Jurnal AL-AZHAR INDONESIA SERI SAINS DAN TEKNOLOGI
Vol. 11, Nomor 01, January 2026, hal. 1- 16
DOI http://dx.doi.org/10.36722/sst.v11i1.5086 https://jurnal.uai.ac.id/index.php/SST

EduCrypt: A Secure File Data Sharing Platform using Hybrid Encryption for Cloud-Based E-Learning

Andi Arniaty[1], Riri Safitri[1], Lambda Sangkala Murbawisesa[1]

[1]Informatics Department, Faculty of Science & Technology, Universitas Al-Azhar Indonesia,
Jl. Sisingamangaraja, Kebayoran Baru, DKI Jakarta, 12110.

Corresponding author/E-mail: andi.arniaty@uai.ac.id

Abstract – **The rise of cloud-based learning environments has transformed education but has also exposed vulnerabilities in data security. Learning Management Systems (LMSs) often handle sensitive academic data without adequate protection, leaving it vulnerable to unauthorized access and data breaches. This study introduces EduCrypt, a secure file-sharing platform that uses hybrid encryption, combining AES for fast data encryption and RSA for secure key exchange. In this research, a Hybrid Waterfall–Iterative method was used as the overarching research methodology to guide requirements analysis, architectural design, system implementation, and evaluation. EduCrypt integrates seamlessly with Moodle LMS via token- and credential-based authentication. Its architecture includes asynchronous RabbitMQ workers for efficient synchronization and a user-friendly web interface for secure file operations. Benchmark tests demonstrate stable encryption and decryption times under 300 ms for files up to 30 MB, showcasing both scalability and efficiency. Security evaluations confirm that EduCrypt effectively mitigates brute-force attacks, SQL injection, and man-in-the-middle attacks. This research resulted in a validated enhancement in file security practices for LMS through the implementation of a hybrid cryptographic model. Furthermore, the outcome includes a fully functional EduCrypt prototype integrated with Moodle, along with performance and security evaluation results.**

Keywords *- Data Security, Hybrid encryption, File sharing, E-learning, Cloud computing*.

INTRODUCTION

In the global digital era, technological transformation has significantly reshaped the educational landscape, particularly through the adoption of cloud-based learning environments, such as Learning Management Systems (LMS). These platforms enable lecturers and students to share materials, assignments, and research documents more conveniently and efficiently. However, as the use of digital platforms expands, data security has become an increasingly critical and complex issue. Incidents of data leakage, file tampering, and unauthorized access to academic documents are now common in educational institutions [1].

Several factors contribute to these vulnerabilities, including weak encryption, insufficient access control policies, and low user awareness of data protection. Many file-sharing systems in academic settings still rely on basic or even no encryption, making them highly susceptible to data misuse and cyberattacks. Previous studies have indicated that cloud systems in the education sector are particularly vulnerable to security breaches due to weak encryption or the lack of additional protection during file transfers. According to Singh and Garg [1], and C. Susmitha et. al [2], nearly 70% of data leakage incidents in the education sector occur due to the absence of robust encryption mechanisms. Within LMS environments, unprotected file-sharing practices create opportunities for threats such as data modification, malware injection, and academic identity theft.

To address these risks, numerous studies have focused on developing secure cloud-based file-sharing systems. One widely adopted approach is hybrid encryption, which combines the strengths of

Received: 16 November 2025, Accepted: 19 December 2025, Published: 07 January 2025

two algorithms: AES (Advanced Encryption Standard) for high-speed symmetric encryption and decryption, and RSA (Rivest–Shamir–Adleman) for secure asymmetric key distribution. Research by Durge and Deshmukh [3] demonstrated that combining AES and RSA significantly enhances data security in cloud computing compared to using either algorithm independently. However, most existing studies focus primarily on algorithmic performance rather than end-to-end system integration within LMS. Many of these works evaluate encryption in isolation rather than as part of a complete educational workflow, creating a research gap regarding the practical deployment of encryption and its impact on real-world teaching and learning ecosystems.

Additionally, most prior studies have focused on general cloud environments, overlooking the specific requirements of educational settings such as integration with LMS platforms, support for efficient teaching and learning workflows, and usability for both lecturers and students. This gap in the literature underpins the present study. To fill this gap, this research develops EduCrypt, a hybrid-encryption-based file security platform specifically designed for digital education. Unlike previous approaches, this study employs a structured research and system development methodology, specifically, a Hybrid Waterfall-Iterative model [8]. This model enables both formal requirements analysis and continuous refinement based on user workflows in real academic environments. This methodological clarity sets the study apart from previous works that focus solely on encryption mechanisms without considering their practical integration.

Several studies have proposed encryption-based methods for ensuring the security of cloud data. However, most of these methods rely on a single encryption technique - either symmetric or asymmetric - without leveraging the strengths of both. Additionally, many file-sharing systems built on Learning Management Systems (LMSs) still rely on basic access controls without robust encryption.

For example, Durge and Deshmukh [3] introduced a hybrid AES-RSA approach that enhances both performance and security in cloud data storage. Singh and Garg [1] enhanced cloud security by combining RSA, AES, and Blowfish with secure One-Time Password (OTP) functionality, resulting in significant improvements in data protection. Abualkas and Bhaskari [4] presented a hybrid ECC-AES approach that focuses on efficient key management in cloud environments. Al-Bayati AS [5] demonstrated a hybrid AES-RSA model that shows increased resistance to brute-force attacks. Fathima and Arumugam [6] developed a new data transmission model integrating RSA with ChaCha20-Poly1305 to ensure data integrity. Lastly, Saini and Sainis [7] proposed a modular hybrid encryption framework that combines AES block encryption with Feistel networks.

Despite these advancements, there is a noticeable gap in the literature regarding the direct integration of hybrid encryption into LMS platforms commonly used in educational settings. Furthermore, most existing approaches have not adequately considered user experience, particularly for non-technical users such as lecturers and students, when adopting secure digital systems. This study differs from current research by not only examining algorithmic performance but also considering LMS-level interoperability, workflow alignment, asynchronous file processing using RabbitMQ, and usability challenges unique to educational environments - areas that have been largely overlooked in the literature.

This research introduces EduCrypt, an innovative secure file-sharing system based on hybrid encryption, designed explicitly for cloud-based e-learning. EduCrypt's key contributions include integrating hybrid encryption into educational environments, using AES for efficient data encryption, and RSA for secure key management. It also optimizes encryption efficiency to ensure the system remains secure and fast while providing a seamless user experience. EduCrypt offers easy integration with existing LMS infrastructures, ensuring that institutional workflows are not disrupted. Additionally, it adopts a Hybrid Waterfall-Iterative development model to facilitate structured yet adaptive system development that aligns with user requirements and continuous security validation. By integrating a clear research method with a system-level implementation that is directly compared with existing studies, this work reinforces the methodological foundation of EduCrypt and firmly situates it within current scholarly discourse.

Based on these motivations, this study aims to explore how an effective secure file-sharing system can be implemented in digital learning environments. It will investigate how hybrid encryption methods can enhance data protection in cloud-based e-learning systems and how such

systems can maintain high processing efficiency for encrypted data without compromising usability or access speed. To meet these objectives, EduCrypt is proposed as a comprehensive, scalable, and user-friendly platform that strengthens data confidentiality and integrity in modern digital education.

