Zero : Jurnal Sains, Matematika, dan Terapan
E-ISSN : 2580-5754; P-ISSN : 2580-569X
Volume 9, Number 3, 2025
DOI: 10.30829/zero.v9i3.26827
Page: 1031-1042

RSA-AES Cryptosystem with Auto-Key Rotation for Cloud Storage

[1] Azanuddin
Departement of Computer Engineering and Informatics, Politeknik Negeri Medan, Indonesia

[2] Asyahri Hadi Nasyuha
Faculty of Information Technology, Universitas Teknologi Digital Indonesia, Indonesia

[3] Ikhwan Ruslianto
Faculty of Mathematics and Natural Sciences, Universitas Tanjungpura, Indonesia

[4] Moch. Iswan Perangin Angin
Faculty of Computer Science and Information Technology, Budi Darma University, Indonesia

[5] Moustafa H. Aly
Arab Academy for Science Technology and Maritime Transport, Egypt

[6] Moses Adeolu Agoi
Lagos State University of Education, Nigeria

Article Info

Article history:

Accepted 26 December 2025

Keywords:

Auto-Rotation Key;
Cloud Storage;
Hybrid Cryptosystem;
Key Lifecycle Modeling;
RSA–AES.

ABSTRACT

The widespread adoption of cloud storage systems has increased the demand for cryptographic mechanisms that ensure data confidentiality while limiting security risks associated with static and long-lived encryption keys. Although hybrid RSA–AES schemes are commonly employed to balance security and computational efficiency, key management—particularly autonomous and quantitatively bounded key rotation—remains insufficiently formalized. This study proposes a hybrid RSA–AES cryptosystem equipped with an autonomous auto-key rotation mechanism defined through explicit analytical constraints. AES-256 is employed for bulk data encryption, while RSA-2048 is used for secure encapsulation of symmetric session keys. Key renewal is governed by inequality-based conditions on elapsed time ($\Delta t \leq 30$ minutes) and encryption usage ($n \leq 10$ operations), yielding a mathematically bounded key lifecycle without manual intervention or external infrastructure. System performance and operational security properties are evaluated in a simulated cloud environment using file sizes ranging from 100 KB to 10 MB. Quantitative metrics include encryption and decryption time complexity, computational overhead relative to AES-only encryption, key variability measured by Hamming distance, and data integrity verification using SHA-256. Experimental results demonstrate linear scalability and a stable average overhead of approximately 12.8%, indicating a bounded constant-factor cost independent of workload size. Successive AES-256 keys exhibit a mean Hamming distance of 127.42 bits, consistent with high key variability and effective key freshness. These findings show that analytically constrained key rotation enables controlled symmetric-key exposure while preserving practical efficiency overall.

Corresponding Author:

Azanuddin,
Departement of Computer Engineering and Informatics,
Politeknik Negeri Medan,
Email: azanuddin@polmed.ac.id

1. INTRODUCTION

The rapid adoption of cloud computing has fundamentally transformed how organizations and individuals store, access, and manage data. As a core component of cloud services, cloud storage provides scalability, accessibility, and cost efficiency[1][2]. However, its decentralized and shared architecture introduces significant security challenges, particularly in ensuring data confidentiality, integrity, and effective cryptographic key management. One of the most critical vulnerabilities arises from the continued use of static or long-lived encryption keys, which increases the risk of unauthorized access, prolonged key exposure, and large-scale data breaches.

Symmetric encryption algorithms such as the Advanced Encryption Standard (AES) are computationally efficient and well suited for encrypting large data volumes. Nevertheless, AES-based systems suffer from inherent key-management limitations, including secure key distribution and uncontrolled key reuse. In contrast, asymmetric cryptographic schemes such as the Rivest–Shamir–Adleman (RSA) algorithm provide secure key exchange but incur substantial computational overhead when applied to bulk data encryption. Consequently, hybrid cryptosystems combining AES for data encryption and RSA for key encapsulation have become a standard solution to balance efficiency and security[3]–[7]. Despite their widespread adoption, many existing RSA–AES hybrid implementations still rely on static or manually refreshed keys, leaving them vulnerable to brute-force attacks and long-term key compromise[8][9].

This study addresses these limitations by designing and evaluating a hybrid RSA–AES cryptosystem equipped with an autonomous key-rotation mechanism[10][11]. he proposed system ensures periodic and event-driven key renewal, reducing symmetric-key exposure while preserving acceptable performance for cloud storage environments. The main contributions of this work are twofold: (1) the development of a replicable hybrid cryptographic framework integrating AES-256 and RSA-2048 with automated key-lifecycle management, and (2) an empirical evaluation of the impact of autonomous key rotation on security-related indicators and computational performance. Recent studies have attempted to improve key management in hybrid encryption systems, yet important gaps remain. Dhamodharan (2023) proposed a dynamic RSA–AES scheme with manual key updates, which improves flexibility but still depends on administrator intervention, introducing security exposure during idle periods[12]. Khalaf and Sagheer (2025) incorporated blockchain technology into hybrid encryption to decentralize key management, achieving improved resilience at the cost of significant computational overhead, limiting applicability in real-time or resource-constrained environments [13]. Other works have explored ECC–AES hybrid models to reduce computational cost; however, the absence of automated key-rotation mechanisms restricts their ability to provide effective forward secrecy [14]. Time-based key-rotation strategies have also been proposed, but reliance on centralized synchronization servers introduces single points of failure and scalability constraints[15]. From an applied-mathematics perspective, these limitations reveal deficiencies in the formal modeling of cryptographic key lifecycles and their associated performance constraints. Many prior studies emphasize implementation and measurement without explicitly formulating key-rotation conditions, bounding key exposure analytically, or modeling overhead behavior. In contrast, this work frames key rotation as an analytically constrained process governed by inequality-based conditions on time and usage. Symmetric-key exposure is explicitly bounded as a function of elapsed time (Δt) and encryption count (n), while computational overhead is modeled as a constant-factor perturbation of baseline AES complexity. Additional mathematical descriptors, including Hamming-distance–based key variability metrics, are used to quantify key freshness in a reproducible manner.

