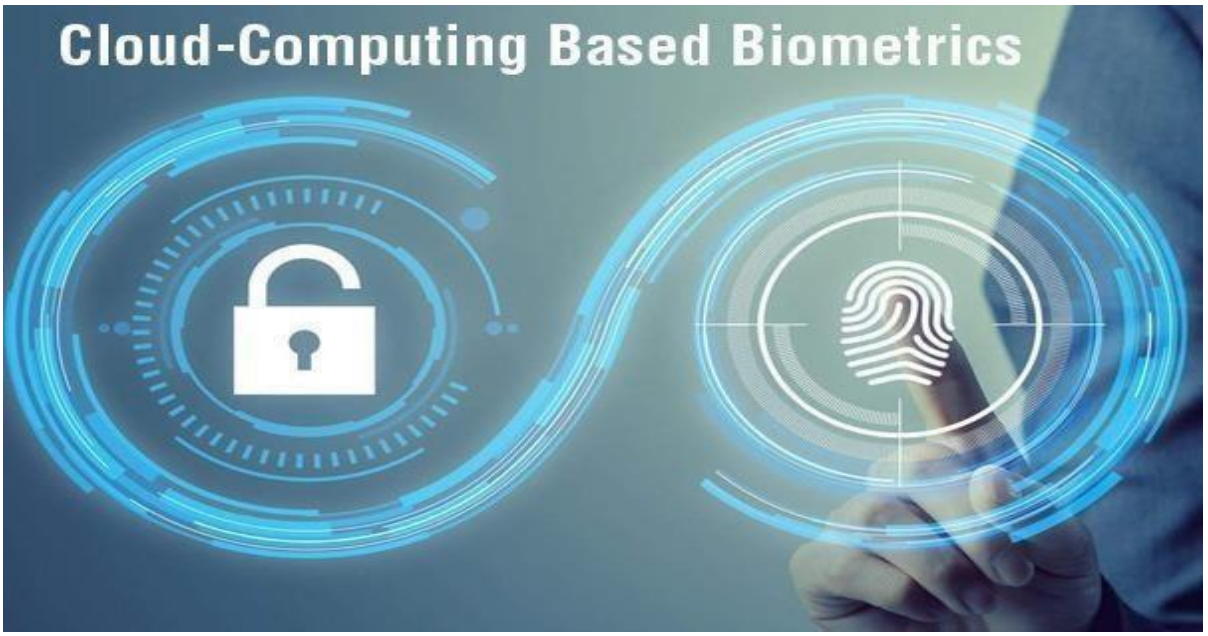
SECURE BIOMETRIC BASED AUTHENTICATION ON CLOUD SERVICES

PROJECT FINAL REPORT



CSE 5325-005 - SOFTWARE ENGINEERING

TEAM 23:

PRATHIMA JAGINI – 1002088570

MEGHANA KATRAJU – 1002088571

DATE: .9-11-2023.

TABLE OF CONTENTS

1. Abstract

The demand for remote data storage and computation services is increasing exponentially in our data-driven society; thus, the need for secure access to such data and services. In this paper, we design a new biometric-based authentication protocol to provide secure access to a remote (cloud) server. In the proposed approach, we consider biometric data of a user as a secret credential. We then derive a unique identity from the user's biometric data, which is further used to generate the user's private key. In addition, we propose an efficient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission. In other words, there is no need to store the user's private key anywhere and the session key is generated without sharing any prior information. A detailed Real-Or- Random (ROR) model based formal security analysis, informal (non-mathematical) security analysis and formal security verification using the broadly-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool reveal that the proposed approach can resist several known attacks against (passive/active) adversary. Finally, extensive experiments and a comparative study demonstrate the efficiency and utility of the proposed approach.

2. Introduction

2.2.1 : Overview of the Project:

Cloud services have become ubiquitous in our society, but ensuring secure access to these services poses significant challenges in terms of authentication, authorization, and accounting. Existing authentication mechanisms, such as those based on Kerberos, OAuth, and OpenID, often rely on the assumption that the remote authentication server is a trusted entity. However, a key limitation lies in storing user credentials on the authentication server, leading to potential security vulnerabilities.

This project addresses the need for a secure and efficient authentication protocol by proposing an alternative to conventional password-based authentication. The objective is to design a mechanism that eliminates the storage of user credentials on the authentication server, enhances security, and minimizes the overhead associated with existing authentication protocols.

2.2.2 : Problem Statement or Need:

The existing authentication mechanisms suffer from weaknesses, as highlighted in various published protocols. Issues include the storage of user credentials on authentication servers, the use of symmetric key cryptography leading to authentication protocol overhead, and vulnerabilities revealed in several established protocols. The need for a secure and efficient authentication protocol becomes evident in the face of these challenges.

This project aims to overcome these limitations by introducing a novel authentication approach. The proposed solution involves using a fingerprint image of the user as a secret credential, generating a private key from the fingerprint image, and securely enrolling the user's credential in the authentication server's database. By doing so, the project addresses the problem of storing user credentials on the authentication server and introduces a more secure and efficient authentication

protocol. The use of biometric data for authentication, the generation of revocable private keys, and the introduction of a novel session key generation mechanism contribute to mitigating the shortcomings of traditional authentication protocols.

3. Project Scope

3.3.1 : Features and Functionalities

The project aims to design a secure and efficient authentication protocol based on biometrics, specifically using a fingerprint image. Key features and functionalities include:

1. **Biometric-Based Authentication:** Utilizing a user's fingerprint image as a secret credential for authentication.
2. **Private Key Generation:** Generating a private key from the fingerprint image for secure enrolment in the authentication server's database.
3. **Secure Communication:** Establishing secure communication between users and servers without storing user credentials on the authentication server.
4. **Session Key Generation:** Proposing a fast and robust approach to generate session keys using two fingerprint data for secure message transmission.
5. **Message Authentication Mechanism:** Introducing a biometric-based message authenticator as an alternative to traditional message authentication protocols.

3.3.2 : Constraints:

1. **No Storage of Private Key:** The project mitigates the limitation of storing user credentials on the authentication server, ensuring no need to store private keys or direct forms of user biometric data.
2. **Avoiding Preloaded Information:** Introducing a mechanism to avoid the need for secret pre-loaded information during the authentication process.
3. **Efficient Data Transmission:** Presenting an effective way to transmit user biometric data securely through unsecured network channels.
4. **Revocable Private Key:** Proposing an approach to generate a revocable private key directly from an irrevocable fingerprint image.

