A major project report on

Attribute-Based Approaches for Secure Data Sharing in Industrial Contexts

submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

Submitted by

Mudireddy Nithin Reddy

(20eg105634)

Jaggari Hari Keerthana

(20eg105644)

Sairam Reddy Mothe

(20eg105660)

Under the guidance of

Mrs. K. Rashmi

Assistance Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG UNIVERSITY

VENKATAPUR– 500088

TELANGANA

Year 2023-24

DECLARATION

I hereby declare that the Report entitled "**Attribute-Based Approaches for Secure Data Sharing in Industrial Contexts** " submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: Anurag University, Hyderabad

Date:

M Nithin Reddy (20eg105634)

J Hari Keerthana (20eg105644)

M Sairam Reddy (20eg105660)

CERTIFICATE

This is to certify that the Report entitled with "**Attribute-Based Approaches for Secure Data Sharing in Industrial Contexts"** that is being submitted by **M Nithin Reddy** bearing roll number **20eg105634**, **J Hari Keerthana** bearing roll number **20eg105644** and **M Sairam Reddy** bearing roll number **20eg105660** in partial fulfillment for the award of B.Tech. in to the Anurag University is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this Report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor.

Mrs. K. Rashmi

Assistant Professor

Department of CSE

Dr. G. Vishnu Murthy

Head of Department, CSE

External Examiner

ACKNOWLEDGMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Mrs. K. Rashmi** for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like express my special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering , Anurag University, for their encouragement and timely support in our B.Tech program. We would like acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy** , Dean, Dept. of CSE ,Anurag University. We also express my deep sense of gratitude to **Dr. V V S S S Balaram** ,Acadamic coordinator ,**Dr .Pallam Ravi** , Project in-Charge , **Dr A Jyothi** .Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage our project work.

ABSTRACT

The sharing of data is becoming increasingly important for the process and manufacturing industries that are using data-driven models and advanced analysis to assess production performance and make predictions, e.g., on wear and tear. In such environments, access to data needs to be accurately controlled to prevent leakage to unauthorized users while providing easy to manage policies. Data should further be shared with users outside trusted domains using encryption. Finally, means for revoking access to data are needed. This paper provides a survey on attribute-based approaches for access control to data, focusing on policy management and enforcement. We aim to identify key properties provided by attribute-based access control (ABAC) and attribute-based encryption (ABE) that can be combined and used to meet the abovementioned needs. We describe such possible combinations in the context of a proposed architecture for secure data sharing. The paper concludes by identifying knowledge gaps to provide direction to future research on attribute-based approaches for secure data sharing in industrial contexts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the dynamic landscape of modern industrial ecosystems, secure data sharing stands as a critical imperative, traversing the realms of trusted and untrusted domains. As Industry 4.0 unfolds, data assumes a pivotal role, driving value generation, innovation, and process optimization. Yet, this reliance on data presents a dichotomy: while efficient sharing fosters collaboration and value creation, safeguarding its integrity and confidentiality remains paramount.

Our investigation embarks on this journey by scrutinizing the evolving needs of data sharing across diverse domains, identifying key properties essential for building a resilient data sharing framework. We navigate the intricate terrain of attribute-based access control (ABAC) and attribute-based encryption (ABE), discerning their respective rationales and methodological advancements within this context.

The challenge lies in integrating ABAC and ABE to fortify data protection mechanisms, particularly in untrusted environments where traditional approaches may falter. Our innovative proposal seeks to address this challenge by harmonizing ABAC and ABE within a novel architecture, bridging the gap between theoretical frameworks and practical implementations. Through this integration, we aim to provide a holistic solution for secure data sharing that ensures both accessibility and confidentiality.

Our contributions extend beyond theoretical exploration; we identify critical research gaps and systematically unravel the intricacies of translating ABAC policies into actionable ABE access structures. Through meticulous analysis and synthesis, we aim to deepen understanding of attribute-based approaches to data security, propelling discourse forward in the pursuit of robust and agile data sharing practices.

Structured to unravel the complexities of data security in Industry 4.0, our paper embarks on a journey through the conceptual foundations, methodological advancements, and pragmatic implications of attribute-based data access control. It is our endeavor to catalyze innovation, stimulate dialogue, and inspire future research endeavors in this vital domain of secure data sharing.

1.1 PROBLEM STATEMENT

The existing system utilizes ciphertext-policy hierarchical attribute-based encryption (CP-HABE), a scheme facilitating efficient data sharing while preserving organizational attribute hierarchy confidentiality. CP-HABE permits users with higher-level attributes to delegate access to those at lower levels, minimizing communication and management overhead. However, the access mechanism's reliance on hierarchical authority levels poses limitations. In large organizations, restricting access to multiple authorities becomes impractical. Furthermore, access keys are user-specific, complicating key management. Thus, the current system faces challenges in accommodating complex data access requirements and managing access keys effectively.

1.2 PROBLEM ILLUSTRATION

Existing methods where developed using ciphertext-policy hierarchical attribute-based encryption CP-HABE method where keys are given based on authority key distribution and access is complex in this method.

Authority	Key	time
Higher	key 1	T1
Medium level	Key1+ key2	T1+t2
Low level	Key1+key2+key3	T1+t2+t3
For low level access data	For three keys K1+k2+k3 = k4	Final time: T1+t2+t3

Table 1.2.1 problem illustration by table

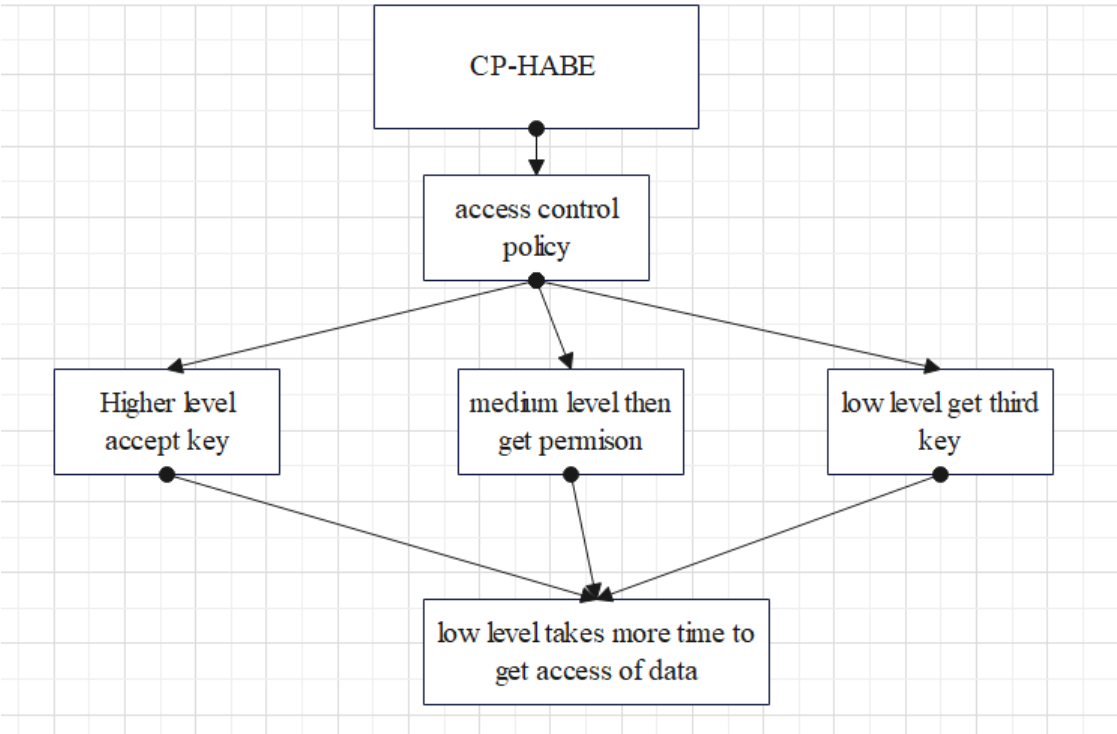


Fig 1.2.1 Problem illustration by flow map

1.3 OBJECTIVE

This project is to address the limitations of the existing data access mechanism based on ciphertext-policy hierarchical attribute-based encryption (CP-HABE). This involves developing an improved system that can efficiently handle complex data access requirements in large organizations while overcoming the challenges associated with user-specific access keys and hierarchical authority levels. The aim is to enhance data sharing capabilities by implementing a more flexible and scalable access control mechanism that allows for granular access control and simplified key management. Ultimately, the project seeks to improve the overall security and efficiency of data sharing within industrial contexts.

2 LITERATURE SURVEY

The landscape of cloud security and access control is rich with innovative solutions designed to address the evolving challenges of privacy, security, and efficient data management. Several notable works contribute significantly to this domain:

Cloud Security Alliance (CSA) Guidance: The third edition of the CSA's Security Guidance for Critical Areas of Focus in Cloud Computing emphasizes security, stability, and privacy in cloud operations. It provides a practical roadmap for managers to safely adopt cloud technology while ensuring corporate privacy in shared environments.

SecCloud Protocol (Wei, Zhu, Cao): The SecCloud protocol aims to discourage privacy cheating and ensure secure computation auditing in cloud environments. It leverages designated verifier signatures, batch verification, and probabilistic sampling techniques to bridge secure storage and computation auditing effectively.

Cryptographic Solutions for Cloud Storage (Kamara and Lauter): This work explores cryptographic architectures tailored for cloud storage, highlighting the benefits for both customers and service providers. It surveys recent cryptographic primitives and their application in enhancing cloud storage security.

Attribute-Based Access Control (ABAC) (Kuhn and Coyne): ABAC evaluates attributes of access event components against predefined rules to determine authorization, offering granular control over access permissions based on various attributes of subjects, objects, and environmental conditions.

Limitations of Role-Based Access Control (RBAC) and Introduction of ABAC (Hu and Kuhn): This work discusses the limitations of RBAC and introduces ABAC as a more flexible alternative, particularly in handling dynamic attributes crucial for determining user permissions in rapidly changing domains.

Extension of ABAC XACML to eXACML for Secure Data Sharing (Tien, et al.): eXACML extends XACML to support obligations, enabling the expression and enforcement of value constraints and improving flexibility in access control policies. However, its feasibility in public clouds without cryptographic or privacy mechanisms remains a challenge.

Semantically Rich Access Control Scheme (Joshi et al.): This scheme employs edge computing and oblivious RAM encryption mechanisms to protect critical organizational documents stored in the cloud. It utilizes a complex ontology and semantic web rule language for access policy expression and enforcement, ensuring data privacy within the organization's trusted domain.

Privacy-Aware Relationship Semantics-Based XACML Access Control Model (Kanwal et al.): PRSX-AC extends XACML-ABAC with a semantic relationship-based access control approach, enabling fine-grained access control and flexible policy creation for electronic health records (EHRs) in hybrid clouds. It employs Anatomy for data privacy, transforming EHR records into quasi-attribute and sensitive-attribute tables to protect privacy.

3 ANALYSIS

3.1 EXISTING SYSTEM

The existing system relies on ciphertext-policy hierarchical attribute-based encryption (CP-HABE) as an encryption scheme for secure data sharing. CP-HABE allows users with higher-level attributes to delegate access to users at lower levels, facilitating efficient sharing of encrypted data while preserving the confidentiality of the organization's attribute model or hierarchy. This approach minimizes management overhead by utilizing a key delegation model, wherein access permissions can be efficiently managed and shared**.** However, the access mechanism's reliance on hierarchical authority levels poses limitations. In this method keys are given based on authority key distribution and access is complex. Furthermore, access keys are user-specific, complicating key management. Thus, the current system faces challenges in accommodating complex data access requirements and managing access keys effectively.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

Complex Access Mechanism: The access mechanism is based on hierarchical attribute-based encryption (HABE), where permissions are granted based on the level of authority. This process can be complex, especially in large organizations with multiple levels of authority. Managing and navigating this hierarchy to grant access can be time-consuming and error-prone.

Limited Access Control: Access keys are restricted to specific users only, which can lead to challenges in managing access control effectively. In scenarios where multiple users or departments require access to the same data, the existing system may struggle to accommodate diverse access needs efficiently.

Key Management Challenges: The existing system faces challenges in key management, particularly in the context of securely distributing and managing access keys. With access keys restricted to specific users, ensuring the integrity and

confidentiality of keys becomes crucial, especially in environments where data sharing is frequent and dynamic.

3.2 PROPOSED SYSTEM

The proposed architecture leverages attribute-based access control (ABAC) components to facilitate encryption and decryption using attribute-based encryption (ABE). This integration ensures that attributes used for access control within trusted domains align seamlessly with those utilized for encryption, establishing a cohesive security framework.

In this project, the focus is on addressing the critical need for secure data sharing within process and manufacturing industries, driven by the utilization of data-driven models and advanced analytics for production optimization and predictive maintenance. The primary objective is to establish robust access control mechanisms to prevent unauthorized data access while streamlining access policy management and ensuring encryption for secure external data sharing. Additionally, the project aims to implement efficient data access revocation mechanisms when necessary. Through a systematic survey, the project evaluates attribute-based approaches, specifically attribute-based access control (ABAC) and attribute-based encryption (ABE), to devise a comprehensive solution. By identifying the essential properties of ABAC and ABE, the project proposes a cohesive architecture tailored to the industrial landscape, emphasizing secure data sharing practices. Furthermore, the project outlines areas for further research to advance the field of secure data sharing within industrial contexts.

