A

Project Report

on

**Secure Smart Healthcare Data with**

**Data Aggregation**

Submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering

by

**Kanchukota Shruthi**

**(20EG105419)**

**Pasupula Sowmya**

**(20EG105441)**

**CH. Yuva Shruthik**

**(20EG105734)**



Under the guidance of

**Mr. Jayendra Kumar**

Assistant Professor,

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ANURAG UNIVERSITY**

**VENTAKAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S - 500088**

**TELANGANA**

**(2023-2024)**

**DECLARATION**

We hereby declare that the Project Report entitled **“Secure Smart Healthcare Data with Data Aggregation”** submitted to **Anurag University** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B. Tech)** in **Computer Science and Engineering** is a record of an original work done by us under the guidance of **Mr. Jayendra Kumar, Assistant Professor,** and this report has not been submitted to any other university for the award of any other degree or diploma.

Place: Anurag University, Hyderabad

Kanchukota Shruthi

(20EG105419)

Pasupula Sowmya

(20EG105441)

CH. Yuva Shruthik

(20EG105734)



**CERTIFICATE**

This is to certify that the Project Report entitled **Secure Smart Healthcare Data with Data Aggregation** that is being submitted by **Ms. Kanchukota Shruthi** bearing the hall ticket number **20EG105419, Ms. Pasupula Sowmya** bearing the hall ticket number **20EG105441** and **Mr. Ch.Yuva Shruthik** bearing the hall ticket number **20EG105734** in partial fulfilment for the award of B.Tech degree in Computer Science and Engineering to Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

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

|  |  |
| --- | --- |
| Signature of The Supervisor  Mr. Jayendra Kumar  Assistant Professor | Signature Dean CSE,  Dr. G. Vishnu Murthy |

|  |
| --- |
| External Examiner 1 |
| External Examiner 2 |

**ACKNOWLEDGMENT**

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Mr. Jayendra Kumar**, Assistant Professor, Department of CSE for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar, Dean School of Engineering, Anurag University**, for his encouragement and timely support in our B. Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy, Dean CSE, Anurag University**. We also express my deep sense of gratitude to **Dr. VVSSS Balaram, Academic co-ordinator, Anurag University** and **Dr. Pallam Ravi, Project in-Charge.** Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage our project work.

Kanchukota Shruthi

(20EG105419)

Pasupula Sowmya

(20EG105441)

Ch.Yuva Shruthik

(20EG105734)

**ABSTRACT**

The evolution of digital technology in the healthcare sector has ushered in the era of smart healthcare, a paradigm shift aimed at enhancing the efficacy and personalization of medical care through the integration of various forms of health data, including patient records, vital statistics, and real-time health monitoring. Despite its potential, smart healthcare systems face significant challenges, notably in the collection and processing of health data. These challenges encompass unrestricted patient access to their data, which poses risks of exposing sensitive information, and the potential for extended delays in transmitting access keys to doctors. These issues highlight the critical need for robust data management and security protocols in smart healthcare systems to safeguard patient information while ensuring efficient and secure access for medical professionals.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **CONTENT** | **Page No.** |
| 1. | Introduction   * 1. Unrestricted Patient Data Access   2. Risks in Data Collection and Delayed Access Key Transmission   3. Motivation   4. Problem Definition   5. Problem Illustration   6. Objective of the Project | 1  1  3  5  6  6  7 |
| 2. | Literature Document | 9 |
| 3. | Secure Smart Healthcare Data with Data Aggregation  3.1. Patient Module  3.2. Doctor Module  3.3. Test Cases | 17  17  21  30 |
| 4. | Implementation  4.1. Functionalities  4.2. Attributes  4.3. Experimental Screenshot  4.4. Dataset | 42  42  43  37  40 |
| 5. | Experimental Setup  5.1. Setup NetBeans  5.2. Setup MYSQL  5.3. Libraries used  5.4. Parameters | 39  39  42  43  49 |
| 6. | Discussion of Results | 51 |
| 7. | Summary, Conclusion, and Recommendation | 54 |
| 8. | Future Enhancements | 56 |
| 9. | References | 57 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No** |
| 2.1 | System architecture based on user-centric viewpoint | 15 |
| 3.1 | Architecture diagram | 17 |
| 3.1.1 | Overview of Patient Module | 18 |
| 3.1.2 | Registration form | 19 |
| 3.2.1 | Class Diagram | 22 |
| 4.3.1 | Patient Upload data | 37 |
| 4.3.2 | View Encrypted data | 37 |
| 4.3.3 | Send a request to Fog Framework | 37 |
| 4.3.4 | Verify Request and send Keys To mail | 37 |
| 4.3.5 | Keys sent to mail | 37 |
| 4.3.6 | Download the data | 38 |
| 4.3.7 | Verify the Keys | 38 |
| 4.3.8 | View Decrypted data | 38 |
| 4.3.9 | Send request To Cloud Framework | 38 |
| 4.4.0 | Mail sent By Cloud Framework | 39 |
| 4.4.1 | Download Each Parameter | 39 |
| 4.4.2 | Total Process time | 39 |
| 4.4.3 | Data Collected | 40 |
| 6.1 | Total Process Time | 52 |
| 6.2 | Recall score of models | 53 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| 1.5.1. | More Time taking for the Total Process Problem in the Existing Approach | 6 |
| 2.1 | Comparison of Existing Methods | 15 |
| 3.3.1 | Registration Verification | 30 |
| 3.3.2 | Login Verification | 31 |
| 3.3.3 | Sending data to fog and cloud framework | 32 |
| 3.3.4 | Downloading Original data | 32 |
| 6.1 | Total Process time of Different Users | 52 |
| 6.2 | Fog Framework vs Cloud Framework | 53 |

**List of Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form** |
| RBAC | Role-Based Access Control |
| SSL | Secure Socket Layer |
| TLS | Transport Layer Security |
| E2EE | End-to-End Encryption |

1. **Introduction**

The evolution of digital technology has brought in a new way of doing things in the healthcare sector, commonly referred to as smart healthcare. This contemporary approach to health management integrates various forms of health data, including patient records, vital statistics, and real-time health monitoring, to enhance the efficacy and personalization of medical care. The promise of smart healthcare systems lies in their ability to process vast quantities of data. However, the collection and processing of health data present two major challenges for smart healthcare systems. Firstly, patients’ information is granted unrestricted access times to their data. Secondly, the data collection process carries the risk of exposing users' sensitive information and may also result in extended delays in transmitting the necessary access keys to the doctor. In smart healthcare systems, two primary challenges arise from the collection and processing of health data.

* 1. **Unrestricted Patient Data Access**

Ensuring patients have unrestricted access to their data while maintaining security and privacy is a delicate balance. On one hand, patients should be empowered with their health information to make informed decisions. On the other hand, this data is highly sensitive, and its accessibility needs to be safeguarded against unauthorized use. A potential approach to address this is the implementation of blockchain technology. By leveraging a decentralized system, patients can gain access to their data through secure, immutable transactions, enhancing data integrity and security while ensuring patient data is accessible whenever needed.

Step-1: User Authentication and Authorization

Role-Based Access Control (RBAC): Define roles for different types of users (e.g., patients, doctors, nurses) and grant permissions based on these roles. This ensures that users can only access the data necessary for their role.

Step-2: Data Encryption and Secure Transmission:

End-to-End Encryption: Encrypt data at the point of creation and decrypt it only at the point of use. This ensures that data is unreadable to anyone who intercepts it during transmission.

Secure Socket Layer (SSL)/Transport Layer Security (TLS): Use SSL/TLS protocols to secure data in transit. This is crucial for protecting health information as it moves between the patient’s device and the website’s servers.

Step-3: Patient-Centric Data Access:

Dynamic Consent: This allows patients to control who accesses their data and for what purpose. Implement a user-friendly interface for patients to give and revoke consent for data access easily

Step-4: Patient Education and Support:

Support Channels: Offer robust support channels, including email, and phone support, where patients can report suspicious activity or seek assistance with accessing their data.

Step-5: Output: Developing a secure website for patient health data access involves implementing role-based access control, end-to-end encryption, and dynamic consent mechanisms. Compliance with data protection regulations, and providing patient education on data security are crucial to balance accessibility with privacy and security.

Advantages:

* Enhanced Security: Authentication ensures that only verified users can access the system, significantly reducing the risk of unauthorized access and protecting sensitive information from potential security breaches.
* Access Control: Authorization allows for fine-grained control over what authenticated users can see and do within the system, ensuring that users only access information and perform actions relevant to their roles.
* User Trust and Confidence: Robust authentication and authorization mechanisms build user trust by demonstrating a commitment to securing personal and sensitive information, essential for maintaining reputation and user confidence in the platform.

Challenges:

* **Evolving Security Threats:** The landscape of cybersecurity threats is continuously evolving, with attackers developing new methods to bypass authentication and authorization mechanisms. Keeping systems secure against these ever-changing threats requires ongoing vigilance, regular updates to security protocols, and user education, all of which demand significant resources and expertise.
  1. **Risks in Data Collection and Delayed Access Key Transmission**

The collection of health data, encompassing patient records, vital statistics, and real-time health monitoring, poses significant privacy risks. Unauthorized access to this data can lead to breaches of privacy and potentially misuse of personal health information. Moreover, delays in transmitting access keys to doctors can hinder timely medical intervention, impacting patient care. To mitigate these risks, advanced encryption methods coupled with efficient key management systems can be employed. For instance, homomorphic encryption allows data to be processed in its encrypted form, ensuring data privacy while still being usable for analysis. Additionally, using a real-time data access management system that utilizes secure, fast, and reliable communication channels can minimize delays in key transmission, ensuring healthcare professionals have timely access to the data they need.

Incorporating these solutions requires a multi-faceted approach, integrating advanced cybersecurity measures, robust data management protocols, and ensuring regulatory compliance. As smart healthcare systems evolve, continuously assessing and adapting to emerging security challenges will be critical in safeguarding patient data and enhancing healthcare delivery.

To mitigate the risks associated with the collection of health data and the potential delays in transmitting access developing a secure website involves several critical steps. Here's a detailed approach:

Step 1: Implement Robust User Authentication

Use Strong Password Policies to encourage users to create complex passwords that are hard to guess.

Step 2: Utilize Advanced Encryption Methods

Implement End-to-End Encryption (E2EE) for all data transmission between patients, healthcare providers, and databases to ensure that data remains encrypted and unreadable to unauthorized parties.

Step 3: Efficient Key Management Systems

Automate Key Distribution to ensure timely access for authorized users, reducing delays in key transmission and access to critical health data.

Step 4: Real-Time Data Access Management

Develop a Dynamic Access Control System that adjusts access rights in real-time based on predefined policies, user roles, and contexts.

Step 5: User Education and Support

Establish a Support and Incident Response Team to assist users with security concerns and respond promptly to any incidents of unauthorized access or data breaches.

Step-6: Output Summary: Smart healthcare promises personalized medical care by integrating various health data, facing challenges in data security and access delays. These issues underscore the need for enhanced data management and security in healthcare systems.

Incorporating these steps in developing a secure website for health data management can significantly mitigate risks associated with data privacy and access delays, ensuring that both patients and healthcare professionals benefit from timely and secure access to health information. This approach requires a careful balance of advanced technology, user-friendly design, and continuous adaptation to emerging security challenges.

Advantages:

* Improved Patient Care: Data collection and efficient access key transmission enable healthcare providers to make informed decisions based on comprehensive patient information, leading to personalized and timely medical interventions.
* Enhanced Health Monitoring: Continuous and real-time health data collection facilitates proactive health management, allowing for early detection of potential health issues and the monitoring of chronic conditions more effectively.

Challenges:

* Privacy and Security Risks: The collection and transmission of health data expose sensitive patient information to potential breaches, posing significant privacy and security challenges that require robust protection measures.
  1. **Motivation**

The smart healthcare project is all about using the latest technology to make healthcare better for everyone. The idea is to take all kinds of health information, like doctor’s notes, test results, and even real-time data from health trackers, and use this to create care that’s tailored just for you. This way, doctors can know exactly what each patient needs and can act quickly. The project aims to make sure that all this private health info is kept safe while making it easy for healthcare professionals to access it when they need to. This could mean better and faster care for patients.

One big reason for this project is to fix problems in the current healthcare system. Right now, there are a lot of delays and confusion because everyone’s health information is scattered and hard to get to. Plus, healthcare tends to wait until people are sick before it does anything. The smart healthcare project wants to change that by keeping an eye on health data all the time. This way, doctors can spot problems early or even prevent them, making everyone healthier in the long run.

By bringing in these high-tech solutions, the project hopes to make healthcare more about keeping people well instead of just treating them when they’re sick. It’s all about making sure doctors and patients are on the same page and working together more smoothly. The end goal is a healthcare system that’s easier for everyone to use, keeps your information safe, and is ready to meet the needs of tomorrow. This could mean a big improvement in how we all experience healthcare and could help us all live healthier, happier lives.

