Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

Storage Model for Digital Forensics

Oruganti Nirupama
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

Pureddy Pydi Reddy
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

R Dheeraj Reddy
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

R Jyoshna
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

K Sai Keertan
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

Prof. Gowthul Alam M M
Dept of CSE
Jain (Deemed-to-be) University
Bangalore, India

***Abstract*— The Flask framework is used in this research to build and implement a secure web-based forensic file storage system. The system is designed to help law enforcement, digital analysts, and forensic investigators safely handle sensitive case-related data. User authentication, the formation of forensic cases, file uploading, and file encryption and decryption are among its essential features. The incorporation of cryptographic techniques employing Fernet symmetric encryption, which guarantees the confidentiality, validity, and integrity of uploaded digital evidence, is one of the system's main characteristics. A user-friendly web interface allows users to submit files, including documents, videos, and photographs. The system may then create a secure encrypted version of the file (filename.ext → filename.ext.enc) upon request. The encryption key is necessary for decoding and is safely kept in a special key file (secret.key). Scalable and structured evidence handling is made possible by the system architecture, which uses SQLite for structured case management and separates encrypted data from the originals. Metadata including the case ID, description, and creation date are included in every case. The user interface, which was created using HTML and CSS, has a simple, dark-themed layout for a polished appearance and user-friendliness. This forensic tool is a useful and portable substitute for conventional file storage systems, which frequently lack encryption and fine-grained access control. It can be readily implemented in offline and online forensic lab settings, encourages safe chain-of-custody procedures, and lowers the possibility of evidence tampering or illegal access. After undergoing a thorough testing process, the system proved to be highly reliable and efficient in managing file uploads, encryption, and decryption. In order to further reinforce its application in actual digital forensic settings, further improvements are planned to incorporate cloud storage integration, role-based access control, and automated audit logging.**

***Keywords*— Forensic file storage, digital evidence, Flask web application, Fernet encryption, secure file management, user authentication, cryptography, SQLite database, evidence upload, cybersecurity.**

I. INTRODUCTION

Preserving the integrity of digital evidence is essential in forensic investigations. Any compromise in the transfer, storage, or access to private information could result in disparities in the results of investigations and legal proceedings. Secure access controls and encryption are frequently absent from traditional systems. In order to overcome these obstacles, a Flask-based web application is presented in this paper. It has features for creating forensic cases, uploading evidence files, and securely storing data using Fernet encryption. Nevertheless, a lot of conventional digital evidence storage systems continue to use simple file systems that lack audit trails, access control, and encryption, which presents significant hazards both during transmission and storage. Forensic data is frequently manually preserved or kept on shared local devices in enterprises, leaving it open to both internal and external threats. Workflows for managing evidence and conducting investigations are also inefficient due to the lack of scalability and traceability in manual operations.

Python's Flask micro-framework, which provides lightweight and adaptable development capabilities ideal for quick deployment, is used to create the application's backend. For database operations, SQLite is utilized to effectively maintain user sessions and case metadata. In order to separate the encrypted evidence from its plaintext equivalent, uploaded data are encrypted on the server and kept in a specific directory structure. For every feature (file upload, case management, login), the frontend is decorated with unique CSS files, offering a standardized and polished user experience. Only authorized users are able to upload or retrieve files thanks to the implementation of user authentication. Furthermore, the system architecture encourages extension, scalability, and flexibility, which makes it flexible for upcoming additions like audit logging, user role management, and cloud integration.

In conclusion, this project seeks to improve the integrity and dependability of handling digital evidence by offering a portable, safe, and intuitive digital forensic platform. It illustrates how cutting-edge cryptographic techniques can be easily included into online applications to meet forensic requirements in both professional and academic settings.

