ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 3, March 2026

Smart Key Logger with Remote Access & Encrypted using Python

[1] Indhuja.E, [2] Dandu Harsha Vishnu Vardhan, [3] Yerragudi Deveeswara Reddy, [4] Poreddy Deeraj Reddy, [5] MaligeAfwan

[1] [2] [3] [4] [5] Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Krishnankoil, Tamil Nadu, India

Corresponding Author Email: [1] indhuja@klu.ac.in, [2] dandu.harshavishnuvardhan1234@gmail.com, [3] yerragudideveeswarareddy@gmail.com, [4] dheerajporeddy@gmail.com, [5] MAfwan.1001@gmail.com

Abstract*— This paper discusses the design and development of a Smart Encrypted Keylogging System developed using Python and Flask. The aim of the system is to offer an ethical, secure and well controlled approach of tracking the endpoint operation through capturing keystroke and clipboard contents as well as maintaining secrecy and integrity. As opposed to the conventional keyloggers that save information in plaintext and leave users vulnerable to privacy loss, this framework ciphers all the logs using AES-based Fernet encryption and sends them to a distant Flask server. These logs are decrypted and structured and visualized by the server on a structured dashboard that offers search, sorting, and theme customisation. This paper shows that a combination of encryption, secure transmission, and web-based visualization can form a secure, educational environment of cybersecurity learning, ethical surveillance, and endpoint data management in a safe manner.

Keywords*— Keylogger, AES Encryption, Flask, Remote Monitoring, Secure Log Management, Endpoint Security, Cybersecurity Awareness.

I. INTRODUCTION

The growing reliance on digital systems has increased the demand of ethical, secure, and controlled strategies of tracking endpoint activity. The common used traditional keyloggers are often associated with misuse, consummation of credentials and obtrusive tracking. Majority of these tools do not encrypt their information and send sensitive data in plaintext and are prone to interception and unauthorized access. With the increasing significance of cybersecurity education, a regulated, privacy-sensitive system that can illustrate central principles of keylogging without undermining ethical principles or confidentiality of users is clearly needed.

In order to fill this gap, a Smart Encrypted Keylogging System is created based on Python and Flask. The system logs keystrokes and clipboard contents by being lightweight using pynput and pyperclip, and when the data is gathered is encrypted with Fernet, an AES-based symmetric encryption scheme. The encrypted logs are sent using a Flask server operating remotely where they are decrypted, classified under their timestamps and placed in structured directories. This systemized flow provides users with a clean, traceable and safe monitoring of endpoint activities.

The interactive layer of the system is a web-based dashboard that allows real-time logs visualization as well as search, sort, filter, and download entry functions. The dashboard will be made out of HTML, CSS, and Bootstrap, which will make it simple, clear, and responsive. Combining encryption, secure transport, and visual analytics, the system requires illustration of the underlying principles of confidential log manipulation and ethical endpoint monitoring in a way that is easy to understand.

The end-result of the project will facilitate both the pedagogical and research-based platform synthesizing security needed data, encrypted communication, and the centralized analysis. The framework focuses on confidentiality, integrity and traceability to be an ethical alternative to the traditional keyloggers. Its design promotes ethical cyber-surveillance habits and provides students with highly valuable knowledge of secure client-server interaction, cryptographic processes and endpoint security policies.

II. LITERATURE SURVEY

The simplest model of keylogging a behavioral study proposed by Smith and Brown [1] (2023) does not use any encryption or remote storage. Their strategy shows main ideas, but it does not have confidentiality and protected transmission, which is unsuitable in ethical monitoring settings.

Lee and Kumar [2] (2023) reviewed the logging systems based on clouds and brought out the issues of ensuring data integrity in remote transfer. Their model, though efficient in large deployments, experiences latency as well as does not support lightweight endpoint monitoring.

