Securing Cloud Storage for Sensitive Medical Data: A File Hosting Solution with Encryption and Access Control

Iyad Najmi Bin Ghazali Yusri1, a), Palanichamy Naveen1, 2, b), Su-Cheng Haw1, 2, c), V. Gomathi3, d) and Prabha Kumaresan1, 4, e)

1Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100, Cyberjaya, Malaysia

2Centre for Digital Innovations, CoE for Immersive Experience, Multimedia University, Persiaran Multimedia, 63100, Cyberjaya, Malaysia.

3Department of Electrical and Electronics Engineering, College of Engineering Guindy, Anna University, Chennai, India

4Center of Natural Language Processing (NLP), CoE for Artificial Intelligence, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Malaysia.

*b) Corresponding author: p.naveen@mmu.edu.my  
a) 1201203090@student.mmu.edu.my*

*c) sucheng@mmu.edu.my*

*d) gomesceg@gmail.com*

*e) prabha.kumaresan@mmu.edu.my*

**Abstract.** As healthcare services continue to digitise, the need for secure and reliable data management platforms, especially within cloud environments, has become more pressing. In Malaysia, the Personal Data Protection Act (PDPA) governs the handling of personal information, but it does not specifically address the protection of sensitive medical data in cloud-based systems. Referencing international standards like HIPAA (Health Insurance Portability and Accountability Act) and PDPA (Personal Data Protection Act) this paper introduces the design and implementation of a secure cloud-based file hosting system tailored for the Malaysian healthcare context. The system is mainly built using FastAPI for the backend, AWS’ S3 for secure file storage, and implements Role-Based Access Control (RBAC) across the three user roles: Patient, Practitioner, and Admin. Patients maintain full ownership of their files, with the ability to grant or revoke access to healthcare practitioners. File permissions are managed through token-based authentication and monitored using an integrated audit log system. The system also utilizes pre-signed URLs to secure file downloads, as well as SMTP-based email notifications for permission requests and informing users in the case of a data breach. This paper outlines the implementation of the system and demonstrates how the system achieves regulatory compliance while maintaining a cloud system architecture. By bridging the present gaps in PDPA through reference to HIPAA, the proposed system serves as a prototype for patient-centric medical file hosting platforms in emerging digital healthcare ecosystems.

# Introduction

In today's times, the use of digital information systems is becoming more widespread in our day-to-day life. Particularly in healthcare, there is an increasing demand and reliance on digital electronic health information systems to manage personal health records of patients. Moreover, with the introduction of cloud computing technologies, the way medical institutions store and manage patients’ medical records has changed. However, with the digitalization of such sensitive personal records, the importance of appropriate safeguarding has become more apparent. Without the appropriate safeguarding, sensitive information is under threat from adversaries such as unauthorized access to data, illegal tampering, as well as data breaches [1]. Ignoring these threats could lead to major consequences such as the loss of the patient's trust, legal and compliance issues, as well as reputational damage.

In hopes to address these challenges, regulatory frameworks such as the HIPAA, which stands for Health Insurance Portability and Accountability Act, were enacted in 1996 by the United States of America [2]. On the other hand, locally in Malaysia, the Personal Data Protection Act (PDPA) was established in 2015 [3]. These acts and their regulations set the standard for handling sensitive personal data. They outline the strict requirements that are required to be adhered to in modern systems. However, on the other hand, currently many existing cloud-based storage systems lack the specialized security requirements needed to secure medical data, particularly in the Malaysian context, as PDPA is still evolving [4].

Studies have explored many different strategies to secure sensitive medical data in cloud environments, for example, using AES encryption along with digital envelopes to secure data transmission and storage. However, many cloud solutions still fail to meet the data protection standards, especially in Malaysia, where HIPAA is not legally compulsory. Most cloud storage systems focus on general file storage rather than sensitive medical data, with many security features lacking such as audit logging, RBAC, and breach notification. This paper outlines the design and implementation of a secure cloud-based file hosting system for medical data. The objectives of this project are:

* Implement secure cloud storage: To develop a cloud-based file hosting application that stores sensitive medical data, utilizing strong encryption algorithms to ensure protection in both transit and at rest.
* Implement robust access control: To design and implement row-based access control on the system, ensuring that only authorized users can access and manage certain data.
* Ensure regulatory compliance: To ensure that the system complies with current healthcare data protection standards like HIPAA and personal data standards such as Malaysia's PDPA.