Runtime cost and storage issues of per-end-user encryption are addressed by attribute-based encryption (ABE). ABE allows for encryption with attributes, so a data owner needs only to predefine enough attributes to support the desired access granularity and thereby does not need to care about the number of users in the system .

3.2.1 ADVANTAGES PROPOSED SYSTEM

Enhanced Security: By leveraging attribute-based access control (ABAC) and attribute-based encryption (ABE), the proposed method offers robust security measures.

ABAC ensures granular control over access permissions based on various attributes, while ABE allows for encryption with attributes, enhancing data confidentiality and integrity.

Efficient Access Control Management: The integration of ABAC components streamlines access policy management, making it easier to define and enforce access rules. This efficiency reduces administrative overhead and minimizes the risk of unauthorized data access.

Secure External Data Sharing: With encryption enabled through ABE, the proposed method ensures secure data sharing with external parties. Data can be shared confidently outside trusted domains, safeguarding sensitive information during transit and storage.

Scalability and Flexibility: Attribute-based encryption (ABE) addresses runtime cost and storage issues associated with per-end-user encryption. The scalability and flexibility of ABE allow for efficient management of access granularity without the need to consider the number of users in the system.

Adaptability to Industrial Contexts: Tailored to the needs of process and manufacturing industries, the proposed method aligns with the requirements of data-driven models and advanced analytics. It provides a cohesive security framework designed to meet the specific challenges of industrial data sharing environments.

3.3 SYSTEM REQUIREMENTS

3.3.1 HARDWARE REQUIREMENTS:

- System : Intel(R) Core (TM) i3-7020U CPU @ 2.30GHz
- Hard Disk : 1 TB.
- Input Devices : Keyboard, Mouse
- Ram : 4 GB.

3.3.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7/10.
- Coding Language : Java
- Tool : Netbeans
- Database : MYSQL
- Cloud : Drive HQ

4 IMPLEMENTATION

4.1MODULES:

4.1.1Owner model:

owner will register into the application by providing all the necessary details and therefore he can login into the application using username and password and owner can upload the files to application and share with the other registered users. Owner can also view the files uploaded by him and can also view the requests for secret key from the other users so that the owner can respond by sending the secret key to user through mail. Using that key, the user can download the file and view the information

4.1.2User Module:

user will register with application and get user name and password. Users need to request the owner for secret key to access the information uploaded by the owner.

4.1.3Cloud Module:

Using this module cloud can register with application and store information of each user and owner who uploads and requests for data. Details which are stored in cloud are uploaded to Drive HQ cloud.

4.2 THE STUDY OF THE SYSTEM

- To conduct studies and analyses of an operational and technological nature, and
- To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design focuses on how users input information into the system and how the system presents information back to them. Data Design deals with how data is represented and stored within the system. Lastly, Process Design addresses how data flows through the system, including validation, security, and transformation processes. Documentation detailing these aspects is produced during the systems design phase for use in subsequent phases.

Physical design, in this context, doesn't pertain to the tangible hardware layout of an information system. Instead, it's analogous to the functional components of a personal computer: input via keyboard, processing within the CPU, and output via monitor or printer. It entails a detailed design of user and product database structures, as well as control processors. The H/S personal specification is developed for the proposed system.

4.2.1 INPUT & OUTPUT REPRESENTATION

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

a. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

b. Select methods for presenting information.

c. Create document, report, or other formats that contain information produced by the system.

4.3 TECHNOLOGIES USED

4.3.1Java Technology

Java technology serves as both a programming language and a platform, providing developers with versatile tools for creating cross-platform applications.

The Java Programming Language

The Java programming language encompasses a multitude of characteristics often described by various buzzwords which are:

- Simple
- Architecture neutral
- Object oriented

- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

In most programming languages, you typically either compile or interpret a program to run it on your computer. However, the Java programming language stands out in that it employs both compilation and interpretation processes. Initially, a program is compiled into an intermediate language known as Java byte codes. These byte codes, which are platform-independent, are then interpreted by the Java platform's interpreter. During execution, the interpreter parses and executes each Java byte code instruction on the computer. Compilation occurs only once, while interpretation happens each time the program is executed. This approach is depicted in the following figure.

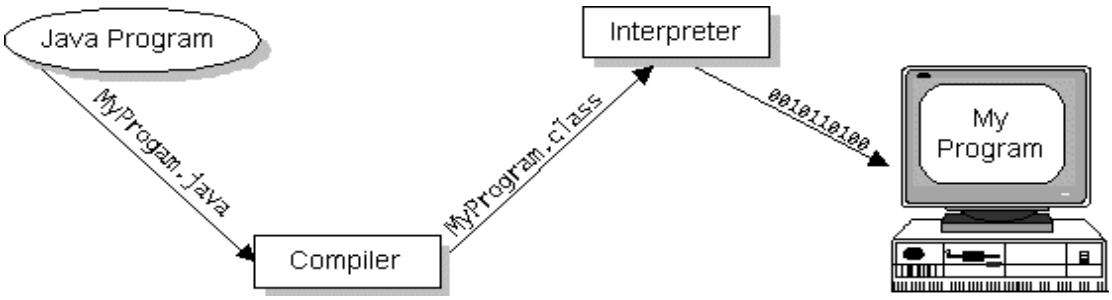


Fig 4.3.1.1 Illustration of how program works

Java byte codes serve as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a web browser capable of running applets, functions as an implementation of the Java VM. The use of Java byte codes facilitates the "write once, run anywhere" principle. You can compile your program into byte codes on any platform with a Java compiler. These byte codes

can then be executed on any implementation of the Java VM. Consequently, a program written in the Java programming language can run on diverse platforms such as Windows 2000, a Solaris workstation, or an iMac, as long as a Java VM is present on the computer.

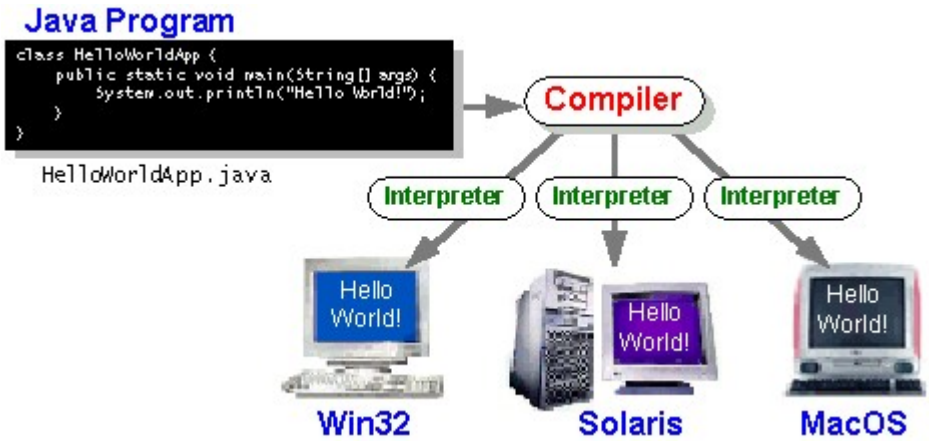


Fig 4.3.1.2 Working Java VM

The Java Platform

A platform encompasses the hardware and software environment in which a program operates, such as Windows 2000, Linux, Solaris, and MacOS. Typically, it's a fusion of an operating system and hardware components. However, the Java platform distinguishes itself by being solely software-based, running atop other hardware-based platforms, and additionally providing a consistent runtime environment for Java applications across diverse systems.

The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

The depicted diagram illustrates a program executing on the Java platform. In this setup, the Java API and the virtual machine act as a protective shield, isolating the program from direct interaction with the hardware.

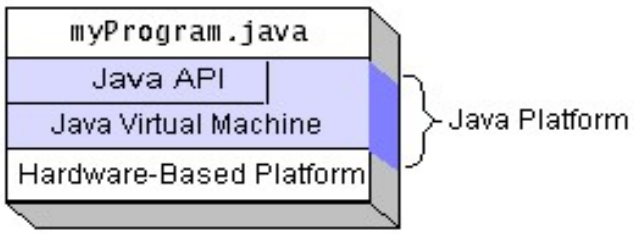


Fig 4.3.1.3 A program running on the Java platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans™, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

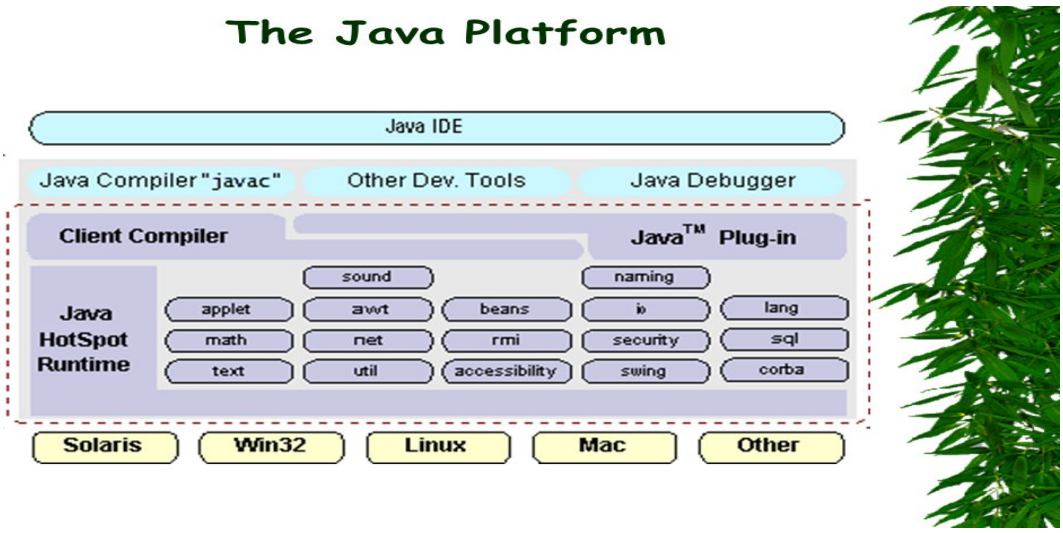


Fig 4.3.1.4 Java 2 SDK

Java Web Development

Web development, also known as website or web application development, involves the creation, maintenance, and updating of applications accessed through a web browser. This process encompasses web design, backend programming, and database management, all facilitated by various software technologies.

Web development relies on servers, which can be web servers or machine servers like CPUs, to host web applications. These applications are built using server-side programming languages or technologies, such as Java, PHP, and others.

Java web development specifically focuses on creating server-side websites and applications. Unlike standalone Java applications, Java web apps are hosted on a web container within the server environment, which serves as a runtime environment akin to the Java Virtual Machine (JVM) for local Java applications.

There are two main types of Java containers: web containers and Java EE containers, each providing additional functionalities such as server load distribution. Web containers support Java servlets and JSP (Java Server Pages), with Tomcat being a common example in Java technology.

Functions of Java Web Development

Java web development creates applications and websites using static and dynamic resources. The static resource refers to HTML pages with images, and a dynamic resource refers to classes, jars, Servlet, and JSP. Java web development uses several packages, files, and online links. Java web development requires web archive files known as a WAR files.

Java web development works on three main factors. These development factors show below

- Front-end web development using Java technology.
- Backend web development using Java server technology.
- Database management using Java database driver.

The above three factors create, update, remove, display and operate data or information.

Front-end web development: The front-end technology interacts with the user and Java interface. It helps to insert and submit data. Java web development uses JavaServer Pages or JSP for the front-end form or table.

Backend web development: The backend technology maintains and updates data of the database. Java uses Servlet, spring, and other advanced technology.

Database management handles or fetches data from the database using the Java database driver. The Java technology uses JDBC, Hibernate to handle the database.

Types of the Java Web Technologies

- Servlet API
- JSP (JavaServer page)
- JDBC Driver
- JAVA Persistence
- JavaServer Faces (JSF)
- JSTL
- JAVA Message Service API

Servlet API (JAVA Web application programming interface)

Servlet, filter, filter chain, servlet config, and other interfaces are available in the javax. Servlet package. The capabilities of servers that host apps are increased by using Servlet.

The request-response model is used in web development applications written with Java servlets. From initialization to garbage collection, a servlet has a life cycle. Servlets are useful for various tasks, including collecting data via web page forms, presenting data from a database or any other third-party source, etc.

JSP (JavaServer Page Web application programming technology)

Developers employ JavaServer Pages or JSP technology to quickly produce platform- and server-independent online content. Normally, the developer works on separate Common Gateway Interface files to embed dynamic elements in HTML pages. Java JSP technology can be used, as it has access to the whole Java API family.

The JSP technology allows embedding bits of servlet code in a text-based document. JSP is a popular Java EE technology that allows programmers to create complex dynamic web pages quickly.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use.

JDBC Driver or Java Database Connectivity

JDBC Driver is a connector between database and Java web application. Java database connectivity helps to update and modify data using queries. The jdbc driver is an essential part of Java web development. This driver helps to send data to the database and retrieve data from the database.

