

Dissertation Title:

**An Efficient and Privacy Preserving Biometric
Identification Scheme in Cloud Computing**

Course No: SSZG628T

Course Title: Dissertation

Dissertation Work Done by:

Student Name: Manam Bharadwaj

BITS ID: 2021MT13176

Degree Program: Master's in software systems

Research Area: Cloud & Enterprise Security

Dissertation Work carried out at:

Dassault Systemes, Bengaluru



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,
PILANI**

VIDYA VIHAR, PILANI, RAJASTHAN - 333031.

September 2023

Abstract

The rapid proliferation of cloud-based services and the escalating volume of sensitive data stored therein have necessitated the development of a robust biometric authentication system. This project addresses the critical need for secure user identification in cloud computing environments. Leveraging a tripartite framework involving database owners, users, and the cloud, our approach ensures the confidentiality of biometric data throughout transmission and storage. In response to a user's identification request, encrypted biometric data is transmitted from the database owner to the cloud. The cloud server, after processing the encrypted query, returns an index to the database owner, enabling the computation of biometric trait similarity. Notably, the scheme encompasses advanced encryption techniques and privacy-preserving measures to safeguard user biometric information. Moreover, this research endeavors to thwart collusion attacks between users and the cloud, thereby fortifying the security of the system. Through rigorous analysis and experimentation, this project aims to set a benchmark for reliable and privacy-enhanced biometric identification systems in the cloud. The outcomes of this study hold promise for a future where cloud-based services afford users both convenience and the assurance of robust data protection.

Keywords: Cloud Computing, Biometric Authentication, Cryptography




		
Signature of Student	Signature of Supervisor	Signature of Additional Examiner
Name: Manam Bharadwaj	Name: Krishna M G	Name: Madanapalli Thousif Hussain

Table of Contents

Chapter 1	1
Introduction	1
1.1 Research Question	1
1.2 Aim and Objectives	2
Aim:	2
Objectives:	2
Chapter 2	2
Related Work & Background.....	2
2.1. Literature Survey.....	2
2.1.1 Gap Analysis	3
2.2 Technical Background	4
2.2.1 JAVA	4
2.2.2 J2EE	4
2.2.3 MVC Architecture.....	4
2.2.4 HTTP	5
2.2.5 JSP	6
2.2.6 JDBC	6
2.2.7 ECLIPSE.....	8
2.2.8 TOMCAT	8
2.2.9 MY SQL.....	8
2.2.10 FTP Protocol	8
2.2.11Cloud Technology.....	8
2.2.12Drive HQ.....	9
2.2.13 Cryptography Technique.....	9
AES Algorithm:	9
HashTag technique:	9
Chapter 3	10
Proposed Methodology	10
3.1 Fingerprint Recognition Systems	11
FEATURE EXTRACTION	11
RIDGE THINNING METHOD	11
MATCHING SCORE	12
3.2 Flow Diagrams.....	13

3.2.1 Data Upload Process	13
3.2.2 Query DATA Retrieval	15
3.2.3 Flow Chart	16
3.3 Modules Description	17
3.3.1 Server Configuration	17
3.3.2 Data Encryption using AES	18
3.3.3 HashTag Technique	19
3.3.4 User Server functionality	19
3.3.5 Fingerprint Comparison	21
3.3.6 Database Connection	22
3.4 Requirements.....	23
a. Software Requirements:	23
b. Hardware Requirements:.....	23
Chapter 4	24
Reference	24

Chapter 1

Introduction

In the realm of user identification, biometric authentication has emerged as a beacon of promise, offering a reliable alternative to conventional methods reliant on passwords and identification cards. The inherent advantages of biometric identification lie in its heightened reliability and unparalleled convenience. This technology leverages distinctive biometric traits, such as fingerprints, iris patterns, and facial features, which can be seamlessly captured by a diverse array of sensors. This ubiquity has facilitated its widespread adoption across a multitude of applications and industries. Within the framework of biometric identification systems, entities like the FBI, entrusted with the management of national fingerprint databases, find themselves contending with the monumental challenge of handling and securing vast troves of biometric data. To mitigate the substantial costs associated with storage and computation, database owners often seek to outsource this data to cloud servers, such as those provided by industry giants like Amazon. However, the paramount concern remains the preservation of biometric data privacy. This imperative mandates that data be encrypted prior to outsourcing, forming a protective veil around individuals' unique biometric signatures.

In practice, this encryption process presents a practical conundrum. When a partner of the FBI, such as a local police station, seeks to authenticate an individual's identity, they turn to the FBI and generate an identification query utilizing the individual's biometric traits - be it fingerprints, irises, voice patterns, or facial features. Subsequently, the FBI encrypts this query and dispatches it to the cloud, where it undergoes a search for a closely matching biometric profile. Thus, the crux of the challenge lies in crafting a protocol that harmonizes efficiency and privacy preservation in the domain of biometric identification within cloud computing.

The urgency of this challenge is underscored by the ever-expanding reliance on cloud-based services and the burgeoning volume of sensitive data held therein. The Telecom Regulatory Authority of India's (TRAI) Chairman, R.S. Sharma, thrust this issue into the limelight with a provocative social media post, daring critics to harm him. In a matter of hours, ethical hackers demonstrated the vulnerability of existing systems by accessing and publishing his sensitive personal information. This incident stands as a stark testament to the critical need for fortified security measures in the handling of sensitive biometric data, particularly within the confines of cloud computing. To date, numerous proposals have been put forth in the pursuit of privacy-preserving biometric identification solutions. However, achieving a delicate balance between efficiency and privacy preservation remains an ongoing challenge. This project seeks to address this critical gap by proposing an innovative biometric identification scheme that promises not only enhanced security and efficiency but also an unwavering commitment to preserving the privacy of user biometric data throughout its journey in the cloud. Through meticulous analysis and empirical validation, this research endeavors to propel the evolution of biometric identification systems, forging a path toward a future where cloud-based services stand as beacons of both convenience and fortified data protection.

1.1 Research Question

- How can biometric data be securely stored and processed in a cloud computing environment while preserving user privacy?
- What are the key components and algorithms required to implement an efficient and privacy-preserving biometric identification scheme in the cloud?

1.2 Aim and Objectives

Aim:

The aim of this research is to develop an advanced biometric identification scheme that ensures both efficiency and privacy preservation within the context of cloud computing environments.

Objectives:

1. Enhance Biometric Recognition Efficiency:

- Develop algorithms and techniques to improve the speed and accuracy of biometric identification processes in cloud-based systems.

2. Privacy Preservation Techniques:

- Investigate and implement state-of-the-art privacy-preserving methods to safeguard sensitive biometric information from unauthorized access or malicious attacks.

3. Secure Data Transmission and Storage:

- Design and implement robust protocols for secure transmission and storage of biometric data in cloud environments, ensuring data integrity and confidentiality.

Chapter 2

Related Work & Background

2.1. Literature Survey

(Yang, X., et.al.2021) The cloud data outsourcing for cost-effective IT services but underscores privacy concerns. It recognizes the necessity of encrypting sensitive data for protection but acknowledges the computational challenges, especially on resource-limited devices like mobile phones. In response, we propose a privacy-focused approach for cloud data outsourcing from these devices. It employs probabilistic public key encryption and ranked keyword searches on encrypted data to enhance both privacy and system usability. This approach sends only the most relevant files, reducing computation and communication overheads. Rigorous security and performance analyses confirm its semantic security and efficiency. Essentially, it addresses privacy issues in cloud data outsourcing while ensuring efficiency and usability for constrained mobile devices.

(C. Xu, et.al.2021) they introduced an approach that prioritizes computational and communication efficiency while safeguarding data privacy. The method involves encrypting both biometric data and query data before sending the cipher text to a cloud server for matching tasks. The cloud server then

returns the index of final matches, allowing the system to verify the legality of the biometric vector. Security analysis demonstrates the scheme's resilience to powerful attacks, and experimental results show its superior efficiency in computation and communication compared to existing biometric identification methods.