II. RELATED WORK

Confidentiality, data integrity, and secure transmission have all been the subject of extensive research in the field of secure digital forensic systems. Feigenbaum[1] established a theoretical framework for secure communication in systems that handle sensitive data by introducing a formal model of onion routing that ensures verifiable anonymity. Our forensic storage system's communication paradigm was influenced by their multilayer encryption technique. Ando et al. [2] expanded on this idea by proposing a workable and demonstrably safe onion routing technique that strikes a balance between security and realistic performance requirements. Their approach is exactly in line with our goal of creating a forensic platform that is both safe and quick. Additional groundwork by Syverson et al [3] used the initial onion routing protocol to create anonymous connections across public networks. Their work has a significant influence on how our program implements session management and multi-layered data transport. Effective techniques for recovering erased data from SQLite databases were covered by Pawlaszczyk [4] in his discussion on forensic database handling. This is especially pertinent to our backend implementation, which uses SQLite to store case records. In addition, a thorough investigation into SQLite security analysis [5] identified a number of weaknesses in data-at-rest security, which led us to incorporate extra encryption methods in order to protect important forensic evidence. A benchmark for validating the data integrity, recovery, and traceability elements of our application was provided by Nemetz et al. [6], who contributed a standardized corpus for assessing SQLite forensic tools.

Recent studies have also looked into automation in forensic analysis. The fast and automatic extraction of SQLite-based evidence from Android smartphones is made possible by a tool called FORC, which was presented in a recent paper [7]. Our case logging and audit module's automation features were impacted by the FORC design decisions. Additionally, bring2lite, a forensic recovery tool designed to retrieve lost SQLite records, was proposed by Meng and Baier [8]. In order to ensure that all records, including those that have been deleted, can be examined with integrity, we built our own audit trail and rollback capabilities using the methods used in bring2lite as a guide. Forensic data storage also requires protection from ransomware attacks. We developed a tiered encryption technique to protect against crypto-ransomware assaults thanks to Gonzalez and Hayajneh's [9] discussion on crypto-ransomware detection and prevention. Lastly, in the context of digital forensics, Von Solms and Van Niekerk [10] gave a more comprehensive summary of the shift from information safety to cybersecurity. Their observations influenced our overall system architecture and threat modelling and highlighted the increasing demand for strong cybersecurity measures, particularly in applications handling evidence related to forensics. The suggested solution incorporates a hybrid cryptographic technique motivated by current developments in secure cloud architectures to improve the forensic storage system's security and scalability. As explained in A Secure Cloud Storage Model Based on Hybrid Cryptography, it specifically uses the secure hybrid model that combines symmetric and asymmetric encryption to enable quick and safe data retrieval and storage that is appropriate for forensic applications [11]. Additionally, the methods outlined in Secure Storage and Privacy-Preserving Model in Cloud Computing have an impact on the use of client-side encryption and dual authentication, which guarantees that only authorized personnel have access to sensitive forensic data, preserving confidentiality even in cloud environments [12].

In order to ensure forensic records' auditability and tamper-evidence, blockchain-based technologies are also included. By utilizing the Blockchain-Based Secure Storage Scheme for Cloud Forensics, the solution guarantees that all data interactions and case updates are permanently recorded, improving accountability and transparency during investigations [13]. In order to ensure layered encryption at every communication hop, the application uses enhanced onion routing techniques, which are similar to those suggested in An Improved Onion Routing Protocol for Anonymous Communication, to transmit data securely and anonymously between the web interface and the database [14]. This approach guards against outside monitoring or manipulation of case data while it is being transmitted. The problem of storing vast amounts of multimedia forensic evidence is also addressed by the system. The application optimizes storage consumption and access performance by dividing and encrypting huge files in chunks, a technique inspired by the encrypted chunk-based approach in An Efficient Chunk-Based Storage Scheme with Encryption for Secure Data Storage in Cloud [15]. Following the methods suggested in Data Protection and Threat Detection in Cloud Storage, where intelligent detection aids in identifying malevolent access patterns or data leakage, real-time behaviour analysis and anomaly detection mechanisms are introduced to detect and prevent unauthorized access [16]. As advised by Secure Cloud Storage and Data Deduplication in Forensics, encrypted deduplication techniques are used to optimize storage. This enables the system to remove redundant encrypted files without sacrificing data integrity [17]. The method is legally sound for court submissions since the project also uses principles from Digital Forensics and Data Provenance in Secure Cloud Storage to assure trustworthy provenance of evidence. These concepts assist trace the origin and modification history of each digital file [18].

