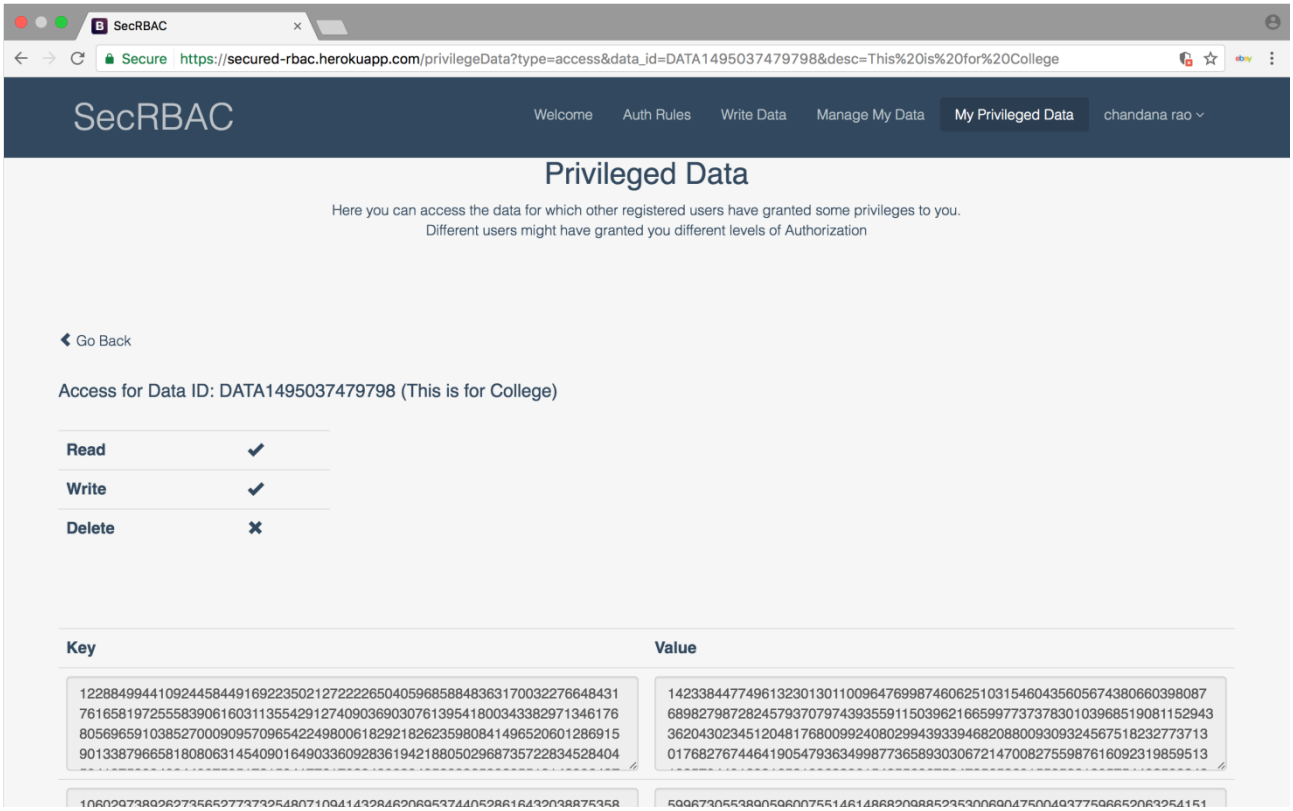


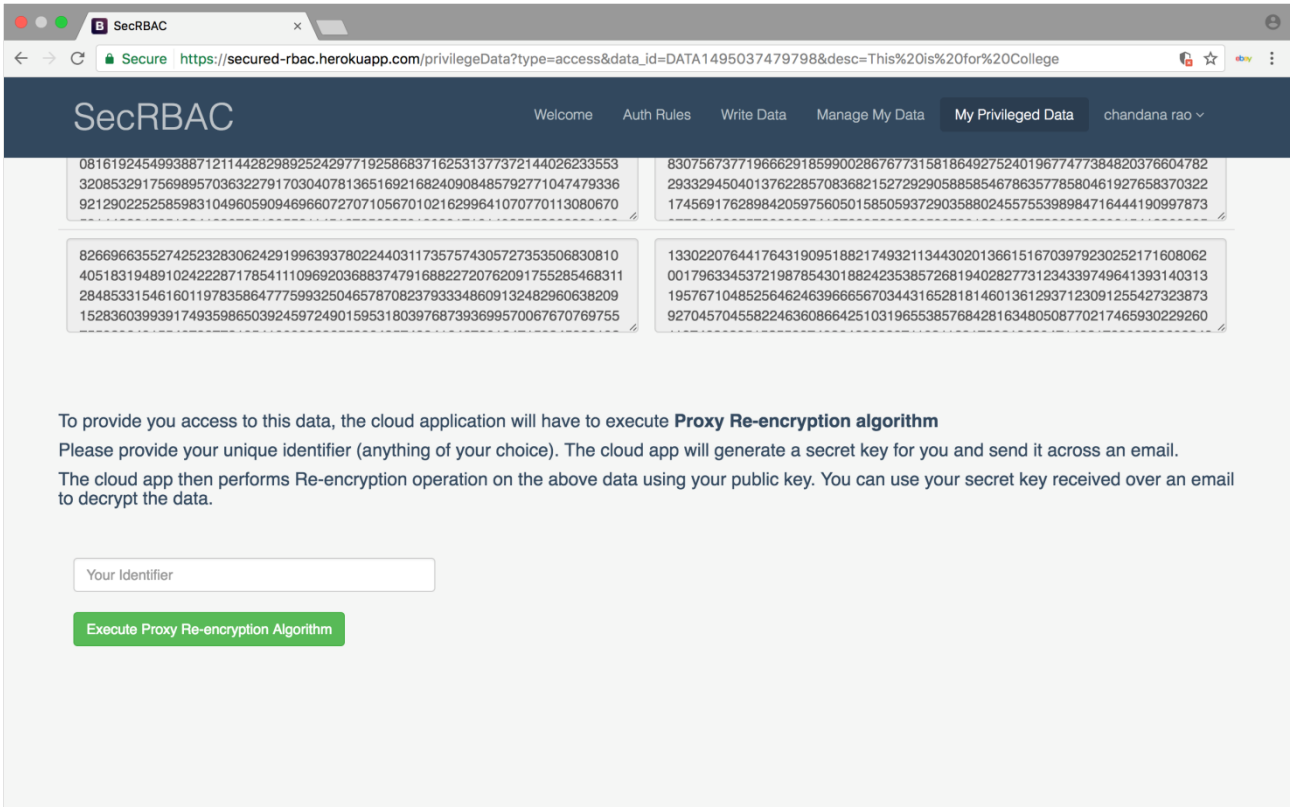
10.6 Manage My Data

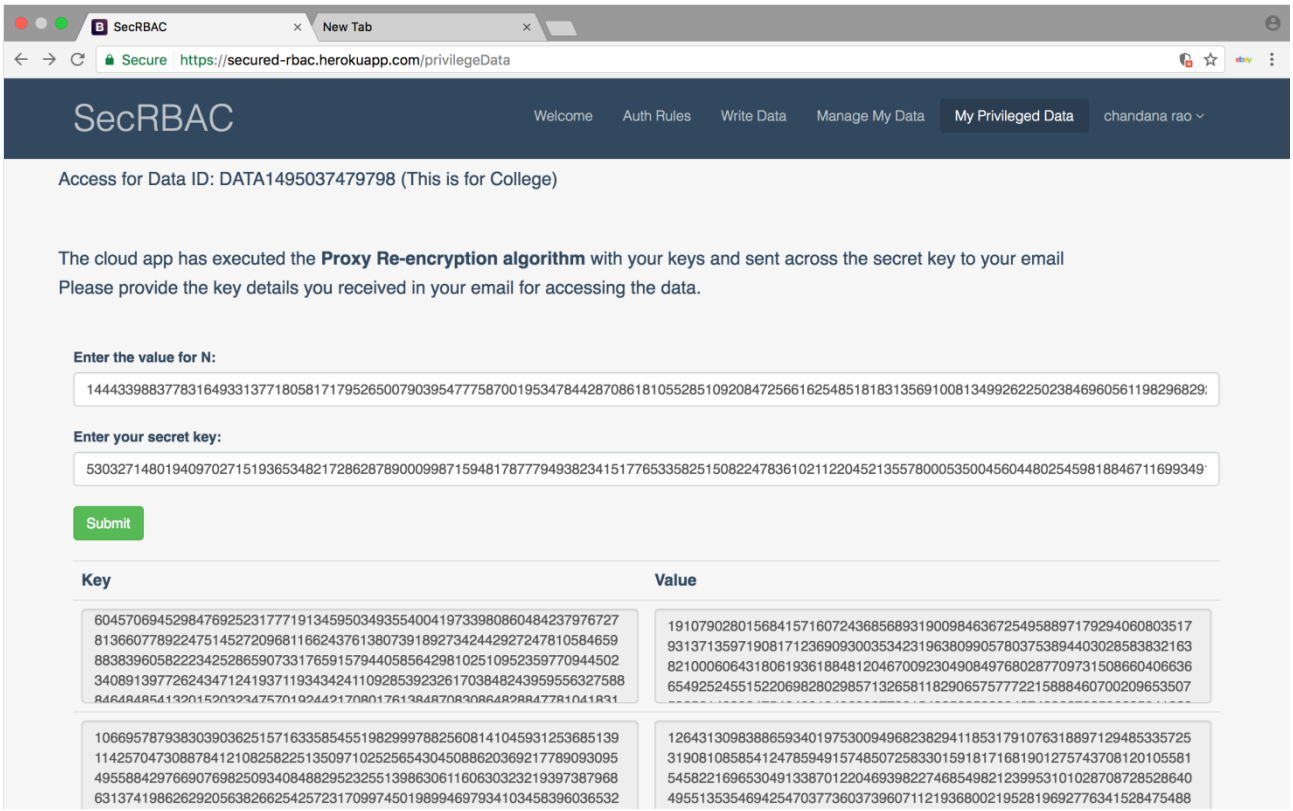


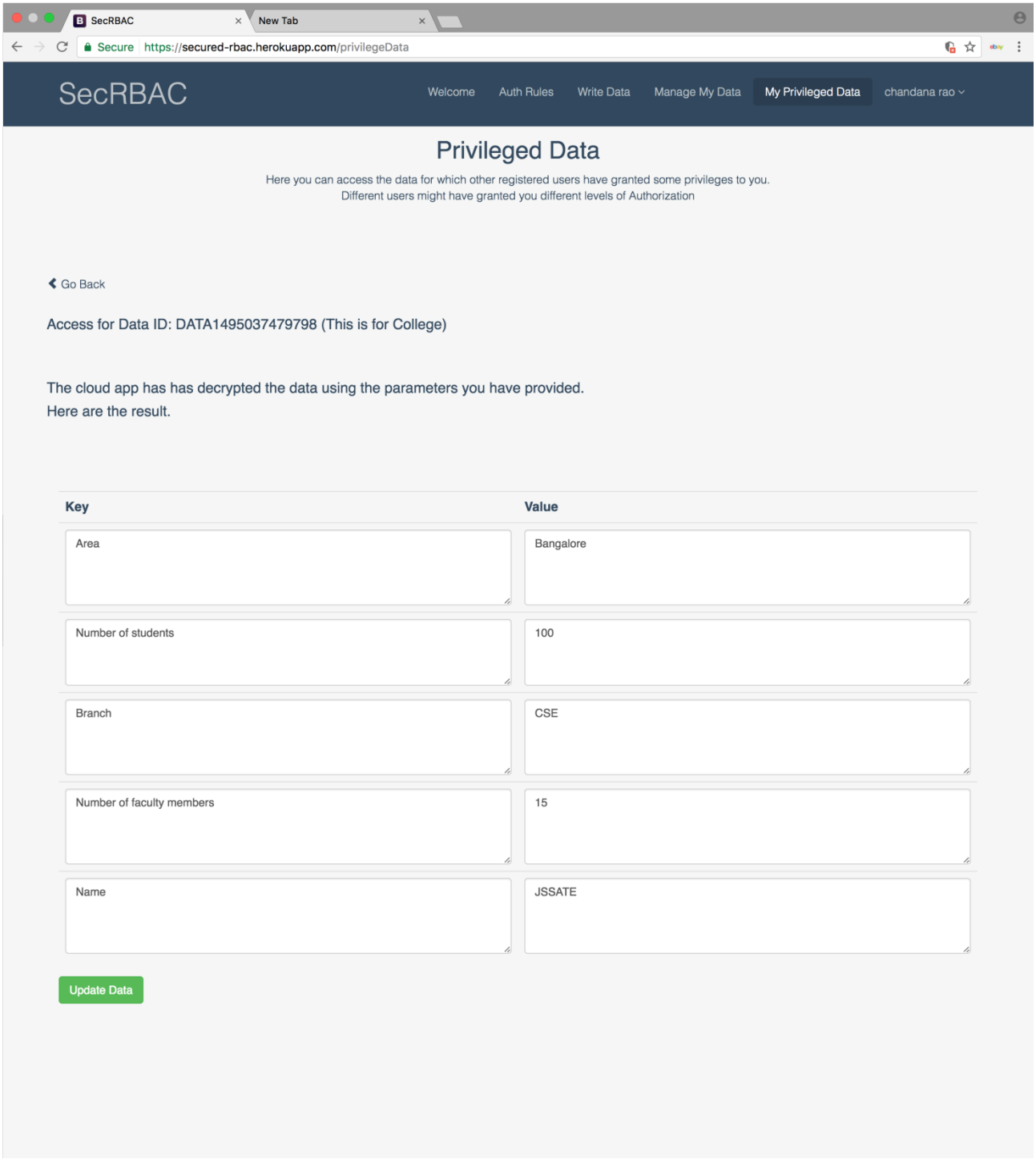
10.7 My Privileged Data






SecRBAC
Secure https://secured-rbac.herokuapp.com/privilegeData?type=access&data_id=DATA1495037479798&desc=This%20is%20for%20College
SecRBAC
Welcome
Auth Rules
Write Data
Manage My Data
My Privileged Data
chandana rao
To provide you access to this data, the cloud application will have to execute Proxy Re-encryption algorithm
Please provide your unique identifier (anything of your choice). The cloud app will generate a secret key for you and send it across an email.
The cloud app then performs Re-encryption operation on the above data using your public key. You can use your secret key received over an email to decrypt the data.
Your Identifier
Execute Proxy Re-encryption Algorithm