(Shahrukh, Mohammed, et.al.2019) The biometric identification has gained popularity. Cloud computing has incentivized database owners to outsource extensive biometric data and identification tasks, reducing storage and computation costs but raising privacy concerns. We present an efficient and privacy-preserving biometric identification outsourcing scheme. Biometric data is encrypted and sent to a cloud server. The database owner encrypts query data for cloud-based identification. A rigorous security analysis ensures the scheme's resilience against attacks and collusion. Experimental results demonstrate improved performance compared to previous protocols. Addressing the need for efficient and private biometric identification, this scheme is particularly relevant with the proliferation of biometric sensor-equipped smart devices.

(Gumaei, Abdu, et.al.2019) In the era of cloud computing, secure storage and processing of sensitive data present significant challenges. Preserving data confidentiality and integrity without any risk of leakage is imperative. Biometric recognition, especially facial recognition, has advanced for privacy protection in cloud computing, but current systems face issues like noise, variations, high processing time, and vulnerability to spoofing attacks. We introduce an innovative multispectral palmprint-based biometric cloud identification approach to address these limitations. It ensures privacy and security through encrypted multispectral palmprint features in both offline enrollment and online identification phases. Experimental results affirm its accuracy and efficiency in safeguarding cloud data.

(L. Zhu.et.al.2018) Biometric identification has gained popularity recently, driven by advancements in cloud computing. Database owners increasingly seek to outsource large biometric datasets and identification tasks to reduce storage and computation costs. However, this shift introduces potential privacy risks. We present an efficient and privacy-focused biometric identification outsourcing approach. In this scheme, query data is encrypted by the database owner and sent to the cloud for processing. The cloud conducts identification on the encrypted dataset and returns results. Rigorous security analysis demonstrates the scheme's resilience against attacks and collusion. Experimental results indicate superior performance in both preparation and identification phases compared to prior protocols.

2.1.1 Gap Analysis

The literature survey highlights several approaches to address privacy concerns in cloud data outsourcing for biometric identification. Yang et al. (2021) emphasize the need for encryption to protect sensitive data, proposing a privacy-focused method employing probabilistic public key encryption and ranked keyword searches. Xu et al. (2021) prioritize computational and communication efficiency while ensuring data privacy through encryption of both biometric and query data. Shahrukh et al. (2019) present an efficient and privacy-preserving biometric identification scheme, addressing concerns of outsourcing biometric data. Gumaei et al. (2019) introduce a multispectral palmprint-based approach, ensuring privacy and security through encrypted features. Zhu et al. (2018) also propose an efficient and privacy-focused biometric identification outsourcing approach, demonstrating superior performance compared to prior protocols. These studies collectively contribute to the ongoing efforts to enhance privacy in cloud-based biometric identification systems.

2.2 Technical Background

2.2.1 JAVA

Java is a computer-programming language that is concurrent, based objects and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most programming languages particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

2.2.2 J2EE

The Java EE stands for Java Enterprise Edition, which was earlier known as J2EE and is currently known as Jakarta EE. It is a set of specifications wrapping around Java SE (Standard Edition). The Java EE provides a platform for developers with enterprise features such as distributed computing and web services. Java EE applications are usually run on reference run times such as micro servers or application servers. Examples of some contexts where Java EE is used are e-commerce, accounting, banking information systems. The J2EE technologies consists of Servlets, Java Server Pages (JSP), Java Database Connectivity (JDBC), Enterprise Java Bean (EJB), Java Message Service (JMS) etc. In our project, we use the MVC architecture containing Servlets, JSP and JDBC technologies.

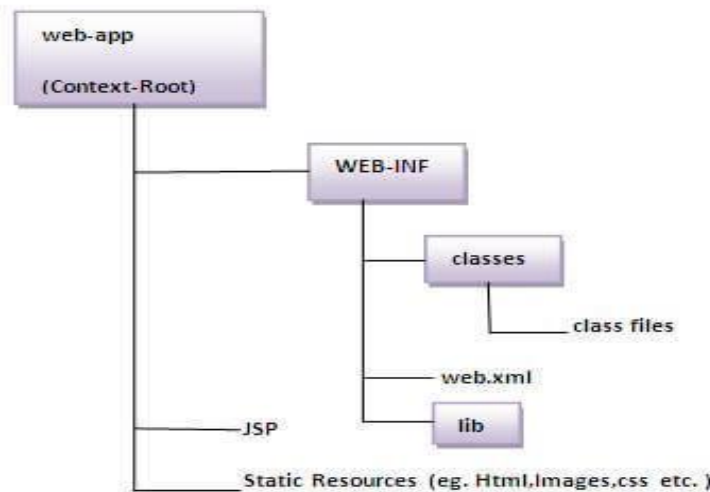


Fig 1: Directory structure of the web application

2.2.3 MVC Architecture

The Model-View-Controller (MVC) design pattern, prevalent in web development, organizes code by emphasizing separate components for data model, presentation, and control information. This project adheres to MVC architecture, implementing it in a web-based application.

In software design with model designs, the application logic is distinct from the user interface. The MVC pattern comprises three layers:

- **Model:** This component embodies the application's business logic, handling data and potentially containing the logic to notify the controller of any changes. In our project, Models are represented by .java files.
- **View:** This element forms the presentation layer, visualizing the data held within the model. In our project, Views are encapsulated in .jsp files, providing the user interface.
- **Controller:** Operating on the model and view, the controller oversees the application's flow, managing data interactions within the model and updating the view when necessary. In our project, the Controller is governed by the web.xml file, which orchestrates our web application.

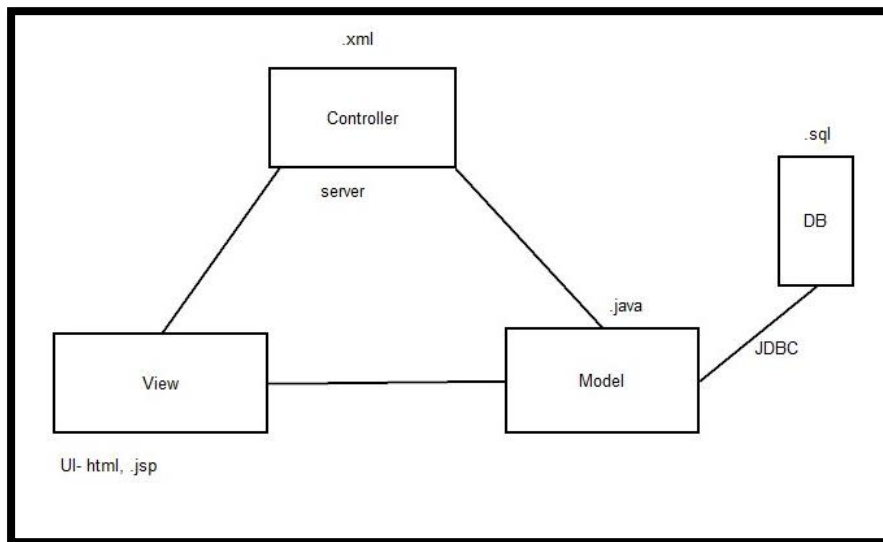


Fig 2: MVC Architecture

2.2.4 HTTP

The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems. It is the data communication protocol used to establish communication between client and server.

There are given 6 steps to create a **servlet example**.

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

These steps are required for all the servers. The servlet example can be created by three ways:

- By implementing **Servlet interface**,
- By inheriting **GenericServlet class**, (or)
- By inheriting **HttpServlet class**
- The mostly used approach is by extending **HttpServlet** because it provides http request specific method such as *doGet()*, *doPost()*, *doHead()* etc.