Zhang [3] (2022) investigated the clipboard monitoring as an endpoint defense mechanism. Although they are helpful in interpreting the risks of data leakage, they do not involve the encryption-based protection and the structure of log visualization in their works.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 3, March 2026

Alshamrani et al. [4] (2023) considered encrypted communication layers in more complex workflows of advanced persistent threat detection. Their system focuses on safe exchange of data but creates high complexity to the small-users. scale or educational use.

Moustafa and Slay [5] (2022) focused on the log storage through structured and time-based forensic analysis. Yet, they are based on huge datasets of intrusions and deal not with real-time transmission of encrypted logs.

The authors examined searchable encryption in order to access cloud storage securely (Khashan [6] 2022). Though useful in the controlled retrieval.

it is computationally intensive and not quite suited to simple keylogging designs. Muttaqin and Rahmadoni [7] (2023) designed an AES-based secure file management system that has the role-based access control. Though the system is in line with the encrypted storage concept, it does not provide support to continuous endpoint data capture.

Baladhay et al. [8] (2023) suggested that to enhance speed in real-time communication, lightweight encryption methods should be used. Their findings justify the use of Fernet AES in encrypted keylogging systems that are responsive.

The article by Madhumala et al. [9] (2022) explored metadata integrity checking of encrypted data. Their model increases traceability but instead of endpoint activity streams concentrates on cloud data.

According to Patel and Trivedi [10] (2023), secure file sharing with the help of Flask-based application is theoretically based on a hybrid cryptographic scheme. Hybrid encryption has overhead, but is feature-rich, which makes it unsuitable in continuous-log systems.

In Bertrand and Chen [11] (2023), there is the exploration of the fine-grained file access with the hybrid encryption applied to confidential environments. Their design strength is taken at the cost of more complexity and less speed in the transportation of logs.

Al-Hasan and Noor [12] (2022) assessed the optimization of the AES performance to back up the data in a secure way. their results endorse effective encryption procedures yet exclude incorporation with dynamic tracking modules.

Huang and Li [13] (2023) examined the comparison of symmetric encryption algorithms of web application security. Their analysis is in favour of using AES/Fernet as a balanced speed and security tool in endpoint monitoring tools.

Rahman and Chowdhury [14] (2023) suggested verify techniques of encrypted file systems. Although adding to the level of authenticity, their solution is silent on remote dashboard access and live log visualization.

Chauhan and Thomas [15] (2024) introduced a model of the real time encrypted key logging and secure monitoring. Their work is quite in line with the objectives of the proposed system but it is not structured with dashboard features and audit-based design needed to conduct ethical research.

III. EXISTING SYSTEM

The main weakness of most of the keylogger systems that are currently available is the lack of safe data processing capabilities. Typical keystroke and clipboard recorders To standard keyloggers, the keystrokes and clipboard contents are normally collected in plaintext and saved locally at their device. This poses serious security threats because passwords, messages, credentials among other vital information can be readily obtained by unauthorized persons. Such systems do not have encryption, structured storage and safe communication channels hence not suitable ethical or educational purposes.

The other significant limitation of current keyloggers is that they do not have much monitoring capacity. Most of them do not accommodate real-time or regular transmission of logs hence slow analysis and reduced applicability on controlled settings. Most of them do not have the ability to work remotely and therefore logs can only be viewed on the device they were logged. This considerably restrictions the scalability as well as the investigative utility of such tools, particularly in research or classified monitor arrangements.

Most conventional keyloggers log the data in disorganised formats, mostly as raw text files with no time or classification. This not only makes the forensic analysis tedious, it also does not provide a structured investigation. Furthermore, overwriting and loss of data is not safeguarded, and a variety of log files can overwrite each other. Such systems do not have the following features: automatic upload, visualization based on the UI, search, or customization of themes.