Target Audience:

- End-users seeking secure access to cloud-based services.
- Administrators managing authentication servers.
- Stakeholders concerned with the security and efficiency of authentication protocols in the context of cloud-based services.

Limitations:

1. **Biometric Data Dependency:** The effectiveness of the proposed approach relies on the accuracy and reliability of biometric data, particularly fingerprint images.
2. **Simulation-Based Demonstration**: The practical demonstration of the proposed approach is done using NS2 simulation, indicating that real-world implementation may require further considerations.
3. **Application to IIoT Environment:** While the focus is on cloud-based Industrial Internet of Things (IIoT) deployment, the applicability may vary depending on specific IIoT use cases and scenarios.

4. Requirements Analysis:

4.4.1. Requirements Gathering Process:

- **Literature Review:** The initial phase involved an in-depth literature survey to understand the existing solutions, challenges, and advancements in biometrics-based user authentication, specifically in the context of cloud-based Industrial Internet of Things (IIoT) deployments.
- **Surveys and Interviews:** Collaborated with experts, including A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues, through interviews and surveys to gather insights into the limitations of current solutions and the specific requirements of a biometric-based privacy-preserving user authentication scheme.
- **Documentation Review**: Analysed papers such as "Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based IIoT Deployment" to extract requirements, focusing on strong authentication, privacy preservation, and low computation and communication overheads.

4.4.2. List of Functional Requirements:

- **Biometric-Based Authentication:**
  - Users should be able to enroll in the system using a biometric, such as a fingerprint.
  - The system must establish a preestablished key agreement between smart devices and the gateway node for strong authentication.
- **Secure Communication:**
  - The protocol should ensure secure communication between users and smart devices without compromising privacy.
  - Users must be able to securely access cloud-based IIoT services using the proposed biometric-based privacy-preserving user authentication (BP2UA) scheme.
- **Session Key Generation:**
  - The system should generate session keys efficiently based on biometric data for secure communication.

4.4.3. List of Non-Functional Requirements:

- **Computation and Communication Costs:** BP2UA should demonstrate lower computation and communication costs compared to existing schemes in the IIoT environment.
- **Security:** The system must undergo formal security analysis using the real-or-random model, and informal analysis to ensure robustness against various known attacks.
- **Efficiency**: BP2UA should be effective in the IIoT environment, considering the limitations of existing solutions.
- **Usability:** The user interface should be intuitive, making it easy for users to enroll and authenticate using their biometrics.

4.4.4. Use Cases/User Stories:

- **User Enrollment:** As a user, I want to enroll in the system securely using my biometric (fingerprint) for subsequent authentication.
- **Accessing IIoT Services:** As a user, I want to securely access cloud-based IIoT services using the proposed BP2UA scheme.

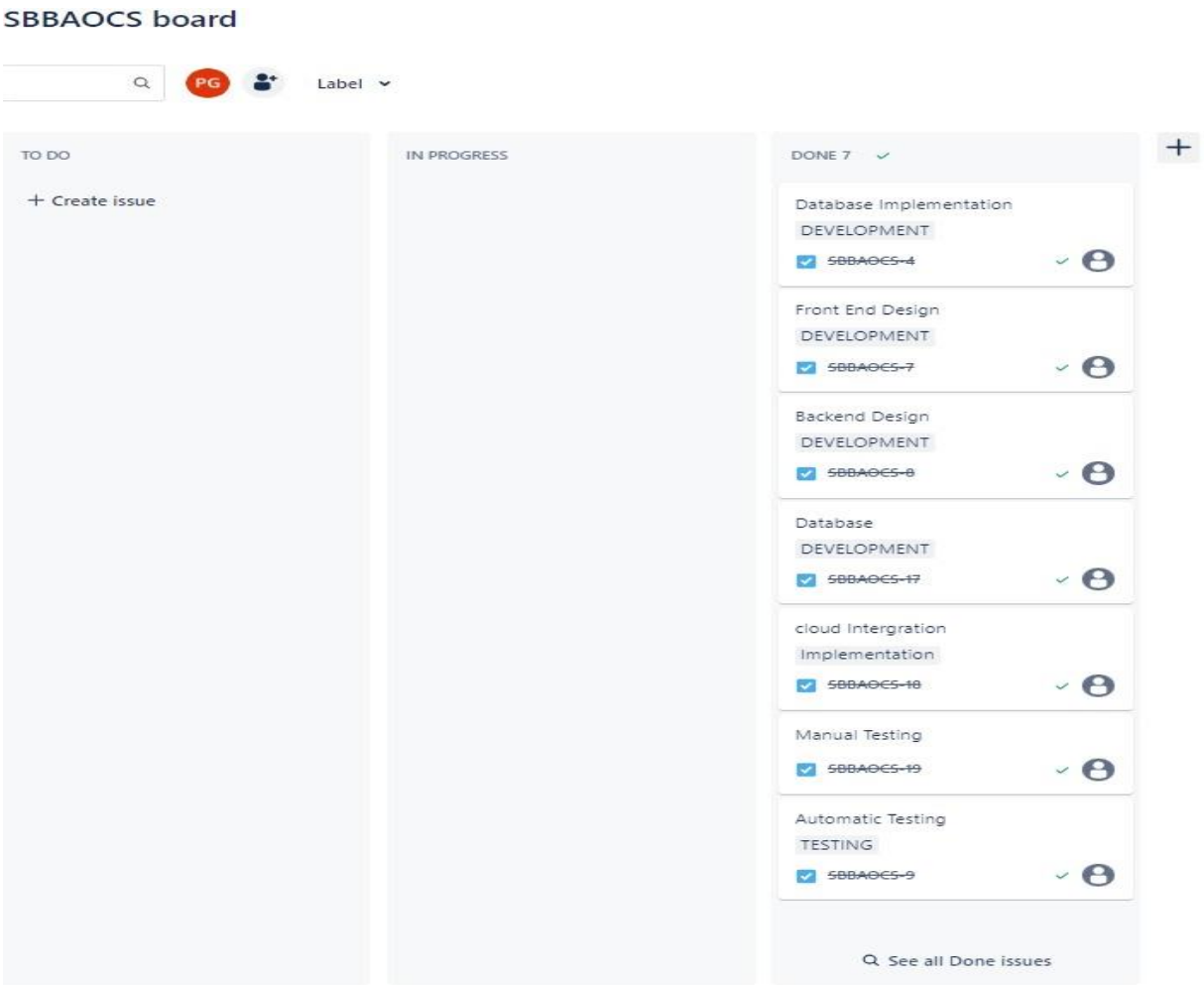
- **Secure Communication:** As a user, I want the system to establish a secure communication channel between my device and the gateway node.

