ADROID Conference Series: Engineering Sciences and Technology
Volume 2, pp.43-52, 2026

An Applied Security Model For Controlled Key-Release In Blowfish-Based Data Retrieval

Vandyala Yukthasri[a)], Manchanpally Arun Kumar[b)], Mohammad Ayaan[c)], S. Satheesh Kumar[d)], Ganesh B Regulwar[e)] and Nikhila Kathirisetty[f)]

Department of Information Technology, Vardhaman College of Engineering, Hyderabad, India
a) Corresponding author: yukthasrivandyala@gmail.com, b) marunkumar0459@gmail.com,
c) mdayaan918268@gmail.com, d) satheeshsmm@gmail.com (https://orcid.org/0000-0001-9007-3665),
e) ganeshregulwar@gmail.com

Abstract. From a cybersecurity standpoint, the security of the data stored in these systems has become a crucial concern as cloud storage continues to proliferate in digital infrastructure. Credential compromise, brute-force attacks, and sophisticated, persistent infiltration tactics have demonstrated the vulnerability of traditional security techniques, especially those that rely on static encryption keys or password-based authentication. This study offers a multi-layered security approach that combines multi-stage authentication with a dynamic key-release method for Blowfish-encrypted file storage, in response to constantly evolving cyber threats. Before allowing the controlled recovery of the Blowfish decryption key, the model described here used sequential verification with user credentials, a time-based one-time password, and a registered device token validation procedure. By keeping encryption keys unreadable until identity is fully confirmed, the system architecture reduces the exposure risks associated with conventional key storage methods. FastAPI, Streamlit, PostgreSQL, and MinIO were used in the implementation; the configuration was simple but deployable in both academic and business environments. According to experimental evaluation, the framework significantly increases resistance to unauthorised access attempts while maintaining low latency for both authentication and decryption. The findings make it clear that the suggested architecture provides a safe and effective way to protect files in cloud environments, balancing computational efficiency and user accessibility while ensuring robust cryptographic security.

Keywords: Cloud Security; Blowfish Encryption; Multi-Stage Authentication; Controlled Key Release; Secure Storage; Cryptographic Key Management.

1. INTRODUCTION

Because cloud-based storage offers scalability, simplicity, and cost-effectiveness, it has become an essential part of contemporary digital infrastructure. However, the risk of targeted cryptographic attacks, illegal access, and credential compromise rises with the amount of sensitive data kept on distributed platforms. A large percentage of cloud breaches occur not because encryption algorithms malfunction but because encryption keys are mismanaged, exposed, or used with shoddy authentication procedures. Conventional systems that depend on single-step authentication or static keys are still extremely susceptible to token theft, brute-force attacks, and persistent infiltration patterns. Although identity verification has been enhanced by recent advancements in time-based one-time passwords (TOTP), multi-factor authentication (MFA), and device-binding techniques, the majority of current cloud-security models regard key management and authentication as distinct procedures. Attackers who evade authentication or obtain legitimate credentials may still be able to access decryption keys kept on the server or in semi-persistent memory thanks to this crucial flaw. Furthermore, popular symmetric encryption methods like AES and Blowfish provide robust protection for data that is not in use, but their overall security largely depends on when and how their keys are made public.

Motivation

Instead of cracking the encryption algorithm directly, modern cyberattacks increasingly target key exposure windows, device spoofing, and authentication bypass techniques. Cloud-storage designs that do not retain encryption keys continually, make them permanently available, or release them until identity verification is finished are becoming more and more necessary. Additionally, many lightweight cloud settings still utilise Blowfish because of its ease of use and adjustable key sizes, even though AES is commonly used because of its speed and uniformity. A thorough security paradigm that combines regulated, ephemeral key release with multi-stage authentication can drastically lower the attack surface for both algorithms.

Objectives

The goal of this project is to provide a useful, flexible, secure storage system that:

- Prevents early or unwanted access by directly connecting multi-stage authentication to the availability of encryption keys.
- Blowfish keys are decrypted into volatile memory only after successful sequential authentication thanks to a dynamic, ephemeral key-release mechanism.
- Evaluates the performance features of AES and Blowfish, offering guidance on cloud platform algorithm selection.
- Minimises latency while upholding stringent cryptographic protection and access control.
- Offers a deployable prototype for academic and enterprise-scale settings using FastAPI, Streamlit, MinIO, and PostgreSQL.