The **deployment descriptor** is an xml file, from which Web Container gets the information about the servlet to be invoked. Web.xml file is the deployment descriptor in our project.

```
<web-app>

<servlet>

<servlet-name>Login</servlet-name>

<servlet-class>com.admin.Login</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Login</servlet-name>

<url-pattern>/Login</url-pattern>

</servlet-mapping>

</web-app>
```

Sample web.xml file structure is given above. The explanation for each line is given below:

- <web-app> represents the whole application.
- <servlet> is sub element of <web-app> and represents the servlet.
- <servlet-name> is sub element of <servlet> represents the name of the servlet.
- <servlet-class> is sub element of <servlet> represents the class of the servlet.
- <servlet-mapping> is sub element of <web-app>. It is used to map the servlet.
- <url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.

The **Request Dispatcher** interface provides the facility of dispatching the request to another resource it may be html, Servlet or JSP. The Forward method: Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

2.2.5 JSP

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than Servlet.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development.

The scripting elements provide the ability to insert java code inside the jsp. There are three types of scripting elements:

1. scriptlet tag
2. expression tag
3. declaration tag

2.2.6 JDBC

The Model manages the business logic and data, potentially updating the Controller. In our project, it's represented by .java files. The View presents data visually, using .jsp files. The Controller, regulated by web.xml, orchestrates data flow and updates the view.

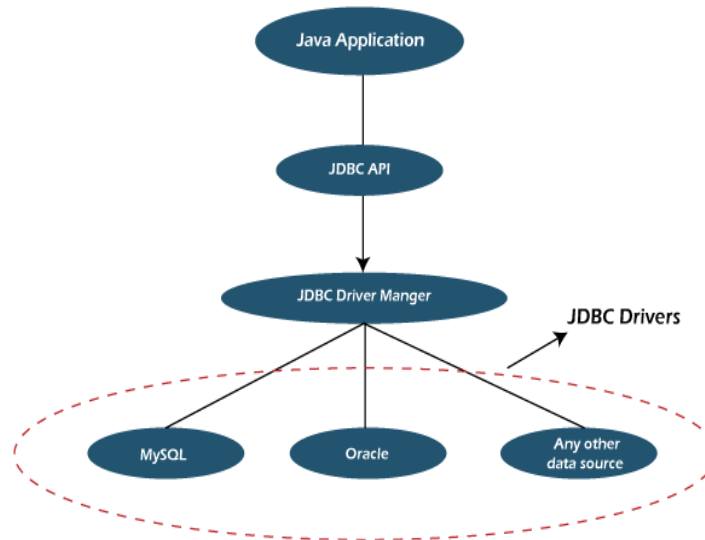


Fig 3: JDBC Architecture

The crucial role in communication between the J2EE application and the database is played by the Driver Manager, utilizing database-specific drivers. As a member of the `java.sql` package, it acts as a bridge, managing available drivers and establishing connections. In our project, we employ the `com.mysql.jar` file as the JDBC Driver.

Connecting a Java Application to a Database using JDBC involves five steps:

1. Driver Registration:

- Start by registering the driver for program usage. This step is essential and needs to be done once in your program.
- Example: `Class.forName("com.mysql.jdbc.Driver");`

2. Establish a Connection:

- After registering the driver, create connections using:
- `Connection con = DriverManager.getConnection(url, user, password);`

3. Formulate SQL Statement:

- Once a connection is established, interaction with the database is possible. Create a JDBC Statement using:
- `Statement st = con.createStatement();`

4. Execute SQL Statement:

- Execute the query, which is an SQL command. Various types of queries are possible, such as updates or inserts, and data retrieval.

5. Closing the Connection:

- Use the `close()` method of the Connection interface to terminate the connection. This step is optional.

- Example: `con.close();`
- The sample code for the JDBC establishment is given below.

```
import java.sql.*;
class AdminDAO
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("com.mysql.jdbc.Driver"); //Loading the driver

            //Creating the connection
            Connection con=DriverManager.getConnection( "jdbc:mysql://localhost:3306/db_name" , "root", "admin");

            //Creating the statement
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeQuery("select * from m_admin"); //Executing the query
            while(rs.next()) {
                System.out.println(rs.getString(1)+" "+rs.getString(2));
            }
            con.close(); //Closing the connection
        }
        catch(Exception e)
        { System.out.println(e);
        }
    }
}
```

2.2.7 ECLIPSE

Eclipse, a widely utilized integrated development environment (IDE) for programming, is primarily tailored for Java application development. However, it supports a diverse range of programming languages through customizable plug-ins, including C, C++, Python, and more. It also caters to LaTeX document creation and Mathematica software package development. Eclipse boasts dedicated tools like JDT for Java/Scala, CDT for C/C++, and PDT for PHP.

2.2.8 TOMCAT

Apache Tomcat, also known as Tomcat Server, is an open-source Java ServletContainer developed by the Apache Software Foundation (ASF). It supports JavaEE specifications like Servlet, JSP, EL, and WebSocket, providing a pure Java environment for HTTP web server operations.

2.2.9 MY SQL

Since 2008, MySQL, short for "My Sequel," has reigned as the world's most widely used open-source relational database management system (RDBMS). It serves as a server enabling multi-user access to multiple databases. MySQL employs Structured Query Language (SQL) for data manipulation. Its development project has released its source code under the GNU General Public License and various proprietary agreements. MySQL is a favored choice for web applications and a crucial component in the prevalent LAMP open-source web application software stack. Despite potential limitations in hard disk performance due to its transactional nature, MySQL is the go-to for projects requiring a reliable, feature-rich database management system.

2.2.10 FTP Protocol

FTP, or File Transfer Protocol, employs a client-server model with distinct control and data connections. Users authenticate with a username and password, or anonymously if permitted. It's the prevalent protocol for Internet file exchanges, utilizing TCP/IP for data transfer. FTP employs SSL/TLS for security, enabling seamless file sharing across remote computers.

2.2.11 Cloud Technology

Cloud computing revolutionizes IT, providing easy access to flexible resources and services. It allows rapid provisioning with minimal management. It operates on resource sharing for efficiency, akin to a

public utility. Third-party clouds liberate organizations from infrastructure concerns, letting them focus on their core operations.

2.2.12 Drive HQ

Drive HQ is a cloud-based service provider specializing in secure online file storage, sharing, and collaboration solutions. Established in 2003, it offers a range of cloud services, including cloud storage, backup, FTP, email hosting, and web hosting. With a focus on data security and privacy, Drive HQ caters to both individual users and businesses, providing a reliable platform for data management and collaboration needs.

2.2.13 Cryptography Technique

Using cryptography technique, we are going to secure our outsourcing data. To secure the data can use asymmetric or symmetric algorithms. In our project we are using the AES algorithm and using SHA1 algorithm for the hash function generation.

AES Algorithm:

AES stands for Advanced Encryption Standard and is a majorly used symmetric encryption algorithm. It is mainly used for encryption and protection of electronic data. It was used as the replacement of DES (Data encryption standard) as it is much faster and better than DES. AES consists of three block ciphers and these ciphers are used to provide encryption of data

HashTag technique:

The HashTag technique involves using a symbol (#) followed by a keyword or phrase to categorize and index content on social media platforms. This practice facilitates content discovery and engagement within specific themes or topics. It originated on Twitter and has since been adopted by various platforms like Instagram, Facebook, and TikTok. Hashtags enable users to search for and participate in discussions related to their interests, making them a powerful tool for viral marketing, activism, and community-building. Marketers and individuals alike leverage hashtags to increase the visibility and reach of their content, effectively connecting with a broader audience in the digital landscape.

Chapter 3

Proposed Methodology

We proposed *An Efficient and Privacy Preserving Biometric Identification Scheme in Cloud Computing* to retrieve data from the cloud storage with privacy preserving biometric identification scheme which can resist the security attack. Find below Fig 4 show the system architecture of the proposed work.