This research was conducted to contribute to the evolving field of cybersecurity, particularly in fields such as healthcare systems. The system also hopes to close the gap between current international standards, such as HIPAA, as well as Malaysia's national data protection acts, PDPA, for cloud-based medical data systems.

# Literature review

With the growing demand for data storage, the healthcare industry has shifted from traditional local systems to cloud-based systems to manage health records. However, this shift introduces many challenges in securing sensitive data, while also maintaining privacy and regulatory compliance. Standards like HIPAA and PDPA play a major role in setting a bar for data security practices for personal data. This literature review covers past research on technologies that were implemented to secure personal data, many of which include concepts such as encryption, access control methods, and compliance strategies.

Cryptographic methods for securing healthcare data are critical for protecting sensitive patient information [6]. AES-256 stands as one of the staple encryption algorithms for data, widely adopted in modern information systems all around the world. In Paper 4, AES encryption was used alongside digital envelopes in a seven-layer cloud framework to secure sensitive data [5]. With it, the research shows significant improvements in execution time and memory usage. Similarly, AES was also used in tandem with the Elliptic Curve Diffie-Hellman (ECDH) for client-side encryption, achieving a secure and efficient way of retrieving medical data [5].

Role-based access control (RBAC) remains one of the most foundational methods in access management for many information systems that handle sensitive information. In Paper 5, RBAC was implemented alongside attribute-based access control to propose their secure healthcare access control system (SHACS), which aims to enhance data protection in healthcare systems [7]. By utilizing this hybrid approach, the system reduced the authentication time by 33%, still being aligned with compliance requirements. In addition, in Paper 8, the author highlighted the effectiveness of using RBAC with encryption and intrusion detection systems in the Advanced Integrated Data Security (AIDS) model [8].

Audit logging and breach detection mechanisms play a major role in complying with regulatory standards. They are also necessary to help with forensic investigation in the case of data breach. The breach notification rule outlined in HIPAA regulations requires a breach detection mechanism. The rule mandates that in the case of a data breach, the affected individuals and parties must be notified within 60 days of the data breach [2]. Additionally, the data integrity principle in PDPA regulations also necessitates audit logging to maintain accountability as well as transparency [3].

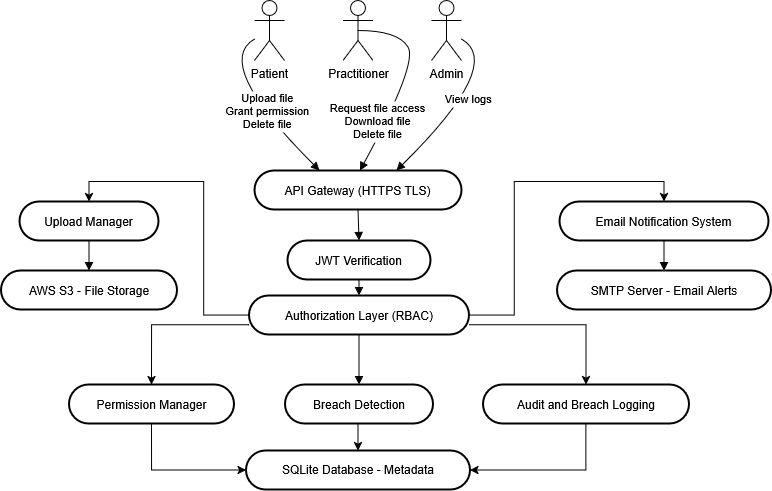
HIPAA sets strict standards for healthcare organizations and cloud service providers, particularly around data encryption, access control, and breach notifications [2]. While PDPA complements this in the Malaysian context, emphasizing user consent, data localization, and safe data handling practices [3]. Kumar et al. and Madavarapu et al. both emphasized encryption and RBAC in their papers as a fundamental aspect to adhere to international health data regulations [4], [8]. In addition, paper [9] also emphasizes the need for multi-layered security mechanisms to protect sensitive medical data, utilizing encryption and strict access controls. The system proposed in this study incorporates those same practices: AES-256 encryption, TLS 1.3, RBAC, audit trails, and real-time breach notifications. Ensuring alignment with both HIPAA and PDPA requirements.