The novelty of this research therefore lies not merely in integrating hybrid encryption with key rotation, but in the explicit formulation of an autonomous, timestamp- and usage-driven key-rotation model supported by analytical constraints and measurable performance bounds. By unifying formal key-lifecycle modeling with empirical evaluation, the proposed system provides a lightweight yet mathematically grounded approach for securing cloud storage environments, consistent with the scope of applied mathematics and computational science.

2. RESEARCH METHODE

This study adopts an experimental and analytical research methodology to design, formalize, and evaluate a hybrid RSA–AES cryptosystem equipped with an autonomous key-rotation mechanism for cloud storage security[16][17][18]. The methodology is structured into four main components.

2.1 System and Threat Model

The proposed system adopts a client-side encryption paradigm, in which all cryptographic operations—including key generation, encryption, decryption, and autonomous key rotation—are executed exclusively on the client. Under this model, cryptographic keys are never transmitted to or stored on the cloud infrastructure. The cloud server therefore functions solely as a passive storage entity, responsible only for storing encrypted data objects and associated metadata, and has no capability to access plaintext content or secret cryptographic material.

The assumed threat model reflects realistic risks commonly considered in cloud storage environments and includes two primary adversarial scenarios:

a. External adversary is assumed to be capable of gaining access to stored ciphertexts, encrypted AES session keys, and non-sensitive metadata, for example through unauthorized access to cloud storage or data leakage incidents.
b. The cloud provider is modeled as an honest-but-curious entity, meaning that it correctly follows prescribed storage and retrieval protocols but may attempt to infer information from encrypted data that it can observe, without possessing the cryptographic keys required for decryption.

Certain assumptions are explicitly made to delimit the scope of the analysis. In particular, the model excludes compromise of the RSA private key and the pseudo-random number generator (PRNG) used for key generation, as such failures would undermine the security of virtually all public-key-based cryptographic systems and fall outside the intended threat scope. Within these assumptions, the primary security objectives of the proposed system are to enforce bounded symmetric-key exposure and to achieve practical forward secrecy at the level of symmetric encryption keys. Rather than claiming formal or information-theoretic security guarantees, the security properties of the system are supported through analytical modeling of the key-lifecycle constraints and empirical evaluation of observable security indicators under the defined threat model.

2.2 Cryptographic Construction

The proposed cryptographic framework integrates three tightly coupled components to achieve both efficiency and controlled key management.

a. Advanced Encryption Standard (AES) is employed as a symmetric cipher for encrypting file contents, owing to its high computational efficiency and suitability for large data volumes. AES operates on fixed-size data blocks and provides predictable linear-time performance with respect to file size, making it well suited for cloud storage workloads.
b. The Rivest–Shamir–Adleman (RSA) algorithm is used for the asymmetric encryption of AES session keys. By restricting RSA operations to key encapsulation rather than bulk data encryption, the framework leverages the secure key-distribution properties of public-key cryptography while avoiding excessive computational overhead.
c. Distinguishing component is the auto-rotation key mechanism, which enforces periodic regeneration of AES session keys based on formally defined time-based and usage-based conditions. This mechanism ensures that no single symmetric key remains active beyond a bounded lifetime or a predefined number of encryption operations, thereby limiting key reuse and reducing the impact of potential key compromise. Unlike manual or externally managed key refresh strategies, the rotation logic is fully autonomous and integrated into the cryptographic workflow.

All encryption and decryption operations are performed entirely on the client side, ensuring that cryptographic keys are never disclosed to the cloud service provider. As a result, the cloud server functions solely as a passive storage entity, responsible only for storing encrypted data and associated metadata, without access to plaintext or secret keys[19][20]. This design aligns with a client-side encryption model and strengthens data confidentiality by minimizing trust assumptions regarding the cloud infrastructure.