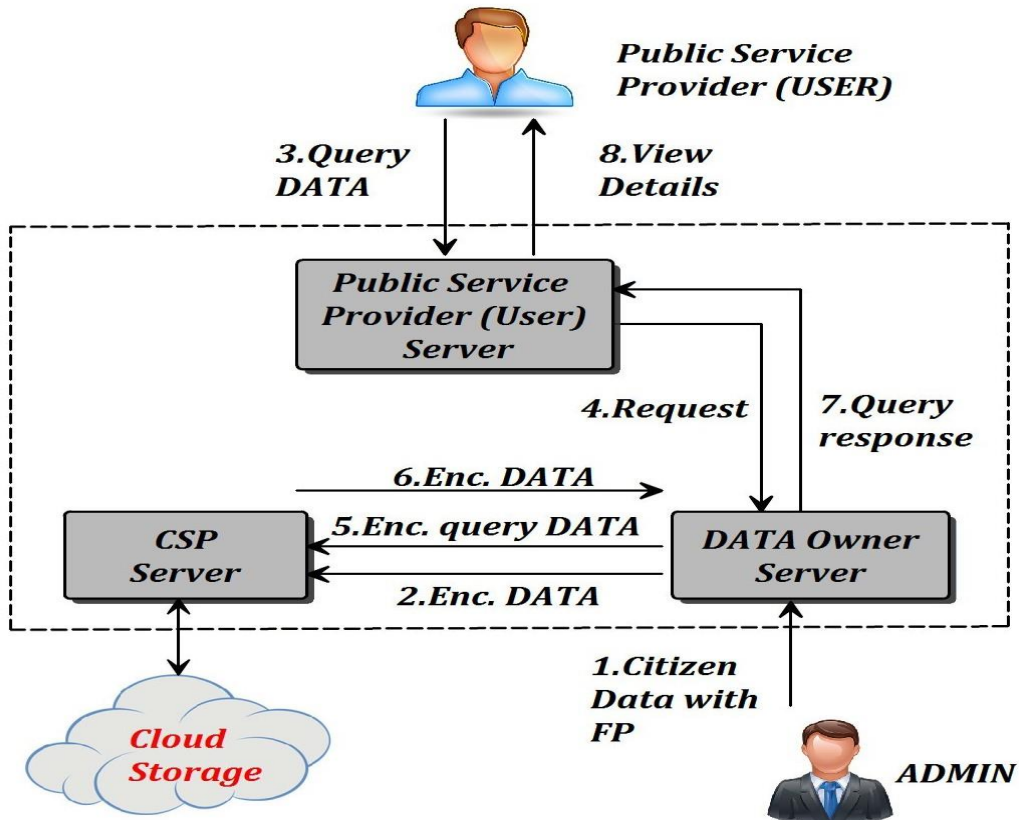


Fig 4: Proposed System Model

In this system, there are three key entities: the Admin (database owner), users (Public Service Providers), and the Cloud Service Provider (CSP). The Admin uploads secure data and biometric data to the Data Owner Server. The Data Owner Server encrypts both the secure data and biometric data, and then converts the encrypted data along with the user ID into a hash code. Subsequently, this data is transmitted to the CSP web server, where it is collected and stored in cloud storage. In this system, Drive HQ is employed for storage purposes.

When a Public Service Provider (User) needs to verify a customer, they initiate the identification verification process by sending a query data package containing the user ID and Finger Print (FP) to the Data Owner Server. The Data Owner Server converts the user ID into a hash tag and forwards it to the CSP server. The CSP server retrieves the corresponding data based on the received hash tag and sends it back to the Data Owner Server.

The Data Owner Server decrypts the received data, extracts the FP, and then compares the extracted FP with the received FP. Based on the comparison result, it determines whether the received query data is genuine or not. This information is then conveyed to the Public Service Provider server for further action.

3.1 Fingerprint Recognition Systems

Fingerprint algorithm used in this system is MINUTIAE-BASED FINGERPRINT RECOGNITION SYSTEMS. Fingerprints are the most used biometrics technique for personal identification. While the purpose of fingerprint verification is to verify the identity of a person, the goal of fingerprint identification is to establish the identity of a person. In the past three decades, automatic fingerprint verification is being more widely than other techniques of biometrics such as face identification and signature identification. Usually associated with criminal identification, now has become more popular in civilian applications, such as financial security or access control.

The most popular matching approach for fingerprint identification is usually based on lower-level features determined by singularities in finger ridge patterns called minutiae. In general, the two most prominent used features are ridge ending and ridge bifurcation (Fig. 5). More complex fingerprint features can be expressed as a combination of these two basic features. Minutiae matching essentially consist of finding the best alignment between the template (set of minutiae in the database) and a subset of minutiae in the input fingerprint, through a geometric transformation.

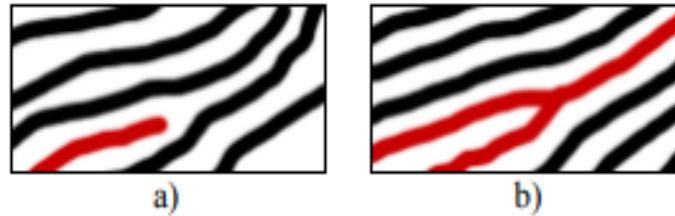


Fig 5: Example of a) ridge ending and b) bifurcation.

FEATURE EXTRACTION

Many approaches of minutia extraction process can be found. The powerful and most used method is based on binarization and ridge thinning stage.

RIDGE THINNING METHOD

The most commonly used method of minutiae extraction is the Crossing Number (CN) concept. The binary ridge image needs further processing, before the minutiae features can be extracted. The first step is to binarize and further to thin the ridges, so that they are single pixel wide (Fig. 6). A large number of skeletonization methods are available in the literature, due to important role in many recognition systems.



Fig 6: Fingerprint image a) binarization and b) skeletonization.

The minutiae points are determined by scanning the local neighborhood of each pixel in the ridge thinned image, using a 3×3 window (Fig. 7).

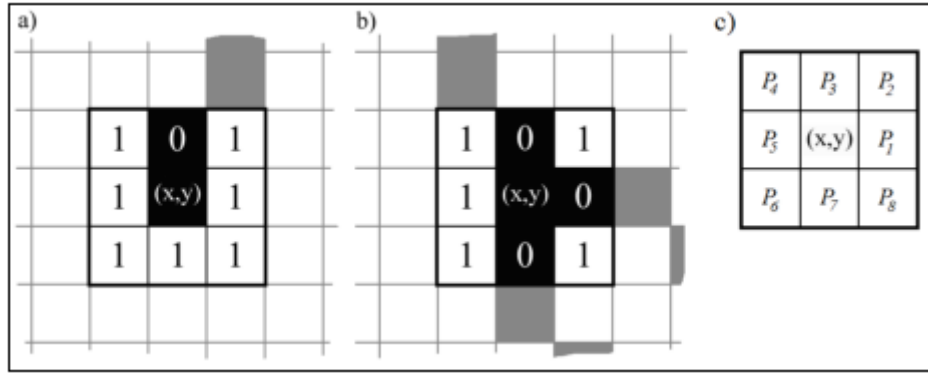


Fig 7: a) Ridge ending and b) bifurcation in c) 3x3 window

The CN value is then computed, which is defined as half the sum of the differences between pairs of neighboring pixels P_i and P_{i+1}

$$CN_{(x,y)} = \frac{1}{2} \sum_{i=1}^8 |P_i - P_{i+1}|, \quad P_9 = P_0$$

Using the properties of the CN as shown in Table (Fig. 8), the ridge pixel can be then classified as a ridge ending, bifurcation or non-minutiae point.

CN	Property
0	Isolated point
1	Ridge ending
2	Continuing ridge
3	Bifurcation
4	Crossing

Fig 8: Properties of the Crossing Number

MATCHING SCORE

In a good quality rolled fingerprint image, there are about 70 to 80 minutiae points and in a latent fingerprint the number of minutiae is much less (approximately 20 to 30).

A minutiae-based fingerprint matching system usually returns the number of matched minutiae on both query and reference fingerprints and uses it to generate similarity scores. According to forensic guidelines, when two fingerprints have a minimum of 12 matched minutiae, they are considered to have come from the same finger.