Within a Java program, the JDBC driver allows to perform the following tasks:

- Make a data source connection
- To the data source, send queries and update statements ◦
  Displays require data from a database.
- Organize application information.

JDBC is a set of methods and queries for accessing databases written in Java. Clients can use web applications using JDBC drivers to update any information in the database. JDBC drivers connect to databases in four ways: JDBC-ODBC Bridge Driver, Network Protocol Driver, Native Driver, and Thin Driver.

Persistence API for Java

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in

conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

   The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.

2. **SQL Conformance**

   SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. **JDBC must be implemental on top of common database interfaces**

   The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. **Provide a Java interface that is consistent with the rest of the Java system**

   Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. **Keep it simple**

   This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

Technology of the JavaServer Faces

JavaServer Faces is called a JSF Technology. This technology provides a framework for developing web-based interfaces. JSF provides a simple model for components in various scripting or markup languages.

The data sources and server-side event handlers are coupled to the User Interface widgets. JSF aids in the creation and maintenance of web applications by minimizing the time and effort required.

- Construct Java web development pages.
- Drop components on a web page by adding component tags to a web page.
- Connect Java web development page components to server-side data.
- Connect component-generated events to application code running on the server.
- Extend the life of server requests by storing and restoring the application state.

Standard Tag Library for JavaServer Pages (JSTL)

The JavaServer Pages Standard Tag Library or JSTL abstracts common functionality of JSP-based applications. We use a single standard set of tags to incorporate tags from various vendors into web applications. This standardization enables the establishment of Java applications on any JSP container. It supports JSTL and increases the tags to optimize during implementation.

JSTL includes iterator and conditional tags for controlling flow. These tags work for manipulating XML documents and tags for internationalization. This JSTL is also used for SQL database access and tags for frequently used functions.

API for Java Message Service

Messaging is a way for software components or apps to communicate with one another. A messaging system is a type of peer-to-peer network. In other words, a messaging client can communicate with and be communicated with by any other client.

Each client establishes a connection with a messaging agent, facilitating the creation, transmission, receipt, and reading of messages.

The Java Message Service (JMS) API provides a strong tool for resolving enterprise computing problems by integrating Java technology and enterprise messaging. Enterprise messaging enables the secure and flexible sharing of business data. The JMS API extends this by providing a uniform API and provider framework that facilitates the building of portable message-based Java applications.

Sockets: A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h> #include
<sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used.. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

J2ME (Java 2 Micro edition):-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. **General J2ME architecture**

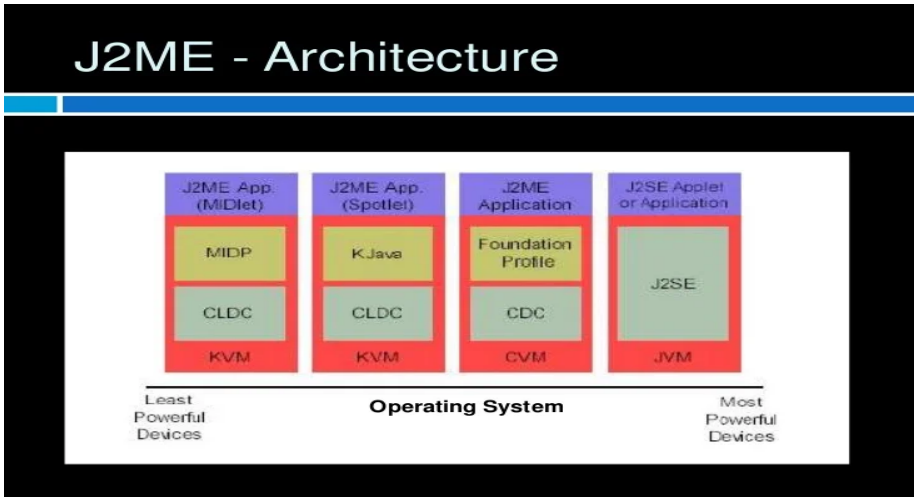


Fig 4.3.1.5 J2ME Architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes.. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role reverification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC..

* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profile

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

* java.lang

* java.io

* java.util

* javax.microedition.io

4.3.2 CRYPTOGRAPHY

Cryptography is the practice of safeguarding information and communications by encoding them with codes, ensuring that only the intended recipients can comprehend and process the data. This prevents unauthorized access to sensitive information. The term "cryptography" derives from "crypt," meaning "hidden," and "graphy," meaning "writing." It employs mathematical concepts and algorithmic rules to convert messages into forms that are difficult to decode. These algorithms play crucial roles in generating cryptographic keys, digital signatures, and data verification, ensuring the privacy of information during activities such as web browsing and confidential transactions like credit card transactions.

Techniques used For Cryptography: In today's age of computers cryptography is often associated with the process where an ordinary plain text is converted to cipher text which is the text made such that intended receiver of the text can only decode it and hence this process is known as encryption. The process of conversion of cipher text to plain text this is known as decryption.

Features Of Cryptography are as follows:

Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.

Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

Non-repudiation: The creator/sender of information cannot deny his intention to send information at later stage.

Authentication: The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

Applications Of Cryptography:

Computer passwords: Cryptography is widely utilized in computer security, particularly when creating and maintaining passwords. When a user logs in, their password is hashed and compared to the hash that was previously stored.

Passwords are hashed and encrypted before being stored. In this technique, the passwords are encrypted so that even if a hacker gains access to the password database, they cannot read the passwords.

Digital Currencies: To safeguard transactions and prevent fraud, digital currencies like Bitcoin also use cryptography. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.

Electronic signatures: Electronic signatures serve as the digital equivalent of a handwritten signature and are used to sign documents. Digital signatures are created using cryptography and can be validated using public key cryptography. In many nations, electronic signatures are enforceable by law, and their use is expanding quickly.

Authentication: Cryptography is used for authentication in many different situations, such as when accessing a bank account, logging into a computer, or using a secure network. Cryptographic methods are employed by authentication protocols to confirm the user's identity and confirm that they have the required access rights to the resource.

Cryptocurrencies: Cryptography is heavily used by cryptocurrencies like Bitcoin and Ethereum to safeguard transactions, thwart fraud, and maintain the network's integrity. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.

End-to-End Encryption: End-to-end encryption is used to protect two-way communications like video conversations, instant messages, and email. Even if the message is encrypted, it assures that only the intended receivers can read the message. End-to-end encryption is widely used in communication apps like WhatsApp and Signal, and it provides a high level of security and privacy for users

4.3.2.1 ABE

Attribute-Based Encryption (ABE) is a cryptographic technique that allows data to be encrypted and decrypted based on attributes associated with the data and the access policies defined by the data owner. In ABE, both the data and the access policies are

associated with attributes, and access to the data is granted to users whose attributes satisfy the access policies.

Attribute-based encryption schemes follow a basic construction structure based on the following 4 primary algorithms: setup, encryption, key generation, and decryption.

- **Setup:** The algorithm that takes any implicit security parameter and creates public parameters and a master key. In this step, the universe of attributes is defined.
- **Encryption:** The algorithm that applies encryption to a message.
- **Key generation:** The algorithm that generates decryption keys based on a set of attributes.
- **Decryption:** The algorithm that decrypts a ciphertext to obtain the underlying message.

The ABE scheme in use determines where the user attributes and the access structure are associated. This is the main difference between the following two main ABE schemes: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE, the ciphertexts are labeled with a set of attributes, while the user's private key is the one associated with the access policy. Hence, in this scheme, the access structure that controls whether a user is allowed to decrypt a ciphertext or not is handled in the key generation algorithm. On the other hand, CP-ABE associates the user's private key with a set of attributes, while the ciphertexts are encrypted by an access policy. In CP-ABE, the encryption algorithm handles the access structure, creating a ciphertext in which only those who possess a specific set of attributes that satisfies this access structure will be able to decrypt.

4.3.3 CLOUD COMPUTING

Imagine a vast digital warehouse, accessible from anywhere. That's cloud data storage in a nutshell. Instead of bulky hard drives, your files reside on remote servers managed by companies like Google Drive or Dropbox.

This remote access is key. With a web browser or app, you can upload, download, edit, and share documents, photos, videos – anything digital – from any device. No more lugging around external drives!

Cloud storage offers scalability too. Need more space? Simply upgrade your plan. No need to constantly buy new hardware. Businesses love this flexibility, easily storing massive datasets or collaborating on projects in real-time.

Security is paramount. Cloud providers employ robust encryption and access controls to keep your data safe from prying eyes. Automatic backups ensure peace of mind – even if your local device fails, your precious files are secure in the cloud.

But the cloud isn't perfect. A strong internet connection is crucial for accessing your data. Outages or slow speeds can be frustrating. Additionally, depending on the amount of data stored and the plan chosen, costs can add up.
For businesses, choosing the right cloud provider is vital. Public clouds offer flexibility and affordability, but some companies might prefer the increased control of a private cloud. Hybrid solutions combine both for a customized approach.

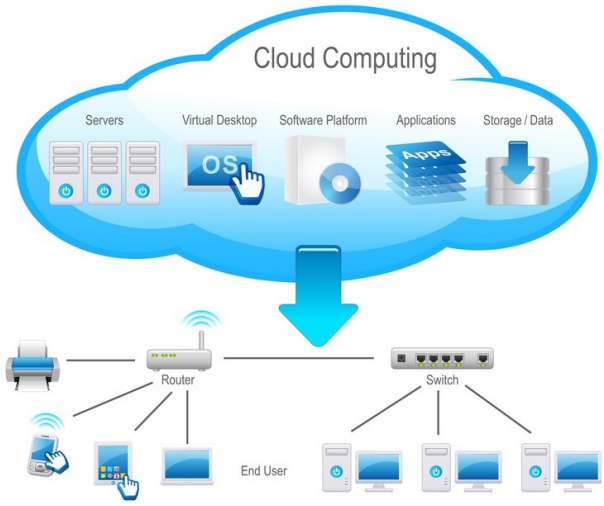


Fig 4.3.3.1 Cloud computing

4.3.3.1 DRIVE HQ

DriveHQ stands out as an experienced provider, prioritizing the requirements of businesses. Established in 2003, DriveHQ offers a secure and feature-rich file server accessible via web browsers and mobile apps from any location. Their expertise lies in ensuring data security through encryption and access controls, delivering peace of mind to users. However, DriveHQ offers more than just storage; they provide automatic backups, controlled file sharing, and cloud-to-cloud backup solutions for added redundancy. For businesses needing specific file transfer capabilities, DriveHQ offers FTP server hosting.

One of DriveHQ's notable strengths is its focus on enterprise functionality. They offer user and group management features, simplifying access control, and seamlessly integrate with Active Directory to align with existing business infrastructure. This, coupled with support for large-scale deployments, positions DriveHQ as an attractive option for companies seeking a robust cloud storage solution. Although DriveHQ offers a free plan for trial purposes, their paid plans offer expanded storage and advanced features tailored to meet specific business requirements.

4.4 SAMPLE CODE

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.Vector;

import javax.swing.*;
public class Attacker extends JFrame implements ActionListener
{
  JFrame f;
  JPanel p;
  JLabel l1,l2,l3;
  JButton b1,b2;
  ImageIcon ic;
  JTextField tc;
  public Font f1 = new Font("Times new Roman", Font.BOLD, 17);
  public JTextArea tf = new JTextArea();
public JTextField fname = new JTextField();
public JScrollPane pane1 = new JScrollPane();

  public Attacker()
  {

  f=new JFrame("Attacker::Security Analysis of Smartphone");
  p=new JPanel();

  p.setBackground(Color.white);
  f.setSize(500, 650);
  f.setVisible(true);
  p.setLayout(null);


  ImageIcon banner = new ImageIcon(this.getClass().getResource("Attacker.jpg"));
  JLabel title = new JLabel();
  title.setIcon(banner);
  title.setBounds(200, 0, 190, 275);
```

```
  b1=new JButton("View CloudFiles");
  b1.setBounds(230, 360, 150, 30);
 // p.add(b1);
  f.add(p);

  b2=new JButton("Attack");
  b2.setBounds(200, 490, 100, 30);
  p.add(b2);

  tc=new JTextField();
  tc.setBounds(160, 250, 250, 200);
  p.add(tc);

//          ic=new ImageIcon(this.getClass().getResource("ps1.jpg"));
//          l1=new JLabel();
//          l1.setIcon(ic);
//          l1.setBounds(150, -30, 250,350);
//          p.add(l1);

  l1=new JLabel("Enter New Key");
  l1.setBounds(30, 330, 150, 30);
  l1.setFont(f1);
  p.add(l1);
  p.add(title);

    tf.setColumns(200);
          tf.setRows(100);
          tf.setName("tf");
          pane1.setName("pane");
          pane1.setViewportView(tf);
          pane1.setBounds(450, 250, 300, 200);

          b1.addActionListener(this);
          b2.addActionListener(this);

  int[] port = new int[] { 401, 1006,201};

          for (int i = 0; i < 3; i++) {
                  Thread th = new Thread(new PortListener(port[i]));
                  th.start();
          }


}
  public static void main(String[] args)
   {
new Attacker();
}
```

```
 class PortListener implements Runnable {
        DataOutputStream dos = null;
        DataInputStream in = null;