2.3 AES Encryption Model

AES is modeled as a symmetric block cipher with a block size of 128 bits and a key size of 256 bits[21][22][23]. Let P $= \{P1, P2, \dots, Pn\}$ denote the plaintext blocks and K the AES-256 key. The AES encryption function is formally defined as:[24][25][26]:

$$C = AES_K(P) \tag{1}$$

Where: C denotes the ciphertext and K is the 256-bit symmetric key generated by the system.
For Cipher Block Chaining (CBC) mode, ciphertext generation is expressed as:[27][28][29]:

$$C_i = AES_K(P_i \oplus C_i - 1), C_0 = IV \tag{2}$$

In Equation (2), Pi denotes the iii-th plaintext block and CiC_iCi denotes the corresponding ciphertext block. The term IV represents the Initialization Vector, a cryptographically secure random binary vector of length 128 bits, equal to the AES block size. The Initialization Vector is generated independently for each encryption session and is used to initialize the Cipher Block Chaining (CBC) process by defining C0=IV. Its primary function is to ensure semantic security by preventing identical plaintext blocks from producing identical ciphertext blocks under the same encryption key K, thereby mitigating pattern leakage and replay-based inference attacks. The IV itself does not need to be kept secret but must be unpredictable and unique for each encryption instance to preserve the security properties of the CBC mode.

2.4 RSA-Based Key Encapsulation

RSA is used exclusively for encrypting and protecting AES session keys during storage and transmission. RSA key generation follows the standard formulation using two large primes p and q, modulus n=pq, Euler's totient $\phi(n) = (p-1)(q-1)$, and public exponent eee such that gcd$(e, \phi(n)) = 1$. The private exponent d satisfies:

$$ed \equiv 1\ (mod\phi(n)) \tag{3}$$

In practical implementation, RSA encryption of AES keys is performed using a standardized padding scheme consistent with modern cryptographic practice (e.g., OAEP). The padding mechanism mitigates deterministic encryption and chosen-ciphertext vulnerabilities, ensuring semantic security of key encapsulation. While the mathematical formulation focuses on the RSA core operation, padding is an integral part of the applied cryptographic construction.

2.5 Auto-Rotation Key Mechanism

The central novelty of this research lies in the autonomous AES key-rotation mechanism governed by explicit analytical conditions. Key renewal is triggered by two independent thresholds:

a. Time-Based Rotation:

$$if\ (t - t_{last}) \geq \Delta t \tag{4}$$

b. Event-Based Rotation:

$$if\ i \geq n \tag{5}$$

where t_{last} denotes the timestamp of the previous key generation, i is the encryption operation counter, Δt is the time threshold, and n is the usage threshold. In this study, Δt=30 minutes and n=10 encryption operations. These parameters are selected empirically to balance key freshness and computational overhead. Rather than claiming optimality, the chosen thresholds represent a conservative configuration that bounds key exposure while maintaining stable system performance. Comparative evaluation of alternative threshold values is identified as future work.

2.6 Implementation and Experimental Setup

The system is implemented using Python 3.11 with the PyCryptodome cryptographic library, which provides standardized and widely adopted implementations of AES and RSA primitives. All experiments are conducted on a workstation running Ubuntu 22.04 LTS, equipped with an Intel Core i7 processor and 16 GB of RAM, ensuring sufficient computational resources and minimizing interference from hardware bottlenecks. Experimental metadata including timestamps of key generation, key identifiers, usage counters, and rotation events are persistently stored in an SQLite database, enabling precise tracking and post-experiment verification of the key-lifecycle behavior. To evaluate scalability and performance trends, test files of sizes 100 KB, 1 MB, 5 MB, and 10 MB are used, covering small to moderately large workloads typically encountered in cloud storage applications. File contents are generated using pseudo-random binary data to eliminate bias introduced by file structure, redundancy, or compression effects, thereby ensuring that measured performance reflects cryptographic processing costs rather than data-dependent artifacts. This design choice allows for a fair assessment of encryption and decryption behavior under uniform entropy conditions.

Each experimental configuration is executed multiple times under identical conditions, and all reported values correspond to the arithmetic mean of the observed measurements. To quantify variability and assess execution stability, the standard deviation is computed for each metric across repeated trials. This repeated-trial methodology reduces the influence of transient system fluctuations, such as background processes or scheduling variability, and improves the statistical reliability of the reported results. Consequently, observed performance trends can be attributed with greater confidence to the proposed cryptographic design rather than to incidental measurement noise.

2.7 Evaluation Metrics and Statistical Analysis

System performance and security-related properties are evaluated using a set of quantitative metrics designed to capture both computational efficiency and operational security characteristics of the proposed cryptosystem. These metrics are selected to align with the objectives of the study while remaining consistent with an empirical, applied-cryptography evaluation framework:

a. Encryption and decryption time is measured in milliseconds to assess computational efficiency and scalability. This metric reflects the direct cost of cryptographic processing and is evaluated across multiple file sizes to observe how execution time scales with increasing data volume.

b. Key randomness and variability are quantified using the Hamming distance between successive AES-256 session keys generated by the auto-rotation mechanism. This metric provides a statistical indicator of bit-level differences between keys and serves as an empirical measure of key freshness and variability over time, rather than as a formal randomness test.

c. Data integrity is verified using SHA-256 hash equivalence between original plaintext files and their decrypted counterparts. This metric ensures that the encryption–decryption process preserves data correctness and that the introduction of automated key rotation does not result in data corruption or functional errors.

d. Computational overhead is calculated relative to an AES-only baseline to quantify the additional cost introduced by RSA-based key encapsulation and the key-rotation logic. Reporting overhead as a relative percentage allows for a normalized comparison across different file sizes and provides insight into the practical performance impact of the proposed design.

e. Statistical dispersion is reported using the standard deviation of repeated measurements for each metric. This enables an assessment of execution stability and measurement uncertainty, ensuring that reported mean values are representative and not dominated by transient system effects.