Matching algorithm compares two minutiae sets: template $T = \{m_1, m_2, \dots, m_j\}$ from reference fingerprint and input $I = \{m_1, m_2, \dots, m_i\}$ from the query and returns similarity score $S(T, I)$.

The minutiae pair m_i and m_j are considered to be match only if difference in their position and directions are lower than tolerance distances:

$$sd(m_i, m_j) = 1 \Leftrightarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq r_0$$

Fingerprint Algorithm:

Step1: Initialize colors

Step2: Open and create the buffered image

Step3: Create the binary picture

Step4: Generate Greymap

Step5: Binary local Result

Step6: Remove Noise (Remove noise from the binary picture using mean algorithm)

Step7: Skeletonization (We skeletonize until there are no changes between two iterations)

Step8: Direction (Convert the direction matrix to direction buffered image)

Step9: Minutiae (intersections-Extract intersection points from the binary image, which is a kind of minutiae)

Step10: Minutiae (endpoints - Extract end points from the binary image, which is a kind of minutiae)

3.2 Flow Diagrams

3.2.1 Data Upload Process

DFD-BioMetric DATA Upload Process

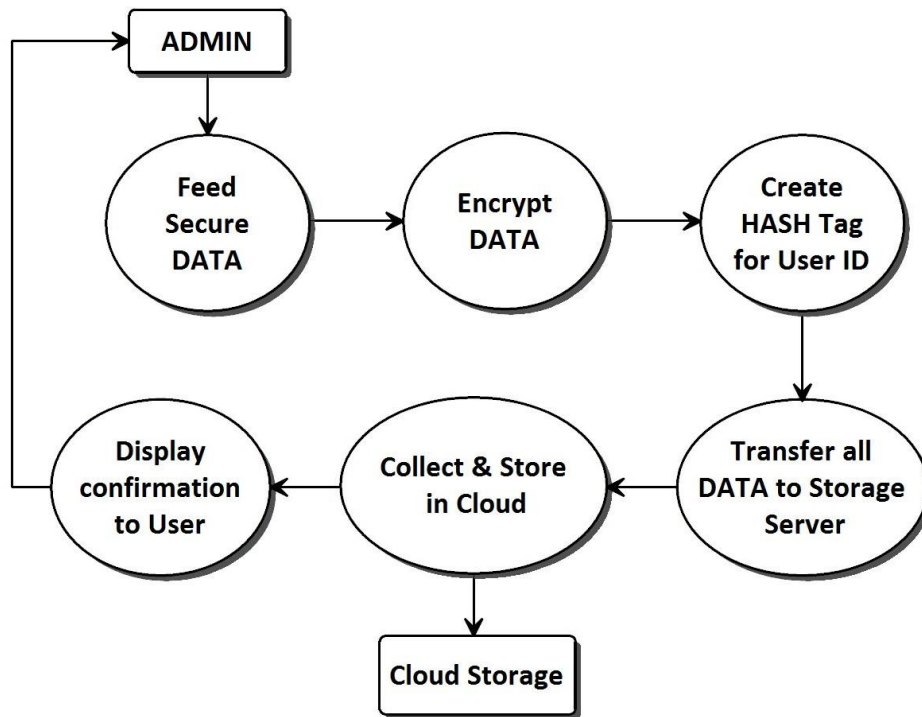


Fig 9: Data Flow Diagram for a biometric data upload process

The image you sent shows a data flow diagram (DFD) of a biometric data upload process in cloud computing. The DFD shows the following steps:

1. Admin feeds secure data to the cloud storage server.
2. The system creates a hash tag for the user ID.
3. The system encrypts the data.
4. The system collects and stores the data in cloud storage.
5. The system transfers all data to the storage server.
6. The system displays a confirmation message to the Admin.

The DFD also shows the following external entities:

- **ADMIN** represents the system administrator who is responsible for uploading biometric data to the cloud storage server.
- **CLOUD STORAGE** represents the cloud storage server where the biometric data is stored.
- The DFD does not show any internal processes, such as how the hash tag is created or how the data is encrypted. However, it does provide a high-level overview of the biometric data upload process and the entities involved.

Here is a more detailed explanation of each step in the DFD:

1. **Admin feeds secure data to the cloud storage server.** This can be done in a variety of ways, such as uploading a file using a web service.
2. **The system creates a hash tag for the user ID.** A hash tag is a unique identifier that is generated from the user ID. It is used to protect the user's privacy and to prevent unauthorized access to the biometric data.
3. **The system encrypts the data.** This is done to protect the data from unauthorized access. The data can be encrypted using a variety of encryption algorithms, such as AES Algorithm.
4. **The system collects and stores the data in cloud storage.** The data is stored in a secure location in the cloud storage server.
5. **The system transfers all data to the storage server.** This step is optional and may not be necessary in all cases. However, it can be used to improve performance and reliability.²
6. **The system displays a confirmation message to the Admin.** This message lets the admin know that their biometric data has been successfully uploaded to the cloud storage server.

This DFD can be used as a starting point for designing a biometric data upload process in cloud computing. It is important to note that the specific steps involved in the process may vary depending on the specific implementation.

3.2.2 Query DATA Retrieval

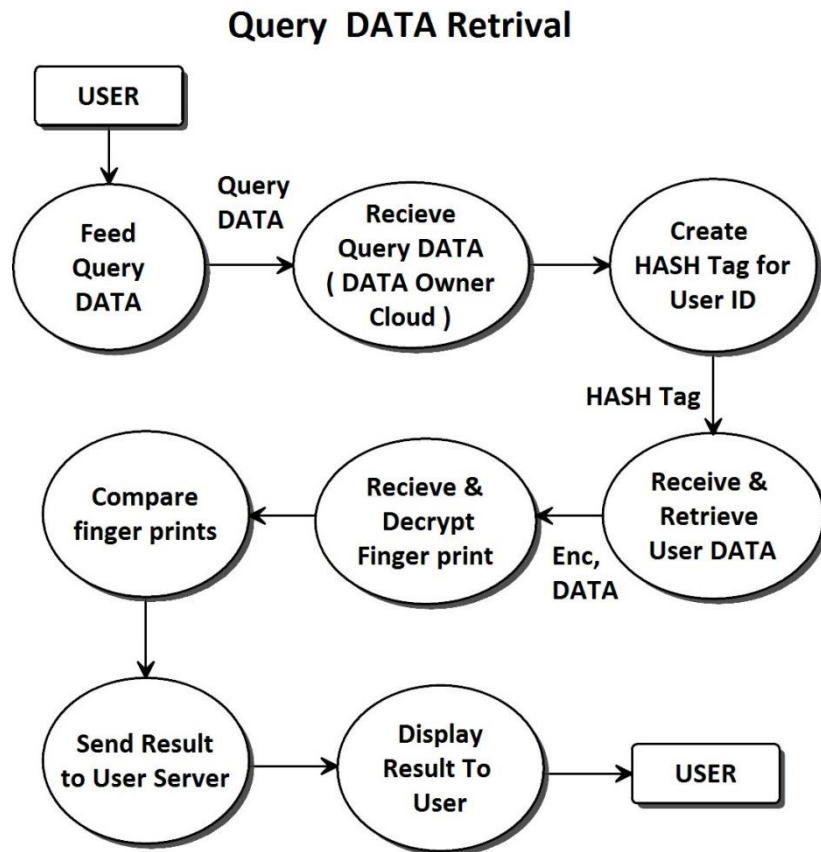


Fig 10: process of query data retrieval

The image shows the process of querying data in a biometric identification scheme in cloud computing. The process is as follows:

1. The user feeds a query data to the system.
2. The system compares the user's fingerprint with the fingerprints stored in the cloud.
3. If the fingerprint match is successful, the system creates a hash tag for the user ID.
4. The system sends a request to the user server to retrieve the user's data.
5. The user server decrypts the user's data and sends it back to the system.
6. The system displays the user's data to the user.