        ServerSocket server;
        Socket connection;
        int i;
        String fileid;
        Connection con;
        Statement stmt;
        int port;

        public PortListener(int port) {
                this.port = port;
        }

        public void run()
        {
    if(this.port==1006)
    {


    }else
          if(this.port==201)
          {

          }


        }

 }

@Override
public void actionPerformed(ActionEvent ae)
    {

if(ae.getSource()==b2)
{
        try
        {
                InetAddress ia = InetAddress.getLocalHost();
                String ip2= ia.getHostAddress();



          String pro=JOptionPane.showInputDialog("Enter  Classifier IP Address");
          Socket s=new Socket(pro,7000);
          DataOutputStream dos=new DataOutputStream(s.getOutputStream());
```

```
            dos.writeUTF(tc.getText());
            dos.writeUTF(ip2);

            System.out.println("    "+tc.getText());
            System.out.println("    "+ip2);
            System.out.println("data send to pattern ");


            DataInputStream diss=new DataInputStream(s.getInputStream());
            String msg=diss.readUTF();
            System.out.println(""+   msg);
            if(msg.equals("Attcker"))
            {
            JOptionPane.showMessageDialog(null,"Fake message injected");
            }



          } catch (Exception e) {
                  // TODO: handle exception
          }
}
    }
}
```

5 SYSTEM DESIGN

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces**.**

5.1 SYSTEM ARCHITECTURE

Architecture Flow:

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

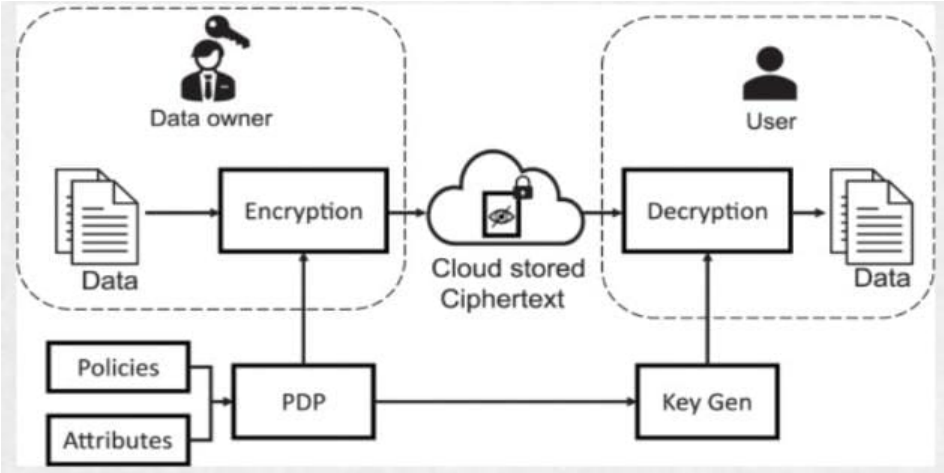


Figure 5.1.1: Architecture diagram

5.2 UML DIAGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

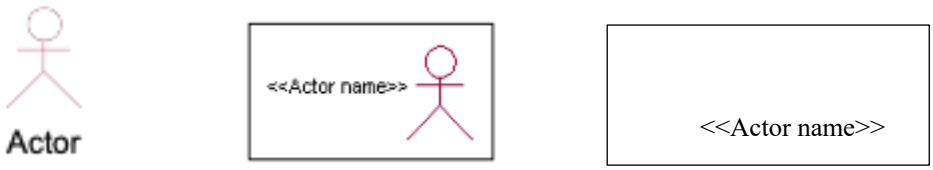


Fig 5.2.1 Graphical representation:

An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

a. System Administrator

b. Customer

c. Customer Care

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:

A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

5.3 Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case

- Normal sequence of events for the use case
- Alternate or exceptional flows

5.3.1Construction of Use case diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

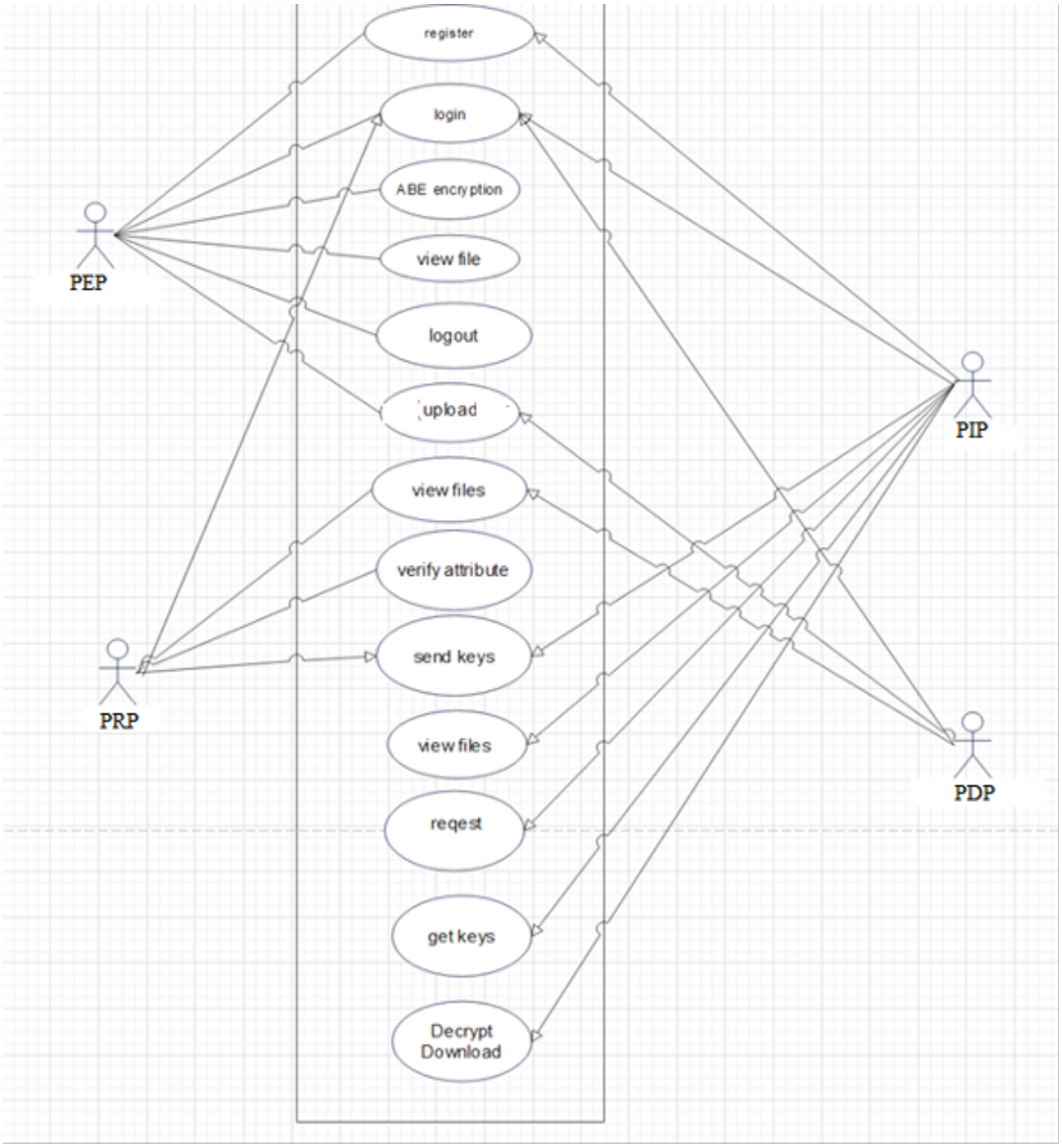


Figure 5.3.1.1 Use Case Diagram

5.3.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

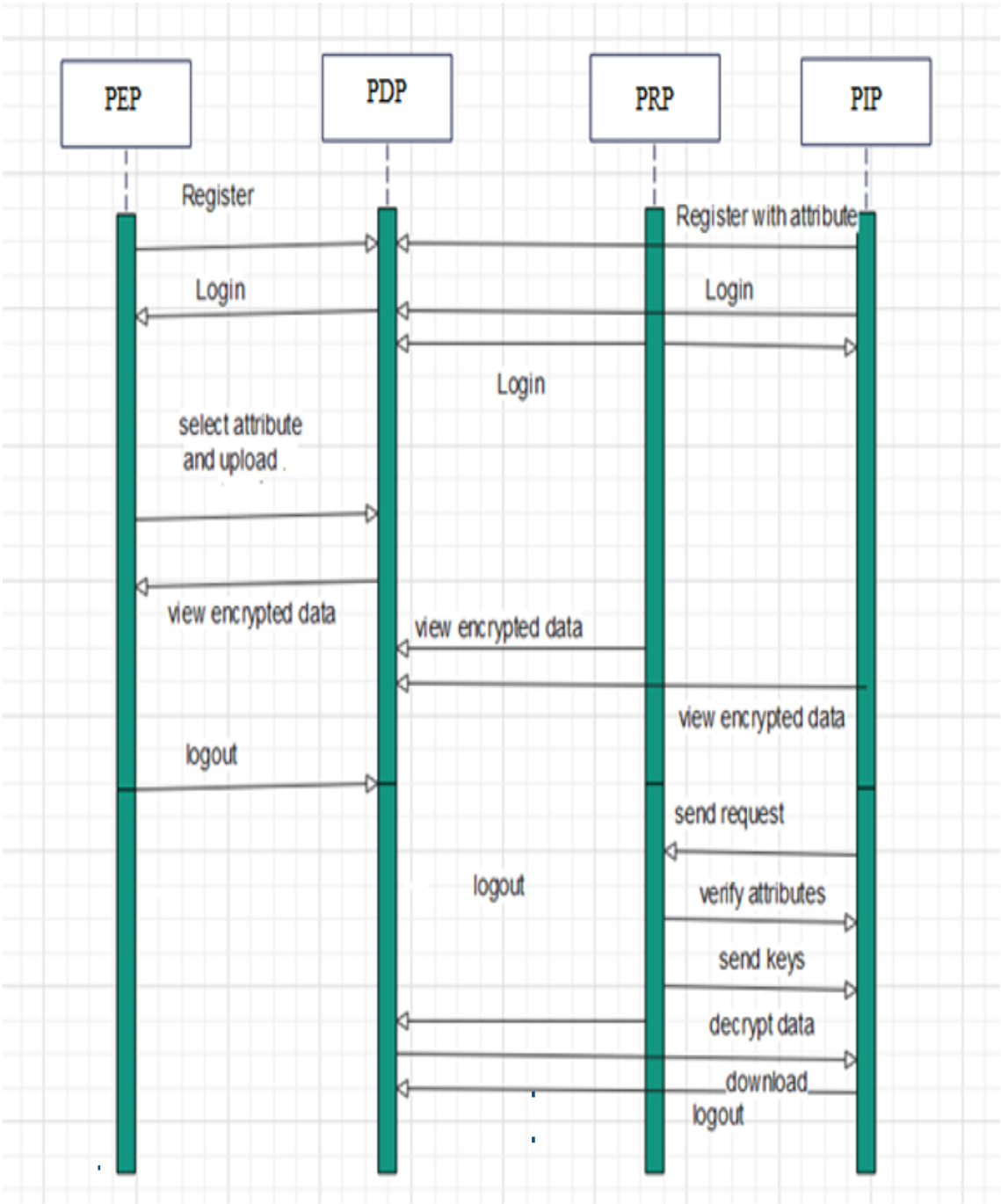


Figure 5.3.2.1 Sequence diagram

5.3.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

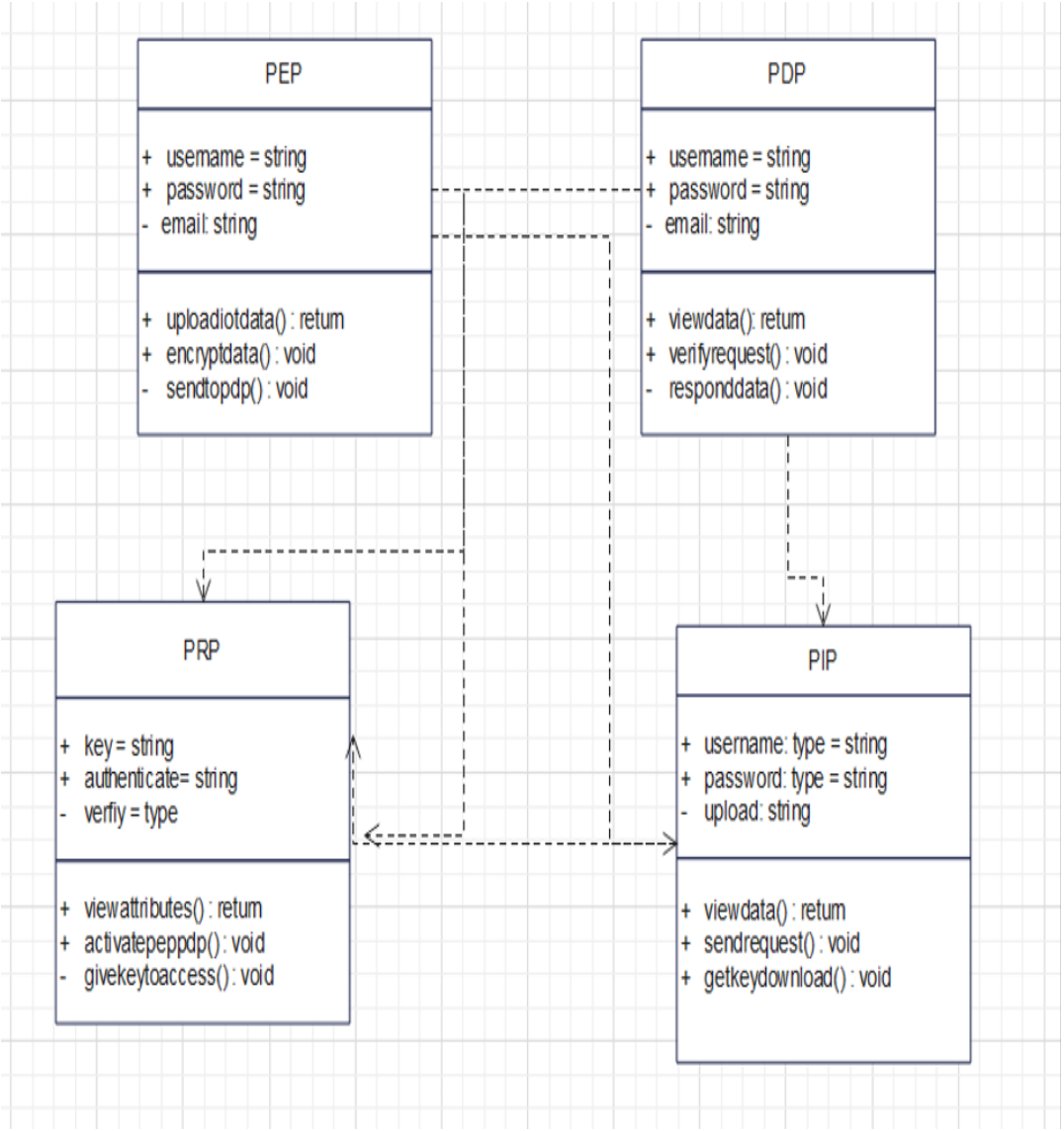


Figure 5.3.3.1: Class Diagram

5.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

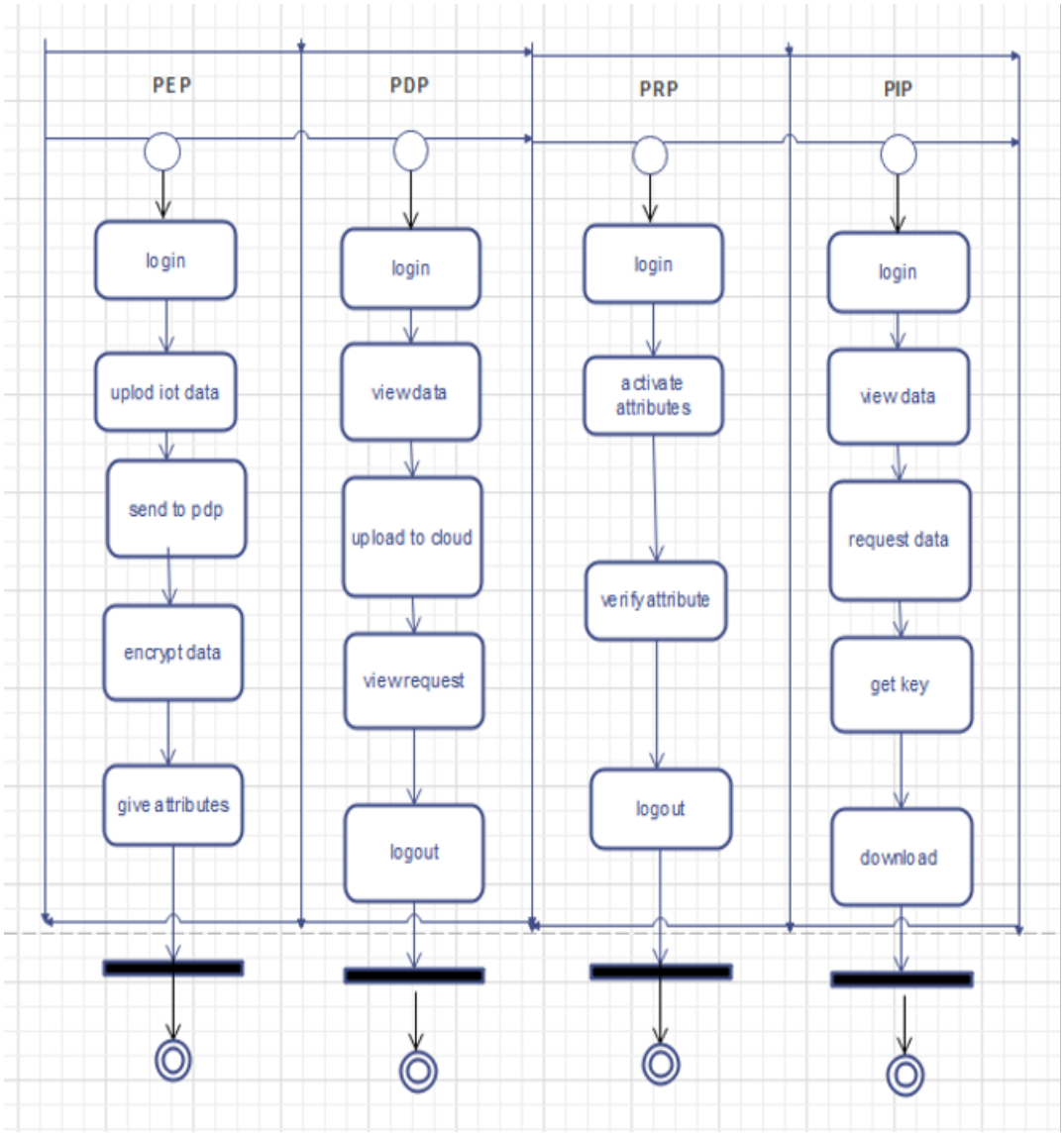


Figure 5.3.4.1: Activity Diagram

6 EXPERIMENT RESULTS

6.1 INTRODUCTION:

Testing and debugging programs are among the most crucial aspects of computer programming. Without functional code, a system cannot produce the intended output. Testing is most effective when developers involve users in identifying errors and bugs. Sample data is utilized for testing purposes, with an emphasis on the quality rather than the quantity of data used. The goal of testing is to ensure the accuracy and efficiency of the system before it goes live.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

6.2. Levels of Testing:

Code testing:

This process scrutinizes the logic within the program. For instance, it involves testing and verifying the logic for updating different sample data, files, and directories. Specification testing entails executing the program based on its specifications to assess how it performs under various conditions. Test cases are created to evaluate different scenarios and combinations of conditions across all modules.

