Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 12, No. 1, December : 2023

DESIGNING SECURE AND EFFICIENT BIOMETRIC BIOMETRIC-BASED SECURE ACCESS MECHANISM FOR CLOUD SERVICES

PARAVADA NARASIMHA MURTY (M.Tech) Avanthi Institute of Engineering and Technology, Makavarapalem, Anakapalle, Visakhapatnam, Andhra Pradesh, India.
Mr.B GANESH M.Tech,(Ph.D)
ASSISTANT PROFESSOR Dept. of Computer science Engineering Avanthi Institute of Engineering and Technology, Makavarapalem, Anakapalle, Visakhapatnam, Andhra Pradesh, India.

Abstract

With the increasing reliance on cloud services for data storage, processing, and collaboration, the need for robust and user-friendly access mechanisms is paramount. Traditional username and password authentication systems are susceptible to various security threats, emphasizing the urgency of adopting more secure and efficient alternatives. Biometric-based access mechanisms have gained attention due to their potential to enhance security and user experience. This research focuses on designing a biometric-based secure access mechanism tailored for cloud services, addressing the unique challenges posed by the cloud computing environment.

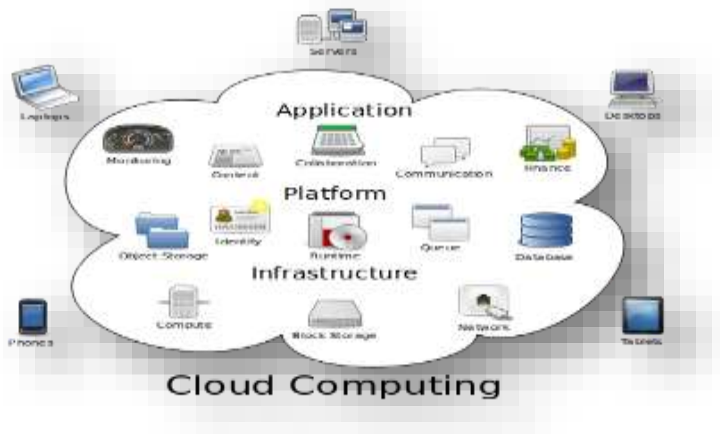
The proposed system leverages biometric authentication, utilizing physiological or behavioral traits unique to individuals, such as fingerprints, iris patterns, or facial features. Biometric data is inherently more secure than traditional credentials and can provide a seamless user experience. However, the integration of biometrics into a cloud environment requires careful consideration of privacy, security, and efficiency concerns.

Kay words: Biometric Data Encryption, Template Protection, Cloud-Integrated Authentication Protocol, Multi-Factor Authentication Integration, Usability and User Acceptance.

Introduction

What is cloud computing?

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.



Structure of cloud computing

How Cloud Computing Works:

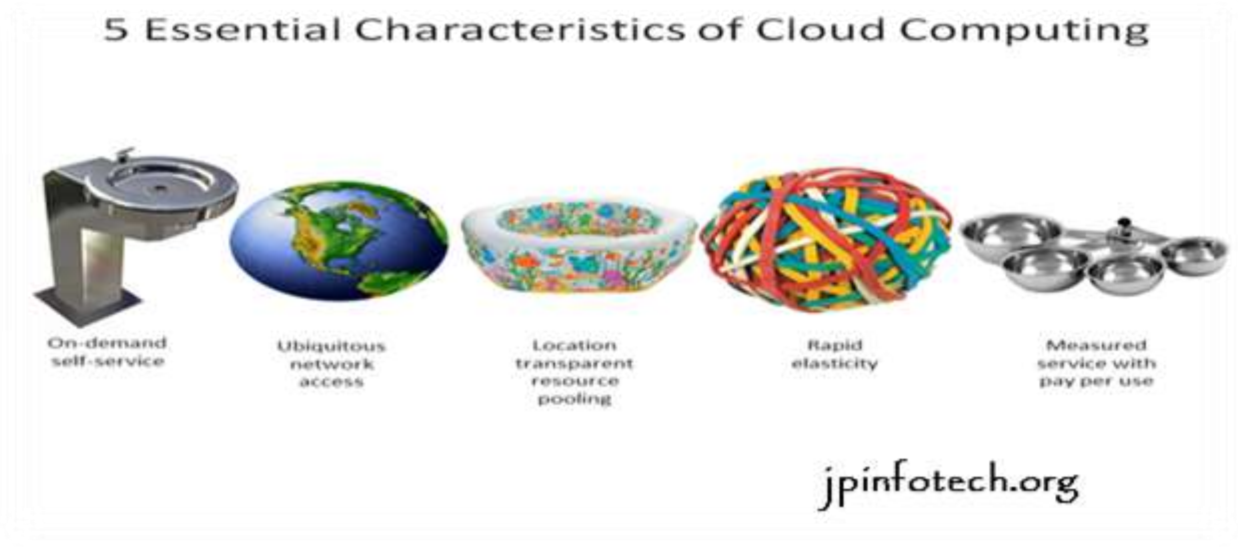
The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.