4.4.5. Changes or Updates to Requirements:

- **Security Enhancements:** Based on the literature review and security analyses, updates were made to enhance the security aspects of the system, ensuring it can withstand various types of known attacks.
- **Efficiency Improvements:** The requirements were refined to ensure that the proposed system is more efficient in terms of computation and communication costs compared to existing schemes.
- **Usability Refinements:** Continuous feedback from experts and potential users led to refinements in the usability aspects, ensuring a seamless and user-friendly experience.

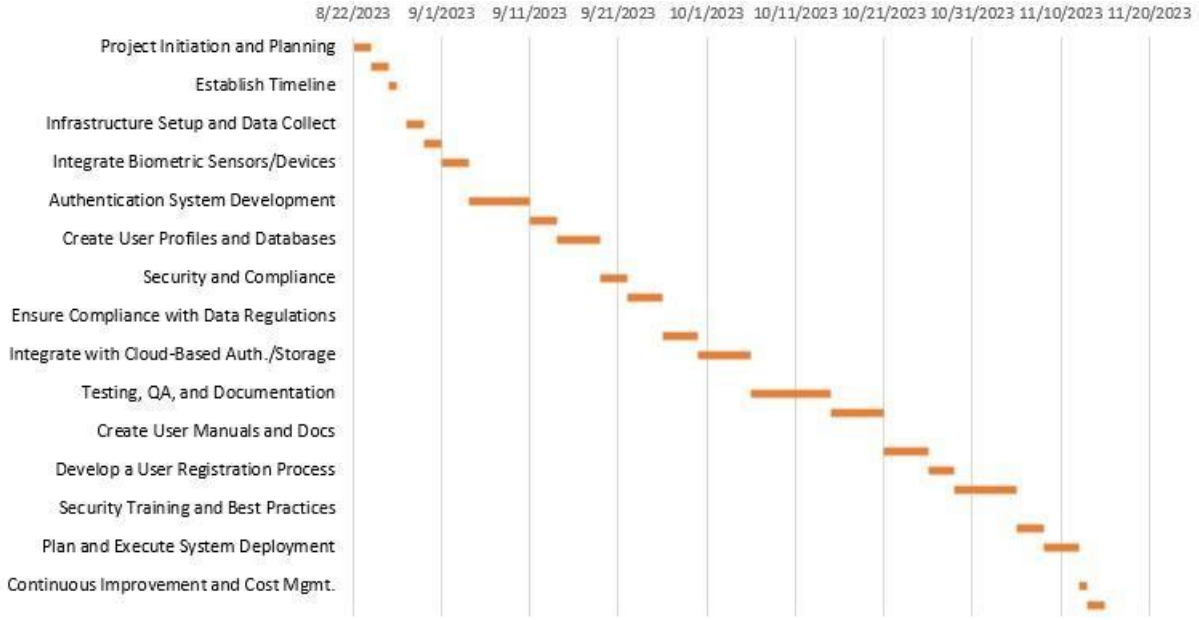
5. Project Management:

Management Tools:
We utilized Jira as our project management tool to track and monitor the project's progress efficiently. Below is an overview of the project's status and milestones achieved.



6. Project Planning:

Updated gantt chart



BURN DOWN CHART:

The reasons we were held back:

1. Initial Software Choice:

- **Problem:** Planned to use MFS biometric software for fingerprint capture due to budget constraints.
- **Challenge:** MFS software did not perform as expected, failing to accurately capture fingerprint images.

2. Impact on Project:
- **Problem:** Inability to capture and match fingerprints compromised the core functionality of the project.
- **Result:** Fell behind schedule as the chosen software did not meet the project requirements.

3. Revaluation and Reimplementation:
- **Solution:** Responded to the software limitations by reassessing requirements and exploring alternative solutions.
- **Action:** Reimplemented the project with a different approach, considering both budget constraints and the need for reliable fingerprint capture.

4. Resource Constraints:
- **Problem:** Being a team of two, the injury of a team member further limited the available human resources.
- **Impact:** The shortage of manpower hindered the pace of project reimplementation, contributing to a delay in the overall schedule.

5. Lesson Learned:
- **Insight:** The importance of thorough evaluation and testing in the initial stages to avoid setbacks during implementation.
- **Adaptation:** Incorporated lessons learned into future decision-making processes to enhance project planning and execution.

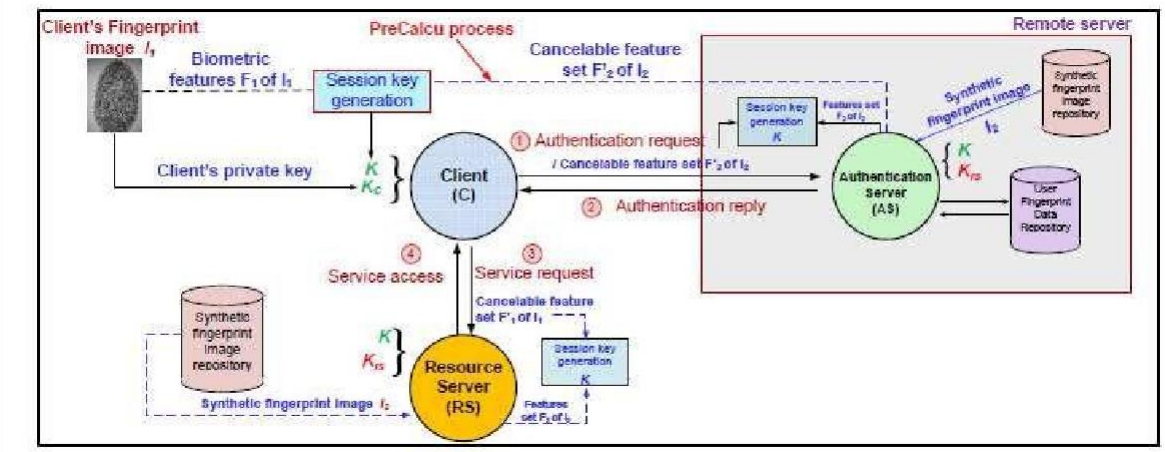
7. Design and Architecture

7.7.1 System Design:

System design is transition from a user-oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analysing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly.