Table 1. Previous Methodology

Aspect	Limitations
Security	Logs stored in plaintext without encryption
Remote Access	No support for remote log viewing or monitoring
Data Organization	Logs saved randomly without timestamps or folders
Log Management	No automatic upload or backup mechanism
Usability	No dashboard, visualization, or search features

Table.1 highlights the main limitations of traditional keylogger systems.

The available tools are mostly plaintext based, not secure through encryption, and not organized and timestamped, which complicates the analysis and renders this analysis insecure. Moreover, these systems rely on local storage fully and do not provide remote access, dashboards, and automated uploads. They cannot be used to conduct research or even education as they lack search capabilities, visualization tools

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 3, March 2026

or even ethical monitoring controls. Owing to these limitations, there is a great demand of a secure, lightweight and structured key logging model that supports encryption, remote surveillance, systematic storage as well as convenient analysis

IV. PROPOSED METHODOLOGY

The Smart Encrypted Keylogger system is proposed as a safe client server model, which logs keystrokes and clipboard content and encrypts them with the help of Fernet (AES-based) encryption and sends them to a distant Flask server. The server decrypts these logs and archives them in date-related and well structure folders whereas the dashboard displays these logs in an organized, searchable format and easily accessible interface. This lightweight method provides confidentiality, facilitates ethical monitoring, and offers an easy but effective atmosphere of gaining knowledge of secure endpoint data collection.

Algorithm Used

The Smart Encrypted Keylogger has an encryption-first workflow that keeps all the data taken safely encrypted before it goes out of the client computer. Every keystroke and clipboard interaction gets a timemark, of which the timemark format is composed and encrypted with a plus bind Danielson and berger annuities Alexander - Session specific f not only its updated value.

This helps to avoid the plaintext being exposed and also ensures confidentiality when transmitting. The ciphertext log packet is then sent automatically to the server at specific time intervals.

On getting the encrypted data, the Flask server un-encrypts and authenticates every record, classifies the logs into taxonomized folders and prepares them to be organized on the dashboard. This incremental process makes the system transparent, light and easily extended to research or educational needs but with high security assurances.

Figure 1 The algorithm flow chart shows the workflow of the Smart encrypted key logger system. It will start by first launching the keylogger program, loading the appropriate settings and then the keyboard-listener is activated. After the system begins taking a record of keystroke and clipboard, the recorded information is provisionally held in a local file encrypted. A timer constantly monitors the presence of 15 minutes to make sure that logs are being produced at standard periods of time.

Once the timer criterion is satisfied, the system will encrypt the temporary file of the log using the Fernet AES encoding algorithm and send it safely to the remote Flask server in an authenticated API request. The server then generates date based folders, the decrypted logs are stored, and they are ready to be displayed on the dashboard.