* 1. **Problem Definition**

The challenge faced in smart healthcare systems; the extensive collection of multidimensional health data raises severe privacy concerns. The challenge lies in safeguarding the privacy of sensitive health information while maintaining the utility of the data for healthcare purposes. The collection and processing of health data present two major challenges for smart healthcare systems. Firstly, information is granted unrestricted access times to their data. Secondly, the data collection process carries the risk of exposing users' sensitive information and may also result in extended delays in transmitting the necessary access keys to the doctor.

**1.5.** **Problem Illustration**

In the existing method of data aggregation and transmission within the healthcare sector involves compiling various pieces of patient data into a single, encrypted message. This approach, while secure, presents several challenges and inefficiencies that impact the overall effectiveness of healthcare delivery.

**Table 1.5.1. More Time taking for the Total Process Problem in the Existing Approach**

Firstly, the process of aggregating diverse healthcare data—ranging from basic patient information and medical histories to complex diagnostic results and real-time monitoring data—into a single message can be exceedingly time-consuming. This is due in part to the need to ensure that all data is accurately captured, formatted, and encrypted before transmission. Given the critical nature of healthcare information, any errors in this process can have significant repercussions, necessitating a thorough and, consequently, slow procedure.

Moreover, the use of a single message for the encryption of diverse data sets introduces complexity in data decryption for healthcare professionals. Upon receiving the aggregated data, healthcare providers must decrypt the message to access the information needed for patient care. This decryption process can be cumbersome, as it requires specialized knowledge and tools. Additionally, because the data is presented as various parameters within a single message, healthcare professionals may find it challenging to quickly locate specific pieces of information. This can delay decision-making and treatment, hindering the ability to provide timely and efficient care.

**1.6. Objective Of The Project**

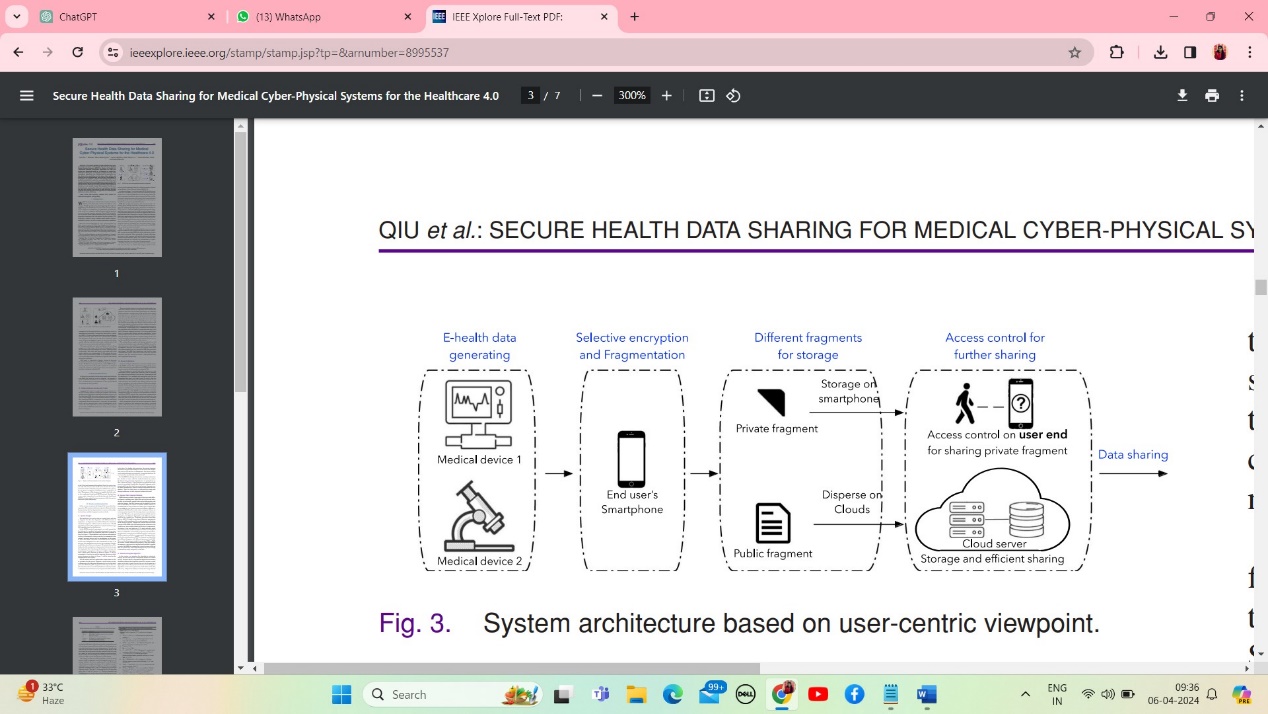
The primary objective of the smart healthcare project is to revolutionize the way healthcare is delivered by harnessing the power of digital technology to create a more efficient, personalized, and proactive healthcare system. By integrating a wide array of health data—from electronic health records and diagnostic tests to real-time monitoring and genetic information—the project aims to enable healthcare professionals to tailor treatments and interventions to the individual needs of each patient. This personalized approach is expected to improve treatment outcomes significantly, reduce healthcare costs by avoiding unnecessary treatments, and enhance patient satisfaction. Moreover, the project seeks to ensure the utmost privacy and security of patient data, employing state-of-the-art encryption and secure transmission methods to protect sensitive information from unauthorized access or breaches.

Additionally, the project addresses the need to overcome existing barriers within healthcare systems, such as inefficiencies, fragmented care delivery, and the reactive nature of traditional health services. By enabling real-time data sharing and analysis, the project aspires to facilitate better coordination among healthcare providers, reduce time delays in accessing vital health information, and shift the focus from reactive treatment to preventive care. This approach not only aims to streamline healthcare processes and make them more responsive but also empowers patients with better access to their health data, encouraging active participation in their health management. Ultimately, the smart healthcare project envisions a future where technology-driven solutions elevate the standard of care, making healthcare more accessible, effective, and aligned with the evolving needs of global populations.

1. **Literature Survey**

[1] S. Wijethilaka and M. Liyanage, “Survey on network slicing for Internet of Things realization in 5G networks,” IEEE Commun. Surv. Tut., vol. 23, no. 2, pp. 957–994, Apr./Jun. 2021.The paper by S. Wijethilaka and M. Liyanage provides a comprehensive survey on the implementation of network slicing for the Internet of Things (IoT) within 5G networks. Network slicing is a crucial concept in 5G, enabling the partitioning of a single physical network infrastructure into multiple virtual networks to meet diverse service requirements.[2] M. A. Siddiqi, C. Doerr, and C. Strydis, “Imdfence: Architecting a secure protocol for implantable medical devices,” IEEE Access,vol.8, pp. 147948–147964, 2020.The paper authored by M. A. Siddiqi, C. Doerr, and C. Strydis presents "Imdfence," a novel approach to architecting a secure protocol tailored specifically for implantable medical devices (IMDs). With the proliferation of IMDs, ensuring their security is paramount to safeguarding patient health and privacy. [3] G. Manogaran, M. Alazab, H. Song, and N. Kumar, “CDP-UA: Cognitive data processing method wearable sensor data uncertainty analysis in the Internet of Things assisted smart medical healthcare systems,” IEEE J. Biomed. Health Informat., vol. 25, no. 10, pp. 3691–3699, Oct. 2021.The study by G. Manogaran, M. Alazab, H. Song, and N. Kumar explores the Cognitive Data Processing Method (CDP-UA) within the context of wearable sensor data uncertainty analysis for smart medical healthcare systems, facilitated by the Internet of Things (IoT). The literature survey in this domain is likely to delve into the intersection of wearable sensor technology, healthcare systems, and IoT-enabled data analytics. [4] K. A. Bhavsar et al., “A comprehensive review on medical diagnosis using machine learning,” Comput., Mater. Continua, vol. 67, no. 2, 2021, Art. no. 1997.The paper authored by K. A. Bhavsar and colleagues provides a comprehensive review on medical diagnosis using machine learning techniques, as published in the journal "Computers, Materials & Continua." Medical diagnosis utilizing machine learning has emerged as a promising area of research, aiming to enhance the accuracy, efficiency, and accessibility of healthcare services. [5] S. A. Chaudhry et al., “An anonymous device to device access control basedonsecurecertificate forinternet of medicalthingssystems,” Sustain. Cities Soc., vol. 75, 2021, Art. no. 103322.The paper authored by S. A. Chaudhry and colleagues explores an innovative approach for ensuring secure access control in Internet of Medical Things (IoMT) systems through the implementation of an anonymous device-to-device access control mechanism based on secure certificates. Within the realm of IoMT, security and privacy concerns are paramount due to the sensitive nature of medical data and the interconnectedness of medical devices. [6] H. Habibzadeh and T. Soyata, “Toward uniform smart healthcare ecosys tems: A survey on prospects, security, and privacy considerations, ” in Connected Health in Smart Cities. Berlin, Germany: Springer, 2020.The work by H. Habibzadeh and T. Soyata provides a comprehensive survey on the prospects, security, and privacy considerations in achieving uniform smart healthcare ecosystems. Within the broader context of connected health in smart cities, the literature survey in this domain likely encompasses a wide array of research efforts aimed at understanding and addressing the complex challenges and opportunities presented by the integration of smart technologies into healthcare systems. [7] P.Gope,Y.Gheraibia,S.Kabir,andB.Sikdar,“AsecureIoT-basedmodern healthcare system with fault-tolerant decision making process,” IEEE J. Biomed. Health Informat., vol. 25, no. 3, pp. 862–873, Mar. 2021. The paper authored by P. Gope, Y. Gheraibia, S. Kabir, and B. Sikdar introduces a secure Internet of Things (IoT)-based modern healthcare system with a fault-tolerant decision-making process, as published in the IEEE Journal of Biomedical and Health Informatics. Within the realm of IoT-enabled healthcare systems, a literature survey in this topic is expected to cover a broad spectrum of research exploring the intersection of IoT technology, healthcare applications, security, and fault tolerance mechanisms. [8] H. Qiu, M. Qiu, M. Liu, and G. Memmi, “Secure health data sharing for medical cyber-physical systems for the healthcare 4.0,” IEEE J. Biomed. Health Informat., vol. 24, no. 9, pp. 2499–2505, Sep. 2020.The paper authored by H. Qiu, M. Qiu, M. Liu, and G. Memmi explores the topic of secure health data sharing for medical cyber-physical systems (MCPS) within the context of healthcare 4.0, as published in the IEEE Journal of Biomedical and Health Informatics. In the landscape of healthcare 4.0, characterized by the integration of digital technologies into healthcare systems, a literature survey in this area would likely encompass a broad spectrum of research focusing on the secure sharing of health data in MCPS environments. [9] N. Tsafack et al., “A new chaotic map with dynamic analysis and en cryption application in internet of health things,” IEEE Access, vol. 8, pp. 137731–137744, 2020.The paper authored by N. Tsafack and colleagues introduces a novel chaotic map with dynamic analysis and encryption application in the context of the Internet of Health Things (IoHT), as published in IEEE Access. Within the realm of IoHT, a literature survey in this area is likely to encompass a diverse range of research efforts exploring the intersection of chaotic systems, security, and healthcare applications. [10] R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan, “Mrc4: A modified RC4 algorithm using symmetric random function generator for improved cryptographic features,” IEEE Access, vol. 7, pp. 172045–172054, 2019.The paper authored by R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan introduce MRC4, a modified version of the RC4 algorithm utilizing a symmetric random function generator to enhance cryptographic features. Published in IEEE Access, this work contributes to the ongoing research in cryptographic algorithms and security. [11] R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan, “Mrc4: A modified RC4 algorithm using symmetric random function generator for improved cryptographic features,” IEEE Access, vol. 7, pp. 172045–172054, 2019.The paper authored by R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan introduces MRC4, a modified version of the RC4 algorithm utilizing a symmetric random function generator to enhance cryptographic features. Published in IEEE Access, this work contributes to the ongoing research in cryptographic algorithms and security. The RC4 algorithm has been widely used in various applications due to its simplicity and efficiency, but it has also been known to suffer from vulnerabilities such as biases in its output and weaknesses in its key scheduling algorithm. [12]. A new method is introduced by Differential Privacy (DP), which involves introducing noise into the data while it is being aggregated. It can be difficult to strike a balance between privacy and data quality when using this approach to protect privacy at the risk of possibly decreasing the accuracy and usefulness of the aggregated data for research [13], [14].The complex and multi-dimensional nature of health data produced by smart healthcare systems is beyond the capabilities of one-dimensional PDA techniques, which are restricted to aggregating single data points. More sophisticated PDA techniques are required because such systems are insufficient for effectively managing the range of data kinds and dimensions involved [15], [16]. This need focuses attention on multi-dimensional PDA methods. These sophisticated techniques are made to protect user privacy while handling the wide range of data types that smart healthcare devices gather, from lifestyle behaviors to physiological measures. [17],[18].To sum up, the integration of IoT devices in the healthcare industry signifies a significant change in the way data is gathered and analyzed, with significant advantages for early disease detection, public health, and tailored medication. However, this change also calls for a serious focus on privacy issues, as each of the current PDA techniques has its own set of drawbacks. To address the current obstacles and realize the full potential of IoT in healthcare, multi-dimensional PDA solutions development is an important area of continuing research. Effectively addressing these privacy concerns will be crucial as this sector develops to guarantee that IoT technologies are widely accepted and used to improve healthcare delivery [19][20].S. Wijethilaka and M. Liyanage provides a comprehensive survey on the implementation of network slicing for the Internet of Things (IoT) within 5G networks. Network slicing is a crucial concept in 5G, enabling the partitioning of a single physical network infrastructure into multiple virtual networks to meet diverse service requirements. In the context of IoT, this survey delves into the specific challenges and opportunities presented by network slicing. [21].The paper authored by M. A. Siddiqi, C. Doerr, and C. Strydis presents "Imdfence," a novel approach to architecting a secure protocol tailored specifically for implantable medical devices (IMDs). With the proliferation of IMDs, ensuring their security is paramount to safeguarding patient health and privacy. The literature surrounding secure protocols for IMDs is multifaceted, encompassing various aspects such as communication security, data integrity, authentication, and privacy preservation. Prior research has highlighted the vulnerabilities inherent in traditional protocols when applied to IMDs, including susceptibility to cyber-attacks and unauthorized access. Through this survey, researchers gain insights into the state-of-the-art approaches, emerging trends, and ongoing efforts to fortify the security of IMDs, ultimately advancing the field toward safer and more resilient healthcare systems[22]. G. Manogaran, M. Alazab, H. Song, and N. Kumar explores the Cognitive Data Processing Method (CDP-UA) within the context of wearable sensor data uncertainty analysis for smart medical healthcare systems, facilitated by the Internet of Things (IoT). The literature survey in this domain is likely to delve into the intersection of wearable sensor technology, healthcare systems, and IoT-enabled data analytics. Previous research has underscored the importance of wearable sensors in facilitating continuous health monitoring and personalized healthcare delivery, thereby enhancing patient outcomes and reducing healthcare costs. [23] K. A. Bhavsar and colleagues provides a comprehensive review on medical diagnosis using machine learning techniques, as published in the journal "Computers, Materials & Continua." Medical diagnosis utilizing machine learning has emerged as a promising area of research, aiming to enhance the accuracy, efficiency, and accessibility of healthcare services. The literature survey within this domain is likely to encompass a broad range of topics, including various machine learning algorithms, data sources, medical applications, and evaluation methodologies. Prior research in this field has explored the effectiveness of machine learning in diverse medical domains such as cardiology, oncology, neurology, and radiology, among others. Studies have investigated the performance of different machine learning models in diagnosing specific medical conditions, detecting abnormalities from medical images, predicting disease outcomes, and optimizing treatment plans. [24] .S. A. Chaudhry and colleagues explores an innovative approach for ensuring secure access control in Internet of Medical Things (IoMT) systems through the implementation of an anonymous device-to-device access control mechanism based on secure certificates. Within the realm of IoMT, security and privacy concerns are paramount due to the sensitive nature of medical data and the interconnectedness of medical devices. A literature survey within this topic likely encompasses a wide range of research efforts aimed at addressing security challenges in IoMT systems. Previous studies have investigated various security mechanisms, including authentication protocols, encryption techniques, access control models, and intrusion detection systems, tailored specifically to the healthcare domain. [25].The work by H. Habibzadeh and T. Soyata provides a comprehensive survey on the prospects, security, and privacy considerations in achieving uniform smart healthcare ecosystems. Within the broader context of connected health in smart cities, the literature survey in this domain likely encompasses a wide array of research efforts aimed at understanding and addressing the complex challenges and opportunities presented by the integration of smart technologies into healthcare systems. Previous studies have explored the potential benefits of smart healthcare ecosystems, including improved patient outcomes, enhanced disease prevention, and more efficient healthcare delivery. [26] P. Gope, Y. Gheraibia, S. Kabir, and B. Sikdar introduces a secure Internet of Things (IoT)-based modern healthcare system with a fault-tolerant decision-making process, as published in the IEEE Journal of Biomedical and Health Informatics. Within the realm of IoT-enabled healthcare systems, a literature survey in this topic is expected to cover a broad spectrum of research exploring the intersection of IoT technology, healthcare applications, security, and fault tolerance mechanisms. Previous studies have investigated the integration of IoT devices and sensors into healthcare environments for real-time patient monitoring, remote diagnostics, and personalized treatment delivery. [27]H. Qiu, M. Qiu, M. Liu, and G. Memmi explores the topic of secure health data sharing for medical cyber-physical systems (MCPS) within the context of healthcare 4.0, as published in the IEEE Journal of Biomedical and Health Informatics. In the landscape of healthcare 4.0, characterized by the integration of digital technologies into healthcare systems, a literature survey in this area would likely encompass a broad spectrum of research focusing on the secure sharing of health data in MCPS environments. Previous studies have examined various aspects of health data sharing, including data privacy, confidentiality, integrity, and interoperability. The survey may delve into existing approaches for securing health data sharing, such as cryptographic techniques, access control mechanisms, data anonymization methods, and blockchain technology. [28] N. Tsafack and colleagues introduces a novel chaotic map with dynamic analysis and encryption application in the context of the Internet of Health Things (IoHT), as published in IEEE Access. Within the realm of IoHT, a literature survey in this area is likely to encompass a diverse range of research efforts exploring the intersection of chaotic systems, security, and healthcare applications. Previous studies have investigated the use of chaotic maps for various cryptographic purposes, including encryption, key generation, and secure communication. [29] R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan introduces MRC4, a modified version of the RC4 algorithm utilizing a symmetric random function generator to enhance cryptographic features. Published in IEEE Access, this work contributes to the ongoing research in cryptographic algorithms and security. The RC4 algorithm has been widely used in various applications due to its simplicity and efficiency, but it has also been known to suffer from vulnerabilities such as biases in its output and weaknesses in its key scheduling algorithm. The literature in this area likely includes a range of studies exploring enhancements and alternatives to RC4, aimed at improving its security while maintaining its performance advantages. [30] K. Sowjanya, M. Dasgupta, and S. Ray, published in the International Journal of Information Security in 2020, introduces an elliptic curve cryptography (ECC) based enhanced anonymous authentication protocol tailored for wearable health monitoring systems. It responds to the escalating demand for secure and privacy-preserving solutions in the realm of IoT-based healthcare by integrating ECC's robust security features with the imperative of anonymous authentication. Building upon existing research, the authors address vulnerabilities in traditional authentication methods, emphasizing the importance of safeguarding sensitive user data while ensuring seamless access for authorized parties..