The menu should be precise and compact.



In designing the software following principles are followed:

1. **Modularity and partitioning:** software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.
2. **Coupling:** modules should have little dependence on other modules of a system.
3. **Cohesion:** modules should carry out in a single processing function.
4. **Shared use:** avoid duplication by allowing a single module be called by other that need the function it provides

MODULE DESIGN:

1. **CLIENT**
2. **AUTHENTICATION SERVER**
3. **ADMIN**
4. **RESOURCE SERVER**

1. CLIENT

Client must register into application with basic details and he can able to login with username password and with fingerprint. Client can able sent request to the resource server. After sending the request he can get the response from the resource server. After getting the response from the server he can be able view the file in the cloud. He can able to see all permission of files.

2. AUTHENTICATION SERVER.

Authentication Server need to login with username and password. After login he can able to view client details and authorize. Authentication server can able to view synthetic finger print images. Server can able to user client images.

3. ADMIN

Admin need to login with basic username and password. After login he can able to upload files those are useful to the user. He can able to view all uploaded files. Admin can able to add synthetic fingerprint images. Admin can able to view the data in the repository.

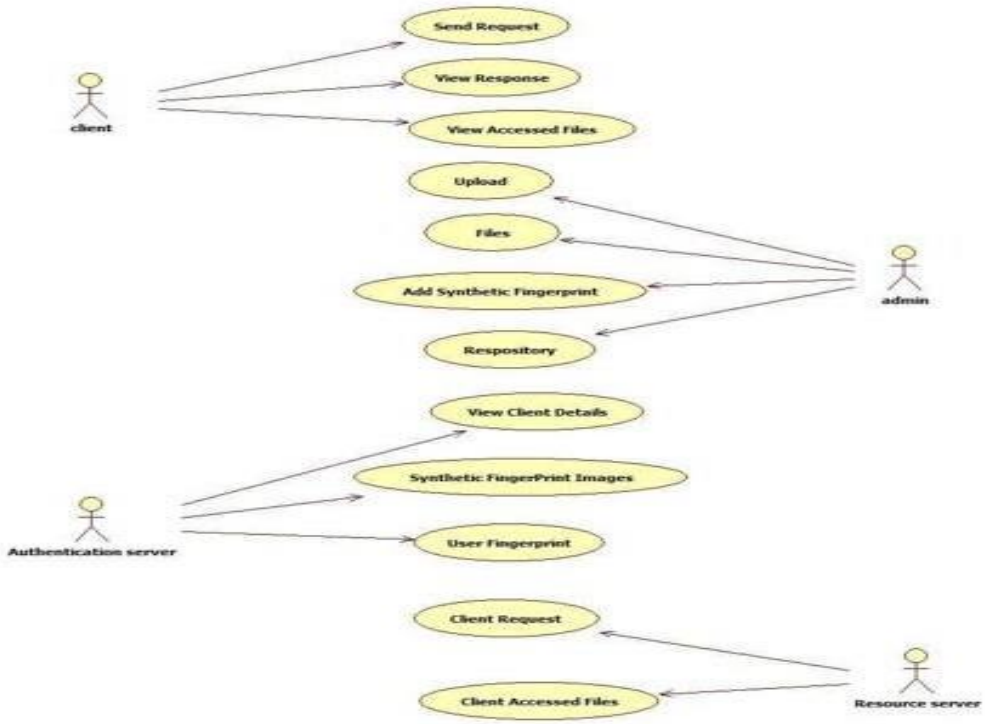
4.RESOURCE SERVER

Resource server need to login into the application using username and password. After login resource server he can able to view all client requests as well as he can able view all users access rights of files.

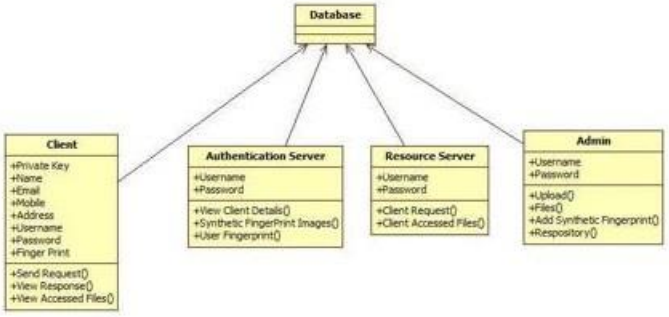
7.7.2 UML DIAGRAMS:

Use Case Diagram: A use case diagram in the Unified Modelling 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

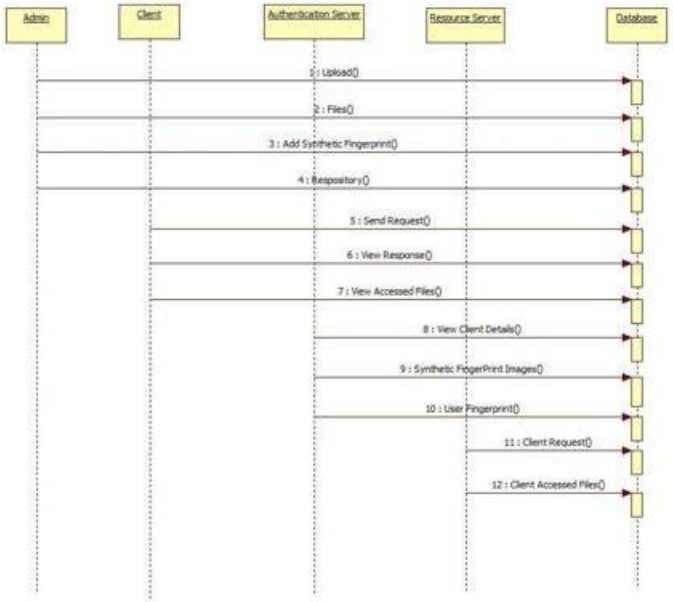
(represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



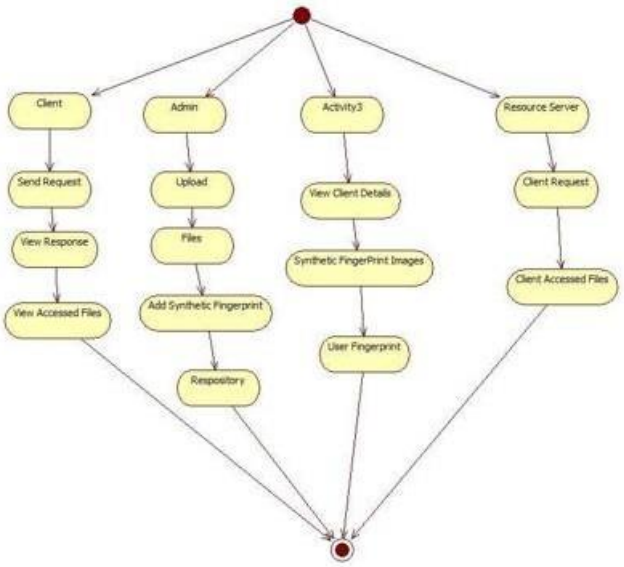
Class Diagram: In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.