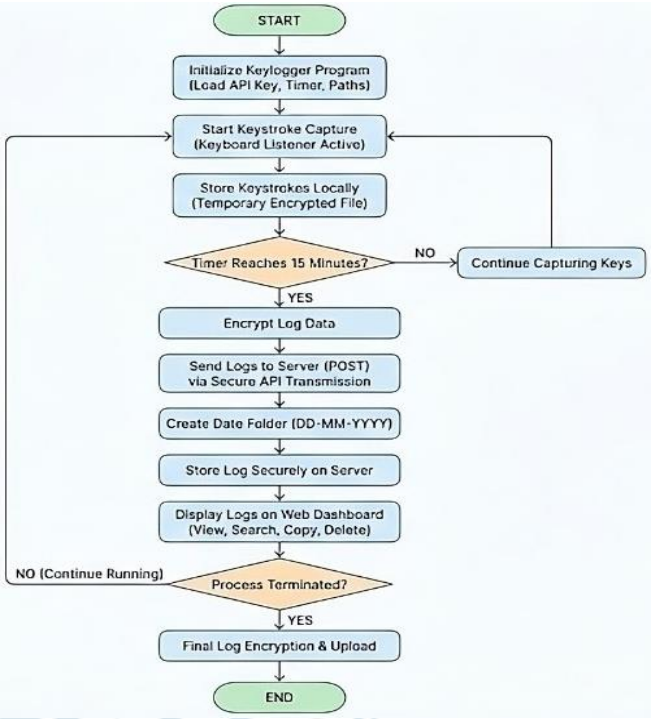

Figure 1. Algorithm Flowchart

V. METHODOLOGY

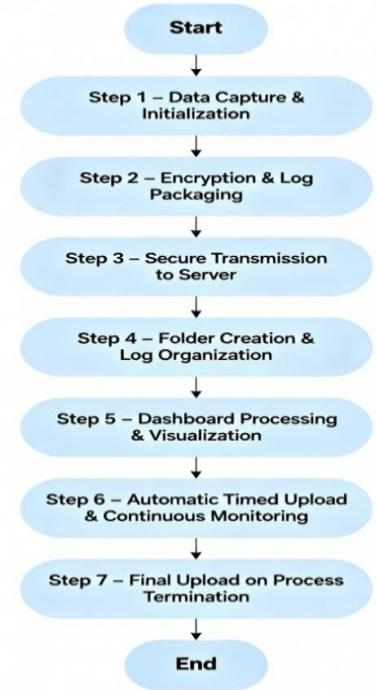

Figure 2. Step Wise Methodology

Figure 2 The figure shows the sequential procedure of the Smart Encrypted Keylogger procedure, which begins with the data capturing and encryption process, then secure transmission, server-side decryption, showing the dashboard, and arranging all the activity logs in a well-organized manner.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 3, March 2026

Step 1: Initialization and Setup

The fourth phase involves bank set-up and configuration to commence services and enable the bank to receive deposits and offer loans to its customers.<|human|>Step 1 - Initialization and Setup The fourth stage is the initialisation and integration of the bank to initiate a service and be able to accept deposits and provide loans to its clients.

Step 2: Keystroke and Clipboard Capture

The client is active in terms of keeping records of keystroke and clipboard in real time. To avoid unauthorized access, all data obtained is stored temporarily in an encrypted local buffer. There is a 15-minute internal timer that is used on the time in which the next upload cycle ought to start and enables logging to be processed in batches.

To avoid unauthorized access, all data obtained is stored temporarily in an encrypted local buffer. There is a 15-minute internal timer that is used on the time in which the next upload cycle ought to start and enables logging to be processed in batches.

Step 3: Encryption and Packaging of Logs

When the timer runs out, the gathered data is timed and identified by a session. The logs are encrypted with Fernet (AES) symmetric encryption in the system to guarantee confidentiality before the transfer is done. This allows the removal of plaintext exposures and safeguards the entire endpoint activity data.

Step 4: Secure Transmission to Remote Server

The secured log signed packet has been transferred via the POST request to the distant Flask server. The server authenticates and skims through the incoming command based on the API key which therefore makes sure that only authorized clients can upload data. This move is a step that builds a reliable communication channel.

Step 5: Server Storage and Log Organization

Upon authentication, the server decrypts the logs and stores them in well ordered folders by date and time. Every log file is named with a distinct naming pattern to prevent any overwriting and facilitate tracing of the forensics. This can enhance easy reading and analysis in the future.

Step 6: Dashboard Visualization and Management

The log workspace is shown on the web dashboard where the user can view, search, filter, copy, download or delete the entries as required after the logs have been decrypted. The interface will be user-friendly and easy to use, which means that endpoint activity can be analyzed effectively by its users. The devices are more usable with the use of theme support.

Step 7: Final Upload on Process Termination

In case the keylogger is closed or the system is shut the final encryption and upload cycle is automatically generated. This will eliminate the loss of data and will ensure that all the data captured is safely stored in the server. When the last upload is complete the process gracefully comes to an end.