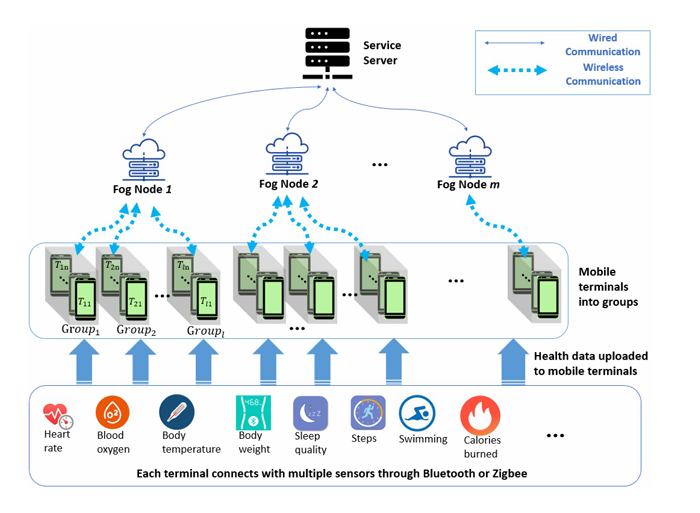
**Figure 2.1.** System architecture based on user-centric viewpoint.



**Table 2.1. Comparison of Existing Methods**

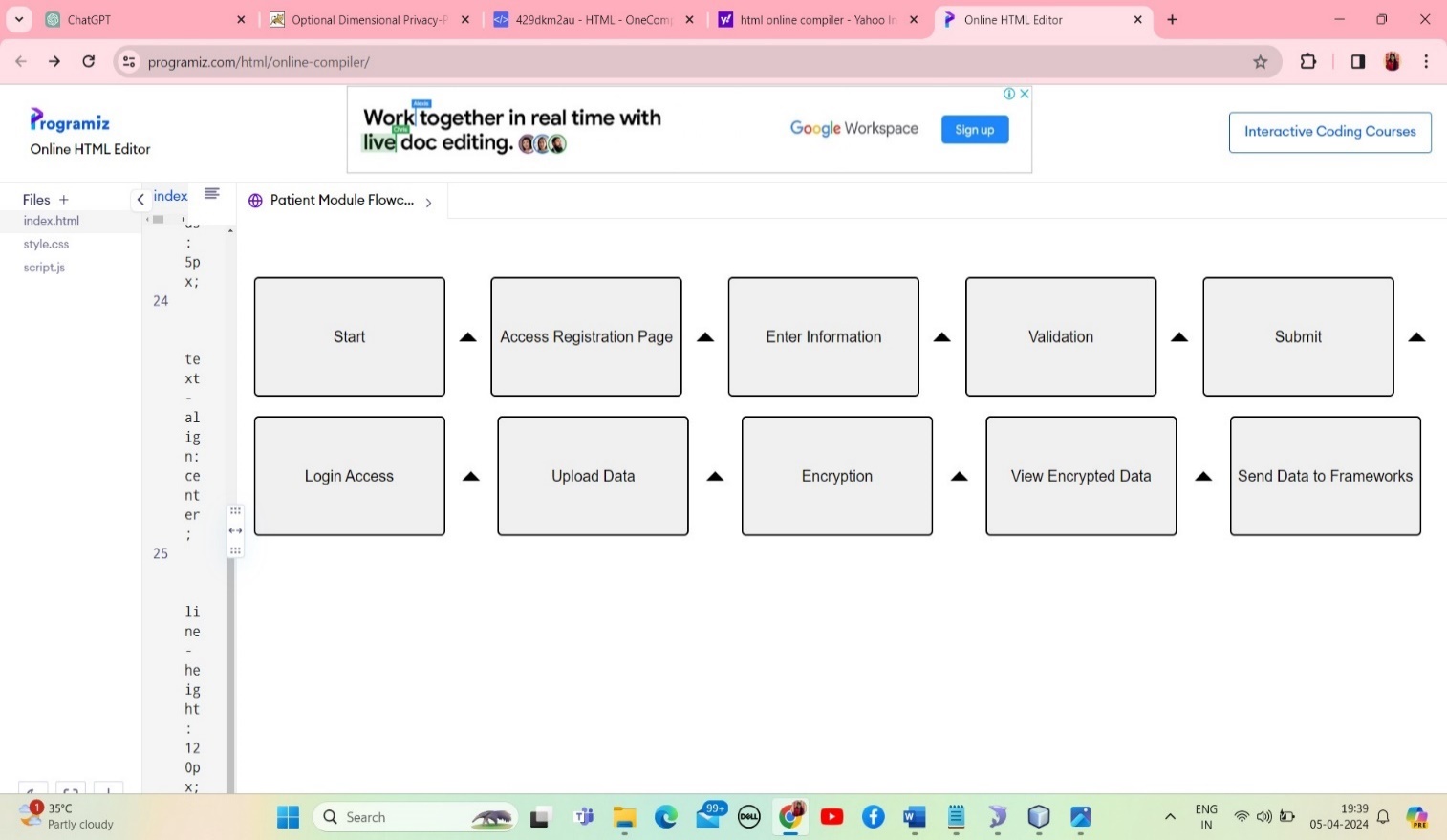
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.no** | **Author (s)** | **Method** | **Advantages** | **Disadvantages** |
| 1 | Alex Adim Obinikpo and Burak Kantarci | The use   of  various   deep learning models like Convolutional Neural Networks (CNNs),Recurrent Neural Networks (RNNs), and their variants  for  analysing                big sensed data. | Ability to process large volumes of data efficiently | High computational requirements and associated costs |
| 2 | Qiong Cai, Hao Wang, Zhenmin Li ,and Xiao Liu | The integration and analysis of multimodal data  using data-driven approaches. | The integration and analysis of multimodal data  using data-driven approaches. | Complexity in integrating and processing diverse                data types. |
| 3 | R. Saha et al. | Novel IoT framework for monitoring oxygen saturation in COVID- 19 patients | Real-time monitoring and early detection of severe COVID-19  Symptoms.Improved patient management          and healthcare response | Enhanced data security and patient privacy  through blockchain integration.  Potential challenges in the deployment and maintenance of IoT devices. |
| 4 | N.Mohamm adzadeh | This underscores wearable sensors'     pivotal role in monitoring vital     signs and disease progression,especially in epidemic control. | Wearable Sensor Significance. Need for Extensive Research. Ethical Compliance.  Technological Advancements | Data security risks, interoperability issues, potential patient resistance, and variability      in sensor accuracy, posing challenges to  effective implementation. |
| 5 | S. Ghosh, M. Dutta and T. Das | Indian Legal Text Summarization: A Text Normalization-based Approach | Uses pretrained models like BART, PEGASUS | Cannot summarize lengthy documents |

1. **Secure Smart Healthcare Data with Data Aggregation**

****The proposed methodology has demonstrated a significant enhancement in data security and access control in healthcare systems. By introducing a fog-based framework with time-based access control, the system ensures that patients' sensitive information is securely encrypted and accessed only during specific times. This approach not only enhances data privacy but also reduces the risk of data exposure and transmission delays. The encryption process, utilizing separate keys for each parameter such as heart rate, blood pressure, body temperature, and body weight, ensures that the data remains confidential throughout the storage and transmission phases. Additionally, the fog framework's ability to aggregate keys and generate a single key for user access requests streamlines the data retrieval process, leading to improved efficiency and reduced response time.