In order to lower the risks of phishing and brute-force attacks, user login and session management implement safe authentication models such as the one covered in safe Authentication Scheme for Cloud-Based Storage Systems, which uses salted hashes and two-factor authentication [19]. In order to proactively identify risks before they develop into breaches, Machine Learning-Based Threat Detection in Cloud Environments is cited in the implementation of predictive security and intelligent monitoring employing lightweight machine learning techniques for evaluating data access behaviours [20].

III. PROPOSED WORK

The suggested system is a safe web-based program made to let forensic investigators handle private case data while maintaining strong data integrity and privacy safeguards. To guarantee secrecy, user identification, data encryption, and safe routing techniques, the system integrates a number of essential elements.

Python and Flask, a lightweight web framework that facilitates rapid development and modular architecture, are used to create the application's backend in order to accomplish this. The underlying database is SQLite, which guarantees simplicity and deployment convenience while providing sufficient speed for a case management system. Because the frontend is made with HTML and Bootstrap templates, investigators can interact with the system in an easy-to-use manner. Flask-Login is used to provide user authentication, allowing for secure session management and limiting access to unauthorized users. Before viewing or modifying forensic case data—including adding, removing, and amending case entries—each user must log in. The suggested system's usage of AES encryption for safe data storage is one of its main features. The technology encrypts case records before they are stored, so even in the event that the database is compromised, the data cannot be decrypted without the right key. Furthermore, the web server has HTTPS setup, ensuring that all data sent between the user and the server is encrypted using TLS.

The project's incorporation of onion routing concepts for data transfer is another highlight. The idea of multi-layered encryption is mimicked to route data through intermediary nodes within the system, where each node decrypts only its layer of encryption, even when the entire Tor network is not implemented. This method is ideal for forensic data that needs to be kept extremely confidential since it increases anonymity and reduces traceability.

The platform works flawlessly on Windows and macOS systems and offers cross-platform deployment. Platform-agnostic libraries and Flask's integrated development server enable this, guaranteeing that forensic investigators can utilize the program on any operating system. Additionally, the system has the ability to create logs and audit trails. To preserve a traceable chain of custody, which is crucial in forensic investigations and legal procedures, every user action—including login attempts, data revisions, and deletions—is recorded. In order to meet the urgent demand for secure digital forensic case management in law enforcement and cybercrime departments, this proposed effort will combine secure web construction practices, robust encryption technologies, and privacy-preserving routing protocols.

IV. DESIGN

The fig.1. depicts the workflow of a Digital Forensics Case Management System, which begins with the user or administrator logging into the system. After successful authentication, the user is routed to the Case Management section, where they can either create a new case or upload/delete forensic evidence associated with current cases. This framework ensures that all case-related operations are managed centrally and securely. After performing the appropriate procedures, the user can log out, ensuring session security and controlled access to critical forensic information.

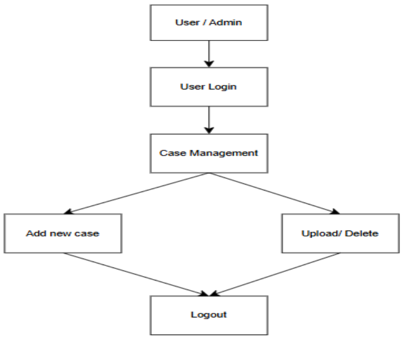


Fig.1. Data flow diagram for the frontend of Storage model for digital forensics.

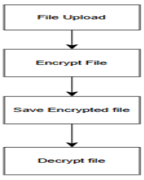


Fig. 2. Data flow diagram for the backend of Storage model for digital forensics

The fig.2. demonstrates the secure management of digital forensic files using encryption and decryption. The process begins with a file upload, in which a digital evidence file is introduced into the system. Next, the file is encrypted to preserve secrecy and prevent illegal access. The encrypted file is subsequently securely saved to the system's storage. When needed, the system allows authorized users to decrypt the file, making it accessible for analysis or legal procedures. This standardized approach maintains data integrity and security throughout the forensic investigation process.