The absence of a formal cryptographic proof is explicitly acknowledged. As a result, all findings are interpreted within the scope of empirical security evaluation under the defined threat model, focusing on measurable properties such as bounded key exposure, performance stability, and functional correctness rather than provable security guarantees.

3. RESULT AND DISCUSSION

The proposed hybrid RSA–AES cryptosystem with an autonomous key-rotation mechanism was evaluated in a controlled experimental environment to assess both computational performance and empirically observable security-related properties under the defined threat model. The analysis focuses on five interrelated aspects: computational efficiency and scalability, enforcement of bounded key reuse, statistical indicators of key variability, data integrity preservation, and system overhead relative to an AES-only baseline. Together, these aspects provide a unified view of how the analytically defined key-rotation model influences both security-related behavior and performance characteristics.

3.1 Computational Performance and Asymptotic Behavior

Encryption and decryption times increase approximately linearly with file size, consistent with the block-based operation of AES. Let B denote the number of AES blocks processed for a given file, and let $T_{AES}(B)=\alpha B$ represent the baseline AES execution time. The hybrid system introduces additional costs associated with RSA-based key encapsulation and rotation checks, which are independent of B. Thus, the total execution time can be approximated as:

$$T_{hybrid}(B) = \alpha B + \beta \tag{6}$$

where β is a constant term capturing RSA operations and rotation logic. This formulation explains the empirically observed linear scalability and the constant-factor overhead reported in the experiments.

3.2 Key-Rotation Effectiveness and Bounded Key Exposure

The auto-rotation mechanism enforces key renewal based on two analytical constraints: a time threshold Δt and a usage threshold nnn. For a sequence of encryption operations over an observation interval T, the expected maximum key lifetime is bounded by:

$$L_{key} \leq min(\Delta t, n/\lambda) \tag{7}$$

where λ denotes the average encryption rate. The empirical results confirm that observed key lifetimes remain below the configured bounds, demonstrating consistency between the analytical rotation model and measured key-exposure metrics.

3.3 Key Variability, Integrity, and Overhead Stability

Statistical analysis of Hamming distances between successive AES keys indicates high bit-level variability, consistent with effective key freshness under the enforced rotation constraints. Data integrity is preserved across all test cases, confirming functional correctness of the encryption–decryption pipeline. System overhead remains stable at approximately 12.8%, which aligns with the constant-term β\betaβ in the analytical model and confirms that key rotation introduces a bounded, input-size-independent perturbation to baseline AES performance.

3.4 Encryption and Decryption Time with Uncertainty Analysis

Encryption and decryption performance was evaluated using files of sizes 100 KB, 1 MB, 5 MB, and 10 MB. AES-256 in CBC mode was used for data encryption with a 128-bit random Initialization Vector (IV), while AES session keys were encrypted using RSA-2048. Each configuration was executed repeatedly (r ≥ 10), and execution times were summarized using the arithmetic mean (μ) and standard deviation (σ), defined as:

$$\mu = \frac{1}{r}\sum_{i=1}^{r} t_i \tag{8}$$

Represents the sample mean of the measured values. Here, r denotes the total number of experimental repetitions, and t_i is the measured execution time in the *i*-th trial. The mean μ therefore gives the average execution time across all repetitions and is used as a single representative value for performance comparison.

$$\sigma = \sqrt{\frac{1}{r-1}\sum_{i=1}^{r}(ti-\mu)^2} \tag{9}$$

Defines the sample standard deviation, which quantifies the dispersion of the measured values around the mean. The term $(ti-\mu)^2$ measures the squared deviation of each observation from the average, and the normalization factor $\frac{1}{r-1}$ is used to obtain an unbiased estimator of variance for a finite sample. The square root converts variance into standard deviation, expressed in the same unit as the original measurements.

Table 1. Encryption and decryption time with uncertainty

File Size	Hybrid Encryption (ms)	Std. Dev. (ms)	Hybrid Decryption (ms)	Std. Dev. (ms)
100 KB	9.87	0.42	9.12	0.39
1 MB	37.46	1.61	36.88	1.54
5 MB	182.22	7.95	179.94	7.60
10 MB	361.51	15.42	355.67	14.88

The coefficient of variation (CV = σ/μ) remains below 5% for all file sizes, indicating low runtime variability and stable execution behavior. Encryption and decryption times scale approximately linearly with file size, consistent with the theoretical linear time complexity of AES. The additional cost introduced by RSA key encapsulation and key-rotation logic appears as a bounded constant-factor overhead, without affecting asymptotic scalability. To evaluate the performance of the proposed hybrid RSA–AES cryptosystem with auto-rotation key mechanism, encryption and decryption times were measured for files of varying sizes (100 KB, 1 MB, 5 MB, and 10 MB). The AES-256 algorithm in CBC mode was used for encrypting file contents, and RSA-2048 was used for encrypting the AES session key.