**Figure 3.1. Architecture diagram**

* 1. **Patient Module**

The patient module within the healthcare system plays a pivotal role in facilitating seamless interactions between patients and the healthcare infrastructure. It encompasses various functionalities, starting from patient registration and login processes to data uploading, encryption, and access control. Patients can register on the web application by providing essential details such as username, password, email, and mobile number, ensuring secure access to personalized healthcare services. Upon registration, patients can log in to the system, where they can input vital medical data like heartbeat rate, temperature, oxygen level, and upload relevant medical reports in various formats. The module ensures data security by encrypting patient data using advanced encryption algorithms and manages access control to safeguard sensitive medical information. Additionally, the patient module enables efficient communication and collaboration between patients and healthcare providers, ultimately enhancing the overall quality and effectiveness of patient care.

**Figure 3.1.1. Overview of Patient Module.**

The Patient Module consists of the following stages:

* + 1. **Patient Registration:**

The patient registration process on the web application begins with accessing the registration page, where the patient fills in essential information such as username, password, email, and mobile number. Stringent validation checks ensure that each field is correctly filled out, including a unique username selection, a strong and secure password, a valid email format, and a correctly formatted mobile number. Upon clicking the "Submit" button, the system processes the data, creating a new patient account if all criteria are met. A confirmation message is then sent to the provided email or mobile number for account verification. Once confirmed, the patient gains access to the web page, where they can log in using their registered credentials to benefit from the healthcare services offered by the system.

   For example:

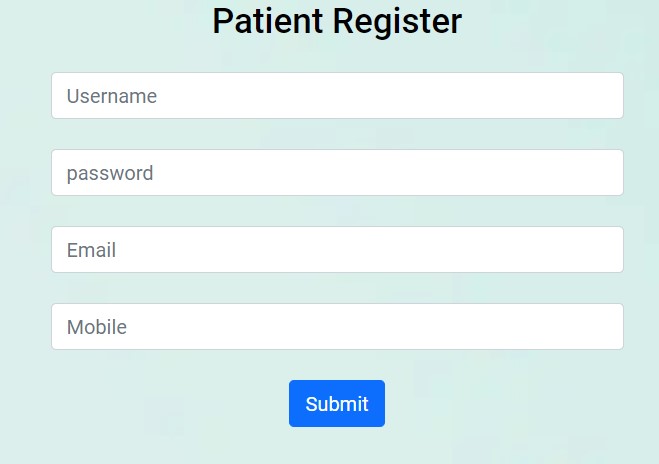
<div class="form-outline mb-4">

<input type="text” name="username" id="form2Example1" placeholder="Username" required class="form-control" />

</div>

<!-- Password input -->

<div class="form-outline mb-4">

 <input type="password" name="password" id="form2Example2" placeholder="password" required="" class="form-control" /> </div>

**Figure 3.1.2. Registration form**

Use the following steps:

Step-1: Display Registration Form: The patient accesses the registration page on the web application.

Step-2: Enter Information: The patient fills in the required information in the registration form, including:

- Username: The desired username for logging into the system.

- Password: A secure password for account authentication.

- Email: The patient's email address for communication and account verification.

- Mobile: The patient's mobile number for contact purposes.

Step-3: Validation: The form includes validation checks to ensure that all required fields are filled out correctly.

Step-4: Submit: After filling in the information, the patient clicks on the "Submit" button to proceed with the registration process.

Step-5: Login Access: Once the account is confirmed, the patient can log in using the registered username and password to access the web page and avail of the healthcare services offered by the system.

* + 1. **Uploading data and View**

After successful registration, the patient logs into the web page using their registered username and password.

Step-1: Patient Fills in Details:

Heartbeat: The patient inputs their current heartbeat rate.

Temperature: The patient provides their body temperature measurement.

Oxygen Level: The patient indicates their oxygen saturation level.

Disease: The patient specifies any specific disease or medical condition they have.

Role: Patient identifies their role or status within the healthcare system.

Time Limit: The patient sets the specified time duration for accessing and viewing the uploaded file.

File Upload: The patient uploads a file in PNG, PDF, or JPEG format containing relevant medical information or reports.

Step-2: Submit Data: After entering all required information, the patient clicks on the "Submit" button to proceed.

Step-3: Encryption: Upon clicking "Submit," each data item (heartbeat, temperature, oxygen level, disease, role, time limit, and uploaded file) is encrypted separately using the Blowfish algorithm. This encryption process generates a separate key for each data item, ensuring data security and confidentiality.

public EncryptFile() { try {

keyGenerator = KeyGenerator.getInstance("Blowfish");

secretKey = keyGenerator.generateKey();

cipher = Cipher.getInstance("Blowfish");} catch (NoSuchPaddingException | NoSuchAlgorithmException ex) {

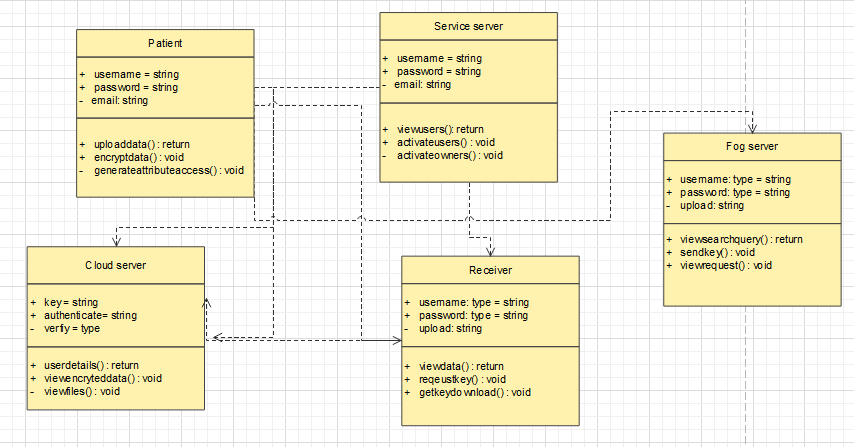
System.out.println(ex); }}

Step-4: View Encrypted Data: The patient can view the encrypted data, represented as ciphertext, reflecting the secure encryption of their sensitive information.

Step-5: Send Data to Fog and Cloud Frameworks: The encrypted data, along with the corresponding keys, is sent to both the fog and cloud frameworks for further processing and storage. The fog framework handles immediate data processing and access control, while the cloud framework ensures long-term storage and availability of the encrypted data.

* 1. **Doctor Module:**

The Doctor Module within the healthcare system encompasses a comprehensive process starting from registration to accessing and decrypting sensitive medical data. Upon registration, doctors provide essential details such as Username, Password, Email, Mobile Number, and Role, followed by account activation facilitated by the Trust Module. Subsequently, doctors initiate requests to the Fog Framework for encryption keys, which are crucial for data access. Upon verification, the Fog Framework sends all necessary keys collectively to the doctor's email. These keys, obtained from the Fog Framework, enable the doctor to decrypt encrypted data, allowing secure viewing of sensitive medical information. Additionally, doctors may request specific keys from the Cloud Framework for more granular access to parameter-specific data. Upon verification, the Cloud Framework sends these keys individually to the doctor's email, facilitating the decryption of parameter-specific data one by one. Ultimately, after successfully decrypting all required data, doctors can view and analyze the decrypted information within the web application, aiding in making informed medical decisions and providing optimal patient care.

****

**Figure 3.2.1. Class Diagram**

* + 1. **Install libraries:**

During the initial phase of workflow, we proceed with the installation of libraries:

i)  Bootstrap: For responsive and mobile-first web design.

ii) jQuery: For simplified DOM manipulation and event handling.

iii) Node.js (with Express.js): For building scalable and efficient server-side applications.

iv)Spring Boot (Java): A powerful framework for building enterprise-level Java applications*.*

* + 1. **Doctor Registration:**

   Step-1. Doctor Registration:

The doctor navigates to the registration page of the web application.

Required details such as Username, Password, Email, Mobile Number, and Role are filled out by the doctor.

Upon completing the form, the doctor clicks on the "Submit" button to register and proceed.

Step-2. Account Activation in Trust Module:

- The registered doctor's account is inactive until the Trust Module activates it.

- Upon registration, an activation request is sent to the Trust Module.

- The Trust Module verifies the request and activates the doctor's account, enabling access to further functionalities.

Step-3. Request to Fog Framework for Keys:

- After successful activation, the doctor sends a request to the Fog Framework to obtain encryption keys for data access.

- The Fog Framework receives and verifies the request, ensuring the legitimacy of the doctor's account and access rights.

- Once verified, the Fog Framework generates and sends all the necessary encryption keys together to the doctor's registered email address.

* + 1. **Downloading keys:**

Step-1. Accessing Keys From Email:

- The doctor receives the email containing all the encryption keys from the Fog Framework.

- To access the keys, the doctor opens the email and clicks on the download button provided.

- Upon download, the doctor can view and copy all the keys from the email for further use.

Step-2. Decrypting Data Using Fog Framework Keys:

- With the encryption keys obtained from the Fog Framework, the doctor navigates back to the web application.

- The doctor pastes the keys into the designated decryption interface and clicks on the submit button to initiate the decryption process.

- The data encrypted using the keys provided by the Fog Framework is decrypted, allowing the doctor to view the sensitive medical information securely.

Step-3. Request to Cloud Framework for Specific Keys:

- In certain cases, the doctor may require specific keys related to individual parameters for more granular access.

- The doctor sends a request to the Cloud Framework specifying the required keys for decryption of particular parameters.

- The Cloud Framework verifies the request and sends the requested keys individually to the doctor's registered email address.

Step-4. Accessing and Decrypting Parameter-specific Data Using Cloud Framework Keys:

- Upon receiving the keys from the Cloud Framework, the doctor accesses the email containing the keys.

- The doctor clicks on the download button for each key to access and copy them for decryption purposes.

- Using the keys obtained from the Cloud Framework, the doctor decrypts parameter-specific data one by one, following the same decryption process as with the Fog Framework keys.

Step-5. Final Decryption and Data Viewing:

After successfully obtaining and applying all necessary encryption keys from both the Fog and Cloud Frameworks, the doctor completes the decryption process.

The decrypted data is now accessible within the web application, allowing the doctor to analyse and make informed medical decisions based on the decrypted information.

* + 1. **Import libraries:**

1. Import Bootstrap for responsive and mobile-first web design:

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

1. Import CryptoJS for cryptographic functions:

<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.0.0/crypto-js.min.js"></script>

1. Import Spring Boot libraries for building Java applications:

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

1. Import MySQL or PostgreSQL for relational database management:

import mysql.connector

import psycopg2

* + 1. **Sample Example:**

The process outlined in the healthcare system involves a seamless flow from patient registration to data encryption and decryption for both patients and doctors. Patients begin by registering on the web application, providing essential information such as username, password, email, and mobile number. Stringent validation checks ensure data accuracy, and upon successful submission, patients receive a confirmation message for account verification. Once verified, patients can log in to access healthcare services and upload vital medical data, including heartbeat, temperature, oxygen level, disease information, and relevant files. This data undergoes encryption using the Blowfish algorithm, generating separate keys for each data item to ensure confidentiality. The encrypted data is then sent to the Fog and Cloud Frameworks for storage and processing. On the doctor's side, registration and account activation follow a similar pattern, with encryption keys obtained from the Fog Framework enabling the decryption of patient data for medical analysis and decision-making. This integrated approach ensures secure data handling and optimal healthcare delivery within the system.

The overall summarization:

Step-1:Patient Registration:

1. Display Registration Form: The patient accesses the registration page on the web application.

2. Enter Information:

- Username: johndoe

- Password: \*\*\*\*\*\*\*\* (secure password)

- Email: johndoe@example.com

- Mobile: +1-555-123-4567

3. Validation: The form ensures all fields are correctly filled out.

4. Submit: After filling in the information, the patient clicks "Submit."

5. Login Access: Upon account confirmation, the patient logs in with username "johndoe" and the password.

Step-2:Uploading Data and View:

1. Patient Fills in Details:

- Heartbeat: 75 BPM

- Temperature: 98.6°F

- Oxygen Level: 98%

- Disease: Hypertension

- Role: Patient

- Time Limit: 24 hours

- File Upload: MedicalReport.pdf

2. Submit Data: The patient clicks "Submit" after filling in all details.

3. Encryption: Each data item (heartbeat, temperature, etc.) is encrypted using Blowfish.

4. View Encrypted Data: Encrypted data appears as ciphertext.

5. Send Data to Fog and Cloud Frameworks: Encrypted data and keys are sent for processing and storage.

Step-3:Doctor Module:

1. Install Libraries:

- Bootstrap for web design.