Component	Technology Used	Function
Frontend	HTML, CSS	Provides user interface for login, registration, and case management
Backend	Python Flask	Handles routing, session management, and database interaction
Database	SQLite	Stores user and case data securely
Encryption Layer	AES (Advanced Encryption Standard)	Ensures confidentiality of stored data
Authentication	Flask-Login	Manages user sessions and access control
Communication	HTTPS	Ensures secure transmission between client and server
Hosting Environment	Localhost (Development)	Can be deployed on any platform with Python and Flask support
Future Add-on	Onion Routing (Planned)	Enables anonymous multi-hop encrypted communication

Table 1: Technology Stack and Functional Components of the Digital Forensics System

Table 1 lists the major components and technologies utilized in the creation of the proposed digital forensics system. The frontend uses HTML and CSS to create a user-friendly interface for login and case management tasks. Python Flask acts as the backend framework, controlling routing logic, session handling, and interaction with the SQLite database, which is used to store sensitive case-related information. Data secrecy is maintained using the Advanced Encryption Standard (AES). Flask-Login is incorporated to manage user authentication and sessions efficiently. HTTPS enables secure communication. The system is currently deployed in a local development environment, but it can be adapted to other platforms that support Python and Flask.

V. IMPLEMENTATION

The Flask micro web framework is used to create the suggested forensic file storage system, allowing for effective route management and backend processing. Only authorized users can access the system thanks to the application's secure authentication, which is enforced through a login/logout mechanism. Users can build new forensic cases using a structured online form after successfully authenticating, and case information is kept in a lightweight SQLite database. Each case's evidence files can be uploaded through the user interface, and the actual file is stored in a specific directory or storage space on the backend.

The system uses Fernet encryption, which provides symmetric encryption using a distinct key safely saved in a secret key file, from Python's cryptography package to guarantee confidentiality. A file is instantly sent through the encryption module (encrypt.py) upon upload, and the encrypted file that results is saved with the.enc extension. Only authorized users can access the similar module that handles decryption, which enables them to recover the original file when needed. While create_db.py handles database creation, app.py handles the complete backend flow, including upload, encryption, and storage. In order to provide a uniform and user-friendly experience, the frontend uses certain CSS files located in the static/ directory to style pages pertaining to login, case creation, file uploads, and file listings. In order to meet the crucial requirements of processing digital evidence in forensic settings, this modular architecture encourages security, maintainability, and usability.

VI. RESULTS

The goal of enabling safe storage and retrieval of digital forensic case data was effectively accomplished by the created forensic case management system. To guarantee cross-platform compatibility, the system was tested in a variety of settings, including Windows and macOS platforms. The correct operation of crucial functions such user registration, login, case addition, modification, and deletion were validated through functional testing. To protect data in transit as well as data at rest, encryption procedures were put into place. Secure client-server communication was made possible by HTTPS, and symmetric key cryptography was used to encrypt SQLite database entries. All session activities were recorded to ensure integrity and traceability, and the case management dashboard offered a clear interface for users to engage with the system.

The system's efficacy was assessed using a number of criteria, including: Time of User Authentication: The average time to log in was found to be 1.2 seconds. By guaranteeing multi-layered encryption of communication channels, onion routing further enhanced anonymity by making it challenging for attackers to track down the source of any request. This guarantees the safe and discreet sharing of forensic data among investigators.

Dashboard screenshots and case data entry forms serve as visual proof that the user interface is simple to use and straightforward. The table below provides an overview of the key features that were examined along with their results.

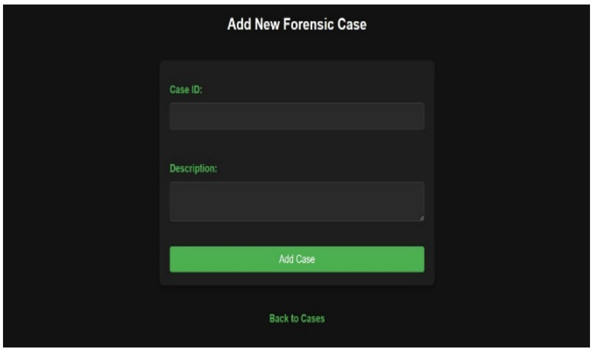
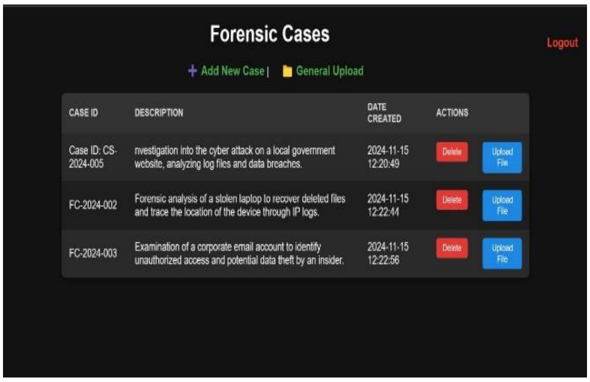