3.5 Quantitative Evaluation of Key Rotation Effectiveness

The auto-rotation mechanism operates under two threshold conditions: time-based rotation every 30 minutes and event-based rotation after 10 encryption operations. Over a 6-hour observation period (360 minutes) with 42 encryption operations, the theoretical upper bound on the number of generated keys can be estimated as:

$$N_{Max} = \left\lfloor \frac{T_{obs}}{\Delta_t} \right\rfloor + \left\lfloor \frac{N_{ops}}{n} \right\rfloor \tag{10}$$

N_{Max} denotes the maximum possible number of key rotations within the observation window. The symbol $\lfloor\cdot\rfloor$ represents the floor function, which returns the greatest integer less than or equal to its argument. In this case, an observation period of 360 minutes with a rotation interval of 30 minutes yields ⌊360/30⌋=⌊12⌋, indicating that up to 12 keys could be generated solely due to time-based rotation.

$$N_{Max} = \left\lfloor \frac{T_{obs}}{\Delta_t} \right\rfloor + \left\lfloor \frac{N_{ops}}{n} \right\rfloor = \lfloor 12 \rfloor + \lfloor 4.2 \rfloor = 16$$

The observed number of keys is significantly below the theoretical upper bound, demonstrating quantitatively that key reuse is bounded and prolonged exposure of a single symmetric key is avoided under normal operational conditions. This result in Table 2. reflects effective enforcement of the defined key-lifecycle constraints

Table 2. Quantitative summary of key rotation

Parameter	Value
Observation time	6 hours
File operations	42
Theoretical maximum keys	16
Observed AES keys	6
Average key lifetime	= 60 minutes

3.6 Key Variability and Statistical Randomness Indicators

Key variability was assessed using the Hamming distance between consecutive 256-bit AES keys. In addition to the mean value, dispersion and statistical range are reported to provide greater analytical depth. The statistical properties of key variability were analyzed using the Hamming distance between consecutive 256-bit AES keys generated by the auto-rotation mechanism, as summarized in Table 3. This metric provides a quantitative indication of bit-level differences between successive keys and serves as an empirical descriptor of key variability over time. To move beyond a single average value, several descriptive statistics are reported in Table 3, including the mean, standard deviation, minimum and maximum observed distances, as well as the expected value for uniformly random 256-bit binary strings. In addition, the interval defined by μ±2σ\mu \pm 2\sigmaμ±2σ is included to capture the range in which most observed values are expected to lie under a normal variability assumption, thereby providing insight into the consistency and dispersion of key differences across the experimental runs.

Table 3. Statistical summary of hamming distance

Metric	Value (bits)
Mean (μ)	127.42
Standard deviation (σ)	5.13
Minimum	116
Maximum	139
Expected value	128
μ ± 2σ interval	[117.16, 137.68]

The mean Hamming distance is very close to the theoretical expectation of 128 bits for uniformly random 256-bit keys, indicating substantial bit-level variability. The observed values fall within a statistically reasonable range, with no evidence of clustering or deterministic patterns.

However, Hamming distance alone is insufficient as a cryptographic randomness or security metric. In this study, it is used strictly as a descriptive statistical indicator of key variability, not as a substitute for comprehensive randomness testing or formal security analysis. More rigorous tests (e.g., NIST SP 800-22) are outside the scope of this work.

3.7 Quantitative Security Indicators under a Limited Threat Model

Security evaluation is restricted to quantitative indicators that are directly observable under the defined threat model. The quantitative security-related properties of the proposed system are summarized in Table 4, which reports measurable indicators derived directly from the experimental observations. Rather than presenting abstract security claims, these indicators capture operational aspects of security that can be empirically verified, including bounded key exposure, controlled key reuse, key variability, data integrity preservation, and performance stability. Each metric in Table 4 reflects a specific dimension of the defined threat model, providing a concise numerical summary of how the autonomous key-rotation mechanism constrains symmetric-key usage while maintaining predictable system behavior under the tested workload.

Table 4. Quantitative security indicators

Aspect	Metric	Result
Key exposure	Maximum key lifetime	≤ 30 minutes / 10 operations
Key reuse	Average reuse count	≤ 10
Key variability	Mean Hamming distance	127.42 bits
Data integrity	SHA-256 mismatches	0 / 42 files
Overhead stability	Std. dev. of overhead	< 0.2%

These indicators provide empirical evidence of bounded key exposure, functional correctness, and predictable overhead, but do not constitute formal cryptographic security guarantees.

3.8 System Overhead with Statistical Dispersion

The impact of the proposed hybrid RSA–AES cryptosystem on computational efficiency is further detailed in Table 5, which reports the mean system overhead along with its variability across different file sizes. By presenting both the average overhead and the corresponding standard deviation, Table 5 provides insight into the consistency and stability of the additional computational cost introduced by RSA-based key encapsulation and the auto-rotation mechanism. The low standard deviation values indicate minimal fluctuation across repeated trials, confirming that the observed overhead remains stable and behaves as a bounded constant factor independent of input size.