METHOD

EduCrypt was developed using a Hybrid Waterfall-iterative model that combines the structured phases of the Waterfall method with the flexibility of iterative development. The research activities within this framework included requirements analysis, system design, and implementation, all integrated with testing. Instead of focusing on full operational deployment, the final stage of this study emphasizes evaluation and validation. This includes performance benchmarking, security testing, and deployment readiness assessment. This methodological approach aligns with the study's research-oriented goals, prioritizing system correctness, security, and performance validation over a large-scale operational rollout. The overall structure uses a Waterfall-style approach to maintain stable requirements and clear architecture. Iterative cycles are integrated within certain phases, such as system implementation and testing. During these phases, components are repeatedly developed, tested, and refined based on performance and security feedback without altering initial requirements. Thus, iteration occurs at the micro level rather than across all phases, in line with hybrid Software Development Life Cycle (SDLC) practices.

The iterative component strengthens the structured framework by promoting continuous improvement in implementation outcomes informed by evaluation results. Feedback from performance measurements and security assessments is used to iteratively refine system components. This ensures that each cycle improves functional correctness, security compliance, and performance efficiency. This iterative refinement process allows the system to evolve systematically while remaining aligned with established research objectives.

The diagram in Figure 1 outlines the implementation and validation workflow of EduCrypt, distinguishing it from a conventional Waterfall model. It illustrates how the structured phases of requirements analysis and system design lead into iterative cycles of implementation, testing, evaluation, and validation. This representation captures the research process employed in this study, underscoring the continuous evaluation and validation required to refine the EduCrypt system before its consideration for real-world deployment.

Requirements Analysis

The research process began with a requirements analysis phase that focused on identifying security risks and assessing user needs within Moodle-based LMS. This phase examined how academic files such as assignments, lecture materials, and research documents are stored, accessed, and transmitted in typical LMS environments.

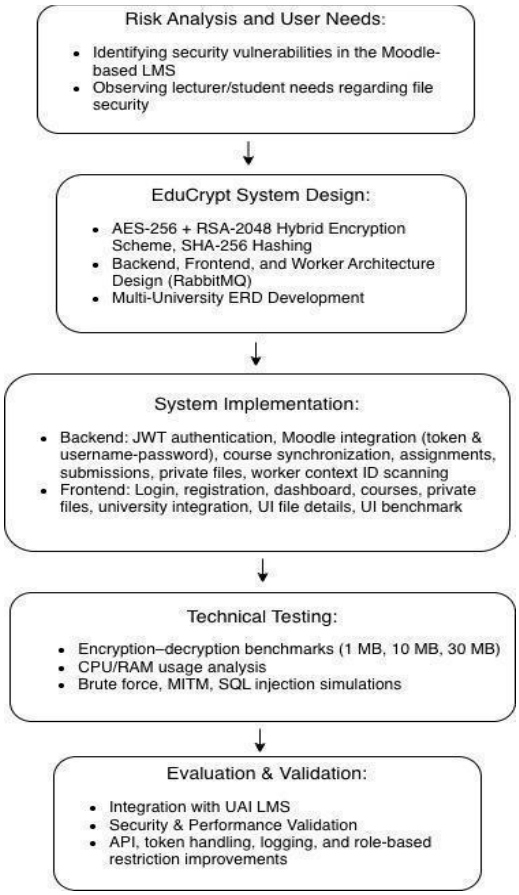

Figure 1. EduCrypt research process and system implementation workflow.

The analysis revealed that many LMS file-sharing mechanisms rely primarily on access control policies and do not implement cryptographic protection at the file level. Consequently, sensitive academic data remains vulnerable to unauthorized access, file tampering, and interception during data transmission.

Simultaneously, user needs were evaluated through direct observations and informal discussions with lecturers and students to understand their expectations and real-world interactions with LMS platforms. The findings indicated that while users prioritize ease of access, efficiency, and seamless integration with existing workflows, they generally have low awareness of file-level security risks. Notably, both lecturers and students emphasized that any additional security mechanisms should operate transparently, without introducing extra complexity or disrupting established teaching and learning processes.

This requirements analysis phase established the core design constraints and functional requirements for EduCrypt. By aligning identified security risks with actual user behavior and institutional workflows, this stage ensured that the proposed system addresses critical vulnerabilities while maintaining usability, efficiency, and compatibility with existing Moodle-based LMS infrastructures. The outcomes of this phase directly informed subsequent design and implementation decisions for the system.

System Design

Following the requirements analysis, the research entered the system design phase, in which the overall architecture and security mechanisms for EduCrypt were clearly defined. During this stage, a hybrid encryption scheme was developed to balance strong security with operational efficiency. AES-256 was chosen as the symmetric encryption algorithm for encrypting file contents due to its high performance and suitability for handling large volumes of data. RSA-2048 was utilized for asymmetric encryption, providing secure protection and distribution of the AES session keys. Additionally, SHA-256 hashing was implemented to verify file integrity and detect unauthorized modifications.

The system architecture was designed using a modular approach that separates core functionalities into backend services, frontend interfaces, and components for asynchronous processing. The frontend component serves as the primary user interface, providing essential functions such as registration, authentication, dashboard navigation, file management, and institutional integration. It is designed to deliver an intuitive, responsive user experience while maintaining strict access controls. The backend serves as the platform's core, managing encryption and decryption, handling API interactions, and facilitating communication with the LMS. It implements user authentication using JSON Web Tokens (JWTs) with a 72-hour expiration period, ensuring secure, time-sensitive access.

Communication between EduCrypt and the Moodle LMS is managed through a dedicated integration layer that uses RESTful APIs supporting both token- and credential-based authentication. This dual approach allows for flexible deployment across various institutional environments. To enhance scalability and efficiency, EduCrypt employs an asynchronous worker system powered by RabbitMQ, enabling parallel processing for context ID scanning and large-scale synchronization of user data and course files.

All encrypted files are stored in a secure repository, where data is protected using the AES-256 algorithm. RSA is used to secure encryption keys and manage key exchange [9], while the SHA-256 hash algorithm ensures data integrity by preventing unauthorized modification or corruption of stored files.

The interaction among various components, including the Moodle API, the integration layer, a hybrid encryption engine using AES-256, RSA-2048, and SHA-256, and secure storage, is illustrated in Figure 2. This figure shows the entire workflow of EduCrypt's system architecture, from file synchronization in Moodle to encryption, storage, and secure end-user retrieval. Figure 3 illustrates the complete file upload and download process, emphasizing the interactions among actors, cryptographic operations, integrity verification, and conditional access control within the system.

A multi-university data model was created to ensure scalability across different institutions. An Entity-Relationship Diagram (ERD) was developed to model multiple universities, users, courses, and file repositories within a single deployment of EduCrypt. This design guarantees data isolation between institutions while enabling centralized management and extensibility. Overall, the system design phase effectively transformed security requirements and user needs into a robust, scalable architecture that is compatible with learning management systems (LMS). This architecture serves as the foundation for the implementation and evaluation stages of the EduCrypt research.