This process ensures that the user's data is protected from unauthorized access, even though it is stored in the cloud. The user's fingerprint is used to authenticate the user and to create a hash tag for the user ID. The hash tag is used to retrieve the user's data from the user server. The user server decrypts the user's data before sending it back to the system. This ensures that the user's data is always encrypted when it is stored in the cloud and when it is being transmitted between the system and the user server.

The system also stores a list of recent timestamps for each user. If the system receives a query data

with a timestamp that is older than any of the recent timestamps for the user, the system rejects the query data.

3.2.3 Flow Chart

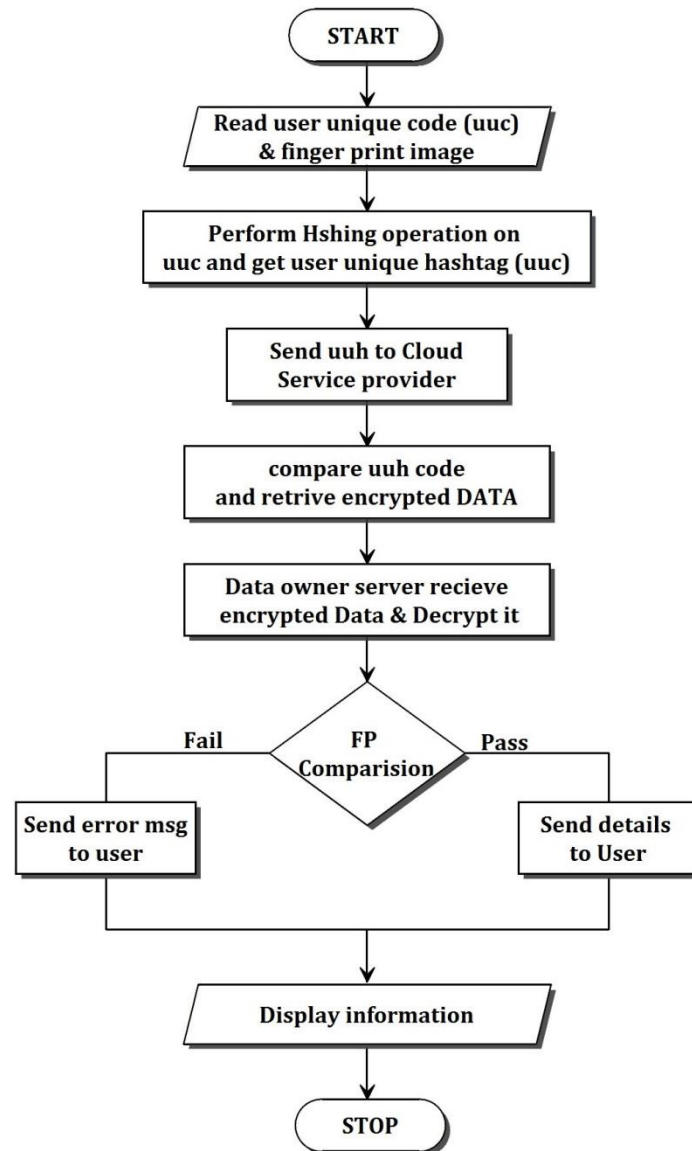


Fig 11: Flow Chart

The flow chart you sent depicts a biometric identification scheme in cloud computing that is designed to be efficient and privacy-preserving. The scheme works as follows:

1. The user provides their unique code (uuc) and fingerprint image to the system.
2. The system performs a hashing operation on the uuc to generate a unique hashtag (uuh).
3. The uuh is sent to the cloud service provider (CSP) server.
4. The CSP compares the uuh to the uuhs of all users in its database. If a match is found, the CSP retrieves the encrypted data associated with the user's account.

5. The CSP sends the encrypted data to the data owner server.
6. The data owner server decrypts the data and performs a fingerprint comparison. If the fingerprint matches, the user is authenticated. If the fingerprint does not match, the user is denied access.
7. The data owner server sends the results of the authentication to the user.

This scheme is designed to be efficient because the CSP only needs to store the uuhs of all users, not their entire biometric data. This makes the scheme scalable and allows it to support a large number of users.

The scheme is also designed to be privacy-preserving because the uuh is a one-way hash of the uuc. This means that it is impossible to reverse the hashing operation and obtain the uuc from the uuh. This protects the user's privacy because the CSP does not have access to the user's uuc.

Overall, the flow chart you sent depicts an efficient and privacy-preserving biometric identification scheme in cloud computing.

3.3 Modules Description

3.3.1 Server Configuration



Fig 12: Cloud Server Configuration



Fig 13: Admin User Interface

Here having three servers Data owner server, Cloud storage server and User server.

3.3.2 Data Encryption using AES

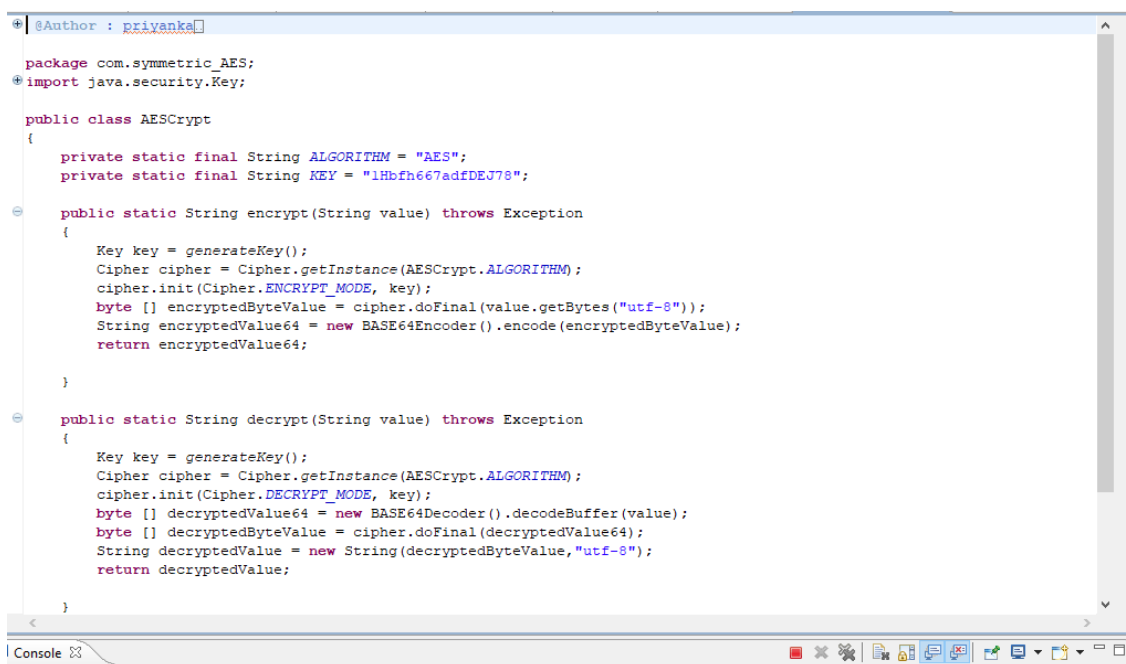


Fig 14: AES Algorithm

Whenever dataowner is uploading the query data, the file has to get encryption by using AES algorithm. So our query data will be secure manner.