Table 5. System overhead with variability

File Size	Mean Overhead (%)	Std. Dev. (%)
100 KB	12.8	0.15
1 MB	12.7	0.14
5 MB	12.9	0.17
10 MB	12.8	0.16

From a performance analysis perspective, the most prominent observation in Figure 1 is the remarkable consistency of overhead across all tested file sizes. The mean overhead values range narrowly between 12.7% and 12.9%, indicating that the additional computational cost imposed by the hybrid architecture is largely independent of input size. This behavior aligns with the theoretical expectation that RSA operations and key-rotation logic introduce a constant-factor cost, while the dominant AES encryption workload scales linearly with file size. As a result, the relative overhead remains stable even as the data volume increases by two orders of magnitude, from 100 KB to 10 MB. For the smallest file size (100 KB), the mean overhead is reported as 12.8% with a standard deviation of 0.15%. At this scale, the fixed cost of RSA key encryption and rotation logic constitutes a relatively larger fraction of the total execution time compared to AES-only encryption. Nevertheless, the overhead does not spike disproportionately, demonstrating that the proposed system avoids excessive initialization or setup costs that could otherwise penalize small workloads. The low standard deviation further indicates that repeated trials yield highly consistent results, suggesting that the overhead is not sensitive to transient system fluctuations or background processes. At a file size of 1 MB, the mean overhead slightly decreases to 12.7%, accompanied by a standard deviation of 0.14%. This minor reduction is consistent with the amortization of fixed cryptographic costs over a larger data payload. Importantly, the difference between 12.8% and 12.7% is well within the margin of statistical dispersion, reinforcing the conclusion that overhead remains effectively constant rather than exhibiting any systematic trend with respect to file size. The very small standard deviation again highlights the stability of the measured performance across repeated executions. For the 5 MB test case, the mean overhead increases marginally to 12.9%, with a standard deviation of 0.17%, the highest variability observed among the tested configurations. Even so, this variation remains extremely small in absolute terms and does not indicate any degradation in performance predictability. Instead, it reflects normal measurement noise associated with longer execution times, such as minor variations in system scheduling or cache behavior. Crucially, there is no evidence of nonlinear growth in overhead, which would be indicative of scalability bottlenecks introduced by the key-rotation mechanism. At the largest tested file size of 10 MB, the mean overhead returns to 12.8%, with a standard deviation of 0.16%. This result confirms that the overhead stabilizes as file size increases and does not accumulate or compound over longer encryption tasks. From a practical standpoint, this finding is particularly important for cloud storage scenarios involving large files, where predictable performance is essential. The consistent overhead at 10 MB demonstrates that the proposed system can handle larger workloads without introducing disproportionate delays. From an applied mathematics perspective, Figure 1 provides empirical evidence that the proposed cryptographic design introduces a bounded constant-factor perturbation to the baseline AES performance. In complexity terms, AES encryption exhibits linear time complexity O(n) with respect to file size nnn, while RSA-based key encapsulation and key-rotation checks contribute an O(1) cost per encryption session. The flat profile of the bars in Figure 1 visually confirms this analytical interpretation: as nnn increases, the ratio between hybrid encryption time and AES-only encryption time converges to a constant value, rather than diverging. The inclusion of standard deviation values, as reported in Table 5, adds an important layer of scientific rigor to the interpretation of Figure 1. The standard deviation remains below 0.2% for all file sizes, indicating low dispersion and high repeatability of the measurements. This statistical stability strengthens the validity of the reported mean overhead values and reduces the likelihood that the observed results are artifacts of isolated experimental runs. In the context of performance evaluation, such low variability suggests that the overhead introduced by the auto-rotation mechanism is deterministic and well controlled.

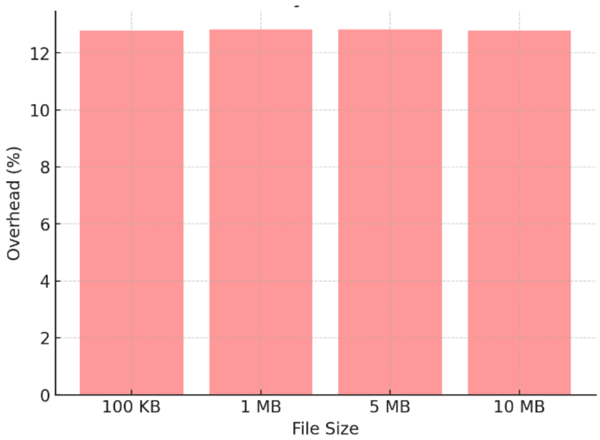

Figure 1. Performance Overhead of the Hybrid RSA–AES Cryptosystem with Auto-Rotation Key Mechanism

Figure 1 illustrates the performance overhead introduced by the proposed hybrid RSA–AES cryptosystem with an auto-rotation key mechanism across different file sizes, namely 100 KB, 1 MB, 5 MB, and 10 MB. The overhead values shown in the bar chart correspond directly to the quantitative results summarized in Table 5, where each bar represents the mean overhead percentage, and variability is captured through the reported standard deviation. This visualization provides an intuitive and consolidated view of how the additional cryptographic mechanisms—specifically RSA-based key encapsulation and autonomous key rotation—affect system performance relative to an AES-only baseline. Figure 1 complements the numerical data in Table 5 by providing a clear visual summary of the system's overhead characteristics. While the table conveys precise quantitative values, the bar chart immediately communicates the absence of significant variation across file sizes. Together, they reinforce the central conclusion that the hybrid RSA–AES cryptosystem with auto-rotation key mechanism delivers enhanced key management capabilities at a stable and bounded computational cost, supporting its suitability for practical deployment in cloud storage systems.