Sequence Diagram: 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.



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.



Client Table:
Attributes:

- **ClientID (Primary Key):** Unique identifier for each client.
- **Username:** User's unique login username.
- **Password**: Encrypted password for user authentication.
- **Fingerprint Data:** Biometric data (fingerprint) for user authentication.

1. **Authentication Server Table:**
   Attributes:

- **AuthServerID (Primary Key)**: Unique identifier for the authentication server.
- **Username**: Username for authentication server login.
- **Password**: Encrypted password for authentication server login.
- **Synthetic Fingerprint**: Storage for synthetic fingerprint images.

2. **Admin Table:**
   Attributes:

- **Admin ID (Primary Key):** Unique identifier for the admin.
- **Username:** Admin's login username.
- **Password:** Encrypted password for admin login.
- **Uploaded Files:** List of files uploaded by the admin.
- **SyntheticFingerprintAdditions**: Records of synthetic fingerprint additions.

3. **Resource Server Table:**
   Attributes:

- **ResourceServerID (Primary Key):** Unique identifier for the resource server.
- **Username:** Username for resource server login.
- **Password:** Encrypted password for resource server login.
- **Client Requests:** Records of client requests for file access.

- **FilePermissions:** Information about file access permissions.
- **UserAccessRights:** Details about user access rights.

Relationships:

1. **Client-Authentication Server Relationship:**
   Description:
- A one-to-one relationship between the Client and Authentication Server tables based on the user's credentials.
   Purpose:
- Ensures a secure and direct link for user authentication, enhancing system security.
2. **Client-Resource Server Relationship:**
   Description:
- Manages user requests and file access permissions.
   Purpose:
- Facilitates secure communication between clients and the resource server, controlling file access.
3. **Admin-Resource Server Relationship:**
   Description:
- Enables file uploads and repository data management.
   Purpose:
- Allows the admin to upload files and manage the repository, maintaining control over system resources.

8. Implementation:

1. **Development Environment and Tools Used:**
- **Programming Language:** Java
- **Web Development Framework:** Java Server Faces (JSF)
- **Database Management System:** MySQL
- **Integrated Development Environment (IDE):** NetBeans 8.1
- **Web Server:** Apache Tomcat 7.0
- **Web Technologies:** HTML, CSS, JavaScript
- **Version Control:** Git

2. **Major Components or Modules:**

   Client Module:
- Responsible for user registration, login, and sending requests to the resource server.
- Manages the biometric data (fingerprint) for user authentication.
   Authentication Server Module:
- Handles user authentication and authorization.
- Stores synthetic fingerprint images for comparison during the authentication process.
   Admin Module:
- Enables file uploads, management of uploaded files, and addition of synthetic fingerprint images.
- Provides administrative control over system resources.
   Resource Server Module:
- Manages client requests for file access.
- Controls file access permissions based on user authentication and admin settings.

- Handles user access rights and permissions.

3. **Challenges Faced During Implementation:**

   Biometric Data Security:
- Ensuring the secure storage and transmission of biometric data posed challenges to prevent unauthorized access and potential breaches.

   Integration of WebAuthn API:
- Implementing the WebAuthn API for secure authentication required careful integration with the existing authentication system.

   User-Friendly Interface:
- Designing an interface that is user-friendly and efficient while ensuring security measures proved to be a balancing act.

4. **Third-Party Libraries or Frameworks Used:**
   WebAuthn API:
- Integrated the WebAuthn API for secure and convenient authentication.
- Utilized the API to enable password less and secure biometric authentication.

   JavaServer Faces (JSF) Libraries:
- Leveraged JSF libraries for the development of a dynamic and interactive web application.

   MySQL Connector/J:
- JDBC driver for connecting Java applications to MySQL databases

5. **WebAuthn API:**
   Description:
- WebAuthn API is a web standard for password less authentication. It allows the use of various authenticators, including biometrics, to securely authenticate users without relying on passwords.

   Integration:
- Integrated the WebAuthn API to enhance the security of user authentication.
- Enabled users to register and authenticate using biometric data through the API.
- Implemented a robust authentication flow that combines traditional and password less methods for increased security.

9. TESTING:

Unit Testing: Unit testing involves evaluating small, isolated parts (units) of code to verify their correctness, typically automated for efficiency. It is a fundamental practice in Extreme Programming (XP) for continual testing and improvement. Unit tests are written to validate that specific methods within classes produce expected outcomes for defined inputs, primarily from a programmer's standpoint.

Integration Testing: Integration testing combines and tests program units together to uncover interface issues before real-world execution. It focuses on testing the interaction between components, helping identify problems early. This practice is integral to Extreme Programming (XP) for rigorous software development through continuous testing and refinement.

System Testing: Ensures seamless integration of biometric data and cloud services, rigorous security measures, optimized performance, and robust error handling mechanisms.

User Acceptance Testing (UAT): Validates user enrollment processes, authentication methods, error scenarios, and user interface intuitiveness. Focuses on collecting valuable user feedback for system improvement.