The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

Characteristics and Services Models:

The salient characteristics of cloud computing based on the definitions provided by the National Institute of standards and terminology (NIST) are outlined below:

- **On-demand self-service**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- **Broad network access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- **Rapid elasticity**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.


5 Essential Characteristics of Cloud Computing
On-demand self-service
Ubiquitous network access
Location transparent resource pooling
Rapid elasticity
Measured service with pay per use
jpinfotech.org

Characteristics of cloud computing

LITERATURE SURVEY:

1. **Privacy-preserving finger code authentication:**

1.1AUTHORS: Mauro Barni, Mario Di Raimondo, Tiziano Bianchi, and Dario Catalano.

We present a privacy preserving protocol for finger print based authentication. We consider a scenario where a client equipped with a fingerprint reader is interested into learning if the acquired fingerprint belongs to the database of authorized entities managed by a server. For security, it is required that the client does not learn anything on the database and the server should not get any information about the requested biometry and the outcome of the matching process. The proposed protocol follows a multi-party computation approach and makes extensive use of homomorphic encryption as underlying cryptographic primitive.

To keep the protocol complexity as low as possible, a particular representation of fingerprint images, named Finger code, is adopted. Although the previous works on privacy-preserving biometric identification focus on selecting the best matching identity in the database, our main solution is a generic identification protocol and it allows to select and report all the enrolled identities whose distance to the user's finger code is under a given threshold.

Variants for simple authentication purposes are provided. Our protocols gain a notable bandwidth saving (about 8−24%) if compared with the best previous work [1] and its computational complexity is still low and suitable for practical applications. Moreover, even if such protocols are presented in the context of a finger print based system, they can be generalized to any biometric system that shares the same matching methodology, namely distance computation and thresholding.

2) **Efficient privacy-preserving biometric identification:**

2.1AUTHORS: Yan Huang, Lior Malka, David Evans, and Jonathan Katz

We present an efficient matching protocol that can be used in many privacy-preserving biometric identification systems in the semi-honest setting. Our most general technical contribution is a new backtracking protocol that uses the by-product of evaluating a garbled circuit to enable efficient oblivious information retrieval. We also present a more efficient protocol for computing the Euclidean distances of vectors, and optimized circuits for finding the closest match between a point held by one party and a set of points held by another. We evaluate our protocols by implementing a practical privacy-preserving fingerprint matching system.

3. Collusion-resisting secure nearest neighbour query over encrypted data in cloud:

3.1AUTHORS Youwen Zhu ; Zhikuan Wang ; Jian Wang

It is a challenging problem to securely resist the collusion of cloud server and query users while implementing nearest neighbour query over encrypted data in cloud. Recently, Cloud BI-II is put forward to support nearest neighbour query on encrypted cloud data, and declared to be secure while cloud server colludes with some untrusted query users. In this paper, we propose an efficient attack method which indicates Cloud BI-II will reveal the difference vectors under the collusion attack. Further, we show that the difference vector disclosure will result in serious privacy breach, and thus attain an efficient attack method to break Cloud BI-II. Namely, Cloud BI-II cannot achieve their declared security. Through theoretical analysis and experiment evaluation, we confirm our proposed attack approach can fast recover the original data from the encrypted data set in Cloud BI-II. Finally, we provide an enhanced scheme which can efficiently resist the collusion attack.