Main Contributions

The following are the main contributions of this paper:

- A single architecture for key release and authentication that does away with static key storage and greatly lowers exposure concerns.
- A three-tiered identity verification pipeline that combines device-token confirmation, TOTP-based second-factor authentication, and password validation.
- RSA-protected controlled Blowfish key-release method that guarantees keys never show up in client memory or persistent storage.
- AES and Blowfish's performance is compared, and responsiveness under cloud workloads is assessed using realistic scaling.
- A completely functional prototype that exhibits strong fault tolerance under stress testing, high attack resistance, and low latency retrieval.
- A workable paradigm with an extendable design that supports more authentication layers and is appropriate for lightweight cloud deployments.

Organisation of the Paper

The rest of the paper is organised as follows: An extensive review of the literature on symmetric encryption performance, key-management strategies, and authentication methods is presented in Section 2. The suggested system architecture, including registration, authentication, key-release logic, and file retrieval procedure, is explained in Section 3. Experimental data, AES-Blowfish comparing graphs, and in-depth comments are presented in Section 4. The study's main conclusions, limitations, and future research directions are summarised in Section 5.

2. LITERATURE REVIEW

Research on symmetric encryption methods, cryptographic key management, and safe authentication has increased due to the explosive rise of cloud-based data storage. Many studies show that poorly protected keys, inadequate authentication procedures, and static key-lifecycle processes are more common causes of contemporary cloud breaches than flaws in encryption methods. According to Shitharth and Khan [1], incorrect cryptographic governance and misconfigured access-control policies are the main causes of key leakage issues in multi-cloud setups. This highlights the need for key-access models that incorporate authentication. One of the most popular methods for stopping unwanted access is still multi-factor authentication, or MFA. According to Singh and Bose [2] and Hu and Feng [3], credential-based attacks are significantly reduced when password-based factors are combined with independent factors such as TOTP, biometrics, or hardware tokens. As demonstrated by Aladadi and Al-Mashaqbeh [4], TOTP-based second-factor systems further improve security by producing time-sensitive codes. Researchers have suggested device-linked authentication and contextual identification systems because even MFA can be broken via token replay, SIM-based attacks, or malware-assisted credential theft [5][6][7].

Because of their effectiveness, symmetric algorithms are important for protecting data when it's at rest, according to cryptographic literature. Because of its ease of use and adjustable key size, Blowfish is still used

in lightweight or resource-constrained environments, even though AES is still the industry standard and is extensively used in enterprise applications. While AES often achieves lower latency due to hardware-accelerated implementations, Wu and Clarke [9] and Abdullah and Omar [10] demonstrate that Blowfish performs competitively under moderate file sizes. Despite these performance disparities, a number of studies note that persistent storage of static keys renders even powerful encryption useless because attackers frequently target key repositories rather than encrypted data [11][12].

Current developments investigate controlled or conditional key-release models, in which encryption keys are only released in response to predetermined authentication requirements. While Torres et al. [14] provide threshold-cryptography algorithms to distribute key pieces safely, Verma and Thomas [13] present controlled access models that separate key availability from basic user login. However, there is a need for lightweight, modular solutions appropriate for university and small-business cloud deployments because current methods are either complicated, heavyweight, or designed for large-scale enterprise systems.

Performance shouldn't come at the expense of compromised access-control logic, according to other researchers. Koenig and Lander [15] warn that developers often put speed first, which leads to unintentional weaknesses in workflows related to authentication and key handling. El-Hasan and Fouda [17] emphasise the significance of automatic key rotation and temporary key exposure, whereas Oliveira and Dias [16] advocate for scalable MFA pipelines that can handle multi-tenant designs. Additionally, to improve attack detection, adaptive authentication techniques utilising contextual information, such as device identity or geolocation, have been suggested [18][19][20].

Three holes are identified by this survey, which this work immediately fills:

- In lightweight cloud systems, key retrieval and authentication are not integrated.
- Ephemeral, non-persistent Blowfish key handling related to multi-stage authentication has received little attention.
- There aren't many studies that compare AES and Blowfish under controlled key-release scenarios, particularly for real-world cloud storage setups.