In conclusion, HIPAA sets strict standards for organizations and cloud service providers, particularly around the protection, access control, and breach notifications [2]. PDPA complements this, emphasizing user consent, data localization, and safe data handling practices [3]. In Paper 3 and Paper 6, the authors both emphasized encryption and RBAC in their papers as a fundamental aspect to adhere to international health data regulations and standards [4], [8]. The system proposed in the study incorporates those fundamental practices: AES-256 encryption, TLS 1.3, RBAC, audit trails, and real-time breach notifications; ensuring alignment with both HIPAA and PDPA requirements.

The review highlights that although several prior systems implement encryption methods and access control, there are still few that integrate them alongside other practices such as audit logs and user consent mechanisms. This project aims to fill that gap by proposing a system that integrates these components into a unified platform for managing sensitive medical data securely in the cloud.

# Methodology

MedSafe is designed as a secure cloud-based system to manage medical file uploads, access control, and audit logging for patients and practitioners. Built using FastAPI and SQLite, it uses JWT authentication [10], RBAC authorization, and AWS S3 for encrypted file storage [11]. The system architecture illustrated in Figure 1 includes a clear flow from user interfaces through secure APIs, logic layers, and into storage and monitoring services. Table 1 outlines the layers in the system’s architecture.



**FIGURE 1.** System architecture

|  |  |
| --- | --- |
| **TABLE 1.** System architecture layers | |
| **Layer** | **Description** |
| Presentation Layer | Interface (web/mobile) for patients, practitioners, and admins |
| API Gateway | HTTPS entry point using TLS 1.3 to FastAPI |
| Authentication Layer | Issues and verifies JWT tokens; manages login |
| Authorization Layer | Enforces role-based access at route level using JWT claims |
| Business Logic Layer | Handles permissions, uploads, downloads, notifications |
| Storage Layer | AWS S3 for encrypted files, SQLite for metadata |
| Audit/Breach Logging Layer | Tracks all activity and violations in audit.log/breach.log |
| Notification Layer | Sends permission request alerts via SMTPw |

## Applied Concepts

The design and development of the MedSafe system were informed by technologies, approaches, and challenges identified in the literature review. The following concepts were selected and implemented to fulfill the project's objectives of secure medical data management, while ensuring HIPAA and PDPA compliance. To ensure comprehensive data protection, MedSafe integrates encryption mechanisms for both data at rest and data in transit. Encryption methods were referenced in [4], [5]

* Data at Rest: All uploaded medical files are automatically encrypted using AES-256 server-side encryption (SSE-S3) provided by AWS S3. When patients or practitioners upload files through the /upload endpoint, the system securely stores them in AWS S3 with a private ACL, ensuring confidentiality and integrity of stored data.
* Data in Transit: All communication between users and the MedSafe backend is protected using HTTPS with TLS 1.3. This secures the transmission of sensitive data such as login credentials, authentication tokens, and file metadata.

MedSafe implements RBAC at the API level, ensuring users can only access endpoints appropriate to their assigned role (Patient, Practitioner, or Admin) [7]. JWT tokens are issued upon login, embedding the user’s role, and route access is enforced via dependency injection in FastAPI.

Under PDPA’s regulations, patients can fully control who can access their uploaded files and practitioners must request permission, which patients explicitly grant or deny. Permissions are stored in the SQLite database and are enforced whenever access to medical data is requested [3]. When a practitioner requests access to a patient's medical file, an automatic email notification is sent to the patient.

MedSafe records all sensitive operations, including file uploads, downloads, permission changes, and authentication events. These are stored sequentially in an append-only audit.log file, ensuring tamper-evident tracking for compliance and forensic analysis. This concept was referenced in [7], [8]

Unauthorized access attempts are automatically classified as breach events. These are logged separately in a breach.log file to distinguish between normal operational activities and potential security incidents. Under HIPAA regulations, data breaches and accountability must be recorded and users affected must be notified accordingly [2], [12]. Table 2 outlines how the concepts work in the system, with the flow being from left to right.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TABLE 2.** Concept flows | | | | | |
| **Encryption** | **RBAC** | **Permission** | **Email Notification** | **Audit Logging** | **Breach Detection** |
| User Upload or Download Request | Practitioner Requests Download | Practitioner Requests File Access | Practitioner Access Request | Sensitive System Event | Unauthorized Access Attempt Detected |
| HTTPS (TLS 1.3) | JWT Token Verified | Request Logged | SMTP Triggered | Event Captured | Breach Logger Triggered |
| API Gateway | Role Check (RBAC) | Patient Receives Email | Patient Receives Notification | Entry Appended to audit.log | Entry Appended to breach.log |
| Upload Manager | Permission Verification | Patient Grants Access | Patient Acts via System | Admin Views Logs Securely via Logs |  |
| AWS S3 (AES-256 Encrypted Storage) | Access Granted or Denied | Access Rights Updated in SQLite |  |  |  |