SecRBAC
New Tab
Secure https://secured-rbac.herokuapp.com/privilegeData
SecRBAC
Welcome
Auth Rules
Write Data
Manage My Data
My Privileged Data
chandana rao
Access for Data ID: DATA1495037479798 (This is for College)
The cloud app has executed the Proxy Re-encryption algorithm with your keys and sent across the secret key to your email
Please provide the key details you received in your email for accessing the data.
Enter the value for N:
Enter your secret key:
Submit
Key
Value


SecRBAC
New Tab
Secure https://secured-rbac.herokuapp.com/privilegeData
SecRBAC
Welcome
Auth Rules
Write Data
Manage My Data
My Privileged Data
chandana rao
Privileged Data
Here you can access the data for which other registered users have granted some privileges to you.
Different users might have granted you different levels of Authorization
Go Back
Access for Data ID: DATA1495037479798 (This is for College)
The cloud app has has decrypted the data using the parameters you have provided.
Here are the result.
Key
Value
Area
Bangalore
Number of students
100
Branch
CSE
Number of faculty members
15
Name
JSSATE
Update Data

Chapter 11

CONCLUSION AND FUTURE WORK

Conclusion

A data-centric authorization solution has been proposed for the secure protection of data in the Cloud. SecRBAC allows managing authorization following a rule-based approach and provides enriched role-based expressiveness including role and object hierarchies. Access control computations are delegated to the CSP, being this not only unable to access the data, but also unable to release it to unauthorized parties. Advanced cryptographic techniques have been applied to protect the authorization model. A re-encryption key complement each authorization rule as cryptographic token to protect data against CSP misbehaviour. The solution is independent of any PRE scheme or implementation as far as three specific features are supported. A concrete IBPRE scheme has been used in this project in order to provide a comprehensive and feasible solution.

A proposal based on Semantic Web technologies has been exposed for the representation and evaluation of the authorization model. It makes use of the semantic features of ontologies and the computational capabilities of reasoners to specify and evaluate the model. This also enables the application of advanced techniques such as conflict detection and resolution methods. Guidelines for deployment in a Cloud Service Provider have been also given, including an hybrid approach compatible with Public Key Cryptography that enables the usage of standard PKI for key management and distribution. A prototypical implementation of the proposal has been also developed and exposed in this project, together with some experimental results.

Future work

Future lines of research include the analysis of novel cryptographic techniques that could enable the secure modification and deletion of data in the Cloud. This would allow to extend the privileges of the authorization model with more actions like modify and delete. Another interesting point is the obfuscation of the authorization model for privacy reasons. Although the usage of pseudonyms is proposed, but more advanced obfuscation techniques can be researched to achieve a higher level of privacy.

BIBILOGRAPHY

[1] Cloud Security Alliance, “Security guidance for critical areas of focus in cloud computing v3.0,” CSA, Tech. Rep., 2003.

[2] Y. Zhang, J. Chen, R. Du, L. Deng, Y. Xiang, and Q. Zhou, “Feacs: A flexible and efficient access control scheme for cloud computing,” in Trust, Security and Privacy in Computing and Communications, 2014 IEEE 13th International Conference on, Sept 2014, pp. 310–319.

[3] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in Public Key Cryptography - PKC 2011, 2011, vol. 6571, pp. 53–70.

[4] B. B and V. P, “Extensive survey on usage of attribute based encryption in cloud,” Journal of Emerging Technologies in Web Intelligence, vol. 6, no. 3, 2014.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proceedings of the 13th ACM Conference on Computer and Communications Security, ser. CCS ’06, New York, NY, USA, 2006, pp. 89–98.

[6] InterNational Committee for Information Technology Standards, “INCITS 494-2012 - information technology - role based access control - policy enhanced,” INCITS, Standard, Jul. 2012.

[7] E. Coyne and T. R. Weil, “Abac and rbac: Scalable, flexible, and auditable access management,” IT Professional, vol. 15, no. 3, pp. 14–16, 2013.

[8] Empower ID, “Best practices in enterprise authorization: The RBAC/ABAC hybrid approach,” Empower ID, White paper, 2013.