- jQuery for DOM manipulation.

- Node.js for server-side logic.

2. Doctor Registration:

- Doctor fills the form with details like:

- Username: drsmith

- Password: \*\*\*\*\*\*\*\*

- Email: drsmith@example.com

- Mobile: +1-555-987-6543

- Role: Doctor

- Account activated by the Trust Module.

3. Request to Fog Framework for Keys:

- Doctor requests encryption keys for data access.

- Fog Framework sends keys to email.

4. Downloading Keys:

- Doctor accesses email and downloads keys.

5. Decrypting Data Using Fog Framework Keys:

- Doctor pastes keys into the decryption interface.

- Encrypted data decrypted for viewing.

**3.2.6. Time Calculation for Total Procedure**

The process of calculating the total process time involves several steps within the web application's backend. Initially, the user's email is retrieved from the session attribute to identify the specific user's data. Subsequently, database queries are executed to fetch relevant time data from the "fogrequest" and "cloudrequest" tables based on the user's email and specific statuses such as "mail\_sent" for Fog requests and "key5\_sent" for Cloud requests. The retrieved time data is then displayed on the web page, showcasing the process times for both the Fog and Cloud frameworks. This mechanism enables users and administrators to track and monitor the time taken for various operations within the system, providing valuable insights into performance and efficiency metrics.

Steps for Time Calculation:

1. Retrieve User Email: Obtain the user's email from the session attribute.

2. Query Fog Request Table: Connect to the database and execute a SQL query to fetch data from the "fogrequest" table where the user's email matches and the status is "mail\_sent". Retrieve the time data from the relevant column (e.g., column index 9 in the ResultSet).

3. Query Cloud Request Table: Similarly, query the "cloudrequest" table to fetch data where the user's email matches and the status is "key5\_sent". Retrieve the time data from the relevant column.

4. Display Process Time: Display the calculated process time for the Fog and Cloud frameworks on the web page.

<%

String email = session.getAttribute("email").toString();

String finalt = "";

String finalt1 = "";

try {

Connection co = databasecon.getconnection();

// Query Fog Request Table

PreparedStatement pst = co.prepareStatement("select \* from fogrequest where user='" + email + "' and status='mail\_sent'");

ResultSet rs = pst.executeQuery();

while (rs.next()) {

finalt = rs.getString(9); // Assuming time data is in the 9th column

}

// Query Cloud Request Table

PreparedStatement pst1 = co.prepareStatement("select \* from cloudrequest where user='" + email + "' and status='key5\_sent'");

ResultSet rs1 = pst1.executeQuery();

while (rs1.next()) {

finalt1 = rs1.getString(9); // Assuming time data is in the 9th column

}

%>

<div class="plans-container container-fluid">

<div class="container">

<div class="row section-title">

</div>

<br><br>

<center>

<h3 style="color:black">Fog time is : <%=finalt%></h3><br>

<h3 style="color:black">Cloud time is : <%=finalt1%></h3><br>

</center>

<%

} catch(Exception e) {

System.out.println(e);

}%><br> </div></div>

In the above code snippet:

- We first retrieve the user's email from the session attribute.

- Then we execute SQL queries to fetch the relevant time data from the database tables for the Fog and Cloud requests.

- Finally, we display the calculated process times for the Fog and Cloud frameworks on the web page.

* 1. **Test Cases**

1. Test case-1:

Test case 1 tests the login of admins. The passwords and usernames are given if the correct password and usernames are entered the login will be successful. If any wrong passwords and usernames the login will be denied.

**Table 3.3.1. Registration Verification**

1. Test case 2:

The test case 2 is for checking the users. Again, if the user’s password and user name are given based on only the correct input the users will be able to log in

|  |  |
| --- | --- |
| Name of Test: - | Login as Patient |
| Items being tested: - | Upload file data |
| Sample Input: - | Text file to encrypt with time access |
| Expected output: - | Encrypted data is shown on the patient page |
| Actual output: - | Encryption successful |
| Remarks: - | Failed. |

**Table 3.3.2. Login Verification**

1. Test case 3:

Test case 3 shows the file upload and it is being stored in a different cloud which is called block generation and storing hence this file is divided and stored. If the storage and block generation are not successful then we can store the file

|  |  |
| --- | --- |
| Name of Test: - | Send data to fog |
| The item being tested: - | Data sent from patient to fog framework |
| Sample Input: - | Click on Send to cloud |
| Expected output: - | Fog can view encrypted data |
| Actual output: - | Data stored in the database and viewed to fog framework |
| Remarks: - | Pass. |

**Table 3.3.3. Sending data to fog and cloud framework**

1. Test case 4:

Test case 4 is for file upload where the uploaded is downloaded. During download, if any one of the cloud fails the recovery should start automatically. If the recovery is successful all the blocks will be generated and merged. With the help of network coding the blocks are stored in the cloud.

|  |  |
| --- | --- |
| Name of Test: - | Download request from receiver |
| The item being tested: - | Get aggregate keys to download |
| Sample Input: - | Request key and get keys to mail |
| Expected output: - | View keys and give them to download data |
| Actual output: - | Data decrypted and downloaded |
| Remarks: - | Pass |

**Table 3.3.4. Downloading the original Data**

1. **Implementation**

The program file is Form uploaded which consists of Heartbeat, Temperature, Oxygen Level, Disease, Role, Time Limit, File Upload

**Input:** Patient form

**Output:** Decrypt Data

**4.1. FUNCTIONALITY:**

**4.1.1. Healthcare Patient Portal:**

The Patient Module, also known as the "Health Portal," facilitates patient registration with stringent data validation.

* **Logging and Auditing**: Implement logging and auditing functionalities to track access to patient data and encryption/decryption activities. Maintain detailed logs to facilitate forensic analysis and compliance with regulatory requirements
* **User Interface:** Design a user-friendly interface for patients and healthcare professionals to interact with the system securely. Consider usability principles and accessibility requirements to ensure ease of use for all users.
* **Data Input:** Patients input vital medical data such as heartbeat rate, temperature, and oxygen level into the system. They can also upload relevant medical reports in formats like PNG, PDF, or JPEG through a user-friendly interface.
* **Encryption:** After validation, each data item is encrypted separately using the Blowfish algorithm or other advanced encryption techniques. Encryption converts the data into a secure format that is unreadable without the corresponding decryption key, ensuring data security and confidentiality.

**4.1.2. Healthcare Doctor Portal :**

* **Account Activation**: Upon successful registration, the doctor's account remains inactive until activated by the Trust Module. The Trust Module processes activation requests and activates the doctor's account, granting access to additional functionalities within the system.
* **Data Decryption:** Upon receiving the encryption keys from the Fog Framework, doctors navigate to the web application's designated decryption interface.
* **Key Management**: The system manages encryption keys generated for data security and ensures that authorized users receive the necessary keys for data access and decryption.

**4.1.3. Time Attribute:**

* **Time Access Control**: The system implements time-based access control mechanisms to safeguard sensitive medical information. Authorized individuals are granted access to patient data only during specific time periods determined by the system. Access permissions are dynamically managed, ensuring that users can view or access patient data within the allotted time frame only. Time-based access control enhances data security and privacy, preventing unauthorized access outside designated time windows.
* **Time Calculation for Total Process Time**: The system incorporates time calculation functionalities to monitor and measure the total process time for various operations. Backend processes such as data encryption, decryption, key generation, and data access requests are timestamped and tracked. The system calculates the time taken for each operation and aggregates this data to determine the overall process time for specific tasks or workflows. Time calculation provides valuable insights into system performance, efficiency metrics, and process optimization opportunities. Administrators and users can access time-related metrics and analytics to evaluate system performance, identify bottlenecks, and make informed decisions for process improvement.

**4.2. Attributes:**

**i.** email : This attribute serves as a primary means of communication and verification within the system. It allows users to receive important notifications, such as account activation links or password resets. Additionally, email verification ensures that user accounts are genuine and helps in maintaining the integrity of user data.

**ii.** Heartbeat\_Rate: This vital sign attribute provides real-time information about the patient's cardiovascular health. Monitoring and recording heartbeat rates enable healthcare professionals to assess cardiac function, detect irregularities, and monitor the effectiveness of treatments or interventions.

**iii.** temperature: Measuring body temperature is essential for assessing a patient's overall health status. It helps in identifying fever, monitoring for signs of infection or inflammation, and tracking changes in body temperature over time, which can be indicative of various medical conditions.

**iv.** Oxygen\_Level: Also known as oxygen saturation (SpO2), this attribute measures the amount of oxygen in the bloodstream. It is critical for evaluating respiratory function, especially in patients with conditions like asthma, or COPD, or during critical care scenarios. Monitoring oxygen levels helps in assessing respiratory efficiency and oxygen delivery to tissues.

**v.** Disease: Patients can input specific medical conditions or diseases they are diagnosed with or experiencing symptoms of. This information is vital for healthcare providers to understand the patient's medical history, tailor treatments, monitor disease progression, and provide appropriate care.

**vi.** Role: Identifying user roles within the healthcare system determines access permissions and functionalities. For example, patients may have access to their medical records and appointment scheduling, while doctors may have additional privileges such as viewing patient histories, ordering tests, and prescribing medications.

**vii.** File\_Upload: Allowing users to upload medical reports or documents in various formats (e.g., PNG, PDF, JPEG) centralizes healthcare information management. It facilitates easy access to patient records, enables healthcare professionals to review diagnostic reports, lab results, and imaging studies, and supports collaborative decision-making.

**viii.** Encryption: Using encryption keys for data encryption and decryption ensures that sensitive information remains secure and confidential. Encryption transforms data into ciphertext, making it unreadable without the corresponding decryption key. This protects patient data from unauthorized access and breaches, maintaining data integrity and privacy.

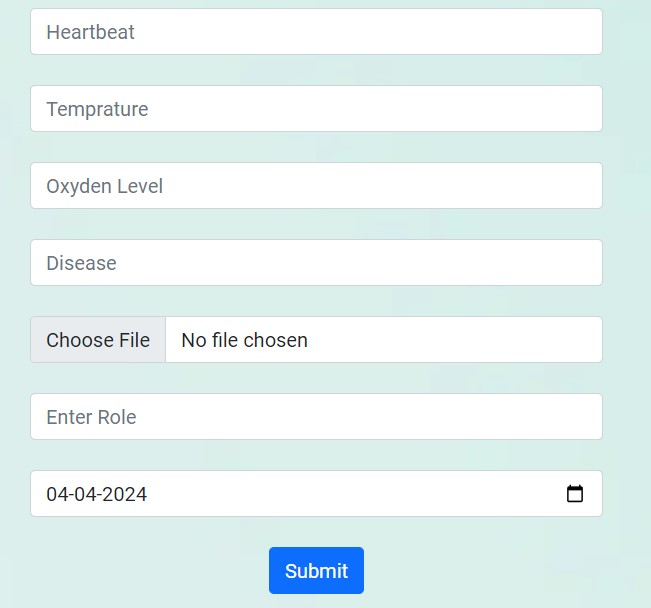
**ix.** Activation: The activation process verifies user credentials and grants access to system functionalities. It involves confirming user identity, validating registration information, and activating user accounts based on predefined criteria, ensuring that only legitimate users can access the system.

**x.** Logging\_Auditing: Logging captures user activities, system events, and access attempts, creating an audit trail for monitoring and analysis. Auditing functionality tracks changes to data, access patterns, and security-related events, aiding in compliance with regulations, identifying anomalies or suspicious activities, and facilitating forensic investigations.

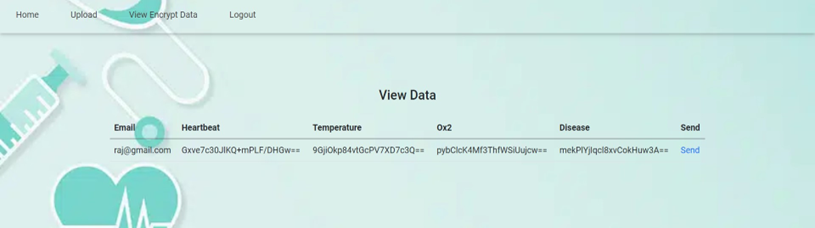
**xi.** Time\_Calculation: Calculating process times for system operations provides insights into performance metrics, efficiency, and optimization opportunities. It helps in identifying bottlenecks, optimizing resource allocation, improving response times, and ensuring timely processing of tasks within the system.