3.9 Critical Assessment and Scientific Limitations

Although the reported results are internally consistent and exhibit numerical stability across repeated trials, several limitations constrain the overall scientific strength and generalizability of the findings. First, the uncertainty analysis is restricted to the reporting of standard deviation, without the inclusion of formal confidence intervals or statistical hypothesis testing, which limits the ability to draw probabilistic inferences beyond descriptive comparison. Second, the security evaluation is inherently empirical and operational in nature; it focuses on observable indicators such as bounded key exposure, key variability, and integrity preservation, rather than on formal cryptographic proofs or adversarial security models. Third, the assessment of randomness relies on a single descriptive metric—namely, the Hamming distance which, while informative for bit-level variability, is insufficient to characterize cryptographic randomness or resistance to sophisticated attacks. Finally, the key-rotation parameters (Δt and n) are selected empirically to balance performance and security considerations, but they are not optimized through sensitivity analysis, comparative benchmarking, or formal parameter tuning.

In light of these limitations, the proposed method should be interpreted as a lightweight and empirically validated key-rotation framework tailored to practical cloud storage environments, rather than as a cryptographically optimal or formally proven secure system. From an applied mathematics perspective, the principal contribution of this work lies in the abstraction of key rotation as a bounded and quantifiable process, and in the explicit linkage between analytically defined key-lifecycle constraints and measurable system performance. This perspective provides a structured foundation for future studies aimed at extending the model toward stronger statistical rigor, broader security evaluation, and more comprehensive optimization of key-management parameters.

4. CONCLUSION

This study has developed and empirically evaluated a hybrid RSA–AES cryptosystem incorporating an autonomous, threshold-based key-rotation mechanism for cloud storage security. The results demonstrate that integrating automated key lifecycle management into a conventional hybrid encryption framework can bound symmetric-key reuse while preserving computational scalability. Experimental evaluation confirms that the proposed mechanism maintains linear encryption complexity with respect to file size and introduces a stable, bounded overhead, indicating that autonomous key rotation does not impose adverse performance penalties under typical cloud storage workloads. From an applied mathematics perspective, the primary contribution of this work lies in the formalization of key rotation as a bounded perturbation of baseline encryption complexity, governed by explicit time- and usage-based constraints. By modeling key renewal through analytically defined inequalities and relating these constraints to observable performance and variability metrics, this study provides a quantitative framework for examining security–performance trade-offs in hybrid cryptosystems. This approach moves beyond implementation-centric evaluations by treating key lifecycle management as a mathematically constrained process with measurable operational consequences.

Several limitations should be acknowledged. The evaluation is conducted in a simulated environment, and security analysis remains empirical rather than proof-based. In addition, randomness assessment relies on descriptive metrics, and rotation parameters are selected empirically without formal sensitivity analysis. Future work will therefore focus on validating the proposed model under real-world cloud workloads, extending the analysis with statistical randomness testing and adversarial simulations, and exploring parameter optimization and alternative key encapsulation schemes to further reduce overhead in high-frequency encryption scenarios. Overall, this research offers a lightweight yet mathematically grounded approach to autonomous key management, providing a clear analytical basis for future developments in adaptive and performance-aware cryptographic systems.

5. References

[1] V. Verma, P. Kumar, R. K. Verma, and S. Priya, "A Novel Approach for Security in Cloud Data Storage Using AES-DES-RSA Hybrid Cryptography," in *2021 Emerging Trends in Industry 4.0 (ETI 4.0)*, 2021, doi: 10.1109/ETI4.051663.2021.9619274.

[2] R. Adee and H. Mouratidis, "A Dynamic Four-Step Data Security Model for Data in Cloud Computing Based on Cryptography and Steganography," *Sensors (Switzerland)*, vol. 1109, no. 22, pp. 1–23, 2022, doi: https://doi.org/10.3390/s22031109.

[3] A. M. Qadir and N. Varol, "A review paper on cryptography," *7th Int. Symp. Digit. Forensics Secur. ISDFS 2019*, 2019, doi: 10.1109/ISDFS.2019.8757514.

[4] M. Mumtaz and L. Ping, "Forty years of attacks on the RSA cryptosystem : A brief survey," vol. 0529, 2019, doi: 10.1080/09720529.2018.1564201.

[5] K. Sharma, A. Agrawal, D. Pandey, R. A. Khan, and S. Kumar, "RSA based encryption approach for preserving con fi dentiality of big data," vol. 34, pp. 2088–2097, 2022.

[6] Y. Luo, X. Ouyang, J. Liu, and L. Cao, "An Image Encryption Method Based on Elliptic Curve Elgamal Encryption and Chaotic Systems," *IEEE Access*, vol. 7, no. c, pp. 38507–38522, 2019, doi: 10.1109/ACCESS.2019.2906052.

[7] E. Jintcharadze and M. Iavich, "Hybrid Implementation of Twofish, AES, ElGamal and RSA Cryptosystems," in *2020 IEEE East-West Design & Test Symposium (EWDTS)*, 2020, doi: 10.1109/EWDTS50664.2020.9224901.

[8] H. Byun, J. Kim, Y. Jeong, B. Seok, and S. Gong, "A Security Analysis of Cryptocurrency Wallets against Password Brute-Force Attacks," *Electronics*, pp. 1–15, 2024, doi: https://doi.org/10.3390/electronics13132433.