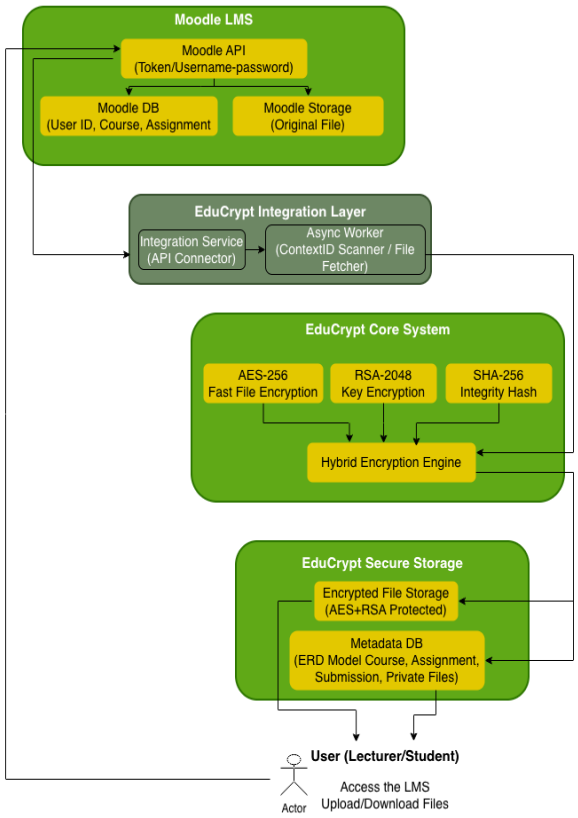

Figure 2. System Architecture and Module Decomposition of EduCrypt

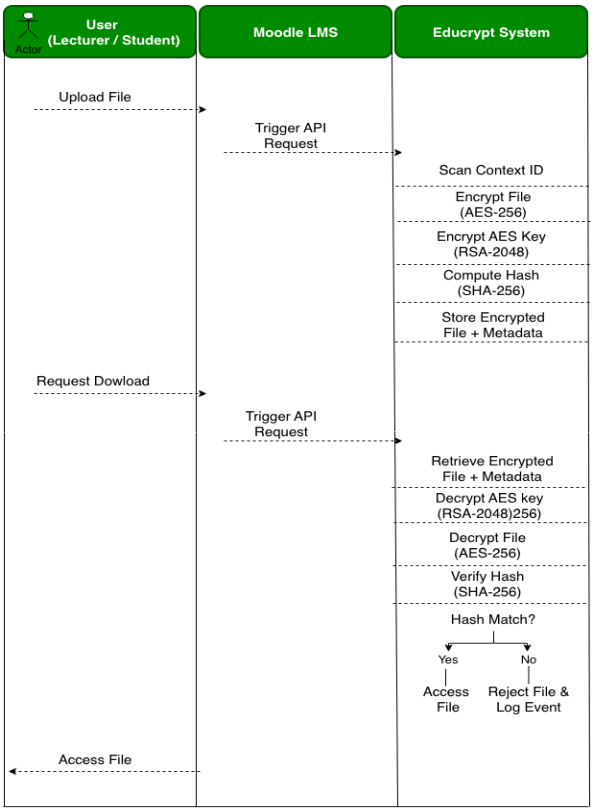

Figure 3. Activity Diagram of Secure File Upload and Download Workflow in EduCrypt

System Implementation

The system implementation stage translated the conceptual design of EduCrypt into a fully functional research prototype. During this stage, all architectural components were developed through concrete backend and frontend processes, ensuring they aligned with the defined security requirements and user needs.

On the backend, EduCrypt implemented a secure authentication mechanism using JSON Web Tokens (JWT) with a designated validity period to manage user sessions and access control. To ensure broad compatibility with Moodle-based LMS, two integration methods were supported: token-based authentication for institutional administrators and advanced users, and credential-based authentication using a username and password for non-technical users. The backend also managed the automated synchronization of academic data, including courses, assignments, submissions, and private user files. To enhance scalability and performance, a RabbitMQ-based worker system was integrated to asynchronously scan and process context IDs, enabling parallel file synchronization and reducing processing latency. All file-related operations were secured through an automated hybrid-encryption workflow, in which files are encrypted upon retrieval from Moodle and decrypted only upon authorization.

On the frontend, EduCrypt provided a web-based interface designed to support common academic workflows without disrupting the user experience. Core interfaces included login and registration pages, a dashboard for system overview, course listings, private file management, and a university integration configuration page. Additional interfaces were created to display detailed file information and performance benchmarks, accessible exclusively to administrative users. File downloads were handled transparently via server-side decryption, ensuring that encrypted files remain protected at rest while users receive plaintext files without requiring any additional technical steps.

This implementation stage signifies the operational realization of EduCrypt as a secure, scalable, and user-oriented file-sharing platform. By integrating cryptographic mechanisms, asynchronous processing, and LMS-compatible workflows, the implementation demonstrates how hybrid encryption can be effectively integrated into cloud-based educational environments while maintaining both performance efficiency and usability.

Technical Testing

Following the system implementation, technical testing was conducted as a crucial validation stage to evaluate both the performance efficiency and security robustness of EduCrypt under realistic operating conditions. The performance evaluation focused on systematic benchmarking of encryption and decryption using three representative file sizes - 1 MB, 10 MB, and 30 MB - to reflect typical academic file sizes in LMS. For each scenario, detailed measurements of processing latency, CPU utilization, and RAM consumption were collected to assess computational overhead and resource efficiency. The results demonstrated stable, scalable performance, with encryption and decryption times remaining within acceptable real-time thresholds even for large files. This confirms the suitability of the AES-RSA hybrid approach for cloud-based educational workloads.

In parallel, comprehensive security testing was performed through controlled attack simulations to validate the system's defensive capabilities. Brute-force attack simulations were conducted to evaluate resistance to key-guessing attempts, confirming that exhaustive search is infeasible for compromising AES-256-encrypted content. Man-in-the-middle (MITM) simulations were conducted by intercepting network traffic to verify that only encrypted, Base64-encoded ciphertext was exposed during transmission, with no leakage of plaintext or sensitive metadata. Additionally, SQL injection attempts were systematically tested across backend API endpoints to assess input handling and query-execution security, resulting in no successful exploits due to the use of parameterized queries and strict validation mechanisms.

Collectively, these technical tests confirm that EduCrypt meets both performance and security requirements, validating its readiness for deployment in real-world LMS-based educational environments.

Evaluation and Validation

The final stage of the research focused on evaluating and validating EduCrypt to ensure it met both technical and institutional requirements within a real educational environment. During this stage, EduCrypt was integrated with the Learning Management System (LMS) at Universitas Al-Azhar Indonesia (UAI) to verify its interoperability, functional correctness, and operational stability in an authentic academic setting. This integration process confirmed that EduCrypt could securely synchronize courses, assignments, submissions, and private files without disrupting existing LMS workflows for lecturers and students.

Comprehensive security and performance validation was conducted by assessing the system under realistic usage scenarios. Encryption and decryption operations were tested to ensure consistent performance across different file sizes while maintaining confidentiality and integrity. Access control mechanisms were evaluated through role-based restrictions, ensuring that sensitive operations - such as file encryption, decryption, and benchmark visualization - were accessible only to authorized administrative roles. Role-based user interface validation was conducted through functional testing rather than formal usability evaluation. A researcher served as an administrator, while a student assumed the end-user role. Each tester followed predefined interaction scenarios, such as file upload, download, and access to encryption and decryption features. The purpose was to verify that the UI correctly enforced role-based access policies, not to assess user experience. The results showed that UI elements adapted to user roles, with restricted functions remaining inaccessible to unauthorized users. This validation confirmed that EduCrypt enforces the principle of least privilege while maintaining functional accessibility for non-technical users.