**xii.** Time-Based Access Control: Implementing time-based access control restricts data access based on predefined time criteria. It enforces access policies that specify when users can access certain data or functionalities, enhancing data security, privacy, and compliance with data protection regulations.

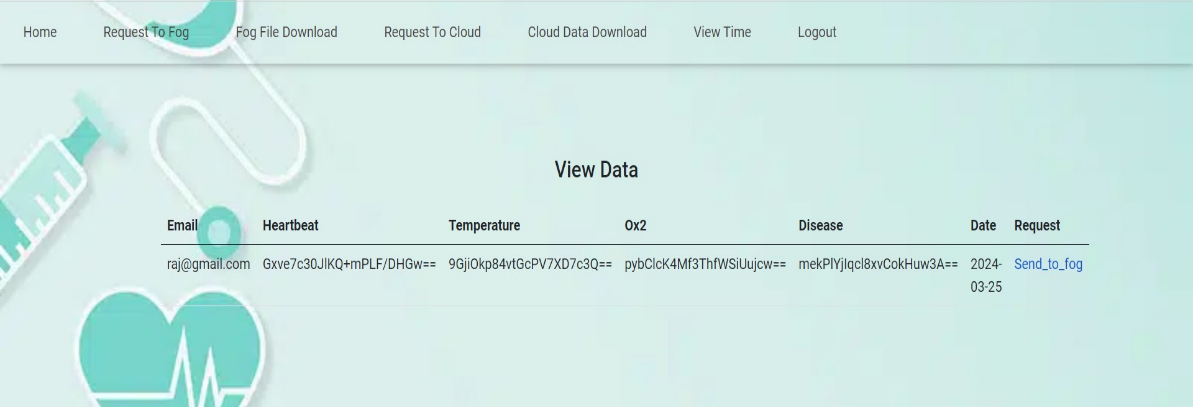
* 1. **Experimental Screenshot**



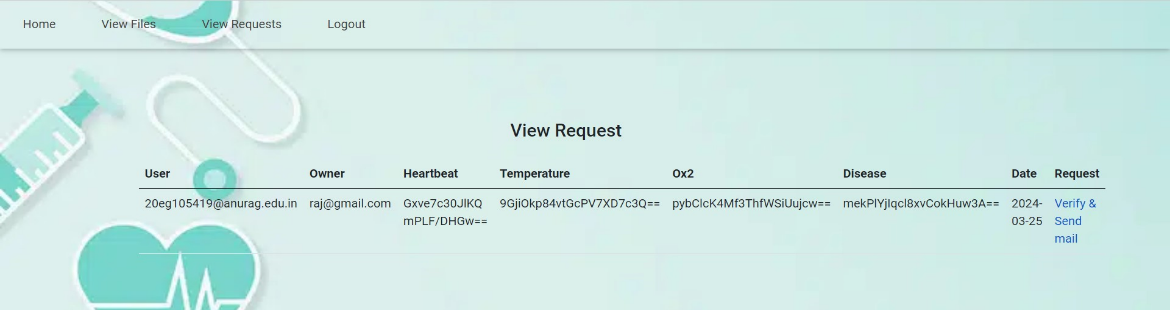
**Figure 4.3.1. Patient Upload data**



**Figure 4.3.2. View Encrypted data**

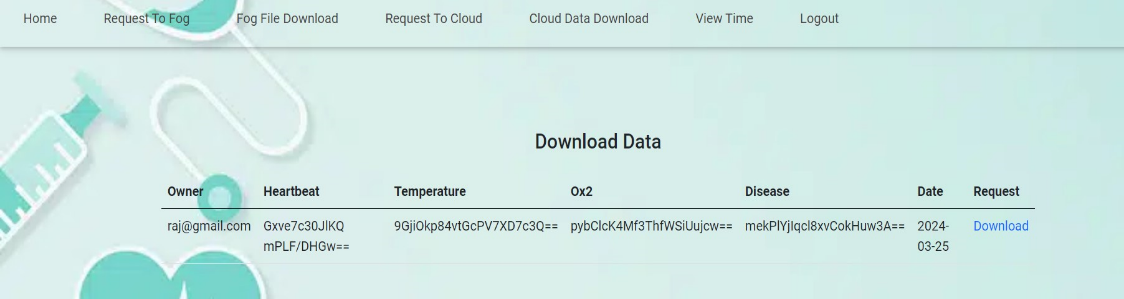


**Figure 4.3.3. Send a request to Fog Framework**

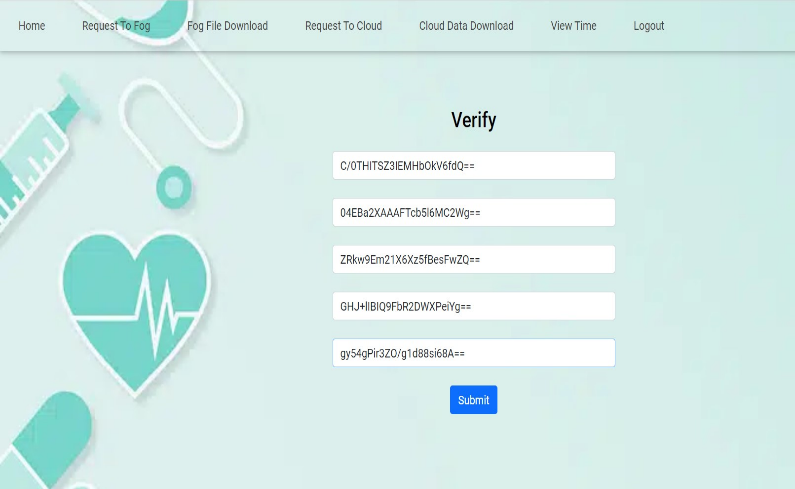
****

**Figure 4.3.4. Verify Request and send Keys To mail**

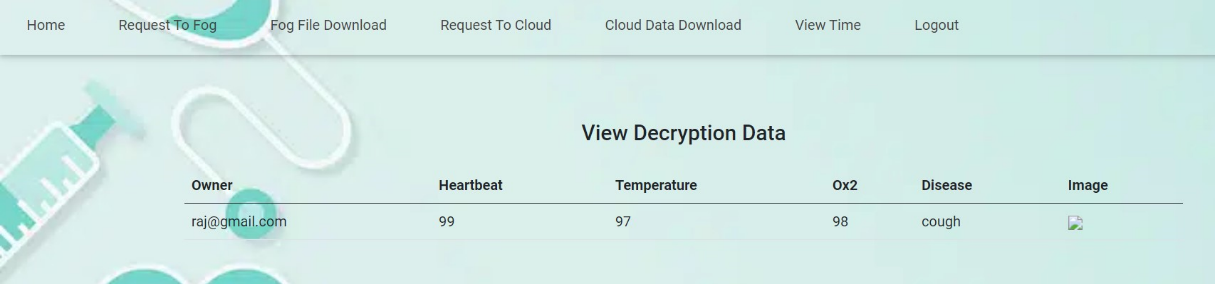
**Figure 4.3.5. Keys sent to mail**

****

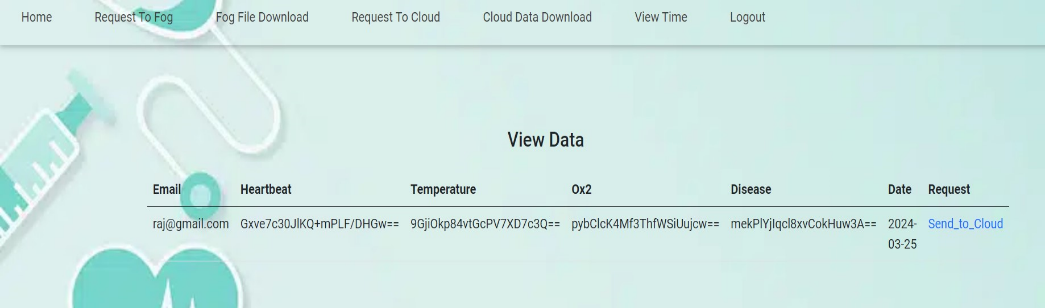
**Figure 4.3.6. Download the data**

****

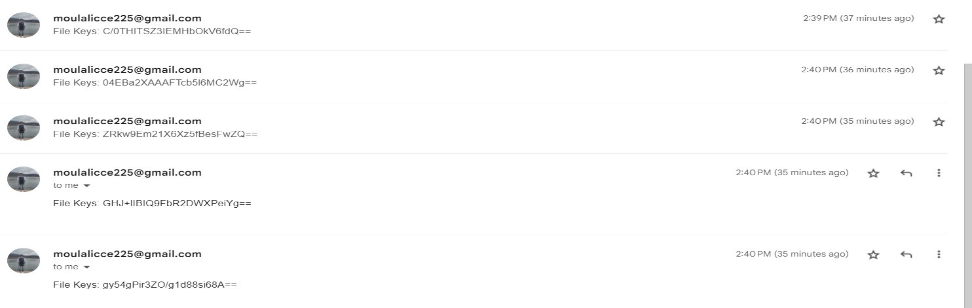
**Figure 4.3.7. Verify the Keys**

****

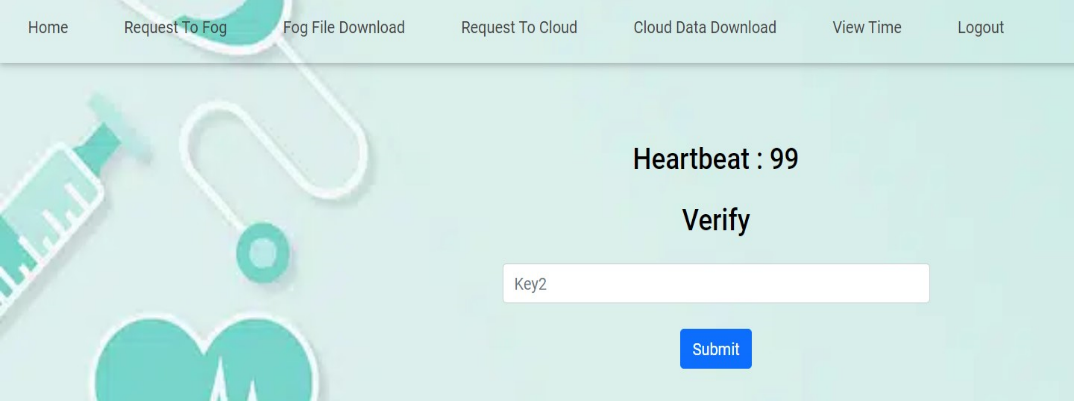
**Figure 4.3.8. View Decrypted data**

****

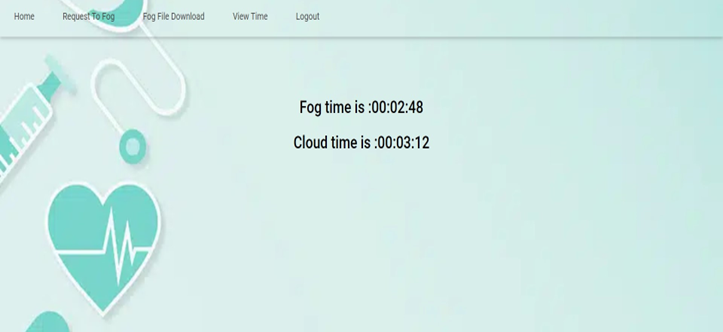
**Figure 4.3.9. Send request To Cloud Framework**

****

**Figure 4.4.0. Mail sent By Cloud Framework**

****

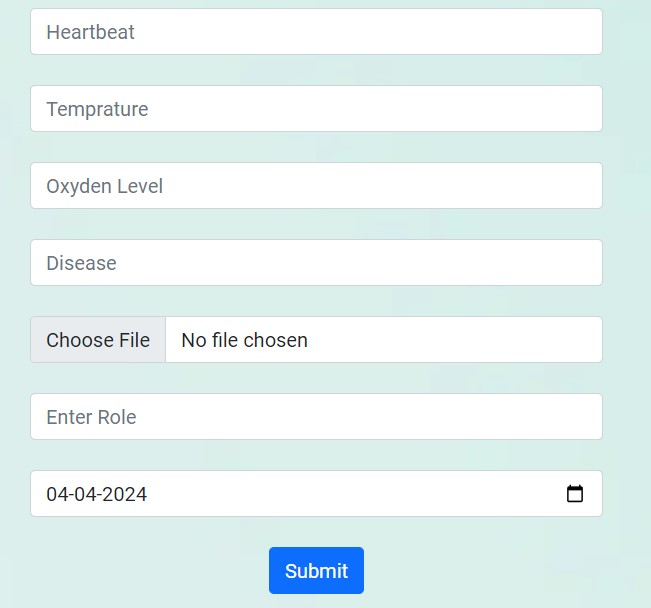
**Figure 4.4.1. Download Each Parameter**

****

**Figure 4.4.2. Total Process time**

**4.4. Dataset Description**

The dataset comprises forms and attached files sourced from esteemed healthcare databases, emphasizing key health parameters such as Heartbeat, Temperature, Oxygen Level, and Disease identification. These records encapsulate encrypted patient data, ensuring confidentiality and security during transmission. Each health parameter is individually encrypted, allowing for secure, targeted access to specific data points as required. The dataset includes a detailed log of the encryption and transmission process, specifically noting the time taken to send encryption keys to the authorized recipient. This aspect of the dataset enables analysis of the efficiency and timeliness of secure data sharing in healthcare communication channels. The comprehensive encryption strategy employed guarantees that patient information remains protected, adhering to strengthen privacy standards while facilitating critical data exchange for healthcare professionals.