Test case1:

Test case for Login form: FUNCTION:	**LOGIN**
EXPECTED RESULTS:	Should Validate the user and check his existence in database
ACTUAL RESULTS:	Validate the user and checking the user against the database
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

Test case2:

Test case for User Registration form: FUNCTION:	**USER REGISTRATION**
EXPECTED RESULTS:	Should check if all the fields are filled by the user and saving the user to database.
ACTUAL RESULTS:	Checking whether all the fields are field by user or not through validations and saving user.
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

Test case3:
Test case for Change Password:

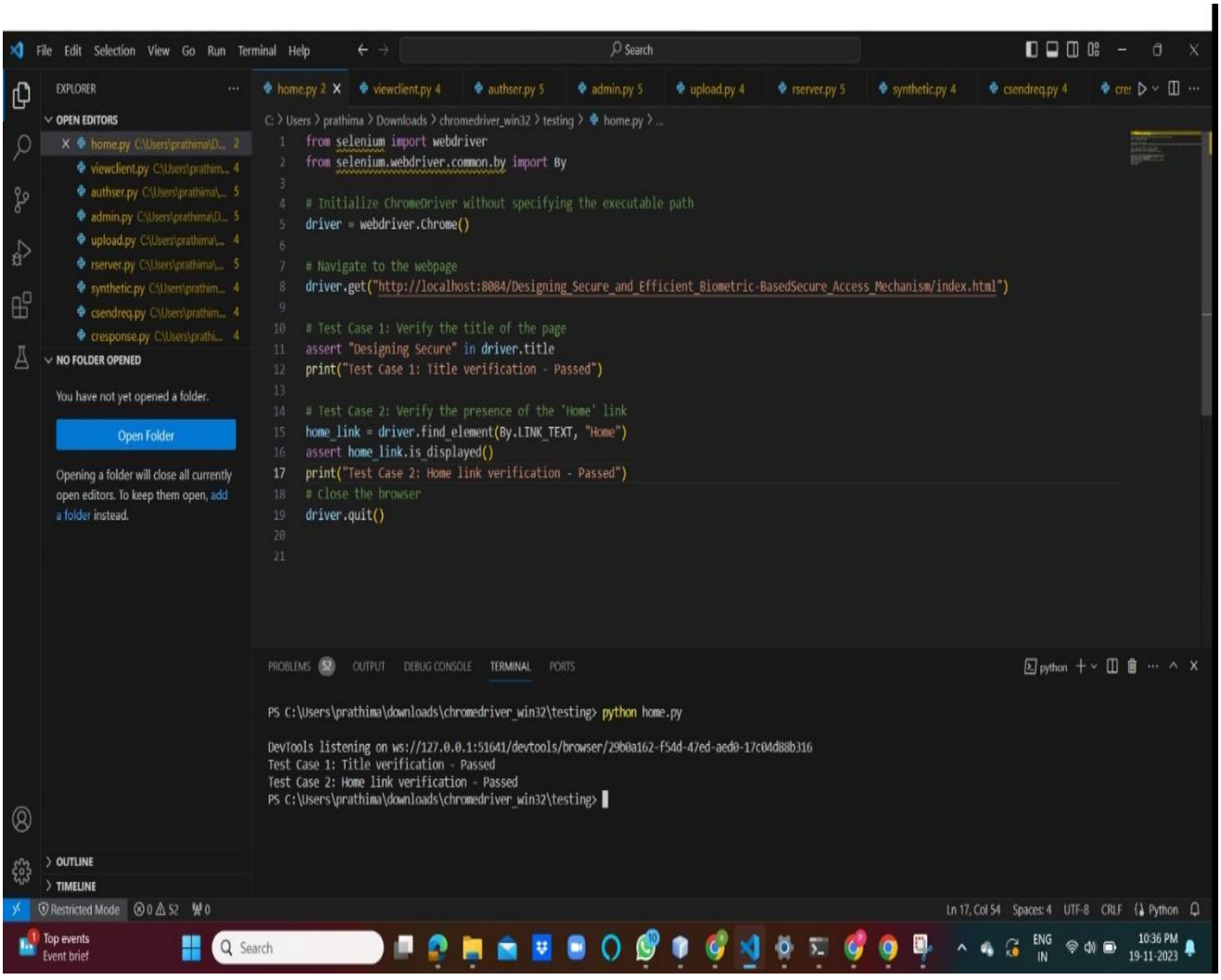
When the old password does not match with the new password , then this results in displaying an error message as " OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD". **FUNCTION:**	**Change Password**
EXPECTED RESULTS:	Should check if old password and new password fields are filled by the user and saving the user to database.
ACTUAL RESULTS:	Checking whether all the fields are field by user or not through validations and saving user.
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

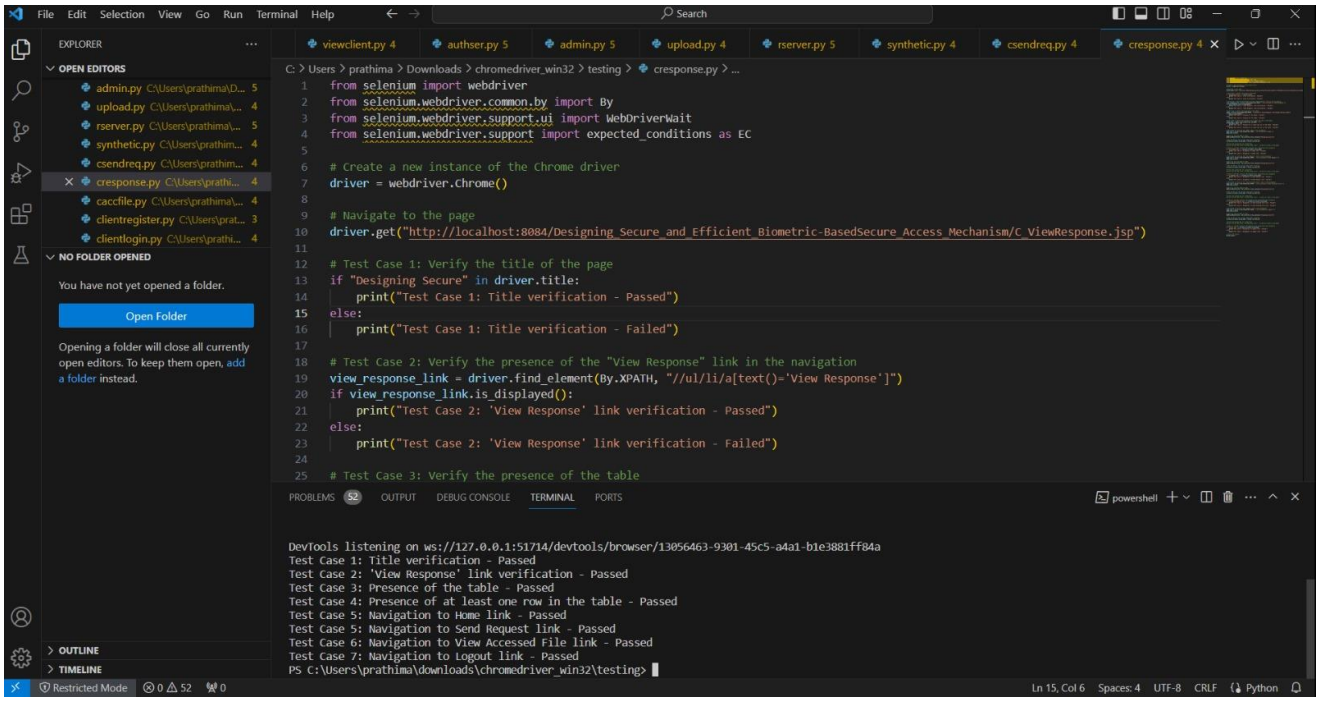
Test case 4:
Test case for Forget Password:
When a user forgets his password, he is asked to enter Login name, ZIP code, Mobile number. If these are matched with the already stored ones then user will get his original password.