Unit testing:

During unit testing, each module is tested individually before integration into the overall system. Unit testing, also known as module testing, focuses on verifying the functionality of the smallest unit of software design within the module. This testing occurs during the programming stage, where each module is tested separately to ensure it operates as expected and produces the desired output. Additionally, validation checks

for fields are conducted, such as verifying the validity of user input data. This thorough testing process facilitates the detection and debugging of errors within the system.

Each Module can be tested using the following two Strategies:

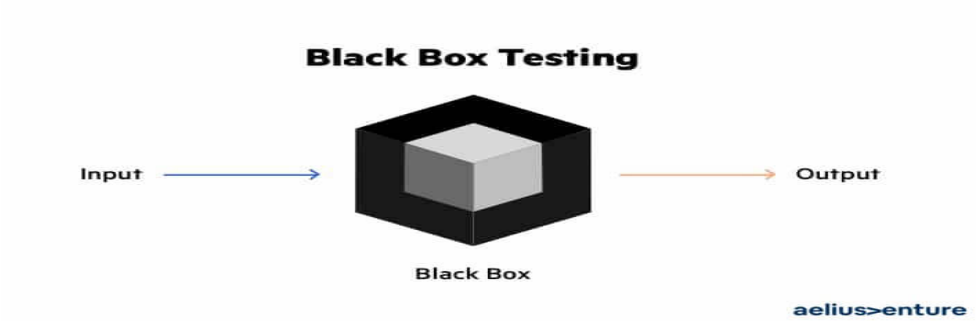
1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing method where the internal structure or workings of the application being tested are not known to the tester. Instead, the tester examines the functionality of the software based on its specifications or requirements. In other words, the tester treats the software as a "black box" and tests its inputs and outputs without knowledge of its internal code or design.

During black box testing, testers typically focus on testing the functionality, usability, and correctness of the software. They design test cases based on the expected behavior of the software, its input conditions, and the desired output. Black box testing helps ensure that the software behaves as expected from the end user's perspective, regardless of its internal implementation details.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

WHITE BOX TESTING

White box testing, also known as clear box testing or structural testing, is a software testing technique where the internal structure, design, and implementation details of the application being tested are known to the tester. Unlike black box testing, where the tester only focuses on the functionality of the software, white box testing involves examining the internal code, logic, and data flow of the software.

During white box testing, testers typically analyze the code and design of the software to create test cases that exercise different paths through the code, ensuring that all statements, branches, and conditions are tested thoroughly. This technique aims to uncover errors or defects in the code, such as logic errors, boundary cases, and optimization issues.

White box testing is commonly used for unit testing, integration testing, and system testing. It helps verify the correctness and completeness of the software's implementation, as well as its compliance with coding standards and best practices. Additionally, white box testing can be automated using tools such as code coverage.

Furthermore, white box testing is not limited to just verifying the correctness of the code; it also involves evaluating the efficiency and maintainability of the software. Testers may assess factors such as code readability, modularity, and adherence to coding standards to ensure that the software is easy to understand, modify, and maintain in the long term.

Overall, white box testing plays a critical role in software quality assurance by providing insights into the internal workings of the software and helping to identify and address potential issues early in the development process.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