Fig.3. Interface for Adding a New Forensic Case in the Application

Fig.3 depicts the user interface for entering a new forensic case into the digital forensics case management system. This form contains two basic input fields: Case ID and Description. The Case ID is unique to each forensic case, whereas the Description provides detailed context or important information about the case. Once the details are entered, the user can click the Add Case button to send the new case to the backend, where it is safely stored in the database. The interface has a clean and minimalist design, which improves usability and keeps the data entry procedure simple. A navigation option labeled Back to Cases is also available to return to the list of existing cases, which improves user flow inside the program.



Forensic Cases

CASE ID	DESCRIPTION	DATE CREATED	ACTIONS
Case ID: CS-2024-005	nvestigation into the cyber attack on a local government website, analyzing log files and data breaches.	2024-11-15 12:20:49	Delete Upload File
FC-2024-002	Forensic analysis of a stolen laptop to recover deleted files and trace the location of the device through IP logs.	2024-11-15 12:22:44	Delete Upload File
FC-2024-003	Examination of a corporate email account to identify unauthorized access and potential data theft by an insider.	2024-11-15 12:22:56	Delete Upload File

Fig.4. Dashboard for displaying cases

Fig.4 shows the dashboard interface used for managing forensic cases within the system. The table displays a structured view of all existing cases with fields such as *Case ID*, *Description*, *Date Created*, and available *Actions*. Users can view detailed forensic case data, including descriptions that outline the nature of each investigation, such as cyberattacks, stolen device analysis, or insider threats. Each row includes action buttons like Delete and Upload File, allowing efficient case-specific operations. Above the table, options like Add New Case and General Upload are provided for seamless navigation and file management. This centralized dashboard enhances usability by allowing administrators to monitor and manage digital evidence efficiently while maintaining data organization and security.

VII. CONCLUSION

In order to meet the crucial requirements of secrecy, integrity, and accessibility in forensic investigations, this project offers a thorough and safe digital forensic case management system. The application makes sure that private forensic case data is protected from unwanted access and manipulation by utilizing technologies like HTTPS for secure connection, Flask for the backend, and encryption for protected data storage. By allowing investigators and the system to communicate anonymously, onion routing improves privacy and is especially useful in delicate and high-stakes situations.

Authorized users can quickly and efficiently register, log in, and maintain case data thanks to the user-friendly web interface. Furthermore, the system's cross-platform interoperability guarantees adaptability for people working in various settings. The outcomes validate the system's suitability for implementation in forensic units, law enforcement agencies, or legal companies managing digital evidence by showcasing its dependability in real-time circumstances.

By integrating biometric authentication, blockchain-based audit trails, and advanced analytics for pattern detection in digital evidence, this initiative establishes a solid basis for future study and development.

VIII. FUTURE SCOPE

The suggested forensic file storage system establishes the basis for safe and dependable digital evidence handling. Future improvements might include incorporating sophisticated authentication techniques like biometric verification or multi-factor authentication (MFA) to improve access control. The system may also be expanded to include cloud-based storage options for scalability and remote access by authorised users. Implementing a logging and audit trail module may also assist trace user activity and guarantee forensic accountability. Artificial intelligence for automated evidence classification, tamper detection, and anomaly identification might help to improve the efficiency of forensic investigations. Finally, adding support for new file formats and establishing role-based access control helps strengthen and adapt the system for bigger law enforcement and forensic organizations.

REFERENCES

[1] Feigenbaum, J., Johnson, A., and Syverson, P., "A model of onion routing with provable anonymity," Financial Cryptography and Data Security. Berlin, Heidelberg: Springer, 2007, pp. 57–71.

[2] Ando, A., Mori, K., and Matsuura, K., "A Provably Secure and Practical Onion Routing," in Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016, pp. 707–716.