SOFTWARE ENVIRONMENT

Java Technology

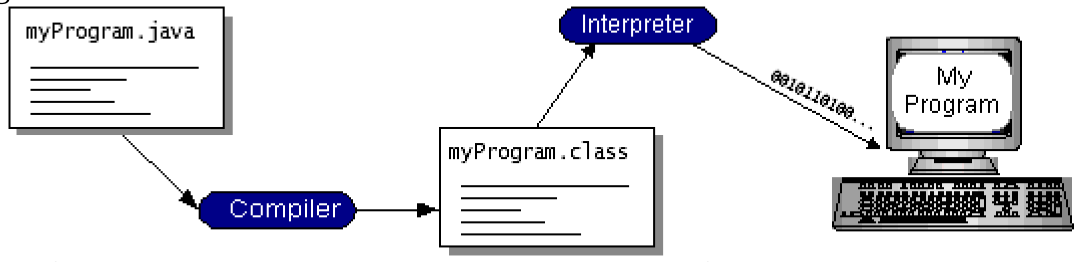
Java technology is both a programming language and a platform.

The Java Programming Language

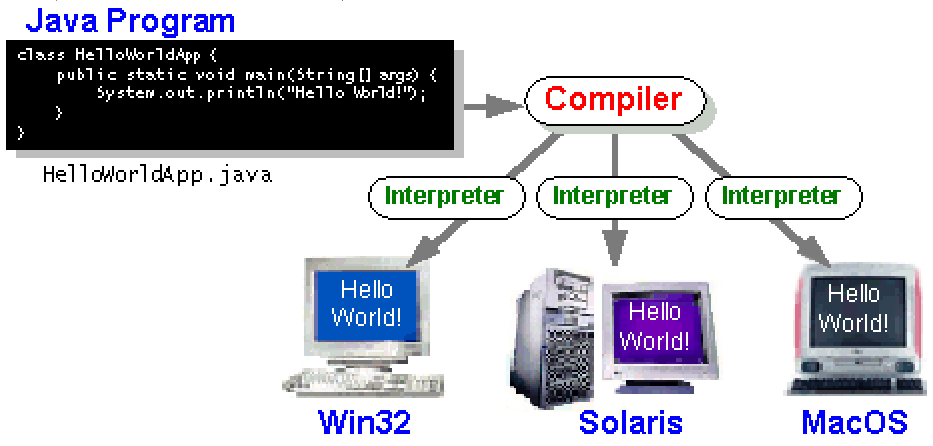
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes 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 that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

HelloWorldApp.java

The Java Platform

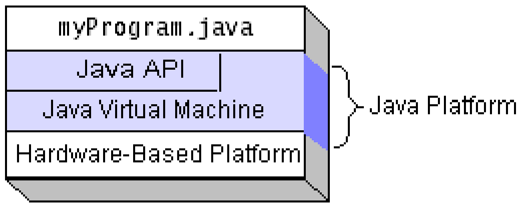
The Java platform has two components:

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

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



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.

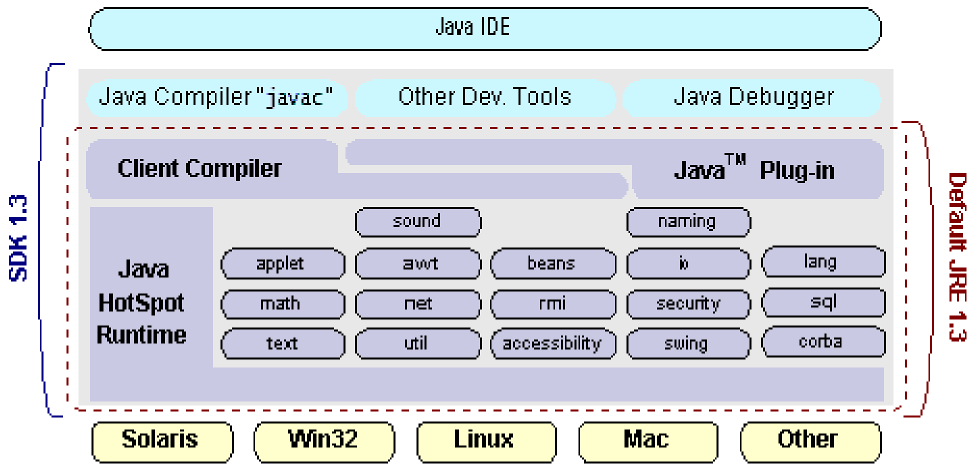
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

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. Servlets are similar to applets in that they are runtime extensions of applications. 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.