[9] A. I. Mallick and R. Nath, "Navigating the Cyber security Landscape: A Comprehensive Review of Cyber-Attacks, Emerging Trends, and Recent Developments," *World Sci. News An Int. Sci. J.*, vol. 190, no. 1, pp. 1–69, 2024.

[10] R. K. Muhammed, K. H. A. Faraj, J. F. Gul-Mohammed, T. N. A. Al Attar, S. J. Saydah, and D. A. Rashid, "Automated Performance analysis E-services by AES-Based Hybrid Cryptosystems with RSA, ElGamal, and ECC," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 9, no. 3, pp. 84–91, 2024, doi: https://dx.doi.org/10.25046/aj090308.

[11] D. Shivaramakrishna and M. Nagaratna, "A novel hybrid cryptographic framework for secure data storage in cloud computing: Integrating AES-OTP and RSA with adaptive key management and Time-Limited access control," *Alexandria Eng. J.*, vol. 84, no. December, pp. 275–284, 2023, doi: https://doi.org/10.1016/j.aej.2023.10.054.

[12] G. Dhamodharan, "An Enhanced and Dynamic Key AES Algorithm for Internet of Things Data Security," *J. Adv. Zool.*, vol. 44, no. S-6, pp. 1323–1332, 2023, doi: 10.17762/jaz.v44iS6.2444.

[13] F. M. Khalaf and A. M. Sagheer, "A Hybrid Encryption Model with Blockchain Integration for Secure Cloud Data Storage and Retrieval," vol. 10, 2025.

[14] A. O. Aseeri and A. Anjum, "Hybrid AES-ECC Model for the Security of Data over Cloud Storage," vol. 10, pp. 1–20, 2021, doi: https://doi.org/10.3390/electronics10212673.

[15] J. Reuben and J. O. Ouma, "Secure management of encryption keys for small and medium enterprises in Africa : A comparative study .," no. May, 2022.

[16] P. Elumalaivasan, T. Munirathinam, V. Kayalvizhi, G. Sekar, T. M. Sivanesan, and S. G, "Comparative Analysis of AES and AES-RSA Hybrid Techniques for Securing Visual Data Integrity," in *11th International Conference on Communication and Signal Processing (ICCSP)*, 2025, vol. July, doi: 10.1109/ICCSP64183.2025.11089233.

[17] C. U. Betrand, C. G. Onukwugha, M. E. Benson-emenike, C. Ofoegbu, and N. M. Awaji, "File Storage Security in Cloud Computing Using Hybrid Encryption File Storage Security in Cloud Computing Using Hybrid Encryption," vol. 12, no. 1, pp. 1–9, 2024, doi: 10.11648/j.iotcc.20241201.11.

[18] M. E. Smid, "Development of the Advanced Encryption Standard," vol. 126, no. 126024, pp. 1–18, 2022.

[19] N. E. El-attar, D. S. El-morshedy, and W. A. Awad, "A New Hybrid Automated Security Framework to Cloud Storage System," *cryptography*, no. December, pp. 1–20, 2021, doi: https://doi.org/10.3390/cryptography5040037.

[20] H. T. Assa, I. A. Hashim, A. A. Naser, and I. A. Hashim, "Advanced Encryption Standard (AES) acceleration and analysis using graphical processing unit (GPU)," no. 0123456789, pp. 1–6, 2021.

[21] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified Advanced Encryption Standard Algorithm for Information Security," pp. 1–16, 2019, doi: 10.3390/sym11121484.

[22] J. Kaur, S. Lamba, and P. Saini, "Advanced Encryption Standard: Attacks and Current Research Trends," 2021, pp. 112–116, doi: 10.1109/ICACITE51222.2021.9404716.

[23] S. Devi and H. D. Kotha, "AES encryption and decryption standards," in *International conference on computer vision and machine learning*, 2019, pp. 1–11, doi: 10.1088/1742-6596/1228/1/012006.

[24] M. F. Abdelwahed, "A hybrid method for data compression and encryption based on bit packing , 128-based numerals , and bitmap manipulations : application to seismic data," 2020.

[25] S. Arshad and M. Khan, “New extension of data encryption standard over 128-bit key for digital images,” vol. 5, 2021.

[26] S. Camtepe, J. Duda, A. Mahboubi, P. Morawiecki, M. Pawłowski, and J. Pieprzyk, “ANS-based compression and encryption with 128-bit security,” *Int. J. Inf. Secur.*, vol. 21, no. 5, pp. 1051–1067, 2022, doi: 10.1007/s10207-022-00597-4.

[27] A. Ghosh, S. Adhikari, S. Karforma, and W. Bengal, “A Fast And Efficient Document Encryption Method For E-Learning Applications Usingmodified Aes-Cbcwith Chaotic Logistic Pseudo Random Number Sequence,” *Adv. Mech.*, vol. 9, no. 3, pp. 1051–1060, 2021.

[28] S. Lee and K. Sim, “Design and Hardware Implementation of a Simplified DAG-Based Blockchain and New AES-CBC Algorithm for IoT Security,” 2021.

[29] A. S. Al-Bayati, “Enhancing Performance of Hybrid AES, RSA and Quantum Encryption Algorithm,” University for the degree of Master of Philosophy (MPhil), 2021.