3.3.3 HashTag Technique

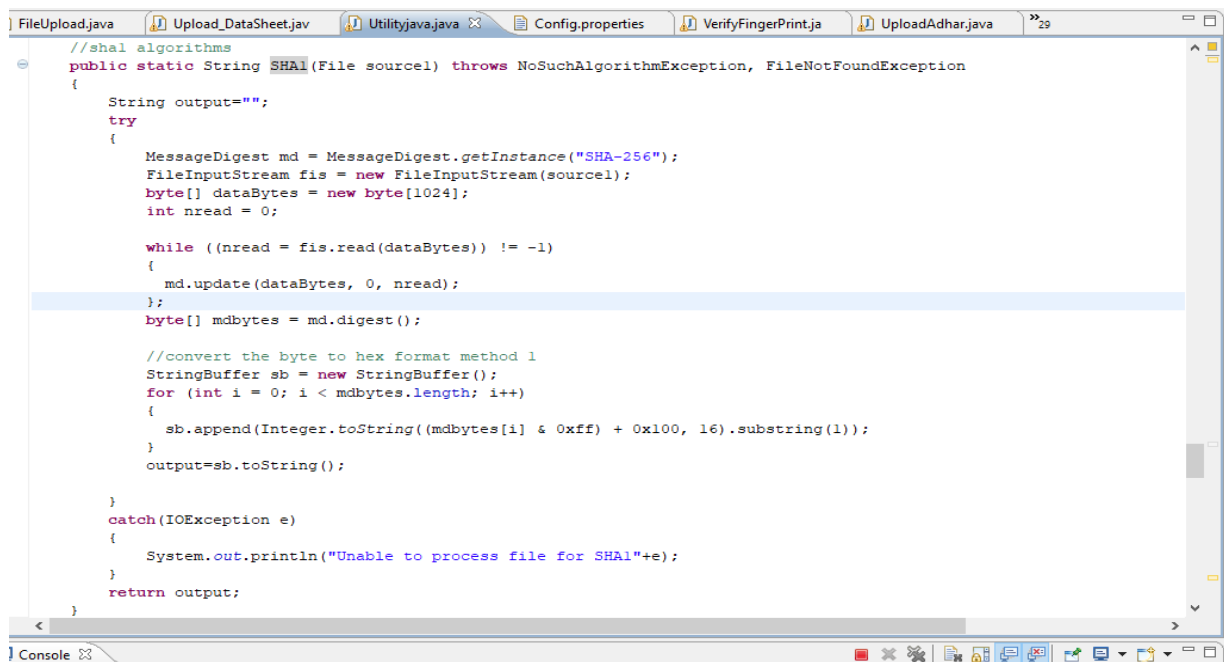


Fig 15: SHA-256 Algorithm

To secure UserId, this system is generating the hash tag by using SHA256 algorithm. Which is helpful to know the data whether it's modified anything or corrupted.

3.3.4 User Server functionality

User has to give their User id so the request will be sent to the dataowner.

A screenshot of a web application interface titled "Admin Session - Data Upload Interface". The interface has a blue header bar with "Welcome to User" on the left and "LOGOUT" on the right. Below the header, there is a sidebar menu on the left with links: "Upload Adhar Details", "Retrieve Adhar Details", "Change Password", and "Logout". The main content area is titled "Upload DataSheet" and contains a form with the following fields: "Adhar No" (a text input field), "Upload Finger Print" (a "Choose File" button with "No file chosen" text), and "Upload Adhar Image" (a "Choose File" button with "No file chosen" text). A "Submit" button is located at the bottom right of the form.

Fig 16: Admin Session – Data Upload Interface

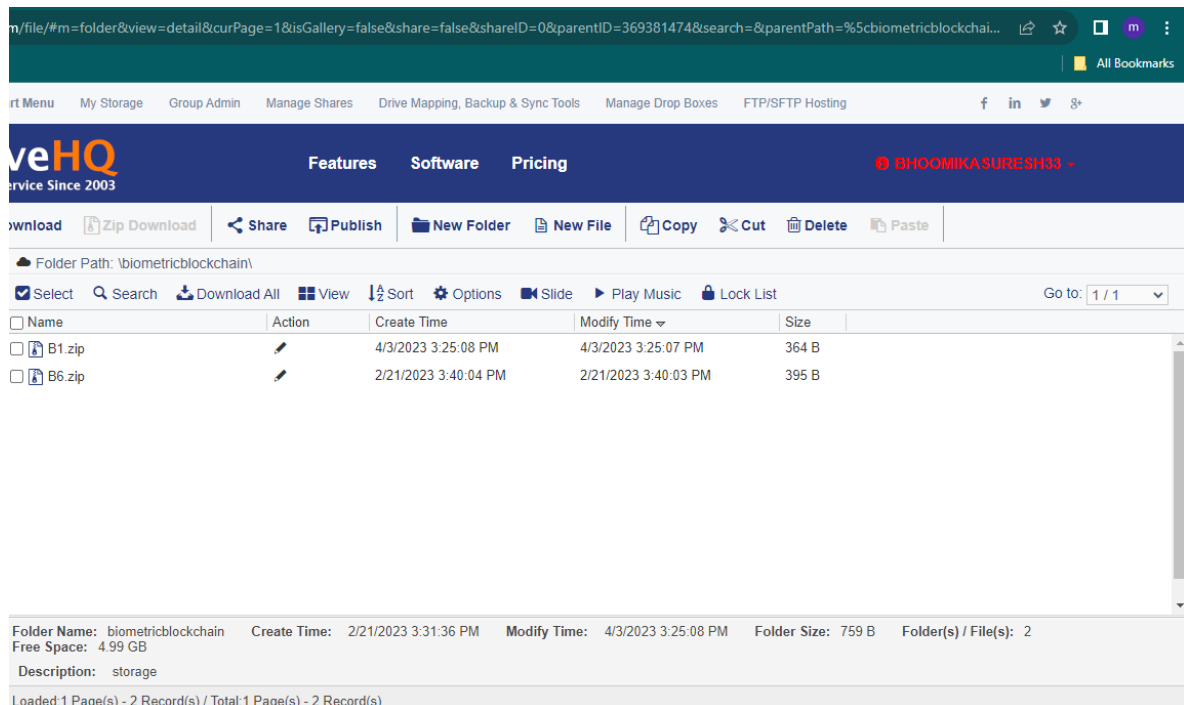


Fig 19: Drive HQ cloud storage

From data owner server to cloud storage server communication used web service concept. So Data owner has to feed the query data, those data has to get encrypted and that encrypted data have to store it in the cloud storage.

3.3.5 Fingerprint Comparison

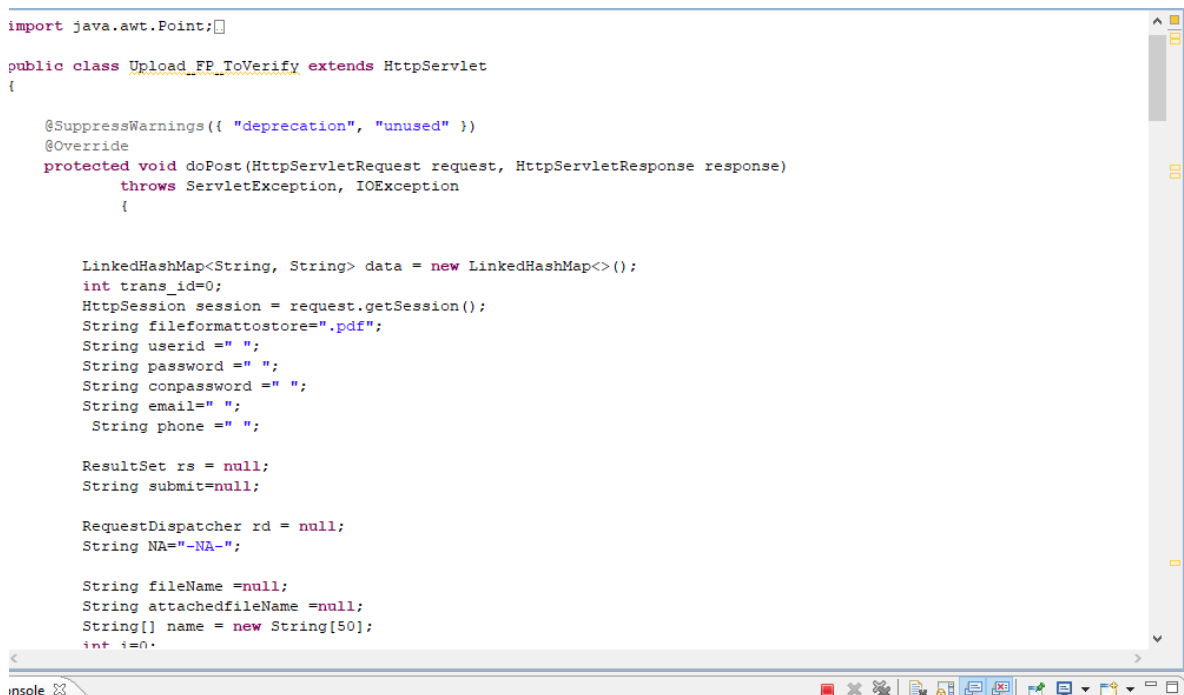


Fig 20: Fingerprint Comparison code

Have to compare with Dataowner finger print feature and user given fingerprint both if its matching the query data retrieval process will be success

3.3.6 Database Connection

This project we developed as web based application which is having model view controller.