Module	**Functionality**	**Test Case**	**Expected Results**	**Actual Results**	**Result**	**Priority**
User	Login Usecase	1. Navigate To Www.Sample.Com 2. 2.Click On Submit Button Without Entering Username and Password	A Validation Should Be As Below “Please Enter Valid Username & Password”	A Validation Has Been Populated As Expected	Pass	High
		1. aNavigate To Www.Sample.Com 1. 2. Click On Submit Button With Out Filling Password And With Valid Username	A Validation Should Be As Below “Please Enter Valid Password Or Password Field Can Not Be Empty “	A Validation Is Shown As Expected	Pass	High
		1. NNavigate To Www.Sample.Com 2. Enter Both Username And Password Wrong And Hit Enter	A Validation Shown As Below “The Username Entered Is Wrong”	A Validation Is Shown As Expected	Pass	High

		1. Navigate To Www.Sample.Com 2. Enter Validate Username And Password And Click On Submit	Validate Username And Password In DataBase And Once If They Correct Then Show The Main Page	Main Page/ Home Page Has Been Displayed	Pass	High

AUTOMATED TESTING USING SELENIUM:





Issue and Bug Management:

1. Tracking:
   - Use an issue tracking system (e.g., Jira, GitHub Issues) for bug tracking.
2. Categorization:
   - Categorize issues by severity, priority, and type.

Bug Resolution:

1. Reporting:
   - Prompt reporting of bugs with detailed descriptions.
2. Triage:
   - Regularly assess and prioritize incoming issues.
3. Ownership:
   - Assign each issue to a responsible team member.
4. Communication:
   - Provide regular updates on issue resolution progress.
5. Resolution Steps:
   - Debug, fix, review, test, and document bug resolutions.

Best Practices:

1. Proactive Measures:
   - Implement monitoring and regular code audits.
2. User Feedback:
   - Actively seek and consider user feedback for issue identification.

By following these practices, issues and bugs can be efficiently managed and resolved in the project.

10. Maintenance:

Ongoing Maintenance Plan:

- **Regular Monitoring:** Implement a system for continuous monitoring of the application's performance, security, and user interactions.
- **Routine Backups:** Conduct regular data backups to prevent data loss in case of unexpected events.
- **Periodic Security Audits:** Perform security audits to identify and address potential vulnerabilities.
- **User Feedback Mechanism:** Establish a mechanism for users to provide feedback, report issues, and suggest improvements.

Updates, Patches, and Bug Fixes:

- **Version Control:** Utilize version control systems, such as Git, to manage the source code and track changes systematically.
- **Automated Testing:** Implement automated testing procedures to detect and address bugs and issues during the development process.
- **Scheduled Updates:** Plan regular software updates to introduce new features, enhancements, and security patches.
- **Bug Tracking System:** Employ a bug tracking system to log, prioritize, and address reported issues efficiently.

Scalability and Future-Proofing:

- **Modular Architecture:** Design the system with a modular architecture to facilitate future enhancements and scalability.
- **Cloud-Based Infrastructure:** Consider migrating to a cloud-based infrastructure to accommodate growing user demands seamlessly.
- **API Standardization:** If applicable, standardize APIs to ensure compatibility with future technologies and ease of integration.
- **Regular Technology Assessments:** Conduct periodic assessments of emerging technologies to evaluate potential upgrades and integrations.
- **User Base Scalability:** Plan for the growth of the user base by optimizing database structures, implementing caching mechanisms, and employing load balancing strategies.

11. Conclusion

Biometric has its unique advantages over conventional password and token-based security system, as evidenced by its increased adoption (e.g., on Android and iOS devices). In this paper, we introduced a biometric-based mechanism to authenticate a user seeking to access services and computational resources from a remote location. Our proposed approach allows one to generate a private key from a fingerprint biometric reveal, as it is possible to generate the same key from a fingerprint of a user with 95.12% accuracy. Our proposed session key generation approach using two biometric data does not require any prior information to be shared. A comparison of our approach with other similar authentication protocols reveals that our protocol is more resilient to several known attacks.

Lessons Learnt:

Key lessons include the importance of rigorous software testing, adaptability in responding to unforeseen challenges, and the need for comprehensive contingency planning. Despite setbacks, the team demonstrated resilience and learned valuable insights for future projects.

12. Reccomendations:

1. Facial Recognition Integration:

- Explore incorporating facial recognition technology as an alternative or complementary biometric method. This could enhance user convenience and add an extra layer of security.

2. Multi-Modal Biometrics:

- Consider implementing a multi-modal biometric system that combines various identification methods, such as fingerprint and iris scans. This approach can improve accuracy and reduce the risk of false positives or negatives.

3. Cloud-Based Authentication:

- Shift towards a cloud-based authentication system to enhance accessibility and scalability. Storing biometric data securely on the cloud allows for seamless access across different devices and locations.

4. Behavioural Biometrics:

- Investigate the integration of behavioural biometrics, such as keystroke dynamics or gait recognition, to supplement traditional biometric methods. This can provide a more comprehensive and personalized authentication approach.

5. Continuous Monitoring and Adaptive Authentication:

- Explore implementing continuous monitoring and adaptive authentication mechanisms. This involves continuously assessing user behaviour and adjusting authentication requirements based on patterns, adding an extra layer of security against evolving threats.