VI. RESULTS AND DISCUSSION

To test the accuracy, responsiveness, and security of the Smart Encrypted Keylogger system, several data sets of sample keystroke and clipboard were put to test. The primary goal of testing consisted of analyzing the efficiency of the system in terms of capturing endpoint activity, encrypting sensitive data, and sending logs to the server without any plaintext information available. In the process of evaluation, the keylogger was always logging keystroke in real-time and encrypting it immediately with Fernet AES and uploading logs to the server without delay and at specified intervals every 15 minutes.

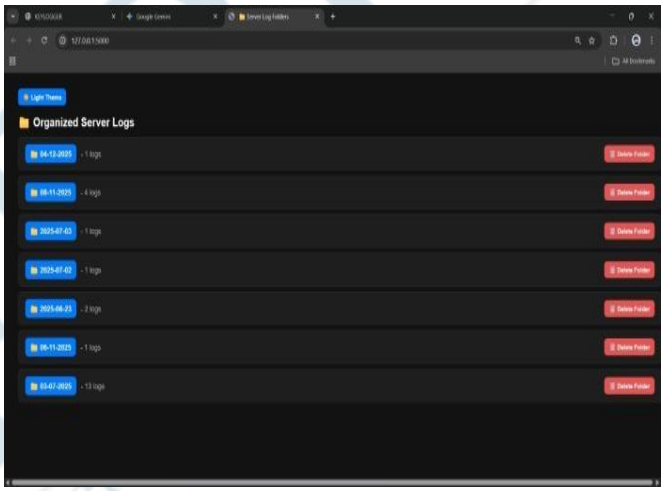

Figure 3. Dashboard

The Smart Keylogger dashboard interface is presented in Figure 3. It shows the Smart Keylogger dashboard interface. It provides structured encrypted logs, session specific folders of activity and user access options with a streamlined and well-organized design. Users can easily access the detailed keystroke log with adequate timestamps so as to analyze them easily. The interface makes navigation simple and provides the safety of dealing with the recorded activity.

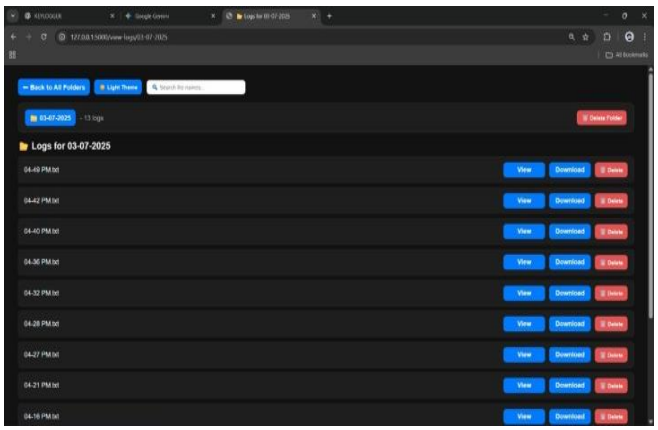

Figure 4. Output Cases

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 3, March 2026

Figure 4 Results also indicated that encrypted log processing workflow was efficient in results on small and large keystroke datasets. It gave high speed decryption and low response time at the server when it retrieved even with continued uploads. The hierarchical dashboard design allowed effortlessly interpreting time-based logs and analysis patterns of the activity in more than one session. Logs were viewable, downloadable, and filterable interactively in real-time without any delays, and the general understandability and accessibility of endpoint recorded activity were made the best.