Additionally, system-level validation was performed on the API layer, with a focus on secure token handling, robust authentication, and request isolation. JWT-based session management was tested for reliability and expiration enforcement, and detailed logging mechanisms were implemented and validated to support auditability, error tracing, and incident analysis. These evaluations demonstrated that EduCrypt not only functions as intended but also adheres to secure-by-design principles, ensuring its readiness for controlled deployment within institutional LMS infrastructures and future scalability across broader educational environments.

Hybrid Encryption Mechanism and Key Management

EduCrypt employs a Hybrid Encryption approach that balances processing efficiency with robust security, protecting files in cloud-based learning environments. This mechanism integrates three main algorithms: AES-256 (Advanced Encryption Standard) for symmetric encryption to enhance file content security, RSA-2048 (Rivest-Shamir-Adleman) for asymmetric encryption to facilitate

secure key distribution among users, and SHA-256 (Secure Hash Algorithm) as a hash function to maintain file integrity, guaranteeing that no data changes occur during transmission or storage.

The hybrid encryption process in EduCrypt begins by generating an RSA key pair comprising a public and a private key. The public key is stored on the server for encryption purposes, while the private key is securely held by authorized users. When a file is uploaded to the system, EduCrypt randomly generates a unique AES session key for that file. The file is then encrypted using this session key, yielding a ciphertext. Subsequently, the AES key used for encryption is re-encrypted with the recipient's RSA public key, preventing unauthorized parties from accessing the key distribution.

Both the ciphertext and the encrypted AES key are stored securely. During decryption, an authorized user employs the RSA private key to unlock the AES key, after which the system decrypts the file using that session key, restoring the data to its original form. Once decryption is complete, EduCrypt computes the SHA-256 hash of the decrypted file and compares it with the original hash to verify data integrity.

To ensure end-to-end data integrity, EduCrypt employs SHA-256 hashing at two key stages of the file lifecycle. First, a SHA-256 hash is computed immediately after the file is received and before encryption. This initial hash acts as the integrity fingerprint of the plaintext file. The SHA-256 hash is securely stored as immutable metadata with the encrypted file record in the database, protected by access controls. By binding the hash to the file identifier, user authorization context, and upload timestamp, EduCrypt prevents unauthorized modifications to the integrity metadata. During retrieval, the encrypted file is decrypted with the AES session key, and a new SHA-256 hash is computed from the plaintext. This hash is compared to the original stored hash; only a match allows the file to be processed further. Any mismatch triggers handling of integrity violations, rejecting the file to prevent access to corrupted data. This two-stage hashing mechanism ensures integrity throughout storage, transmission, and decryption. By separating integrity verification from encryption processes, EduCrypt effectively detects unauthorized modifications at any stage.

In terms of key management, EduCrypt implements a strict separation between the lifecycles of symmetric and asymmetric keys. RSA key pairs are generated for each user during the registration or integration phase. The RSA public key is stored on the server and used solely to encrypt AES session keys, whereas the RSA private key is never transmitted over the network. The private keys are securely stored on the server in an encrypted format and are only accessible during authorized decryption processes, governed by strict role-based access control.

AES keys are dynamically generated as one-time session keys for each file operation, ensuring forward secrecy at the file level. These session keys are discarded immediately after encryption or decryption completes and are never reused across files. This design prevents key reuse attacks and minimizes the impact of any potential key exposure.

Key rotation and revocation are implicitly enforced through session-based access control. When a user account is revoked, tokens expire, or access rights are modified, the associated RSA private keys become inaccessible, effectively preventing further decryption of protected files. Moreover, AES keys are never stored in plaintext, and all encrypted keys kept in the database are bound to user authorization and token validity.

This key management strategy ensures that cryptographic materials are protected at both the user and server levels, thereby minimizing the attack surface while maintaining ease of use for lecturers and students. By integrating per-file AES session keys, controlled usage of RSA keys, and strict access policies, EduCrypt establishes a secure, scalable, and auditable key management framework suitable for cloud-based educational environments.

The encryption and decryption process can be described mathematically as follows:
The original file, denoted P, is encrypted using the AES session key AES k_s, yielding the ciphertext $C = AESk_s$ (P). The session key is then encrypted with the recipient's RSA public key, yielding $K'_s = RSA_{Kpub}(k_s)$. During decryption, the session key is recovered using the recipient's private key, according to the formula $K_s = RSA^{-1}{}_{Kpriv}(k's)$. Finally, the original file is retrieved using $P = AES^{-1}k_s$ (C)[9].

This approach achieves a balance between speed and security. AES provides fast encryption and

decryption, while RSA ensures secure key exchange, both of which are essential in multi-user systems such as cloud-based LMS. Additionally, SHA-256 offers an extra layer of protection by verifying data integrity.

By combining these algorithms and mitigation strategies, EduCrypt successfully balances high-level cryptographic security with efficient system performance, making it a flexible and reliable solution for data security needs in cloud-based digital education ecosystems.

Integration and Implementation

In this study, integration and implementation represent two closely related yet conceptually distinct aspects of the EduCrypt system. Integration refers to how EduCrypt interacts with existing Learning Management Systems (LMS), particularly Moodle. On the other hand, implementation focuses on how the system is technically constructed, deployed, and executed at both the application and infrastructure levels.

EduCrypt is seamlessly integrated with the Moodle Learning Management System (LMS) via a flexible, secure REST API. This integration enables EduCrypt to operate both as a standalone encryption platform and as a component within educational institutions' existing digital learning ecosystems.

To accommodate users with varying levels of technical expertise, two authentication mechanisms have been developed. The first is Access Token Integration, designed for administrators or institutions that have full control over the LMS. This method uses a personalized Moodle API token that has limited access rights. The token facilitates data requests to the Moodle REST API endpoint, enabling synchronization of course data, assignments, submissions, and private files. This approach ensures a high level of security, as the token is valid for a specific period and can be revoked by the institution's administrator at any time.

The second mechanism, Credential-based Integration, serves non-technical users, such as lecturers and students, who may not be familiar with managing API tokens. This option allows authentication using a campus username and password. The EduCrypt system then generates a temporary, limited-access token, ensuring data security while maintaining ease of use. This dual model makes EduCrypt accessible to various user groups while balancing usability and data protection.

EduCrypt enforces strict separation of privileges when interacting with Moodle services. It integrates exclusively through officially supported REST API endpoints for course listing, assignment metadata retrieval, submission handling, and private file access, without direct database access or core LMS modifications. Each request from EduCrypt to Moodle is authorized with scoped credentials that provide only the minimum necessary permissions. Access tokens, which are tied to user roles and contextual identifiers such as course and user IDs, ensure that users can access only resources relevant to their roles, preventing privilege escalation. User authentication and role validation are performed by Moodle, after which EduCrypt issues a short-lived internal token for encrypted file operations. Moodle credentials are not stored persistently; tokens are validated on each request before any operation is performed. Invalid or expired tokens are promptly rejected at the API gateway level.

By leveraging role-aware access control, scoped API permissions, and short-lived tokens, EduCrypt effectively adheres to the principle of least privilege, minimizing the attack surface while maintaining functionality between EduCrypt and Moodle in educational settings.