13. References

[1] C. Neuman, S. Hartman, K. Raeburn, “The kerberos network authentication service (v5),” RFC 4120, 2005.

[2] “OAuth Protocol.” [Online]. Available: http://www.oauth.net/

[3] “OpenID Protocol.” [Online]. Available: http://openid.net/

[4] G. Wettstein, J. Grosen, and E. Rodriguez, “IDFusion: An open architecture for Kerberos based authorization,” Proc. AFS and Kerberos Best Practices Workshop, June 2006.

[5] A. Kehne, J. Schonwalder, and H. Langendorfer, “A nonce-based protocol for multiple authentications,” ACM SIGOPS Operating System Review, vol. 26, no. 4, pp. 84–89, 1992.

[6] B. Neuman and S. Stubblebine, “A note on the use of timestamps as nonces,” Oper. Syst. Rev., vol. 27, no. 2, pp. 10–14, 1993.

[7] J. Astorga, E. Jacob, M. Huarte, and M. Higuero, “Ladon : endto-end authorisation support for resource-deprived environments,” IET Infomration Security, vol. 6, no. 2, pp. 93–101, 2012.

[8] S. Zhu, S. Setia, and S. Jajodia, “LEAP: efficient security mechanisms for large-scale distributed sensor networks,” Washington D.C., USA, October 2003, pp. 62–72.

[9] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler, “SPINS: security protocols for sensor networks,” ACM Wireless Networking, vol. 8, no. 5, pp. 521– 534, 2002.

[10] P. Kaijser, T. Parker, and D. Pinkas, “SESAME: The solution to security for open distributed systems,” Computer Communications, vol. 17, no. 7, pp. 501– 518, 1994.

[11] G. Wettstein, J. Grosen, and E. Rodriguez, “IDFusion: An open architecture for Kerberos based authorization,” Proc. AFS and Kerberos Best Practices Workshop, June 2006.

[12] M. Walla, “Kerberos explained,” Windows 2000 Advantage Magazine, 2000. [13] Q. Jiang, J. Ma, X. Lu, and Y. Tian, “An efficient two-factor user authentication scheme with unlinkability for wireless sensor networks,” Peer-toPeer Networking and Applications, vol. 8, no. 6, pp. 1070–1081, 2015. [14] O. Althobaiti, M. Al-Rodhaan, and A. Al-Dhelaan, “An

efficient biometric authentication protocol for wireless sensor networks," International Journal of Distributed Sensor Networks, vol. 2013, pp. 1–13, 2013, Article ID 407971, http://dx.doi.org/ 10.1155/2013/407971. [15] K. Xue, C. Ma, P. Hong, and R. Ding, "A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 316 – 323, 2013.

[16] M. Turkanovic, B. Brumen, and M. Holbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," Ad Hoc Networks, vol. 20, pp. 96 – 112, 2014. [17] M. Park, H. Kim, and S. Lee, "Privacy Preserving Biometric-Based User
Authentication Protocol Using Smart Cards," in 17th International Conference on Computational Science and Engineering, Chengdu, China, 2014, pp. 1541–1544. [18] P. K. Dhillon and S. Kalra, "A lightweight biometrics based remote user authentication scheme for IoT services," Journal of Information Security and Applications, vol. 34, pp. 255 – 270, 2017.

[19] S. D. Kaul and A. K. Awasthi, "Security Enhancement of an Improved
Remote User Authentication Scheme with Key Agreement," Wireless Personal Communications, vol. 89, no. 2, pp. 621–637, 2016.

[20] D. Kang, J. Jung, H. Kim, Y. Lee, and D. Won, "Efficient and Secure
Biometric-Based User Authenticated Key Agreement Scheme with Anonymity," Security and Communication Networks, vol. 2018, pp. 1–14, 2018, Article ID 9046064, https://doi.org/10.1155/2018/9046064.

[21] D. Dolev and A. C. Yao, "On the security of public key protocols," IEEE Transactions on Information Theory, vol. 29, no. 2, pp. 198–208, 1983.

[22] A. K. Das, "A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks," Peer-to-Peer Networking and Applications, vol. 9, no. 1, pp. 223–244, 2016.

[23] "A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor," International Journal of Communication Systems, vol. 30, no. 1, pp. 1–25, 2017.

[24] C. T. Li, C. Y. Weng, and C. C. Lee, “An advanced temporal credentialbased security scheme with mutual authentication and key agreement for wireless sensor networks,” Sensors, vol. 13, no. 8, pp. 9589–9603, 2013.

[25] D. He, N. Kumar, and N. Chilamkurti, “A secure temporal-credentialbased mutual authentication and key agreement scheme for wireless sensor networks,” in International Symposium on Wireless and pervasive Computing (ISWPC), Taipei, Taiwan, 2013, pp. 1–6.

[26] M. Turkanovic and M. Holbl, “An improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks,” ELEKTRONIKA IR ELEKTROTECHNIKA, vol. 19, no. 6, pp. 109 – 116, 2013.

[27] R. Amin and G. P. Biswas, “A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks,” Ad Hoc Networks, vol. 36, pp. 58–80, 2016.

[28] C.-C. Chang and N.-T. Nguyen, “An Untraceable Biometric-Based Multiserver Authenticated Key Agreement Protocol with Revocation,” Wireless Personal Communications, vol. 90, no. 4, pp. 1695–1715, 2016.

[29] Z. Xia, C. Yuan, R. Lv, X. Sun, N. N. Xiong, and Y. Shi, “A Novel Weber Local Binary Descriptor for Fingerprint Liveness Detection,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, doi: 10.1109/TSMC.2018.2874281.

[30] C. Yuan, X. Sun, and Q. M. J. Wu, “Difference co-occurrence matrix using BP neural network for fingerprint liveness detection,” Soft Computing, vol. 23, no. 13, pp. 5157–5169, 2019.

[31] W. Yang, S. Wang, J. Hu, G. Zheng, and C. Valli, “Security and Accuracy of Fingerprint-Based Biometrics: A Review,” Symmetry, vol. 11, no. 2, 2019. [Online]. Available: https://www.mdpi.com/2073-8994/11/2/141

[32] X. Huang, X. Chen, J. Li, Y. Xiang, and L. Xu, “Further Observations on Smart-Card-Based Password-Authenticated Key Agreement in Distributed Systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp. 1767–1775, 2014.