Model is the mysql which is stored in database



```
public class Database
{
    private static Database database=null;
    private Database ()
    {

    }

    public static Database newInstance()
    {
        if(database==null)
        {
            database=new Database();
        }
        return database;
    }

    public Connection connector()
    {
        Connection con=null;
        try {

            Class.forName(Global.JDBC_DRIVER);
            System.out.println("Driver has loaded");
            con = DriverManager.getConnection(Global.JDBC_HOST_URL_WITH_DBNAME,Global.DATABASE_USERNAME, Global.DATABASE_PASSWORD);
            System.out.println("Connected" + con);

        } catch (Exception e) {
            System.out.println("Exception in serverconnector-->connector(): " + e);
        }
        return con;
    }
}
```

Fig 21: JDBC Connection (java class file)

View is the Jsp which giving the user interface of our application



```
</button>
<div class="navbar-brand">
    <span><h1 style="color:black;">Biometric Identification Using Finger print</h1></span>
</div>
</div>
<!-- <div class="navbar-collapse collapse"> -->
<div class="menu">
    <ul class="nav nav-tabs" role="tablist" style="margin-top:10px;">
        <!-- <li role="presentation" style="margin-top:20px;"><i class="fa fa-home fa-lg fa-3x" ></i><button type="button" class="btn btn-primary"><a href="#">Home</a></button></li>
        <li role="presentation" style="margin-top:20px;"><button type="button" class="btn btn-primary"><a href="#">Log Out</a></button></li>
        <li role="presentation" style="margin-top:20px; margin-right:20px;"><button type="button" class="btn btn-secondary"><a href="#">Register</a></button></li>
        <li role="presentation" style="margin-top:20px;"><button type="button" class="btn btn-primary"><a href="#">Log In</a></button></li>
    </ul>
</div>
</div>
</nav><!--</nav-->
</header><!--</header-->

<div class="slider">
    <div id="about-slider">
        <div id="carousel-slider" class="carousel slide" data-ride="carousel">
            <!-- Indicators -->
            <ol class="carousel-indicators visible-xs">
                <li data-target="#carousel-slider" data-slide-to="0" class="active"></li>
                <li data-target="#carousel-slider" data-slide-to="1"></li>
                <li data-target="#carousel-slider" data-slide-to="2"></li>
            </ol>
        </div>
    </div>
</div>
```

Fig 22: Sample JSP Code

Controller is the servlet which is controlling our web application

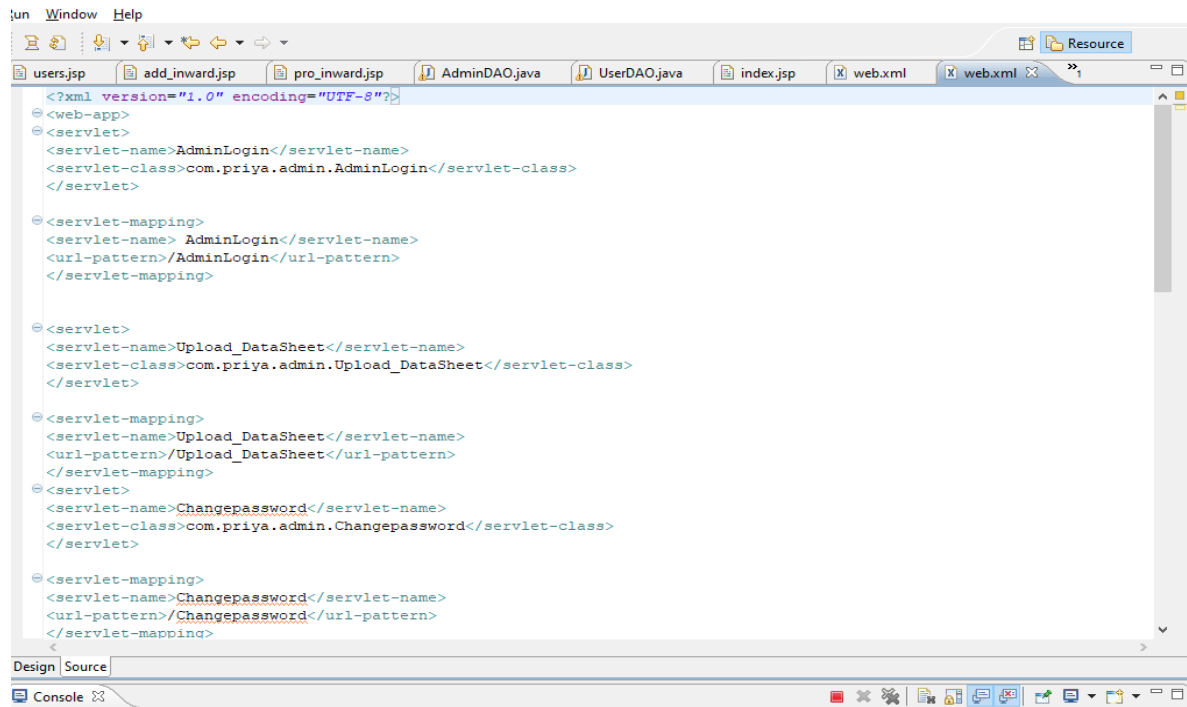


Fig 23: Web.xml Partial View

3.4 Requirements

a. Software Requirements:

Operating system	:	Windows OS
Coding Language	:	Java (Jdk 1.7)
Web Technology	:	Servlet, JSP
Web Server	:	TomCAT 6.0
IDE	:	Eclipse Indigo
Database	:	My-SQL 5.0
UGI for DB	:	SQLyog
JDBC Connection	:	Type 4

b. Hardware Requirements:

System	:	Intel I3 Core IV 2.4 GHz.
Hard Disk	:	500 GB.
Ram	:	8 GB

Chapter 4

Reference

1. Yang, X., Zhu, H., Wang, F. et al. MASK: Efficient and privacy-preserving m-tree based biometric identification over cloud. *Peer-to-Peer Netw. Appl.* 14, 2171–2186 (2021). <https://doi.org/10.1007/s12083-021-01120-7>
2. <https://ieeexplore.ieee.org/abstract/document/9724472>
3. Xu, L. Zhang, L. Zhu, C. Zhang and K. Sharif, "Achieving Efficient and Privacy-preserving Biometric Identification in Cloud Computing," 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 2021, pp. 363-370, doi: 10.1109/TrustCom53373.2021.00063.
4. Shahrukh, Mohammed, Rafi Hussain, and Abdul Rais Shaikh Farhan. "An efficient and privacy preserving biometrics identification scheme in cloud computing." *J Eng Sci* 10.1 (2019): 13-15.
5. Gumaei, Abdu, et al. "Anti-spoofing cloud-based multi-spectral biometric identification system for enterprise security and privacy-preservation." *Journal of Parallel and Distributed Computing* 124 (2019): 27-40.
6. L. Zhu, C. Zhang, C. Xu, X. Liu and C. Huang, "An Efficient and Privacy-Preserving Biometric Identification Scheme in Cloud Computing," in *IEEE Access*, vol. 6, pp. 19025-19033, 2018, doi: 10.1109/ACCESS.2018.2819166.