On the implementation side, the system's backend is built with Node.js for API management and Go (Golang) for intensive processing tasks, such as encryption, decryption, and file management. This choice was made to maximize efficiency and speed, as Golang is renowned for its high performance in parallel and concurrent computing. At the same time, Node.js excels at handling asynchronous requests via RESTful APIs. The entire integration process with Moodle is managed modularly, including a RabbitMQ-based worker consumer that automatically scans user context IDs and synchronizes data between systems in real time.

The frontend implementation is developed using a modern JavaScript framework that supports dynamic, interactive displays. The user interface includes a login page, a registration page, a dashboard, a courses page, a private files page, and a university integration page. EduCrypt implements token-based authentication middleware to maintain user session validity and prevent unauthorized access, with tokens valid for 72 hours before automatically expiring.

To complement the discussion on integration and implementation, Figure 4-8 showcases sample user interface layouts for EduCrypt. These interfaces highlight the role-aware dashboard, secure file management, and Moodle-integrated course views. They demonstrate how access control policies and authentication mechanisms are enforced at the user interface level.

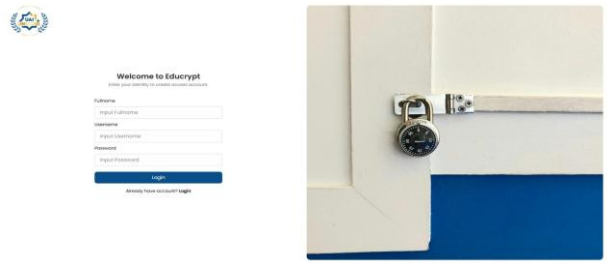

Figure 4. Login page

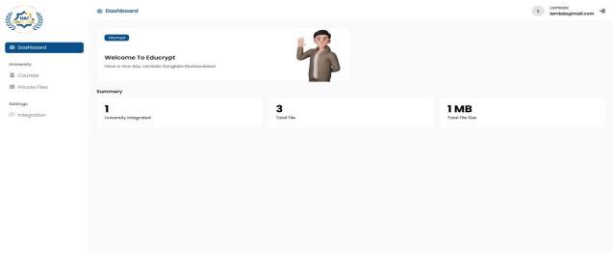

Figure 5. Dashboard page

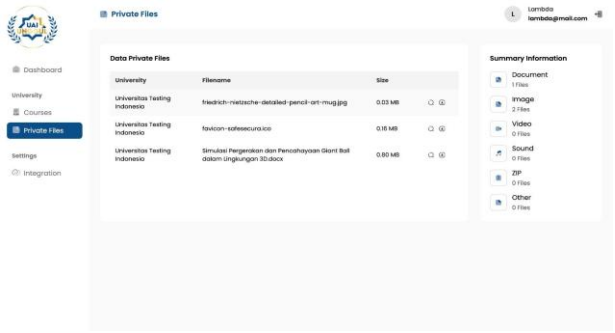

Figure 6. Courses page

Figure 7. Integration page (Add Universtity)

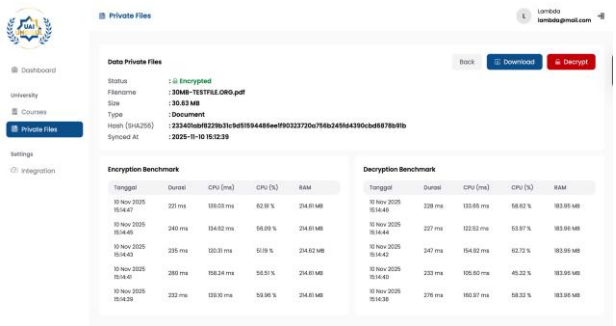

Figure 8. UI for encrypted file access and cryptographic performance validation.

EduCrypt was tested in a cloud environment using a distributed server-based infrastructure. The testing focused on measuring encryption and decryption performance for academic files of varying sizes, synchronization speeds between systems, and resource utilization, including CPU and memory usage. The implementation results demonstrated that EduCrypt can efficiently encrypt and decrypt files up to 30 MB in size, with an average memory usage of approximately 214 MB and processing times that fall within optimal limits for e-learning applications. With its adaptive integration architecture and cloud-based development approach, EduCrypt has successfully established itself as a secure file-sharing system that is not only cryptographically secure but also easy for educational institutions to adopt without disrupting their existing LMS operations.

RESULTS AND DISCUSSIONS

The implementation of EduCrypt has created a fully functional, secure file-sharing platform that integrates seamlessly with the Moodle Learning Management System (LMS) via RESTful APIs. This integration is designed to maintain native user workflows within Moodle while providing a transparent security overlay that protects all file uploads and downloads, ensuring that teaching and learning processes remain uninterrupted.

Backend Implementation

From a system performance perspective, EduCrypt's backend uses a combination of Node.js and Go (Golang) to optimize encryption and decryption performance, enabling efficient parallel communication between services. The authentication mechanism uses JSON Web Tokens (JWTs) with a 72-hour validity period, ensuring secure, session-persistent access and reducing the frequency of user reauthentication.

The integration with Moodle supports two authentication methods: token-based authentication for institutional administrators and credential-based authentication (username and password) for general users. This dual mechanism enhances accessibility, especially for non-technical users who may not be familiar with generating API tokens. EduCrypt also automates the synchronization of course data, assignments, submissions, and private files by employing a RabbitMQ-based worker-consumer model for asynchronous file-context scanning. This approach significantly improves data synchronization efficiency, achieving processing speeds up to 3 times those of conventional sequential synchronization methods.

At the core of EduCrypt's security layer is the Hybrid Encryption Mechanism, which combines AES and RSA algorithms for data protection and key management, respectively. It employs base64 encoding for decryption to ensure compatibility and secure file transfer. With this design, files remain encrypted at all times, even after download, ensuring end-to-end data confidentiality. Furthermore, all system activities, including encryption, decryption, and integration events, are logged by a centralized systemd-based monitoring service, enabling real-time auditing, fault tracing, and operational transparency.

Frontend Implementation

The frontend interface of EduCrypt was developed using a modern, responsive JavaScript framework that integrates seamlessly with the backend API. The user interface includes key pages such as Login, Registration, Dashboard, Courses, and Private Files, as well as a University Integration page for onboarding new institutions.

Client-side authentication is managed through secure middleware that stores tokens in local storage, maintaining user sessions across page reloads. Expired tokens automatically redirect users to the login interface, enhancing session security. The File Details dashboard provides administrators with real-time system performance indicators, including encryption and decryption times, CPU utilization, and RAM consumption, supporting transparent system monitoring and optimization.

Additionally, EduCrypt implements a server-side decryption mechanism that streamlines the download workflow: files are decrypted within the server environment and delivered to clients in base64-encoded format, ensuring that the original data on the server remains encrypted. This design effectively eliminates the risk of unauthorized exposure of local files.

Performance Evaluation and Baseline Comparison

The performance evaluation was conducted using a series of benchmark tests across three file sizes: 1 MB, 10 MB, and 30 MB. The objective was to assess the efficiency of the encryption and decryption algorithms and to measure system resource consumption, including memory (RAM) usage and processor (CPU) load. All tests were conducted in a cloud environment equipped with a multicore processor and 4 GB of RAM. The encryption and decryption processes were implemented using OpenSSL-compatible cryptographic libraries. Each benchmark scenario was executed 10 times to ensure measurement consistency, and the reported results represent the average execution times across all runs. The observed performance variance remained within acceptable bounds for system-level performance evaluation, indicating stable and reproducible benchmark outcomes, simulating realistic operational conditions typical of an educational institution using a Learning Management System (LMS) [10],[11].