Initially, a tester typically familiarizes themselves with the source code of the application, which is a crucial step in white box testing. Given that this approach delves into the internal mechanisms of the application, the tester must possess extensive knowledge of the programming languages employed within the application. Additionally, proficiency in secure coding practices is essential, as security stands as a paramount concern in software testing. The tester's role extends to identifying security vulnerabilities and thwarting potential attacks from hackers or unwitting users who might inadvertently introduce malicious code into the application.

Step 2) CREATE TEST CASES AND EXECUTE

The next fundamental step in white box testing entails scrutinizing the application's source code to ensure its proper flow and structure. One approach involves writing additional code specifically designed to test various processes or sequences within the application. This method necessitates a deep understanding of the code and is frequently undertaken by the developer. Alternatively, other methods such as manual testing, trial and error testing, and the utilization of testing tools, as elaborated further in this article, can also be employed..

7 DISCUSSION OF RESULTS

7.1 Test Cases

Below stable shows the test case for the admin. The details if admin are stored in the database such as admin name, admin name etc. Admin is responsible for handling the transactions. The admin sets password and user name. Weather all the passwords and usernames are correct not is checked. Because by using these details the user will login in into the cloud and can view the details of file stored.

Test case 1

The test case 1 tests the login of admin's. The passwords and username is given if the correct password and usernames are entered the login will be successful. If any wrong in passwords and username the login will be denied.

Sl # Test Case : -	UTC-1
Name of Test: -	**Register BMS**
Items being tested: -	**Registration process**
Sample Input: -	**Fill form enter details**
Expected output: -	**Registration data store in database and validation message**
Actual output: -	**Registration successful**
Remarks: -	**Pass.**

Table 7.1.1 Test Case1

Test Case 2

The test case 2 is for checking the users. Again for the user's password and user name is given based on only the correct input the users will be able to login

Sl # Test Case : -	UTC-2
Name of Test: -	**Upload file with attributes**
Items being tested: -	**Data stored with permissions**
Sample Input: -	**Text medical data**
Expected output: -	**Data stored in cloud and database with confirmation**
Actual output: -	**File upload success**
Remarks: -	**success**

Table 7.1.2 Test Case2

Test case 3

The test case 3 shows for the file upload and it is being stored in different cloud which is called block generation and storing and hence this file is divided and stored .If the storage and block generation is not successful than we can store the file

Sl # Test Case : -	ITC-1
Name of Test: -	NM Key generate
Item being tested: -	**Key is generated for BMS data or not**

Sample Input: -	**Text file data**
Expected output: -	**Key must be generated**
Actual output: -	**Key generated and given to bms**
Remarks: -	**Pass.**

Table 7.1.3 Test Case 3

Test case 4

Test case 4 is for file upload where the uploaded is downloaded. During download if any one of the cloud fails the recovery should start automatic. If the recovery is successful all the blocks will be generated and merged together.

Sl # Test Case : -	STC-1
Name of Test: -	**Complete task**
Item being tested: -	**Update status of Completion**
Sample Input: -	**Task completed status update**
Expected output: -	**Task completed message**
Actual output: -	**Result displayed toKPS**
Remarks: -	**Pass**

Table 7.1.4 Test Case 4

7.2 OUTPUT SCREENSHOTS

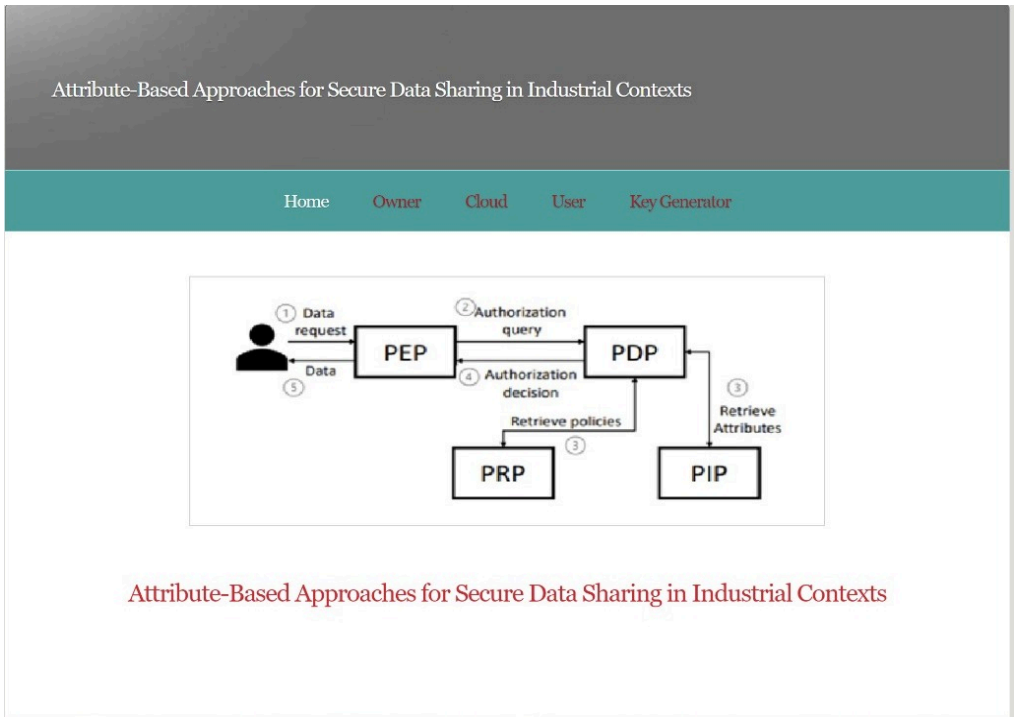


Fig 7.2.1 Home page