**Figure.4.4.3. Data Collected**

1. **Experimental Setup**

Used **NetBeans and MYSQL** to develop this Secure smart healthcare data with Data Aggregation. **NetBeans** will be used to Develop the development of web applications that use the Java and HTML5 platforms, and **MYSQL** will be used to create a relational database management system (RDBMS). MySQL is ideal for both small and large applications.

**5.1. Setup NetBeans:**

To Install NetBeans, you need to follow these steps:

**Prerequisites**

Ensure you have a compatible version of the Java Development Kit (JDK) installed on your computer. NetBeans requires JDK to run.

Verify your system requirements to make sure your computer meets the minimum specifications for NetBeans.

Step 1: Download NetBeans IDE

1.Visit the official NetBeans site: [https://netbeans.apache.org/download/index.html](https://netbeans.apache.org/download/index.html).

2. Choose the version that suits your development needs. If you're primarily interested in Java development, the Java SE bundle would be appropriate.

3. Click on the download button for the chosen version to start the download process.

Step 2: Install Java Development Kit (JDK)

NetBeans requires Java Development Kit (JDK) to be installed on your computer.

1. Check if you already have JDK installed by opening a terminal or command prompt and typing `java -version`. If you get a version number, JDK is installed.

2. If you don't have JDK installed, download it from Oracle's website: [https://www.oracle.com/java/technologies/javase-jdk15-downloads.html](<https://www.oracle.com/java/technologies/javase-jdk15-downloads.html>) or adopt OpenJDK: [https://adoptopenjdk.net/](https://adoptopenjdk.net/).

3. Follow the installation instructions on the website to install JDK.

Step 3: Install NetBeans IDE

1. Locate the downloaded NetBeans installer file (it should have an extension `.exe` for Windows).

2. Double-click the installer file to begin the installation process.

3. Follow the on-screen instructions. You will be asked to accept the license agreement, select the installation path, and choose which components to install.

4. The installer will also ask you to specify the JDK installation path. If JDK was correctly installed in the previous step, the installer should automatically detect it. Otherwise, you may need to browse and select the JDK installation directory manually.

5. Click on the Install button to start the installation.

Step 4: Launching NetBeans IDE

1. Once the installation is complete, you can launch NetBeans IDE from the Start menu (on Windows) or through the Applications menu (on macOS or Linux).

2. When you first start NetBeans, it might take a few moments to initialize. You might also be prompted to import settings from a previous version, if applicable.

Step 5: Configuring NetBeans IDE (Optional)

1. You can customize NetBeans to suit your preferences. Go to Tools -> Options to open the Options dialog.

2. Here, you can configure the IDE's appearance, editor settings, fonts, colors, and more.

3. You can also install additional plugins by going to Tools -> Plugins. This allows you to extend the functionality of NetBeans to support more programming languages or frameworks.

Step 6: Create a New Project

1. To start a new project, go to File -> New Project.

2. Select the type of project you want to create (e.g., Java Application, PHP Application, etc.).

3. Follow the project creation wizard, and specify the project name, location, and any other required settings.

**5.2. Setup MYSQL:**

To install and set up **MYSQL**, you can follow these steps:

1. Download MySQL Installer: Visit the official MySQL website and download the MySQL Installer appropriate for your operating system (Windows, macOS, or Linux).

2. Run the Installer: Once the installer is downloaded, run it and follow the on-screen instructions. Choose the option to install MySQL Server, which will also include other necessary components like MySQL Workbench.

3. Select Installation Type: During the installation process, you'll be prompted to choose an installation type. Select the option that best suits your needs, such as Developer Default, Server Only, or Custom.

4. Configure MySQL Server: After selecting the installation type, you'll need to configure MySQL Server. Set a root password for the MySQL Server, which will be required for administrative tasks.

5. Choose Start Menu Folder: Specify the Start Menu folder where MySQL shortcuts will be placed. You can choose the default location or customize it according to your preference.

6. Complete the Installation: Proceed with the installation process, allowing the installer to complete the installation of MySQL Server and associated components.

7. Verify Installation: Once the installation is complete, verify that MySQL Server is running correctly. You can do this by opening MySQL Workbench or using the command-line interface and logging in with the root account and the password you set during installation.

**5.3. Libraries Used:**

**5.3.1. JSP (JavaServer Pages) codes**

**1. Java I/O Libraries:**

- `java.io.InputStreamReader`: InputStreamReader is used for reading bytes and decoding them into characters using a specified charset.

- `java.io.BufferedReader`: BufferedReader reads text from a character-input stream, buffering characters to provide efficient reading of characters, arrays, and lines.

- `java.io.InputStream`: InputStream is an abstract class representing an input stream of bytes.

**2. Java SQL Libraries:**

- `java.sql.\*`: This package contains interfaces and classes for JDBC (Java Database Connectivity) API, which enables Java programs to interact with databases.

**3. Custom Database Connection Class:**

- `novelefficient.Dbconnection`: This appears to be a custom class for establishing a database connection. It likely contains methods for connecting to the database using JDBC and managing database connections.

**4. Session Management Directive:**

- `<%@ page session="true" %>`: This directive indicates that session management is enabled for the JSP page, allowing you to access session attributes and maintain user sessions across multiple requests.

**5. HTML, CSS, JavaScript Libraries:**

- `templatemo\_style.css`: This CSS file is used for styling the HTML elements of the web page, providing visual layout and design.

- `css/ddsmoothmenu.css`: This CSS file is likely used for styling dropdown menus, providing a smooth navigation experience.

- `js/jquery.min.js`: This JavaScript library is jQuery, a popular JavaScript framework used for simplifying client-side scripting and DOM manipulation.

- `js/ddsmoothmenu.js`: This JavaScript file contains code for initializing and configuring the Smooth Navigational Menu, a dropdown menu plugin based on jQuery.

- `js/jquery.ennui.contentslider.css`: This CSS file styles the content slider plugin used for displaying sliding content sections on the web page.

- `js/jquery.easing.1.3.js`: This JavaScript file contains easing functions for smooth animation transitions used by the content slider.

- `js/jquery.ennui.contentslider.js`: This JavaScript file contains the code for the content slider plugin used on the web page.

**5.3.2. Encryption & Decryption**

**1. `com.sun.org.apache.xerces.internal.impl.dv.util.Base64`:**

- This library provides utility methods for encoding and decoding data using the Base64 encoding scheme. Base64 encoding is commonly used to represent binary data as ASCII text, making it suitable for tasks like encoding binary data for transmission over text-based protocols or storing binary data in text-based formats.

**2. `java.io.ByteArrayOutputStream`:**

- The `ByteArrayOutputStream` class from the `java.io` package is used for writing binary data into a byte array. It's often used in conjunction with other stream classes to collect binary data in memory before further processing, such as encryption or serialization.

**3. `java.io.FileInputStream`:**

- This library is used to read data from files. The `FileInputStream` class from the `java.io` package reads bytes from a file and is often used to supply data to processes that require input from files, such as encryption or reading configuration files.

**4. `java.io.FileWriter`:**

- The `FileWriter` class from the `java.io` package is used for writing character-oriented data to files. While not directly related to encryption, it could be used to log or write encrypted data to a file for storage or further processing.

**5. `java.util.Scanner`:**

- The `Scanner` class from the `java.util` package is used for parsing primitive types and strings from input streams. It can be helpful for reading input from various sources, including files or user input, which may be necessary for providing data to encryption processes.

**6. `javax.crypto.Cipher`:**

- The `Cipher` class from the `javax.crypto` package is a fundamental class in the Java Cryptography Architecture (JCA). It provides encryption and decryption capabilities and supports various cryptographic algorithms. In this context, it's used to perform AES encryption on data.

**7. `javax.crypto.KeyGenerator`:**

- The `KeyGenerator` class from the `javax.crypto` package is used to generate secret keys for symmetric encryption algorithms like AES. It provides a convenient way to generate random keys of the appropriate length for use in encryption operations.

**8. `javax.crypto.SecretKey`:**

- The `SecretKey` interface from the `javax.crypto` package represents a cryptographic secret key used for encryption and decryption. It's a common interface for handling symmetric keys generated by `KeyGenerator` or created from raw key bytes.

**9. `javax.crypto.spec.SecretKeySpec`:**

- The `SecretKeySpec` class from the `javax.crypto.spec` package provides a way to create a `SecretKey` from raw key bytes. It's useful for specifying keys in encryption and decryption operations when the key material is known in advance.

**10. `javax.swing.JOptionPane`:**

- The `JOptionPane` class from the `javax.swing` package is used to create dialog boxes for displaying messages or prompting the user for input. While not directly related to encryption, it could be used for displaying messages or interacting with the user during encryption operations.

**11. `sun.misc.BASE64Encoder`:**

- The `BASE64Encoder` class from the `sun.misc` package is used to encode binary data as a Base64-encoded string. Base64 encoding is commonly used for representing binary data as text, making it suitable for scenarios where binary data needs to be transmitted or stored as text.

**12. sun.misc.BASE64Decoder:**

The BASE64Decoder class from the sun.misc package is used to decode Base64-encoded data into its original binary form. It complements the Base64 encoding functionality provided by the Base64 class, enabling the decoding of Base64-encoded strings back into their original binary representation.

**5.3.3. Database Connection**

**1.java.sql.Connection:**

This class represents a connection to a database. It is part of the JDBC (Java Database Connectivity) API and is used to establish a connection to a database server.

**2.java.sql.DriverManager:**

The DriverManager class is part of the JDBC API and is used to manage a set of JDBC drivers. It provides methods for registering drivers, establishing database connections, and controlling the drivers' behavior.

**3.java.lang.Class:**

This class is part of the core Java API and is used for obtaining information about classes loaded in the Java Virtual Machine (JVM). In this context, it's used to load the MySQL JDBC driver dynamically at runtime.

**4.com.mysql.jdbc.Driver:**

This class is the MySQL JDBC driver implementation provided by MySQL. It allows Java applications to connect to MySQL databases using JDBC. The Class.forName() method is used to load this driver class dynamically.

**5.4. Parameters**

* Time Attribute Access Control:

Time attribute access control is a security model that uses time as a critical factor in deciding whether a user can access a specific resource. This decision-making process takes into account various conditions related to the user, the resource in question, and the situational context. The unique aspect of this model is the dynamic adjustment of access permissions based on changing conditions over time, which may include:

Priority: Reflects the importance of the resource or user at a given time.

Congestion: Indicates the level of busyness or demand for the resource.

Criticality: Measures the urgency of the access request based on situational factors.

These labels (priority, congestion, and criticality) are not static; they adapt over time and according to different situations. The access control system continuously evaluates these factors to determine the current state of users, resources, and environmental conditions, thereby making informed decisions on granting or denying access. This method is innovative because it integrates the temporal dynamics of the environment, which most traditional access control models do not consider.

* Total Process Time Calculation

When calculating the total process time, especially in the context of email transactions, it's crucial to consider all the components that contribute to the delay from the request initiation to the final email delivery. The formula provided is:

T\_{total} = T\_{processing} + T\_{email sending} + T\_{email delivery}

(T\_{processing}): This is the time taken by the server to process the request. It includes all the operations required to prepare for sending the email, such as generating any necessary keys, formatting the email content, and other pre-send processing tasks.

(T\_{email sending}): Refers to the time taken by the email server specifically to send the email out from the server to the recipient's email server. This step is distinct from the processing time, focusing solely on the act of sending the email.

(T\_{email delivery}): This is the time from when the email is sent until it is received by the recipient. This duration can vary significantly based on several factors, including the recipient's email server processing time, network conditions, and any filters or rules that may delay delivery.

To accurately calculate the total process time, one must individually measure each component and sum them up. This calculation helps in identifying bottlenecks and understanding the efficiency of the email communication process.

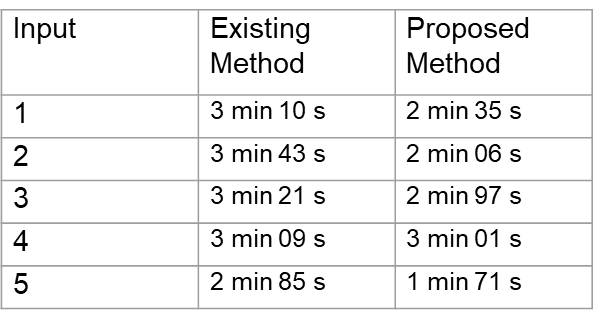
Implementing time attribute access control effectively requires a deep understanding of the dynamic nature of the factors involved and the ability to measure and analyze process times accurately. This approach not only enhances security by adapting to the current context but also helps in optimizing processes by identifying and mitigating delays.