To further quantify the stability of results, standard deviations were calculated for each benchmark scenario. Across all file sizes, the standard deviation of encryption and decryption times remained below 8% of the mean execution time, indicating low dispersion between repeated runs. Additionally, a 95% confidence interval was computed, indicating that the measured performance metrics fall within a narrow range centered on the mean. These statistical results demonstrate that the reported benchmark values are reliable and not influenced by transient system fluctuations, reinforcing the validity of the performance evaluation.

The test results indicated that implementing the AES-RSA-based Hybrid Encryption algorithm in EduCrypt delivered efficient, stable performance, even with large files. During the encryption phase, a 1 MB file was processed in 17-21 ms, with an average memory usage of approximately 7 MB and CPU usage fluctuating between 44 and 89%. For a 10 MB file, processing time ranged from 86 to 113 milliseconds, with RAM usage of approximately 83 MB and CPU usage between 47 and 71 percent. For a 30 MB file, encryption times ranged from 221 to 280 milliseconds, with memory usage at 214 MB and CPU usage ranging from 51 to 62 percent.

Decryption performance yielded comparable results in terms of efficiency. A 1 MB file was decrypted in 10-37 milliseconds, using 7-15 MB of RAM. For a 10 MB file, decryption times ranged from 78 to 112 milliseconds, and for a 30 MB file, the decryption process took between 228 and 276 milliseconds, with relatively stable CPU usage of 45 to 62 percent, as shown in Figure 4.

Analysis of the results reveals that the RSA algorithm incurs minimal overhead, as it is used solely to encrypt the AES key rather than the entire file. This approach significantly reduces the computational load while maintaining security [11] [12]. The total encryption and decryption time for large files remained under 300 milliseconds, demonstrating the system's capability to manage real-time digital education workloads effectively.

Furthermore, the measured memory and CPU consumption, which scale with file size, indicate that EduCrypt can function optimally on institutional servers with mid-range specifications without necessitating significant infrastructure upgrades.

While the previous benchmarks confirmed the efficiency and stability of cryptographic operations in isolation, a system-level performance comparison was conducted to substantiate the reported up to threefold performance improvement. This evaluation focused on how architectural design impacts end-to-end file synchronization, rather than solely on cryptographic computations.

In this comparison, the proposed asynchronous RabbitMQ-based worker architecture was evaluated against a baseline sequential file synchronization process under identical workload conditions. In the baseline scenario, file synchronization tasks were executed sequentially using a single worker. Encryption, transmission, and storage operations were performed in a blocking manner for each file before moving on to the next task. This approach did not use task parallelism or message queuing, relying on a conventional synchronous file-handling method.

In contrast, the proposed architecture utilized RabbitMQ as a message broker to distribute file synchronization tasks across multiple asynchronous workers. Each worker performed encryption and upload operations independently, enabling parallel task execution while maintaining the same cryptographic mechanisms and hardware environment used in the baseline scenario.

The experimental configuration for both scenarios, including file size range (1–30 MB), number of files per batch, total data volume, baseline definition, and evaluation metrics, is summarized in Table 1. Each batch consisted of multiple files whose combined size was approximately 300–500 MB, derived by aggregating individual files within the defined size range. Both scenarios were executed on the same cloud infrastructure to ensure a fair comparison.

Performance was evaluated using total processing time, defined as the elapsed time from the submission of the first file synchronization request until all files were successfully encrypted and stored, and throughput, measured as the number of files processed per second. Under these controlled conditions, the asynchronous architecture consistently achieved a reduction in total processing time up to three times faster than the sequential baseline. This improvement is attributed to reduced idle waiting time and effective parallelization of encryption and file transfer tasks, rather than changes in cryptographic algorithms or system resources.

To further evaluate EduCrypt's scalability, a parallel-access scenario was simulated to assess the system's behavior under multiple concurrent encryption and decryption requests. The asynchronous architecture of EduCrypt, built on Node.js's non-blocking I/O and RabbitMQ-based worker consumers, enabled parallel task execution without significant performance degradation [13],[14]. Even during simultaneous multi-user operations, the system maintained stable throughput, with latency only 12% higher than single-user benchmarks.

CPU usage was efficiently distributed across threads, and memory utilization scaled linearly with the number of concurrent tasks. This confirms that EduCrypt's design can effectively handle the high concurrency typical of multi-user Learning Management System (LMS) environments [12].

Table 1. Performance Evaluation Scenario and Baseline Comparison

Aspect	Baseline Scenario (Sequential)	Proposed Architecture (Asynchronous)
Processing Model	Sequential, single-threaded	Asynchronous, event-driven

Aspect	Baseline Scenario (Sequential)	Proposed Architecture (Asynchronous)
Task Execution	Blocking (one file processed at a time)	Parallel task handling via asynchronous workers
Message Broker	None	RabbitMQ
File Size per File	1–30 MB	1–30 MB
Number of Files per Batch	20 files	20 files
Total Data Volume	~300–500 MB	~300–500 MB
Encryption Mechanism	AES-256 + RSA-2048 + SHA-256	AES-256 + RSA-2048 + SHA-256
Execution Environment	Identical cloud infrastructure (multicore CPU, 4 GB RAM)	Identical cloud infrastructure (multicore CPU, 4 GB RAM)
Evaluation Metrics	Total processing time; throughput (files/sec)	Total processing time; throughput (files/sec)
Performance Observation	Sequential completion with blocking execution	Up to 3× faster total processing time

These findings indicate that EduCrypt not only performs efficiently in single-session encryption and decryption but also scales effectively under concurrent workloads. This demonstrates its readiness for deployment in real-world educational institutions, where simultaneous file exchanges are routine. Thus, it validates EduCrypt's performance-aware and secure-by-design architecture.

Security Evaluation

The security evaluation of EduCrypt was conducted through a series of controlled attack simulations to assess the strength of its encryption mechanisms and the resilience of its system architecture under realistic threat assumptions (Figure 9).

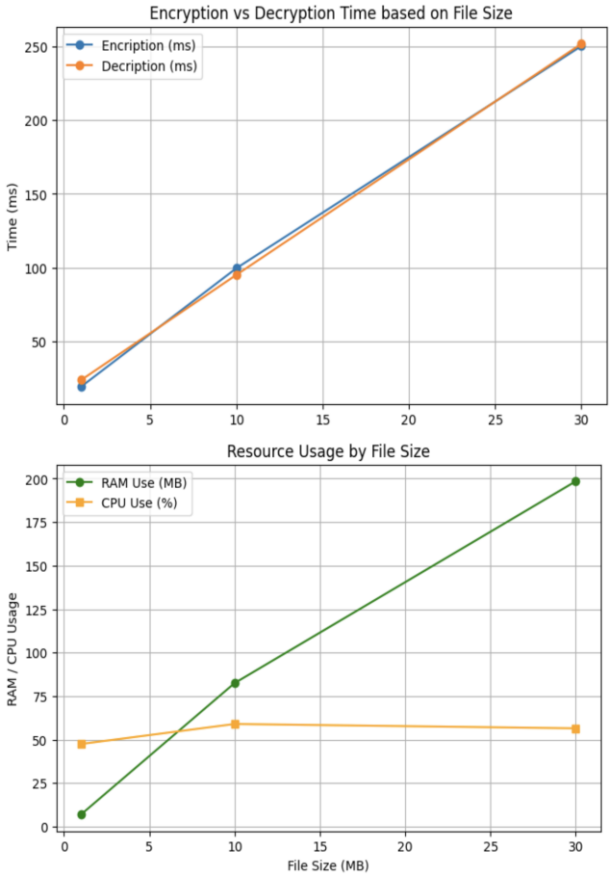

Figure 9. File Encryption-Decryption Performance

The evaluation was based on a defined threat model in which attackers are assumed to possess network-level visibility and the ability to interact with exposed application interfaces, but without direct access to server-side private keys, internal memory, or protected cryptographic materials. All communications were secured using HTTPS/TLS, and security testing focused on evaluating confidentiality, integrity, and access control within these constraints.