# Results and Discussion

Authentication and Authorization: Successful user registration and login were validated through JWT token issuance for Patients, Practitioners, Admins. Access to protected endpoints was correctly restricted based on user roles, with unauthorized access attempts returning 403 errors as shown in Figure 2.

|  |  |
| --- | --- |
|  | |
| (a) | |
|  |  |
| (b) | (c) |

**FIGURE 2**. (a) JWT token issued after successful login. (b) Invalid credential attempt (c) Unauthorized attempt returning 403 error

Secure File Upload and Storage (AWS S3, AES-256): Files uploaded via /upload and /upload-file APIs were securely stored in AWS S3 with AES-256 server-side encryption, ensuring confidentiality at rest. Figure 3 shows the medical files uploaded in the AWS S3 bucket.

A black screen with green and white text

AI-generated content may be incorrect.

**FIGURE 3**. Upload API success and encrypted file in AWS S3

Permission Management and Secure File Download: Practitioners could only download patient files after explicit permission was granted through the request-approval flow or through patient consent as shown in Figure 4. Unauthorized download attempts were blocked, and breach logs were generated, while approved accesses generated secure pre-signed URLs as shown in Figure 5.

|  |  |
| --- | --- |
| A computer screen with green text  AI-generated content may be incorrect. | A black rectangle with white text  AI-generated content may be incorrect. |
| (a) | (b) |

**FIGURE 4**. (a) Practitioner finds file by patient email (b) Practitioner sends permission request for file

|  |  |
| --- | --- |
| A computer screen shot of a black screen  AI-generated content may be incorrect. | A black screen with white text  AI-generated content may be incorrect. |
| (a) | (b) |
| A screen shot of a computer  AI-generated content may be incorrect. | |
| (c) | |

**FIGURE 5**. (a) Patient sees permission request. (b) Patients approve permission. (c) Practitioner can now download file

Audit Logging and Breach Detection: All security-sensitive events including uploads, permission changes, and file access were logged in an append-only audit.log as shown in Figure 6 (a). Breach attempts (e.g. unauthorized downloads or route access) were separately recorded in a security.log as seen in Figure 6 (b). This audit infrastructure supports forensic analysis and meets HIPAA/PDPA requirements for post-incident review.

|  |
| --- |
|  |
| (a) |
|  |
| (b) |

**FIGURE 6**. (a) Audit log txt viewed by Admin (b) Security log txt viewed by Admin

## Performance Evaluation

To validate MedSafe's real-time performance, basic benchmarks were recorded into Table 3.

|  |  |
| --- | --- |
| **TABLE 3.** Performance metrics | |
| **Operation** | **Time Measured** |
| File Upload (2 MB) | 4.5 seconds |
| File Upload (13.5 MB) | 8.0 seconds |
| Generate Pre-signed Download Link | 0.5 seconds |

Encryption (AES-256 server-side via AWS S3) added a minor processing overhead of approximately 0.5 to 0.6 seconds per file, depending on file size and system load. These timings suggest the system maintains practical responsiveness for end users.

## Limitations

MedSafe performed well during testing, with key features like access control and secure data handling functioning as intended. There are a few areas that could be improved or expanded upon in future development:

* Scalability: The current use of SQLite is used due to the paid nature of Amazon’s AWS database service, under real usage and deployment, the Medsafe system could expand its toolset to include more AWS integrated services.
* **Real-Time Alerts:** currently, unauthorized access attempts and breach attempts aren’t automatically investigated by the system, it requires manual inspection. Adding breach detection technology to conduct active detection can help improve response to attacks.

## Threat Modeling and Security Resilience

A basic threat model was carried out using the STRIDE framework outlined in Table 4, which evaluates threats under six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS) and Elevation of Privilege [13],[14]. While MedSafe provides a secure baseline through RBAC, audit logging, and strong encryption, it currently lacks full defense-in-depth mechanisms for more advanced threats (e.g., insider attacks, APTs). Future work could implement anomaly detection systems, IP blacklisting and so on to enhance system resilience.