1. **Discussion of Results**

The implementation of a fog-based framework for healthcare data processing, as proposed, yields significant efficiency and security enhancements over traditional cloud-based solutions. The innovative approach to data handling, which integrates a time-based access control for file retrieval, ensures a more secure yet accessible system for managing sensitive healthcare information. Notably, the method's ability to process data at the edge, closer to where it is generated, substantially reduces the overall time required for data transmission and retrieval. This is crucial in healthcare settings where timely access to patient data can be critical. Furthermore, the fog framework's expedited process for sending keys presents a clear advantage over cloud-based alternatives, demonstrating its effectiveness in facilitating quicker, more secure data access.

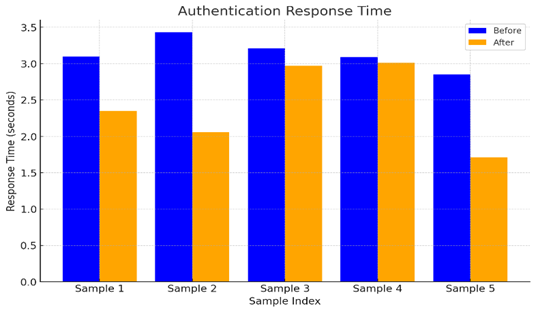
The layered structure of the system, encompassing patient, cloud framework, fog framework, data receiver, and trust framework modules, creates a comprehensive and secure data handling ecosystem. Patients and data owners benefit from a secure platform to upload and encrypt medical data, with the assurance that their information remains confidential through batch file encryption. The cloud framework's role in secure storage and key management complements the fog framework's efficient data and key distribution capabilities, ensuring data is both secure and readily accessible to authorized users. The inclusion of a trust framework further enhances the system's integrity, providing a robust mechanism to verify user permissions and access rights. Overall, the results indicate that the proposed fog-based framework not only minimizes the time taken for data processing and retrieval but also strengthens data security and accessibility, making it a valuable solution for modern healthcare data management challenges.

Table I displays the Total Process Time for different Users applied to a given set of Information and attachment. Examining the table, it is evident that the the proposed method consistently shows significantly faster key transmission times across all five measured instances



**Table 6.1. Total Process Time of Different Users**

The Total Process Time for all users is illustrated in Fig. 2.



**Figure 6.1. Total Process Time**

Table II displays the Comparison of Time Taken between Fog based frame work and cloud based frame work. Examining the table, it is evident that the fog based frame work takes less time than cloud based frame work. The total time taken to download patient data varies with each attempt. In most cases, the process takes several seconds, with some instances showing a notably faster download time.

|  |  |  |
| --- | --- | --- |
| Input | Fog based framework | Cloud-based framework |
| 1 | 2.3 | 1.35 |
| 2 | 1.2 | 0.35 |
| 3 | 3.7 | 2.3 |
| 4 | 5.6 | 2.5 |
| 5 | 6.2 | 2.3 |

**Table 6.2. fog frame work Vs cloud Frame work**

The Comparison of Time Taken between Fog based frame work and cloud based frame work are illustrated in Fig. 3.

**Figure. 6.2. Recall score of models**

**7. Summary, Conclusion and Recommendation**

The proposed fog-based framework for healthcare data processing demonstrates a significant advancement in the management and security of health data. Through the implementation of this framework, healthcare data is processed at the edge, closer to the point of data collection, which markedly reduces the time required for data processing compared to traditional cloud-based systems. This approach not only accelerates the accessibility of health data for medical professionals but also enhances the overall efficiency of healthcare delivery. The framework employs a time-based access control mechanism for file retrieval, ensuring that data remains secure yet promptly accessible within set time limits. This feature is crucial for maintaining the confidentiality of patient information while providing timely access to healthcare providers for informed decision-making.

Furthermore, the method of data encryption and key management presented in the framework addresses the critical challenge of securing sensitive health parameters such as heart rate, blood pressure, body temperature, and body weight. By encrypting each parameter with a unique key and then transmitting the encrypted data to the fog framework, the system ensures a high level of data security. The generation of an aggregated key for decrypting the data streamlines the process, making it more efficient than the segmented key distribution and retrieval process typical of cloud-based frameworks. This method not only simplifies the decryption process but also significantly reduces the time taken to access critical health data.

The comparative analysis between the fog framework and traditional cloud-based alternatives highlights the efficiency of the fog framework in terms of key transmission and data access times. The fog framework's ability to aggregate keys and transmit a single key for decrypting all health parameters at once presents a clear advantage over cloud frameworks, which require multiple keys sent at different times. This efficiency is particularly evident in the reduced download time of keys sent to users, where the fog server provides a single key for all data, enabling faster and more efficient data retrieval.

Based on these findings, it is recommended that healthcare systems consider the adoption of fog-based frameworks for data management and security. The enhanced processing speed, security features, and efficiency of data access provided by the fog framework can significantly improve healthcare delivery and patient care. Additionally, further research and development into optimizing the fog framework for healthcare applications could uncover additional benefits and efficiencies, potentially setting a new standard for digital health data management. This shift towards fog computing could be instrumental in addressing the evolving needs of the healthcare sector, offering a more responsive, secure, and patient-centered approach to healthcare information technology.

**8. Future Enhancements**

For future enhancements of the fog-based healthcare framework, there's potential to integrate advanced machine learning algorithms and artificial intelligence (AI) to further personalize and improve patient care. By analyzing real-time health data collected at the edge, these technologies can predict health issues before they become severe, enabling preemptive medical intervention. Additionally, incorporating blockchain technology could enhance data security and patient privacy by creating an immutable ledger for health data transactions, ensuring tamper-proof storage and sharing of sensitive information. Improvements in interoperability between different healthcare systems and devices are also crucial, allowing for seamless data exchange and a more unified healthcare ecosystem. These advancements promise to make healthcare delivery more efficient, secure, and tailored to individual patient needs, paving the way for a new era in smart healthcare.

**9. Reference**

[1]  Huadong LiU , OPERA: Optional Dimensional Privacy-Preserving Data Aggregation for Smart Healthcare Systems, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 19, NO. 1, JANUARY 2023 (857)

[2]  M. I. Pramanik, R. Y. Lau, H. Demirkan, and M. A. K. Azad, “Smart health: Big data enabled health paradigm within smart cities,” Expert Syst. Appl., vol. 87, no. 2017, pp. 370–383, 2017

[3]  Q. Cai, H. Wang, Z. Li, and X. Liu, “A survey on multimodal data-driven smart healthcare systems: Approaches and applications,” IEEE Access, vol. 7, pp. 133583–133599, 2019.

[4]  R. Saha et al., “Internet-of-Things framework for oxygen saturation monitoring in COVID-19 environment,” IEEE Internet of Things J., vol. 9, no. 5, pp. 3631–3641, Mar. 2022.

[5]  N. Nasser, Q. Emad-ul Haq, M. Imran, A. Ali, I. Razzak, and A. Al-Helali, “A smart healthcare framework for detection and monitoring of COVID-19 using IoT and cloud computing,” Neural Comput. Appl., vol. 7, pp. 1–15, 2021.

[6]  N. Mohammadzadeh, M. Gholamzadeh, S. Saeedi, and S. Rezayi, “The application of wearable smart sensors for monitoring the vital signs of patients in epidemics: A systematic literature review,” J. Ambient Intell. Humanized Comput., vol. 1, 2020, Art. no. 1.

[7]  Z. Ahmadi, M. H. Kashani, M. Nikravan, and E. Mahdipour, “Fog-based healthcare systems: A systematic review,” Multimedia Tools Appl., vol. 80, pp. 36361–36400, Dec. 2021.

[8] C. Guo, P. Tian, and K. K. R. Choo, “Enabling privacy-assured fog-based data aggregation in E-healthcare systems,” IEEE Trans. Ind. Informat., vol. 17, no. 3, pp. 1948–1957, Mar. 2021.

[9] L. J. Laffin et al., “Rise in blood pressure observed among US adults during the COVID-19 pandemic,”Circulation, vol. 145, no. 3, pp. 235–237, 2022.

[10]  R. D. Brook, “The environment and blood pressure,”Cardiol. Clin., vol. 35, no. 2, pp. 213–221, 2017.

[11]  O. R. Merad-Boudia and S. M. Senouci, “An efficient and secure multidimensional data aggregation for fog-computing-based smart grid,” IEEE Internet of Things J., vol. 8, no. 8, pp. 6143–6153, Apr. 2021.

[12] Z. Xia, Y. Zhang, K. Gu, X. Li, andW. J. Jia, “Secure multi-dimensional and multi-angle electricity data aggregation scheme for fog computing-based smart metering system,” IEEE Trans. Green Commun. Netw., vol. 6, no. 1, pp. 313–328, Mar. 2022.

[13] X. Zuo, L. Li, H. Peng, S. Luo, and Y. Yang, “Privacy-preserving multidimensional data aggregation scheme without trusted authority in smart grid,” IEEE Syst. J., vol. 15, no. 1, pp. 395–406, Mar. 2021.

[14]  A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.

[15]  C. C. Yang, T. Y. Chang, and M. S. Hwang, “A (t, n) multi-secret sharing scheme,” Appl. Math. Comput., vol. 151, no. 2, pp. 483–490, 2004.

[16] I. Khan, “Thousands of Coinbase wallets drained by hackers,” Tom’s Guide, Oct. 2021. Accessed: Jan. 20, 2022. [Online]. Available: https://www.tomsguide.com/news/coinbase-hacked-6000-accountssee-cryptocurrency-drained

[17]  M. O’Brien, E. Tucker, K. Freking, and B. Ortutay, “Experts say Twitter breach troubling, undermines trust,” FOX 13 News, Jul. 2020. Acessed: Jan. 20, 2022. [Online]. Available: https://www.fox13news.com/news/ experts-say-twitter-breach-troubling-undermines-trust

[18] J. Daemen and V. Rijmen, “The advanced encryption standard process,” in The Design of Rijndael. Berlin, Germany: Springer, 2020, pp. 1–8.

[19] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn., 1999, pp. 223–238.

[20]  L. S. Hughes, R. L. Phillips, J. E. DeVoe, and A. W. Bazemore, “Com munity vital signs: Taking the pulse of the community while caring for patients,” J. Amer. Board Fam. Med., vol. 29, pp. 419–422, May 2016.

[21] J.Xu,G.Yang,Z.Chen,andQ.Wang,“Asurveyontheprivacy-preserving data aggregation in wireless sensor networks,” China Commun., vol. 12, no. 5, pp. 162–180, 2015.

[22] S. Han, S. Zhao, Q. Li, C. H. Ju, and W. Zhou, “PPM-HDA: Privacy preserving and multifunctional health data aggregation with fault toler ance,” IEEE Trans. Inf. Forensics Secur., vol. 11, no. 9, pp. 1940–1955, Sep. 2016.

[23] Y. Liu, W. Guo, C. I. Fan, L. Chang, and C. Cheng, “A practical privacy preserving data aggregation (3PDA) scheme for smart grid,” IEEE Trans. Ind. Informat., vol. 15, no. 3, pp. 1767–1774, Mar. 2019.

.

[24]  C. Peng, M. Luo, P. Vijayakumar, D. He, O. Said, and A. Tolba, “Multi functional and multi-dimensional secure data aggregation schemes in WSNs,”IEEEInternetofThingsJ.,vol.9,no.4,pp. 2657–2668,Feb.2022..

[25]  H. Shen, M. Zhang, and J. Shen, “Efficient privacy-preserving cube-data aggregation scheme for smart grids,” IEEE Trans. Inf. Forensics Secur., vol. 12, no. 6, pp. 1369–1381, Jun. 2017.

[26] H. Liu, T. Gu, Y. Liu, J. Song, and Z. Zeng, “Fault-tolerant privacy preserving data aggregation for smart grid,” Wireless Commun. Mobile Comput., vol. 2020, 2020, Art. no. 8810393.

[27] R. K. Pathinarupothi, P. Durga and E. S. Rangan, "IoT-Based Smart Edge for Global Health: Remote Monitoring with Severity Detection and Alerts Transmission," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2449-2462, April 2019

[28] K. Sowjanya, M. Dasgupta, and S. Ray, “An elliptic curve cryptography

based enhanced anonymous authentication protocol for wearable health

monitoring systems,” International Journal of Information Security,

vol. 19, no. 1, pp. 129–146, 2020.

[29] R. Saha, G. Geetha, G. Kumar, T.-H. Kim, and W. J. Buchanan,

“Mrc4: A modified RC4 algorithm using symmetric random function

generator for improved cryptographic features,” IEEE Access, vol. 7,

pp. 172045–172054, 2019.

[30] H. Qiu, M. Qiu, M. Liu, and G. Memmi, “Secure health data sharing for

medical cyber-physical systems for the healthcare 4.0,” IEEE J. Biomed.

Health Informat., vol. 24, no. 9, pp. 2499–2505, Sep. 2020