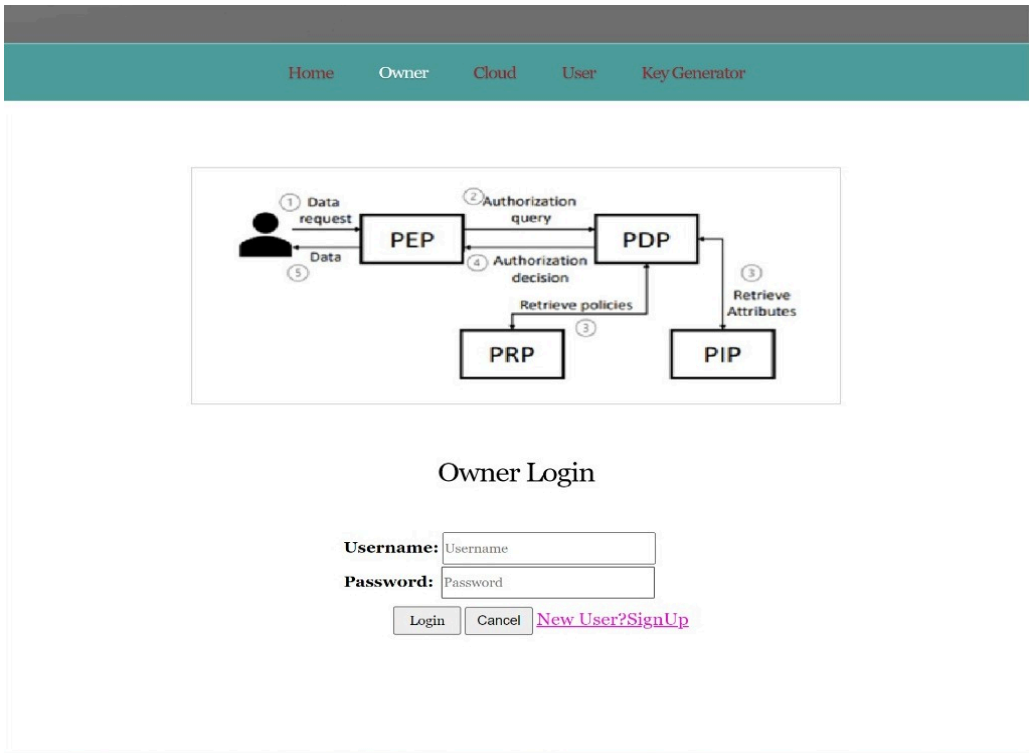


Fig 7.2.2 Owner Login page

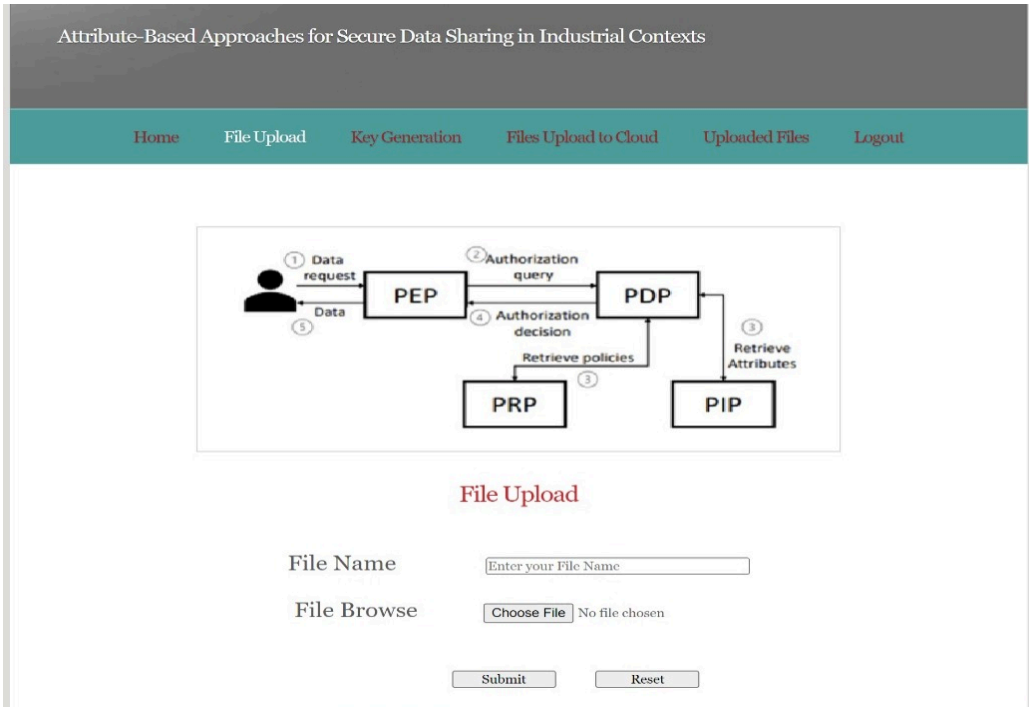
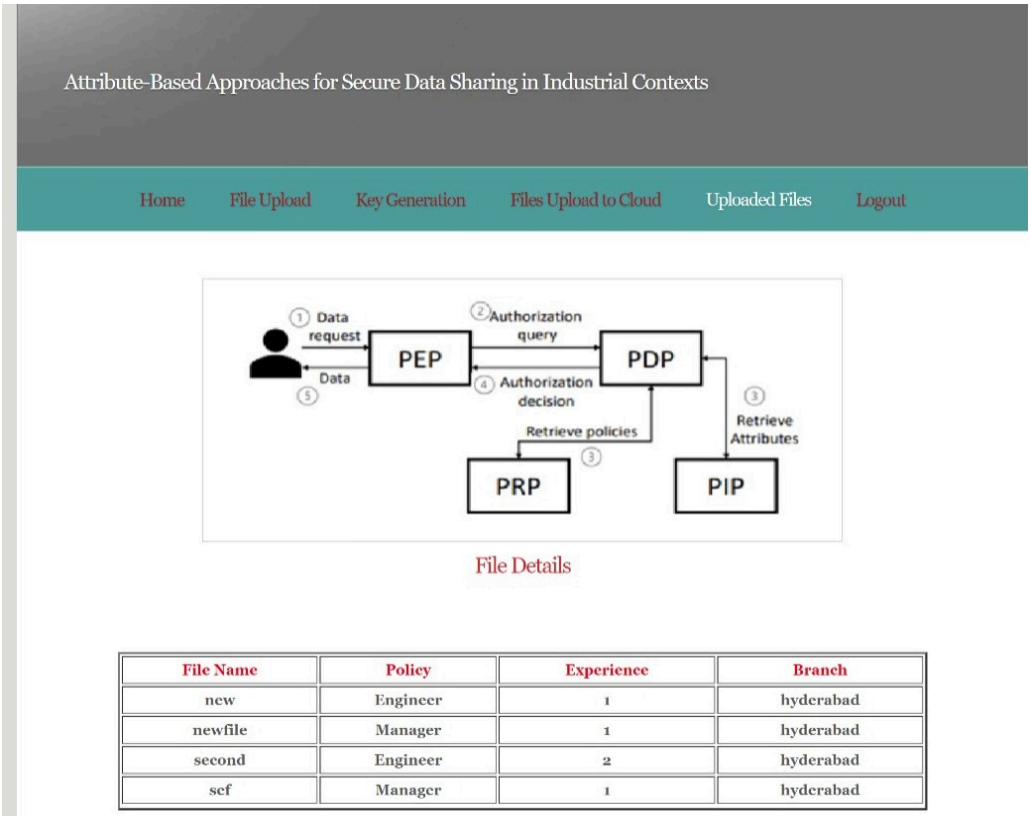


Fig 7.2.3 Owner file upload page



File Name	Policy	Experience	Branch
new	Engineer	1	hyderabad
newfile	Manager	1	hyderabad
second	Engineer	2	hyderabad
sef	Manager	1	hyderabad

Fig 7.2.4 All uploaded files

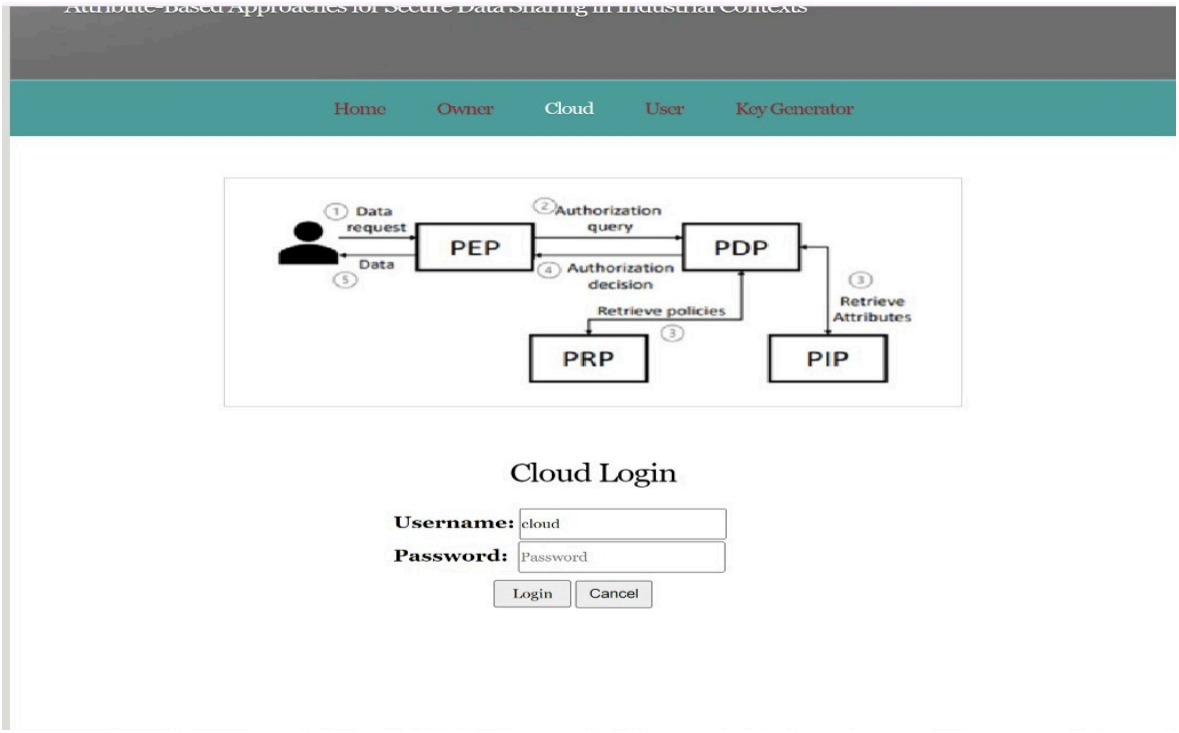
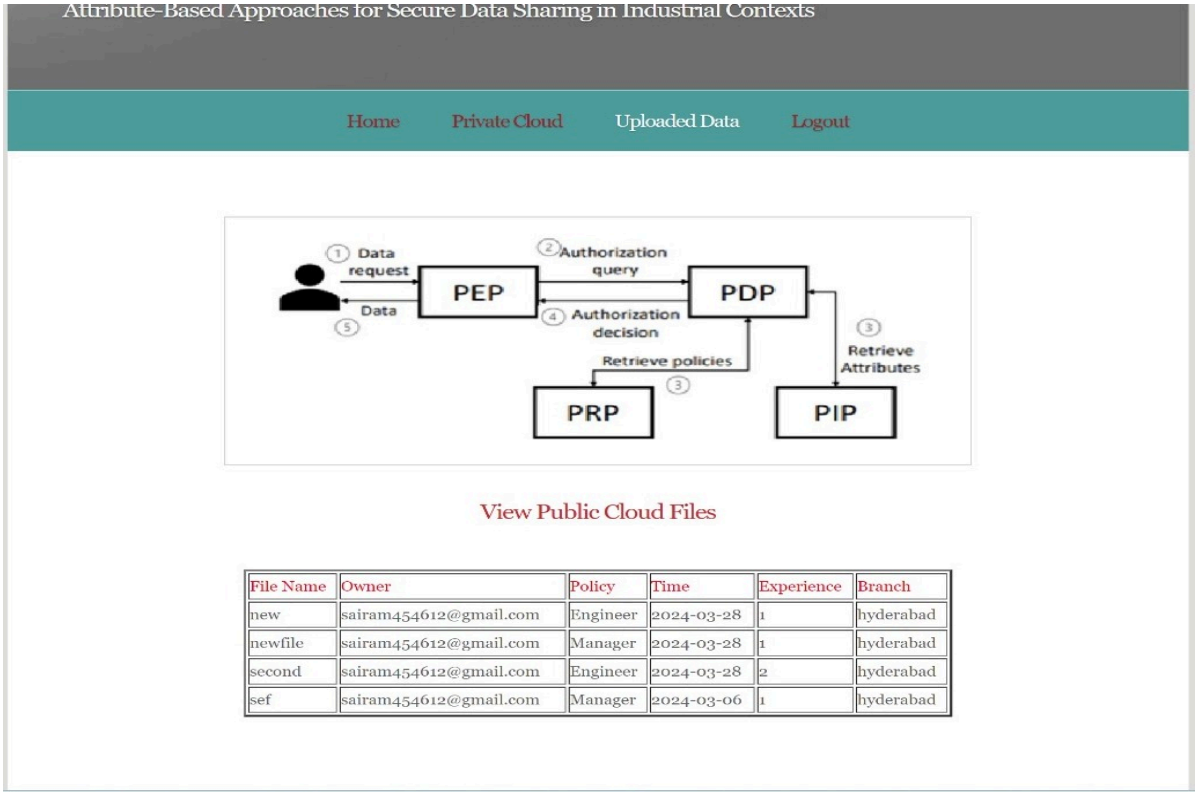


Fig 7.2.5 Cloud Login page



File Name	Owner	Policy	Time	Experience	Branch
new	sairam454612@gmail.com	Engineer	2024-03-28	1	hyderabad
newfile	sairam454612@gmail.com	Manager	2024-03-28	1	hyderabad
second	sairam454612@gmail.com	Engineer	2024-03-28	2	hyderabad
sef	sairam454612@gmail.com	Manager	2024-03-06	1	hyderabad

Fig 7.2.6 Cloud page

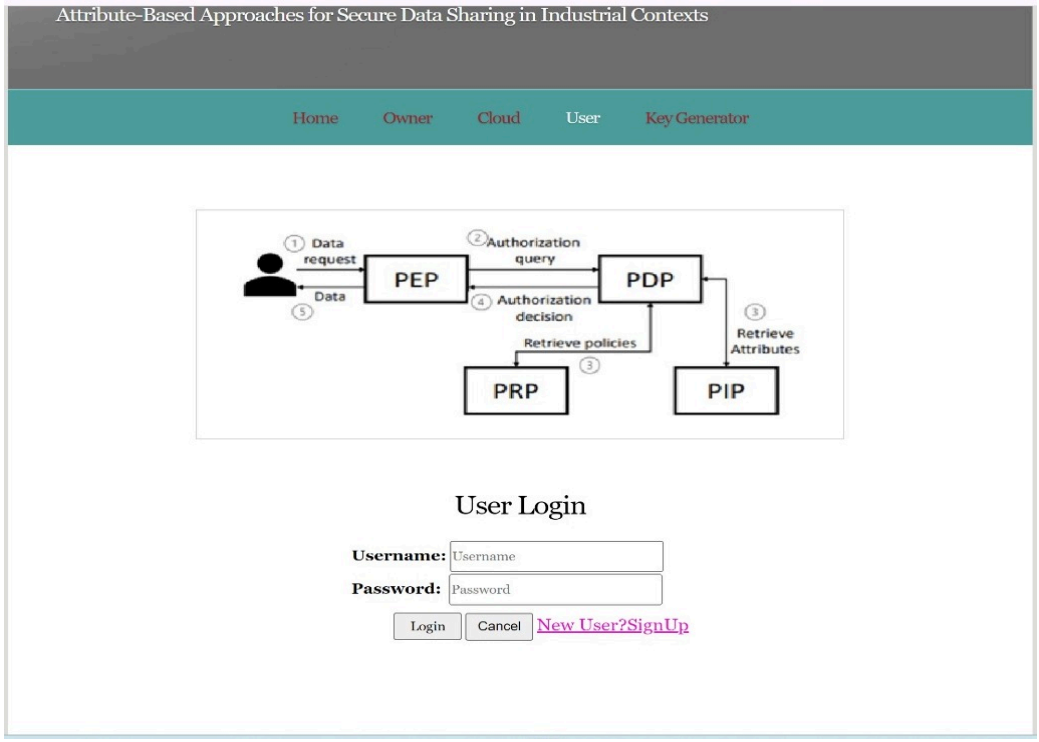
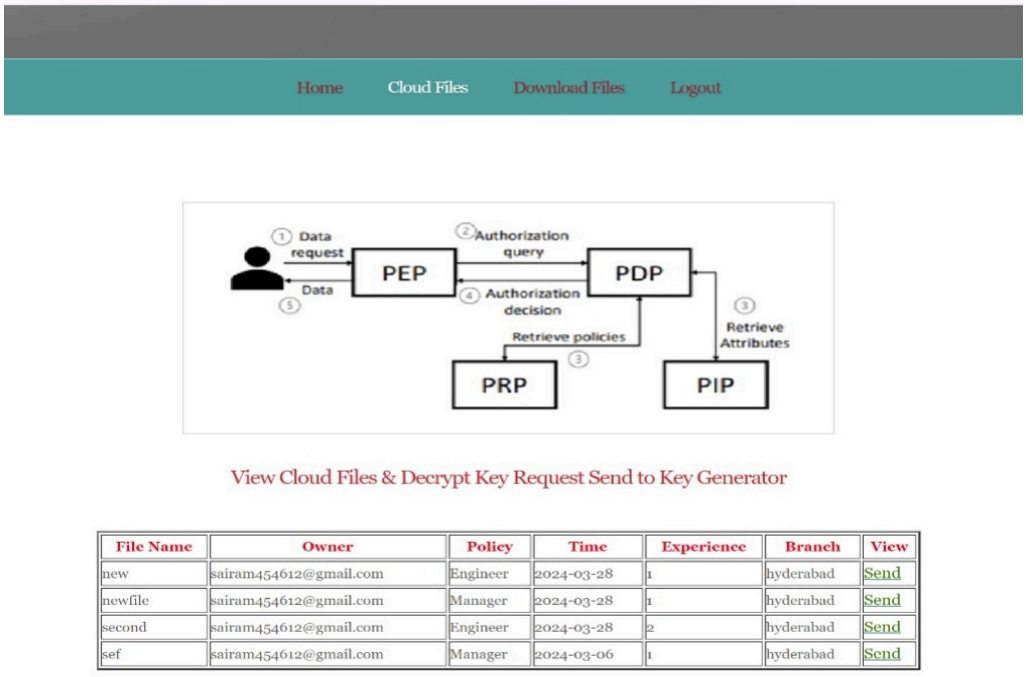


Fig 7.2.7 User Login/Registration page



File Name	Owner	Policy	Time	Experience	Branch	View
new	sairam454612@gmail.com	Engineer	2024-03-28	1	hyderabad	Send
newfile	sairam454612@gmail.com	Manager	2024-03-28	1	hyderabad	Send
second	sairam454612@gmail.com	Engineer	2024-03-28	2	hyderabad	Send
sef	sairam454612@gmail.com	Manager	2024-03-06	1	hyderabad	Send

Fig 7.2.8 Send Request For Secret Key

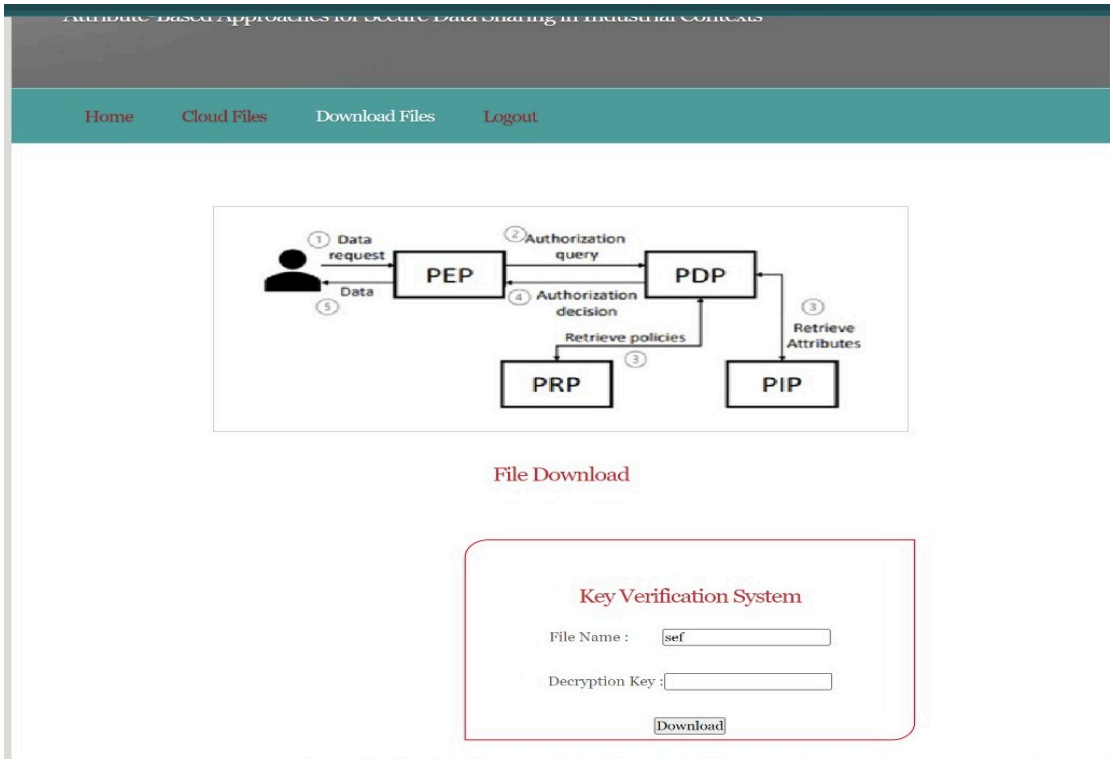


Fig 7.2.9 User File Downlaod

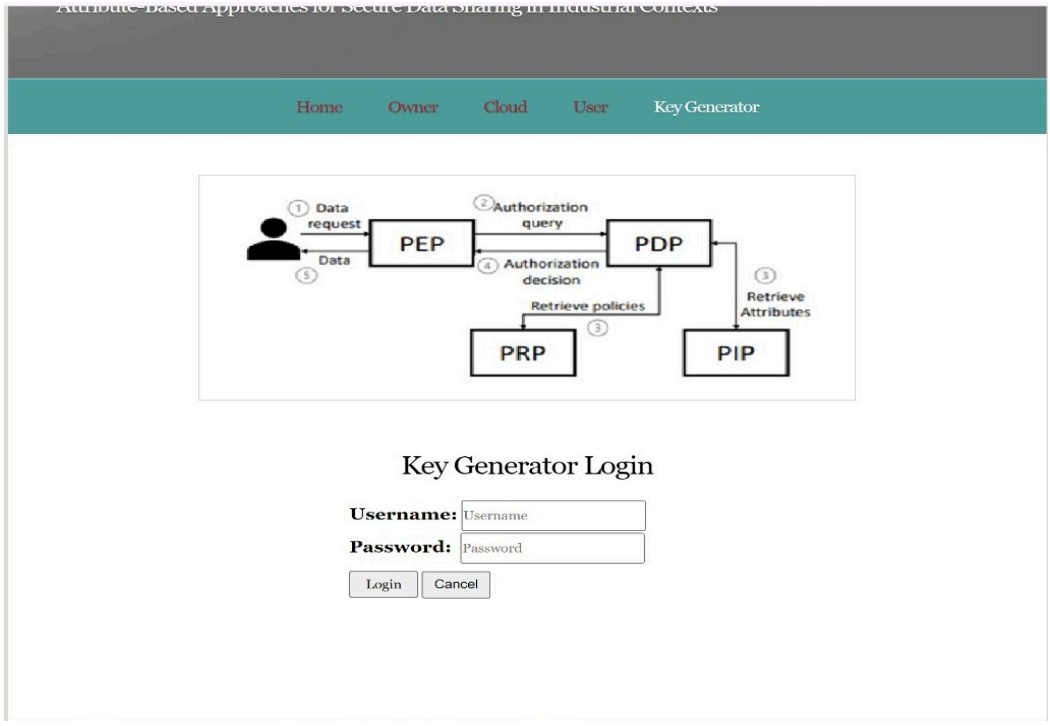
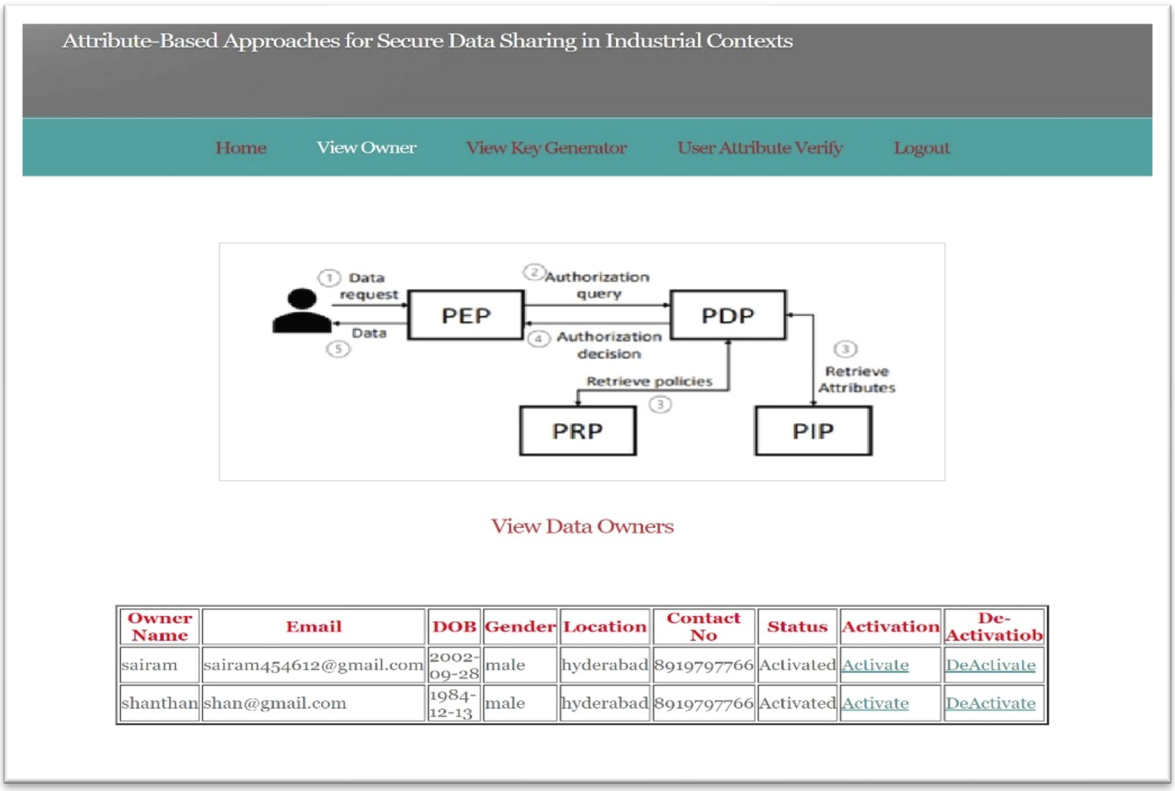
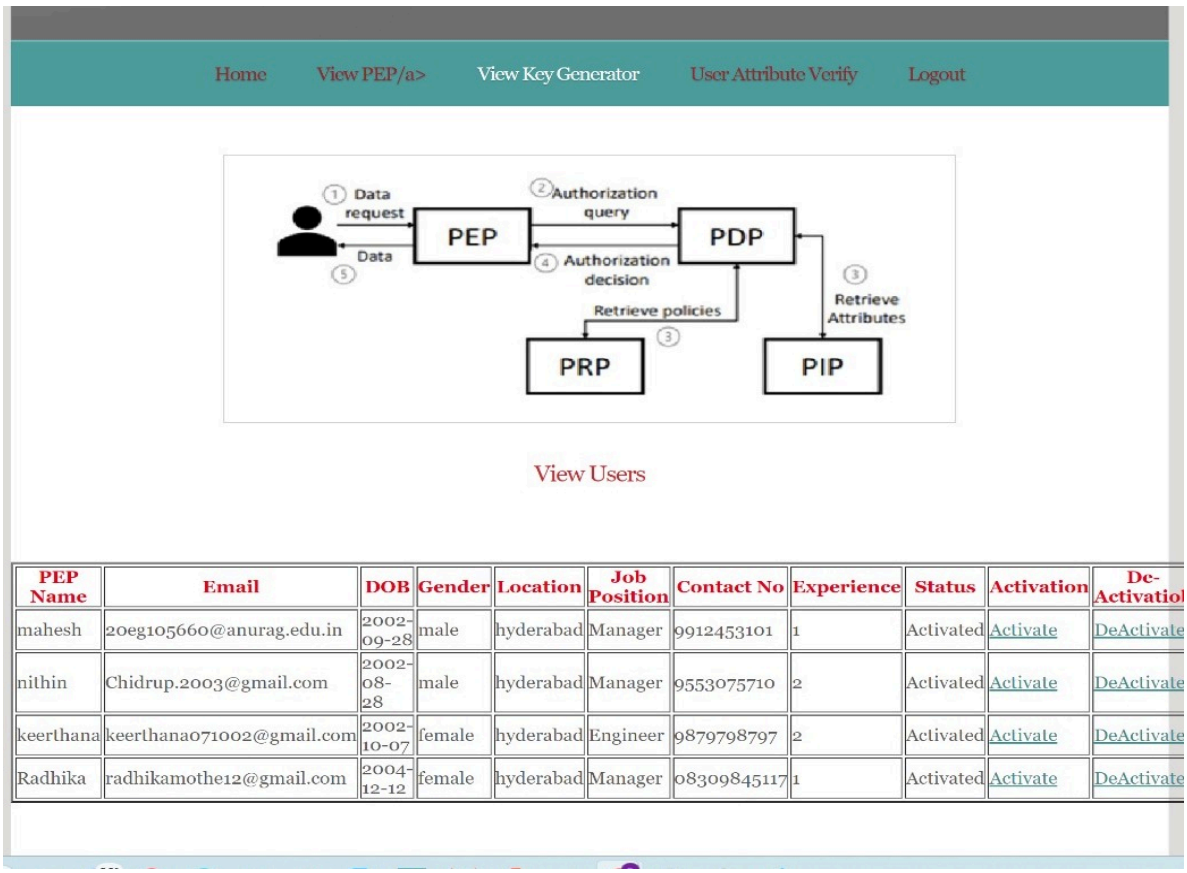


Fig 7.2.10 Key Generator Login Page



Owner Name	Email	DOB	Gender	Location	Contact No	Status	Activation	De-Activatiob
sairam	sairam454612@gmail.com	2002-09-28	male	hyderabad	8919797766	Activated	Activate	DeActivate
shanthan	shan@gmail.com	1984-12-13	male	hyderabad	8919797766	Activated	Activate	DeActivate

Fig 7.2.11 Owner Date



PEP Name	Email	DOB	Gender	Location	Job Position	Contact No	Experience	Status	Activation	De-Activatio
mahesh	20eg105660@anurag.edu.in	2002-09-28	male	hyderabad	Manager	9912453101	1	Activated	Activate	DeActivate
nithin	Chidrup.2003@gmail.com	2002-08-28	male	hyderabad	Manager	9553075710	2	Activated	Activate	DeActivate
keerthana	keerthana071002@gmail.com	2002-10-07	female	hyderabad	Engineer	9879798797	2	Activated	Activate	DeActivate
Radhika	radhikamothe12@gmail.com	2004-12-12	female	hyderabad	Manager	08309845117	1	Activated	Activate	DeActivate

Fig 7.2.12 User Data

CONCLUSION

In this project, we address the specific needs of data sharing in industrial environments by defining key properties essential for secure data exchange: fine-granularity, expressiveness, dynamic adaptability, ease of management, access revocation, and ease of implementation. To achieve these properties, we advocate for the use of ABE-enabled ABAC, combining attributes from ABAC models with attribute-based cryptographic elements to ensure data privacy in untrusted domains. Through a thorough exploration, we identify gaps and relevant mechanisms in both ABAC and ABE approaches, particularly focusing on enforcing expressive and fine-grained policies efficiently. We present a comparative analysis of state-of-the-art approaches in data sharing using ABE, pinpointing gaps in expressiveness and ease of implementation. Additionally, we propose a conceptual architecture that integrates relevant mechanisms to address these research gaps. This architecture leverages the capabilities of an ABAC policy decision point as a central element in the encryption process, enabling ABE mechanisms to be controlled by a single entity capable of utilizing the flexibility of ABAC policy languages. Lastly, we analyse knowledge gaps within our architecture and discuss deployment implications for data owners, highlighting the need for translating ABAC access policies into actionable parameters for ABE encryption and decryption algorithms.