# COnclusion

This paper presented the design and development of MedSafe, a secure cloud-based file hosting system for managing sensitive medical data. The system integration of many mechanisms and concepts such as encryption through AES and TLS, Role-Based Access Control (RBAC), audit logging, and breach notification. In turn, the system successfully addresses the key compliance challenges for cloud services to adhere to HIPAA and PDPA regulations. By providing the patients data sovereignty and enforcing strict access management, MedSafe increases the trust and transparency of cloud-based healthcare systems. The results demonstrate that secure file storage, controlled access, and effective monitoring are much achievable within a scalable cloud architecture, offering a practical model for future healthcare data management solutions in Malaysia and beyond.

|  |  |
| --- | --- |
| **TABLE 4.** STRIDE threat modelling | |
| **Threat Type** | **Mitigation Strategy** |
| Spoofing | JWT-based authentication with strong password policies |
| Tampering | AES-256 encryption for data at rest; TLS 1.3 for transit |
| Repudiation | Comprehensive audit logs with timestamped user actions |
| Information Disclosure | Role-Based Access Control (RBAC), file-specific permissions |
| Denial of Service | Basic request throttling (planned); AWS S3 redundancy |
| Elevation of Privilege | Strict role enforcement and token verification at every route |

# References

1. P.A. Deshpande, P.H. Unki, J. Bhagyashree, and H. Misbanaaz, “Fortified Sharing of Confidential Health Records in Cloud Environment,” in 2024 International Conference on Innovation and Novelty in Engineering and Technology (INNOVA), (IEEE, Vijayapura, India, 2024), pp. 1–5.
2. C. Codington-Lacerte, “Results - OpenURL Connection - EBSCO,” Health Insurance Portability and Accountability Act of 1996., (n.d.).
3. “Standard Perlindungan Data Peribadi 2015 • Perlindungan Data Peribadi,” (2024).
4. K. Pradeep Kumar, B.R. Prathap, M.M. Thiruthuvanathan, H. Murthy, and V. Jha Pillai, “Secure approach to sharing digitized medical data in a cloud environment,” Data Science and Management **7**(2), 108–118 (2024).
5. K. Pradeep Kumar, B.R. Prathap, M.M. Thiruthuvanathan, H. Murthy, and V. Jha Pillai, “Secure approach to sharing digitized medical data in a cloud environment,” Data Science and Management **7**(2), 108–118 (2024).
6. K. Selvakumar, and S. Lokesh, “A cryptographic method to have a secure communication of health care digital data into the cloud,” Automatika **65**(1), 373–386 (2024).
7. S.K.B. Sangeetha, C. Selvarathi, S.K. Mathivanan, J. Cho, and S.V. Easwaramoorthy, “Secure Healthcare Access Control System (SHACS) for Anomaly Detection and Enhanced Security in Cloud-Based Healthcare Applications,” IEEE Access **12**, 164543–164559 (2024).
8. J.B. Madavarapu, R.K. Yalamanchili, and V.N. Mandhala, “An Ensemble Data Security on Cloud Healthcare Systems,” in *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, (IEEE, Trichy, India, 2023), pp. 680–686.
9. I. Patel, S. Jain, J.K. Vishwajeet, V. Aggarwal, and P. Mehra, “Securing Electronic Healthcare Records in Web Applications,” IJEAT 10(5), 236–242 (2021).
10. L. Zhang, C. Zhou, and J. Wen, “APSH-JWT: an authentication protocol based on JWT with scalability and heterogeneity in edge computing,” Wireless Netw **31**(3), 2939–2953 (2025).
11. “HIPAA Compliance,” Amazon Web Services, Inc., (n.d.).
12. A.K. Hebballi, B. J, A. Agarwal, and M. Challa, “Securing Medical Data Records using Blockchain in a Cloud Computing Environment,” in *2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, (IEEE, Bhilai, India, 2023), pp. 1–5.
13. H. Raza, R. Ali, J. Iqbal, and M. Awais, “Secure Room-Sharing Decentralized App Development on Ethereum Block Chain Using Smart Contracts,” *Journal of Informatics and Web Engineering* 3(2), 146–158 (2024).
14. A. A. Zainuddin, A. Q. Amerolazuam, N. A. Nordin, A. F. Abdul Rajat @ Abdul Razak, and N. A. A. Abdul Rahman, “Understanding beyond the block: Overcoming scalability challenges for blockchain-powered healthcare IoT,” in *Highlights in Web Engineering (HIWE)*, eds. S.-C. Haw, S. Y. Ooi, Y. J. Chew, and J. J. (MMU Press, 2025), pp. 3–30.