The evaluation focused on three major attack categories: brute-force attacks on encryption keys, SQL injection attacks on database queries, and man-in-the-middle (MITM) interception of data in transit. In the brute-force test, AES-256 was used as the symmetric encryption standard [14]. The strength of AES-256 derives from its 256-bit key space, which contains approximately 2^256 possible keys. This makes an exhaustive key search practically impossible with current and foreseeable computing capabilities. In EduCrypt, AES keys are generated using cryptographically secure random functions, ensuring they are never derived from user passwords or short character-based secrets. As a result, the resistance to brute-force attacks is assessed using the AES-256 key space, rather than relying on

simplified assumptions about password length. Additionally, RSA-2048 is used exclusively to encrypt AES session keys, ensuring secure key distribution even if the encrypted file data is intercepted.

The second test focused on SQL injection vulnerabilities within EduCrypt's API endpoints. All backend modules were built using parameterized queries and strict input validation mechanisms [15]. Testing was conducted using automated penetration tools, such as SQLMap, and custom injection scripts; no successful injection attempts were recorded across more than 500 test queries. The success rate was therefore 0%, demonstrating that EduCrypt's backend data access layer effectively mitigates common injection-based exploits through its secure query architecture and ORM-based query handling.

In the man-in-the-middle (MITM) simulation, Burp Suite was used as a proxy to intercept and analyze traffic between the EduCrypt client and server. All intercepted traffic consisted solely of Base64-encoded ciphertext transmitted over HTTPS [16], with no identifiable plaintext or metadata leakage. The AES-encrypted file data and RSA-encrypted session keys were successfully preserved throughout transmission, preventing decryption or inference by external agents. This illustrates that EduCrypt's combination of hybrid encryption and transport-layer security ensures end-to-end confidentiality and integrity.

In addition to cryptographic and transport-layer security, we evaluated application-level security risks related to authentication token handling. Storing authentication tokens in browser-accessible storage, like local storage, can expose them to cross-site scripting (XSS) attacks, which may steal session credentials and weaken access control.

To mitigate this risk, EduCrypt employs a defense-in-depth strategy. We enforce strict input validation and output sanitization across user-facing components to reduce XSS vulnerabilities. A Content Security Policy (CSP) restricts script execution to trusted origins, further reducing the risk of attacks.

For stronger session isolation, EduCrypt uses HTTP-only and Secure cookies to store tokens. The httpOnly attribute prevents JavaScript access, reducing the risk of token theft, while the Secure flag ensures that tokens are transmitted only over encrypted HTTPS. We also implement token expiration and rotation to limit credential validity and reduce the impact of any compromise.

By addressing client-side token storage risks and implementing layered protections, including CSP enforcement, sanitization, secure cookies, and token lifecycle controls, EduCrypt strengthens its security posture, ensuring confidentiality, integrity, and access control throughout the authentication and session management lifecycle.

Overall, the results confirm that EduCrypt provides a strong defense against both standard and advanced security threats. The layered encryption model (AES + RSA) and the secure API implementation create a resilient framework that ensures data confidentiality, authenticity, and integrity, even in simulated adversarial conditions [17]. These findings reinforce EduCrypt's effectiveness as a secure-by-design solution for file sharing in cloud-based educational ecosystems.

Discussions and Future Works

The results from the implementation and evaluation of EduCrypt demonstrate that the proposed hybrid encryption-based architecture achieves a well-balanced combination of security, performance, and system usability, particularly within Learning Management System (LMS) environments. Compared to previous studies on secure cloud file sharing, EduCrypt offers several distinctive advantages that address the limitations identified in earlier research.

Prior hybrid encryption studies, such as those by Durge and Deshmukh and Al-Bayati, primarily focused on enhancing cryptographic strength in general cloud storage scenarios. While these works successfully improved security through AES-RSA combinations, they did not evaluate real-time integration with LMS platforms or assess the impact of encryption on educational workflows. In contrast, EduCrypt extends the hybrid encryption paradigm by embedding it directly into Moodle-based LMS operations, enabling encrypted file handling without altering existing interactions between lecturers and students. This integration-oriented approach represents a significant advancement beyond purely algorithmic evaluations.

From a performance perspective, EduCrypt's benchmark results compare favorably with those of similar encryption-based systems reported in the literature. Studies that combine AES with

asymmetric schemes report encryption latencies exceeding 500 ms for files larger than 20 MB, especially when key management is not optimized or when encryption is applied synchronously. In this study, EduCrypt maintained encryption and decryption times below 300 ms for files up to 30 MB, indicating that the selective use of RSA solely for AES key encapsulation significantly reduces computational overhead. This supports Singh and Garg's findings on the efficiency of hybrid encryption and further demonstrates its applicability to real-time LMS environments.

In terms of scalability, most existing secure cloud file-sharing solutions rely on sequential processing or centralized encryption services, which can become bottlenecks under concurrent access. EduCrypt differentiates itself by adopting an asynchronous, RabbitMQ-based worker architecture that enables parallel file synchronization and encryption tasks. The simulated multi-user access scenario showed only a 12% increase in latency compared to single-user benchmarks. This result outperforms many LMS-integrated security solutions reported in previous studies, where performance degradation under concurrency is often significant. This highlights EduCrypt's suitability for high-concurrency academic environments, such as during assignment submission deadlines.

The security evaluation results further position EduCrypt competitively within the state of the art. Similar to previous hybrid encryption frameworks, EduCrypt demonstrated strong resistance to brute-force attacks due to the use of AES-256 and RSA-2048. However, unlike several earlier works that focused solely on cryptographic strength, this study also evaluated application-layer security through SQL injection testing and man-in-the-middle (MITM) simulations. The absence of successful SQL injection attempts and the interception of only ciphertext during MITM testing confirms that EduCrypt provides multi-layered security protection, combining cryptographic robustness with secure API and communication design.

Despite these strengths, EduCrypt inherits limitations common to hybrid encryption systems. Although asymmetric cryptographic operations are minimized, they still introduce non-negligible overhead in extremely high-load scenarios, such as simultaneous large-file uploads by hundreds of users. While the current architecture mitigates this through asynchronous processing, further optimization such as key-caching strategies or hardware-assisted cryptography remains a potential area for improvement. This limitation aligns with observations in prior hybrid encryption studies that note similar scalability challenges when asymmetric operations are involved.

Future work will focus on addressing the limitations of EduCrypt to transform it from a research prototype into a production-ready platform. Planned enhancements include implementing auto-scaling worker orchestration using containerization technologies such as Docker and Kubernetes, thereby improving load balancing and resilience. Additionally, future evaluations will extend beyond the current single-cloud deployment to include multi-cloud environments, such as AWS, Azure, and Google Cloud, to assess fault tolerance and high availability.