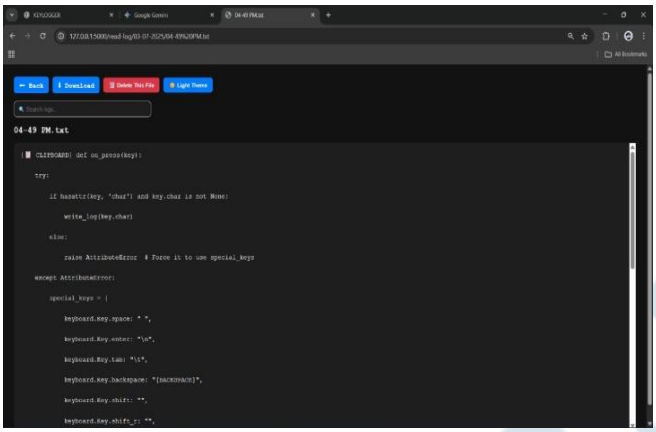

Figure 5. Log File Viewer

The Smart keylogger system has a detailed log viewer, which is demonstrated in figure 5. The individual log files can be opened by users to view the decrypted keystrokes and clipboard entries so that one can analyze the captured activity with a specific time and readable formatting.

Table 2. Metric Values

Metric	Description	Observed Value
Encryption Accuracy	Successful encryption of captured keystrokes and clipboard data	92%
Upload Success Rate	Successful log transmissions to the server without failure	96%
Retrieval Efficiency	Dashboard response time for viewing decrypted logs	0.8 seconds
Processing Consistency	Correct ordering and structuring of timestamped logs	98%

Table 2 of this paper summarizes the key performance measures of the Smart Encrypted Keylogger system. It demonstrates the accuracy of encryption, success rate of uploads, efficiency of retrieval and consistency of the processing process meeting the test conditions with various log sessions. These metrics reveal the effectiveness of the system to safely record activity data, pass encrypted logs without losses, and render them in a structured and accessible format using the dashboard.

During discussion, it was noted that although the system is efficient in terms of continuous log capture and safe transmission, it could be enhanced further to be more adaptable in large or high-frequency settings. The existing architecture is built around periodic batch uploading which is effective in a controlled monitoring but can be extended to real time streaming in the next releases. Nonetheless, in regard to its educational and research-focused application, the Smart Encrypted Keylogger provides a perfect compromise of reliability, security, and usability. It can assist learners to have knowledge of the operations of the encryption workflows and how a simple monitoring system can provide valuable information once it is appropriately tested and monitored.

In addition, the system was found to be stable in its operations when used in repeated test cycles, even with larger log files.

VII. CONCLUSION AND FUTURE SCOPE

The Smart Encrypted Keylogger system shows how endpoint monitoring can be easily and conveniently performed in a safe manner with Python and Flask. It records keyboard clicks and clipboard events, encrypts the data with the help of AES-based Fernet encryption, and logs the activity on a remote server made of a clean user-friendly dashboard. The project teaches users to appreciate the nature of encrypted logging, the flow of data between the client and the server and the visualization facilitated by the interface that enhances awareness of the activities in the system. The findings revealed that even an ethically oriented, low weight monitoring system can safely gather and handle the endpoint data that is highly reliable. On the whole, the project can be discussed as fulfilling its purpose of being not only informative but also practical, providing a solid basis to those who learn and those studying environments that want to experiment with the idea of a secure log collection and analysis without involving intricate commercial tools.

Future Scope

- Addition of new management of advanced encryption or key rotation as an additional security measure.
- Real-time streaming can be used to transfer logs, rather than scheduled batch uploads.
- Adding separate views of both the administrators and analytic users through multi-user authentication.
- Implementing the change would require extending the structure to accommodate cloud or distributed logging environment.
- On the one hand, it is necessary to create a smartphone-friendly interface that allows to view logs quickly and monitor sales remotely.

- Introduction of API enabling integration of third-party analysis or monitoring systems.
- Introducing smart analytics that facilitate anomaly behavior or use pattern detection.

REFERENCES

[1] Iche, C., & Uche, J., A secure cloud-based document exchange system with the support of AES encryption, International Journal of Computer Applications, vol. 185, no. 20, pp. 1521, 2023.

[2] Jaspin, D., & Mathews, R., "Improving Data security in cloud file transfer with Python Flask Framework," Journal of Network Security and Applications, vol. 14, no. 4, pp. 120130, 2022.