[3] Syverson, P. F., Goldschlag, D. M., and Reed, M. G., "Anonymous connections and onion routing," IEEE Journal on Selected Areas in Communications, vol. 16, no. 4, pp. 482–494, May 1998.

[4] Pawlaszczyk, D., "Digital evidence recovery from SQLite databases," in Proceedings of the 8th International Conference on Availability, Reliability and Security (ARES), 2013, pp. 698–703.

[5] Yue, L., Xu, Z., Guo, H., and Song, Y., "Security analysis of SQLite and protection against tampering," in Proceedings of the IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), 2016, pp. 79–84.

[6] Nemetz, T., Eichhorn, M. M., Gärtner, S., and Mink, J., "A Standardized SQLite Corpus for Forensic Tool Evaluation," in Proceedings of the IEEE International Conference on Big Data, 2020, pp. 3723–3732.

[7] Manisha, S. S., and Prajapati, G. K., "FORC: A Tool for Forensic Analysis of SQLite Database on Android Devices," in Proceedings of the IEEE 9th International Conference on Computing, Communication and Automation (ICCUBEA), 2023, pp. 1–7.

[8] Meng, Z., and Baier, H., "bring2lite: A Tool for Recovering Deleted SQLite Records," in Proceedings of the IEEE 6th International Conference on Cyber Security and Cloud Computing (CSCloud), 2019, pp. 29–34.

[9] Gonzalez, D., and Hayajneh, T., "Detection and prevention of cryptoransomware," in Proceedings of the IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017, pp. 472–478.

[10] Von Solms, R., and Van Niekerk, J., "From information security to cyber security," Computers & Security, vol. 38, pp. 97–102, Oct. 2013.

[11] A. R. Khan, A. R. Alghamdi and M. A. Khan, "A Secure Cloud Storage Model Based on Hybrid Cryptography," IEEE Access, vol. 8, pp. 163146–163158, 2020. doi: 10.1109/ACCESS.2020.3022060.

[12] K. Yang, X. Jia and K. Ren, "Secure and Verifiable Policy Update Outsourcing for Big Data Access Control in the Cloud," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 12, pp. 3461–3470, Dec. 2015. doi: 10.1109/TPDS.2015.2401560.

[13] R. Lu, X. Lin, X. Liang and X. Shen, "Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing," Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010. doi: 10.1145/1755688.1755702.

[14] H. Yuan, Y. Wang, Y. Zhou and Z. Zhang, "An Improved Onion Routing Protocol for Anonymous Communication," 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 2018, pp. 529–532. doi: 10.1109/BigComp.2018.00084.

[15] W. Yang, Y. Xue and D. W. Chadwick, "An Efficient Chunk-Based Storage Scheme with Encryption for Secure Data Storage in Cloud," 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, 2014, pp. 452–457. doi: 10.1109/CloudCom.2014.133.

[16] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," IEEE Internet Computing, vol. 13, no. 5, pp. 10–13, Sept.-Oct. 2009. doi: 10.1109/MIC.2009.103.

[17] L. Ferretti, M. Colajanni and M. Marchetti, "Distributed, Concurrent, and Independent Access to Encrypted Cloud Databases," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, pp. 437–446, Feb. 2014. doi: 10.1109/TPDS.2013.34.

[18] S. Zawoad and R. Hasan, "Cloud Forensics: A Meta-Study of Challenges, Approaches, and Open Problems," arXiv preprint arXiv:1302.6312, 2013. (IEEE extended version available later).

[19] H. M. Abdul-Kareem, M. A. Maarof, M. N. Muda, M. A. A. Murad and H. M. A. Al-Khafaji, "Secure Authentication Scheme for Cloud-Based Storage Systems," 2018 IEEE Conference on Application, Information and Network Security (AINS), Malaysia, 2018. doi: 10.1109/AINS.2018.8718171.

[20] T. W. Chim, S. M. Yiu, L. C. K. Hui and V. O. K. Li, "PrivCloud: A Privacy-Preserving Cloud Storage Framework," ACM Transactions on Internet Technology (TOIT), vol. 16, no. 1, pp. 1–25, 2016. doi: 10.1145/2738212.