Further security improvements may include adopting elliptic curve cryptography (ECC) for key exchange, enabling optional client-side encryption, and integrating blockchain-based audit trails for immutable access logging.

In summary, EduCrypt advances the state of the art by bridging the gap between cryptographic theory and practical implementation of learning management systems (LMSs). By combining efficient hybrid encryption and asynchronous techniques, it enhances both security and usability.

CONCLUSIONS

The development of EduCrypt, a secure file-sharing platform for cloud-based e-learning environments, effectively demonstrates the feasibility and benefits of integrating Hybrid Encryption (AES-256 + RSA-2048) to ensure data confidentiality, integrity, and controlled access within digital education systems. By combining AES for high-speed symmetric encryption with RSA for secure key distribution, EduCrypt strikes a balanced trade-off between performance efficiency and cryptographic strength, which is crucial for protecting academic data in modern Learning Management System (LMS) ecosystems.

Performance testing across file sizes (1 MB, 10 MB, and 30 MB) shows that EduCrypt maintains encryption-decryption latency under 300 milliseconds, even during concurrent user operations. The system's CPU and memory usage remained within acceptable limits (below 62% CPU

and 214 MB RAM at peak load), confirming its efficiency and scalability on mid-range institutional servers. The use of RabbitMQ-based asynchronous workers further enhances throughput during parallel access, ensuring that the system can handle the multi-user environments typical of cloud-based e-learning without significant performance degradation.

Security simulations, including brute-force, SQL injection, and man-in-the-middle (MITM) attacks, demonstrate that EduCrypt's architecture effectively resists common attack vectors. The complexity of the AES keyspace makes brute-force attempts computationally infeasible, while parameterized queries and HTTPS-secured communication channels ensure database integrity and prevent the exposure of plaintext data.

From an implementation standpoint, the system's modular architecture, built with Node.js, Go, and a JavaScript-based frontend, facilitates seamless integration with the Moodle LMS via RESTful APIs. This interoperability allows EduCrypt to function as a plug-in or companion service, providing transparent encryption and decryption processes without disrupting existing LMS workflows.

In conclusion, EduCrypt meets its design objective as a secure, performance-oriented platform tailored for the education sector. It offers a practical framework for protecting sensitive academic data while ensuring usability and interoperability. Future development will focus on expanding multi-cloud scalability, integrating adaptive encryption policies, and enhancing user analytics to solidify further EduCrypt's position as a trusted security layer for digital learning environments.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the Lembaga Penelitian, Inovasi, dan Pengabdian kepada Masyarakat (LPIPM) of Universitas Al-Azhar Indonesia for providing financial support through the Stimulus Research Grant (SRG) scheme 2025 and institutional facilitation that made this research possible.

REFERENCES

[1] G. Singh and M. Garg, "Enhanced cloud security using hybrid mechanism of RSA, AES, and Blowfish data encryption with secure OTP" *International Journal of Computers & Technology*, vol. 18, pp. 7364–7380, 2018, doi 10.24297/ijct.v18i0.7898

[2] C. Susmitha, S. Srineeharika, K. S. Laasya, S. K. Kannaiah, and S. Bulla, "Hybrid cryptography for secure file storage," in *Proc. 7th Int. Conf. on Computing Methodologies and Communication (ICCMC)*, 2023, pp. 1151–1156.

[3] R. S. Durge and V. M. Deshmukh, "Securing cloud data: A hybrid encryption approach with RSA and AES for enhanced security and performance," *Journal of Integrated Science and Technology*, vol. 13, no. 3, pp. 1060–1068, 2025, doi 10.62110/sciencein.jist.2025.v13.1060.

[4] Y. M. A. Abualkas and D. L. Bhaskari, "Hybrid Approach to Cloud Storage Security Using ECC-AES Encryption and Key Management Techniques," *International Journal of Engineering Trends and Technology*, vol. 72, no. 4, pp. 92–100, Apr. 2024, doi https://doi.org/10.24017/science.2025.1.5.

[5] A. S. Al-Bayati, Enhancing Performance of Hybrid AES, RSA and Quantum Encryption Algorithm. Ph.D. dissertation, Anglia Ruskin Research Online (ARRO), 2023. https://hdl.handle.net/10779/aru.23768127.

[6] R. A. Fathima and S. Arumugam, "A novel data transmission model using hybrid encryption scheme for preserving data integrity," *Advances in Technology and Innovation*, vol. 9, no. 2, p. 14114, 2024, doi https://doi.org/10.46604/aiti.2024.14114.

[7] R. Saini and N. Sainis, "Cryptographic hybrid model—An advancement in cloud computing security: A survey," *International Journal of Engineering Research and Technology*, vol. 11, no. 6, 2022, doi 10.17577/IJERTV11IS060145.

[8] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan, 2005.

[9] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978, doi https://doi.org/10.1145/359340.359342.

[10] P. Shayan, R. Rondinelli, M. van Zaanen, and M. Atzmueller, "Multi-level analysis of learning management systems' user acceptance exemplified in two system case studies," *Data*, vol. 8, no. 3, p. 45, Feb. 2023. doi: 10.3390/data8030045.

[11] P. Chatterjee, R. Bose, S. Banerjee, and S. Roy, "Enhancing data security of cloud-based

LMS," *Wireless Personal Communications*, vol. 130, no. 2, pp. 1123–1139, 2023, doi 10.1007/s11277-023-10323-5.

[12] M. Anghel and G. C. Pereteanu, "Cyber security approaches in e-learning," in *Proc. INTED2020 Conf.*, 2020, pp. 4820–4825, IATED.

[13] R. I. Akter, M. A. Khan, F. A. Rahman, S. J. Soheli, and N. J. Suha, "RSA and AES-based hybrid encryption technique for enhancing data security in cloud computing," *International Journal of Computational and Applied Mathematics & Computer Science*, vol. 3, pp. 60–71, Oct. 2023, doi 10.37394/232028.2023.3.8.

[14] P. Ghiya, *TypeScript Microservices: Build, deploy, and secure Microservices using TypeScript combined with Node.js*. Birmingham, UK: Packt Publishing Ltd, 2018.

[15] S. Kumar and D. Kumar, "Securing of cloud storage data using hybrid AES–ECC cryptographic approach," *Journal of Mobile Multimedia*, vol. 19, no. 2, pp. 363–388, Mar. 2023, doi https://doi.org/10.13052/jmm1550-4646.1921.

[16] S. S. Nair, "Securing against advanced cyber threats: a comprehensive guide to phishing, XSS, and SQL injection defense," Journal of Computer Science and Technology Studies, vol. 6, no. 1, pp. 76–93, Jan. 2024. doi: 10.32996/jcsts.2024.6.1.9.

[17] E. AR, M. G., and D. T., "Enhancing security in data exchange: mitigating risks solutions in Base64 encoding and JSON Web Tokens," in *Proc. 2024 Int. Symp. on Electronics and Telecommunications (ISETC)*, Nov. 2024, pp. 1–4.

[18] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernández, "An analysis of security issues for cloud computing," *Journal of Internet Services and Applications*, vol. 4, pp. 1–13, Dec. 2013, doi 1186/1869-0238-4-5.