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

IMPLEMENTATION

MODULES:

- ❖ Database Owner
- ❖ Data
- ❖ Cloud Server

MODULES DESCRIPTION:

Database Owner:

- ➢ The database owner holds a large size of biometric data (i.e., fingerprints, irises, voice, and facial patterns etc.), which is encrypted and transmitted to the cloud for storage .
- ➢ After receiving the request, the database owner generates a ciphertext for the biometric trait and then transmits the ciphertext to the cloud for identification.
- ➢ Database owner computes the similarity between the query data and the biometric data associated with the index, and returns the query result to the user.

Data User:

- ➢ When a user wants to identify himself/herself, a query request is be sent to the database owner.

Cloud Server:

- ➢ The cloud server figures out the best match for the encrypted query and returns the related index to the database owner.

Code:

Ftpcon.java

```
package com.register;
import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.net.ftp.FTPClient;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author java2
 */
public class Ftpcon {

   FTPClient client = new FTPClient();
   FileInputStream fis = null;
   boolean status;

   public boolean upload(File file,String fname) {
      try {
         client.connect("ftp.drivehq.com");
         client.login("drive05", "drive15");
         client.enterLocalPassiveMode();
         fis = new FileInputStream(file);
         status = client.storeFile(" /kk/" + fname, fis);
         client.logout();
         fis.close();

      } catch (Exception e) {
         System.out.println(e);
      }
      if (status) {
         System.out.println("success");
         return true;
      } else {
         System.out.println("failed");
         return false;
      }
   }
}
```

Regaction.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.register;

import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

@WebServlet("/RegAction")
@MultipartConfig(maxFileSize = 16177215)
public class RegAction extends HttpServlet
 {
    privateStringdbURL=  "jdbc:mysql://localhost:3306/biometric_identification";
    private String dbUser = "root";
    private String dbPass = "root";

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        String id = request.getParameter("id");
        String username = request.getParameter("username");
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String gender = request.getParameter("gender");
        String address = request.getParameter("address");
         HttpSession session = request.getSession();

    InputStream inputStream = null;
        Part filePart = request.getPart("image");
        if (filePart != null) {
            System.out.println(filePart.getName());
            System.out.println(filePart.getSize());
            System.out.println(filePart.getContentType());
            inputStream = filePart.getInputStream();
        }
        Connection conn = null;
        String message = null;
        try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            conn = DriverManager.getConnection(dbURL, dbUser, dbPass);
            String sql = "INSERT INTO user (id,username, email, password, gender, address, image) 
values (?,?, ?, ?, ?, ?, ?)";
            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setString(1, id);
            statement.setString(2, username);
            statement.setString(3, email);
            statement.setString(4, password);
            statement.setString(5, gender);
            statement.setString(6, address);
              session.setAttribute("id", id);
            if (inputStream != null) {
               statement.setBlob(7, inputStream);
            }
            int row = statement.executeUpdate();
            if (row > 0)
            {
         System.out.println("image upload sucess");
               response.sendRedirect("User.jsp?msg= Registration Completed");
            } else {
               response.sendRedirect("User.jsp?msg=Failed");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Userregistration.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.register;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

@WebServlet("/UserRegAction")
@MultipartConfig(maxFileSize = 16177215)
public class UserRegAction extends HttpServlet {

   private String dbURL = "jdbc:mysql://localhost:3306/biometric_identification";
   private String dbUser = "root";
   private String dbPass = "root";