Future enhancements of the project entail integrating emerging technologies like blockchain or homomorphic encryption, refining access control policies for dynamic adaptability, automating policy management for flexibility, improving key management practices, ensuring compliance with industry standards, optimizing scalability and performance, designing user-friendly interfaces, implementing granular data revocation mechanisms, integrating AI/ML for predictive analytics and adaptive access control, and sustaining continuous research and development efforts to stay innovative in the field. These enhancements aim to bolster security, scalability, and user experience while addressing the evolving challenges of secure data sharing in industrial contexts.

REFERENCES

[1] G. S. Mahmood, D. J. Huang, and B. A. Jaleel, ‘‘A secure cloud computing system by using encryption and access control model,’’ J. Inf. Process. Syst., vol. 15, no. 3, pp. 538–549, 2019.

[2] T. Kanwal, A. A. Jabbar, A. Anjum, S. U. Malik, A. Khan, N. Ahmad, U. Manzoor, M. N. Shahzad, and M. A. Balubaid, ‘‘Privacy-aware rela- tionship semantics–based XACML access control model for electronic health records in hybrid cloud,’’ Int. J. Distrib. Sensor Netw., vol. 15, no. 6, pp. 1–24, 2019.

[3] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, ‘‘CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme,’’ IEEE Access, vol. 7, pp. 5682–5694, 2019.

[4] D. Wee, R. Kelly, J. Cattel, and M. Breunig, ‘‘Industry 4.0-how to navigate digitization of the manufacturing sector,’’ McKinsey Company, vol. 58, no. 58, pp. 7–11, Apr. 2015.

[5] International Data Space (IDS). (2019). International Data Space—Reference Architecture Model. [Online]. Available: https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf

[6] What is Considered Personal Data Under the Eu GDPR?.[Online]. Available: https://gdpr.eu/eu-gdpr-personal-data/ (Feb. 2019)

[7] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, Microsoft Azure. New York, NY, USA: Apress, 2015.

[8] S. Granneman, Amazon Web Services, AECCU Guide, Washington Univ. St. Louis, St. Louis, MO, USA, Dec. 2012.

[9] C. Alliance, ‘‘Security guidance for critical areas of focus in cloud comput- ing V3. 0,’’ Cloud Secur. Alliance, vol. 15, no. 15, pp. 1–176, Nov. 2011.

[10] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, ‘‘Security and privacy for storage and computation in cloud comput- ing,’’ Inf. Sci., vol. 258, pp. 371–386, Feb. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025513003320

[11] S. Kamara and K. Lauter, ‘‘Cryptographic cloud storage,’’ in Proc. Int. Conf. Financial Cryptography Data Secur. Berlin, Germany: Springer, 2010, pp. 136–149.

[12] M. Falkenthal, F. W. Baumann, G. Grünert, S. Hudert, F. Leymann, and M. Zimmermann, ‘‘Requirements and enforcement points for policies in industrial data sharing scenarios,’’ in Proc. 11th Symp. Summer School Service-Oriented Comput., 2017, pp. 1–14.

[12] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, ‘‘Role-based access control models,’’ Computer, vol. 29, no. 2, pp. 38–47, 1996.

[13] D. R. Kuhn, E. J. Coyne, and T. R. Weil, ‘‘Adding attributes to role-based access control,’’ IEEE Comput., vol. 43, no. 6, pp. 79–81, Jun. 2010.

[14] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, ‘‘Attribute-based access control,’’ Computer, vol. 48, no. 2, pp. 85–88, Feb. 2015.

[15] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, ‘‘Guide to attribute based access control (ABAC) definition and consid- erations (draft),’’ NIST Special Publication, vol. 800, no. 162, pp. 1–54, 2013.