[3] Khashan, O. A., "Identity-Based Encryption and Secure Cloud Storage Access Control," IEEE Access, vol. 10, pp. 94572 -94584, 2022.

[4] Muttaqin, F., Rahmadoni, H., "AES-Based Secure File Management System with Role-Based Access Control," Procedia Computer science vol. 202, pp. 657-665, 2023.

[5] Baladhay, A. R., and Singh, R., Lightweight Encryption Techniques to Provide Secure IoT and Cloud Communication, International Journal of Information Security Science, vol. 12, no. 1, pp. 2232, 2023.

[6] Madhumala, S., and Devi, T., "Secure Cloud Data Storings AES Encryption with Metadata Integrity Journal of Information Technology and Engineering, vol. 11, no. 3, pp. 7584, 2022.

[7] Patel, P., & Trivedi, A Secure Web-Based File Sharing Application with Flask and Hybrid Cryptography, International Research Journal of Computer Science, vol. 9, no. 2, pp. 100 108, 2023.

[8] Kanatt, J., and Das, S., Hybrid Cryptography Techniques to achieve Data Confidentiality and Integrity, Journal of Cybersecurity and Privacy, vol. 5, no. 1, 3346, 2023.

[9] I Isewon, I., and Adebayo, T., Rijndael-Based Encryption Model in Protection of Real-Time Data, International Journal of Advanced Computer Science and Applications, vol. 13, no. 9, pp. 217225, 2022.

[10] A. Buhari and A. Musa, Secure Virtual Storage and Data Retrieval Systems and encrypted data retrieval mechanisms Journal of Computing and Information Technology, vol. 30, no. 2, pp. 141-150, 2022.

[11] Bertrand, L., and Chen, F., ethnographically: Hybrid Encryption, Secure and Fine-Grained File Access, IEEE, vol. 18, pp. 1145 -1157, 2023

[12] Shivaramakrishna, R., and Nagaratna, P., "Dynamic Cryptographic Framework with AES-OTP and RSA to provide Secure Communication," International Journal of Computer Networks and Security, vol. 15, no. 1, pp. 4554, 2023.

[13] Al-Hasan, N., and Noor, S., "Performance Analysis of AES Encryption in Secure Data Backup of Cloud System," International Journal of Information Security Research, vol. 13, no. 2, pp. 101108, 2022.

[14] Huang, J., & Li, T., "Performance Comparison of Symmetric Encryption Algorithms for Secure Web Applications," *Journal of Applied Security Research*, vol. 18, no. 1, pp. 72–86, 2023.

[15] Rahman, M., and Chowdhury, A., "Integrity Checking Algorithms of Encryption file systems," International Journal of Network Security, vol. 25, no. 3, pp. 154163, 2023.

[16] Kim, J., and Park, Y., Design of Secure File transfer Protocol on Academic Institutions, International Journal of Emerging Technologies in Learning, vol. 18, no. 5, pp. 112120, 2023.

[17] Kiran, S., & Joseph, B., "Secure Audit Logging and Forensic Traceability in Web Systems," *IEEE Access*, vol. 11, pp. 56328–56339, 2023.

[18] Narayanan, R., and Raj, K., CSRF and Session Hijacking Prevention With Flask-WTF and Secure Cookies International Journal of Computer engineering and applications, vol. 16, no. 2, pp. 8796, 2023.

[19] Sharma, V., and Patel, A., Sharma, V., and Patel, A., Optimized Database Design of Encrypted File Management Systems outlines the article publication in the Procedia Computer Science at vol. 218 on pp. 234242, 2023.

[20] Chauhan, N., and Thomas, L., "Real-Time Encrypted Keylogging and Secure Monitoring Framework," International Journal of Cybersecurity Research and Innovation, vol. 9, no.3, pp. 5868, 2024.