   protected void doPost(HttpServletRequest request,
         HttpServletResponse response) throws ServletException, IOException {
      String id = request.getParameter("id");
      String username = request.getParameter("username");
```

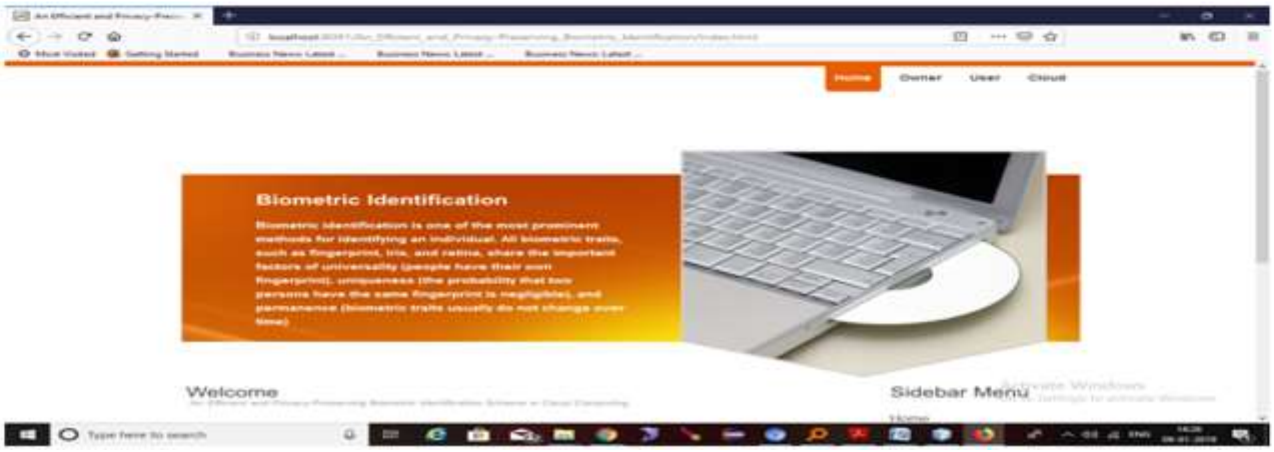
```
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String gender = request.getParameter("gender");
        String address = request.getParameter("address");
         HttpSession session = request.getSession();

         InputStream inputStream = null;
        Part filePart = request.getPart("image");
        String fname=filePart.getSubmittedFileName();
        if (filePart != null) {
            System.out.println(filePart.getName());
            System.out.println(filePart.getSize());
            System.out.println(filePart.getContentType());
            inputStream = filePart.getInputStream();
        }
        Connection conn = null;
        String message = null;
        try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            conn = DriverManager.getConnection(dbURL, dbUser, dbPass);
            String sql = "INSERT INTO userreg (id,username, email, password, gender, address,
image,imgname) values (?,?, ?, ?, ?, ?, ?,?)";
            PreparedStatement statement = conn.prepareStatement(sql);
             statement.setString(1,id);
            statement.setString(2, username);
            statement.setString(3, email);
            statement.setString(4, password);
            statement.setString(5, gender);
            statement.setString(6, address);
              session.setAttribute("id", id);
            if (inputStream != null) {
                statement.setBlob(7, inputStream);
            }
            statement.setString(8, fname);
            int row = statement.executeUpdate();
            if (row > 0)
            {
         System.out.println("image upload sucess");
                response.sendRedirect("userregsucess.jsp?reg=success");

            } else {
                response.sendRedirect("userreg.jsp?regg=Failed");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Output:

Home Page


Home
Owner
User
Cloud

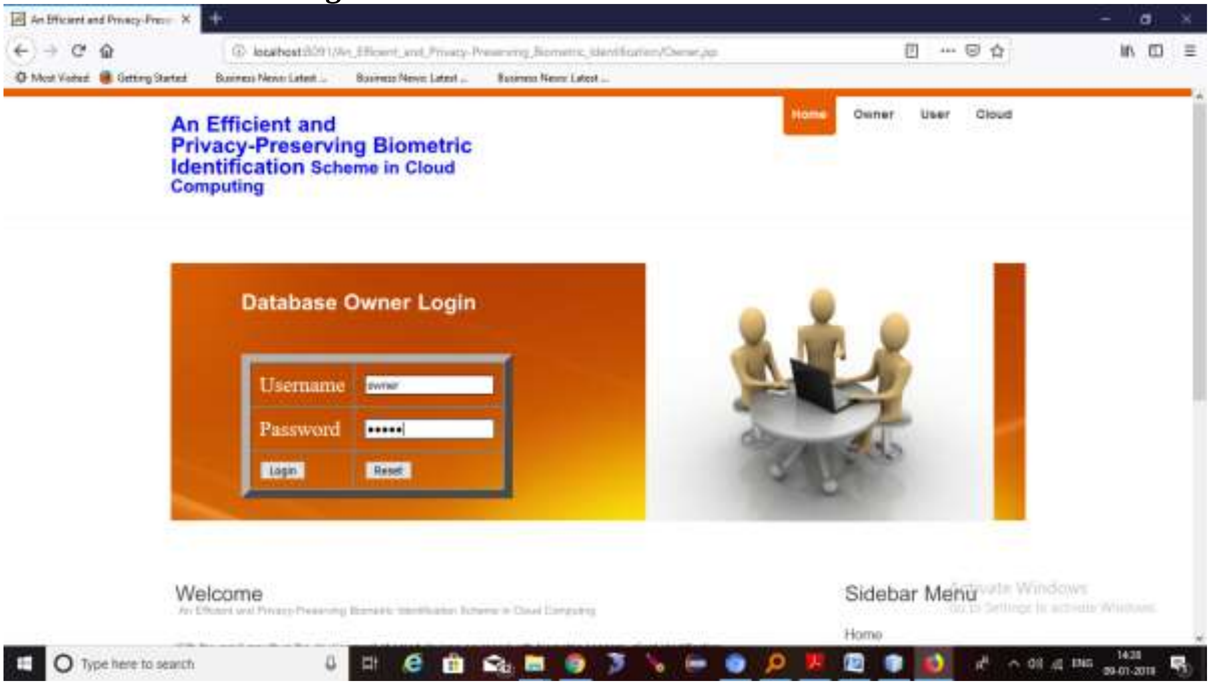
Biometric Identification

Biometric identification is one of the most prominent methods for identifying an individual. All biometric traits, such as fingerprint, iris, and retina, share the important factors of universality (people have their own fingerprint), uniqueness (the probability that two persons have the same fingerprint is negligible), and permanence (biometric traits usually do not change over time)

Welcome
An Efficient and Privacy-Preserving Biometric Identification Scheme in Cloud Computing

Sidebar Menu

Database Owner Login


Home
Owner
User
Cloud

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

Database Owner Login

Username	owner
Password	•••••
Login	Reset

Welcome
An Efficient and Privacy-Preserving Biometric Identification Scheme in Cloud Computing

Sidebar Menu
Home

Conclusion

In this paper, we proposed a novel privacy-preserving biometric identification scheme in the cloud computing. To realize the efficiency and secure requirements, we have designed a new encryption algorithm and cloud authentication certification. The detailed analysis shows it can resist the potential attacks. Besides, through performance evaluations, we further demonstrated the proposed scheme meets the efficiency need well.

References

1. Jain, A. K., Nandakumar, K., & Nagar, A. (2008). Biometric template security. EURASIP Journal on Advances in Signal Processing, 2008(1), 1-17.
2. Rathgeb, C., & Busch, C. (2011). A survey on biometric cryptosystems and cancelable biometrics. EURASIP Journal on Information Security, 2011(1), 1-17.
3. Ratha, N. K., Connell, J. H., & Bolle, R. M. (2001). Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal, 40(3), 614-634.
4. Zhang, D., & Jain, A. K. (2004). Authentication of multispectral palmprint images. Pattern Recognition, 37(6), 1293-1304.
5. Mistry, P., & Shih, T. K. (2011). A Survey of Biometric Recognition Methods. International Journal of Computer Applications, 17(7), 18-25.

6. Rane, S., Boult, T. E., & Gao, G. (2003). Personal authentication using palmprint and hand geometry biometric. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2003, 321-324.
7. Wayman, J. L., Maltoni, D., Maio, D., & Jain, A. K. (2005). Biometric Systems: Technology, Design, and Performance Evaluation. Springer.
8. Kaur, G., & Rani, M. (2016). A Survey of Cloud Computing Security Management. Procedia Computer Science, 85, 1135-1142.
9. Mather, T., Kumaraswamy, S., & Latif, S. (2009). Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media.
10. Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009). Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), 199-212.