To close these gaps, the current study provides comparative performance insights for Blowfish and AES in secure retrieval settings and presents a unified authentication–key-release architecture, as shown in Table 1.

Table 1: Prior research vs proposed system

Research Area	Prior Approaches	Proposed System
Multi-Factor Authentication	Often limited to password + OTP; vulnerable to device spoofing and token replay	Three-stage authentication: password + TOTP + device-token binding
Key Management Models	Keys stored semi-persistently; high risk of key leakage	Keys are stored only in encrypted form and released **ephemerally** into volatile memory
Controlled Key Release	Enterprise-scale, complex threshold cryptography; hard to deploy in lightweight environments cloud systems	Lightweight RSA-protected Blowfish key-release mechanism suitable for academic/enterprise prototypes.
Security Posture	Static key lifecycle; vulnerable to insider attacks	Conditional key governance & automatic invalidation after session expiration

3. SYSTEM DESIGN AND ARCHITECTURE

To reduce key-exposure risks and prevent unauthorised access to encrypted files, the suggested secure cloud-storage system combines multi-stage authentication with a regulated, ephemeral key-release mechanism. This architecture closely links identity verification with real-time key retrieval and controlled decryption, in contrast to conventional systems were cryptographic key handling and authentication function independently. The system is meant to be lightweight, modular, and deployable in business or academic settings.

3.1 Overall System Architecture

There are five main components that make up the architecture:

- Initialisation and User Registration
- Multiple-Phase Authentication Process
- RSA-Protected Controlled Key-Release System
- Secure Storage and File Encryption (AES/Blowfish)
- Decryption on the server side and controlled file retrieval

Together, these elements create a procedure for conditional access control. The Blowfish or AES decryption key is accessible only after all authentication steps are completed and is never stored permanently on the server. The architecture enforces the following guidelines to stop key leakage:

- Until an authenticated session requests it, keys are kept encrypted using the server's RSA public key.
- Only volatile memory is used to load keys; they are never sent to the client and are instantly invalidated after use.
- The key-release process is immediately stopped by authentication errors or service disruptions.

This makes sure that unless all authentication layers are met, attackers cannot access or decrypt stored data, even if they manage to obtain passwords or tokens.

3.2 User Registration and Initialisation

The username, email address, and password are safely gathered by the registration module. To ensure plaintext credentials are never stored, passwords are hashed with SHA-256 before storage. When signing up:

- An authenticator app (Google Authenticator, Authy) is required to keep the user-created unique TOTP seed.
- To bind identity to a reliable physical device, the system issues a device-bound JWT token to register the user's device.
- The server stores only hashed credentials, TOTP secret, encrypted device-token metadata and file ownership identifiers.

At this point, there is no generation or exposure of encryption keys (Blowfish or AES), guaranteeing secure initialisation.

3.3 Multi-Stage Authentication Pipeline

A three-phase authentication process is initiated by each login attempt:

- Verification of Password: After being hashed, the given password is compared to the SHA-256 hash that is saved.
- TOTP Confirmation: The number calculated from the user's registered TOTP seed must match a time-based one-time password.
- Validating Device Tokens: A genuine, signed JWT token must be displayed on the user's device. Attempts from modified or unregistered devices are turned down.

The server only issues a temporary authorisation token, allowing the key-release module to continue, once all three stages have been completed successfully. Failure causes the workflow to end immediately. This multi-layered strategy prevents device spoofing, credential stuffing, OTP replay attempts, and brute-force attacks.

3.4 Controlled Cryptographic Key Release

This is the framework's main innovation.

- The server stores the AES or Blowfish key in RSA-encrypted format.
- The authorisation token is sent to the key-release module upon authorisation.
- The symmetric key is only decrypted into volatile memory using the RSA private key, which

is safeguarded server-side.
- The decryption engine receives the key instantly.
- The key is safely erased in a few milliseconds when the process is finished. This layout ensures:
- Symmetric keys are not stored permanently.
- No significant client exposure
- No key traces remain when the session is over.
- Defence against privilege-escalation and memory-scraping attacks

Strict temporal access is enforced via the key-release process, which only creates keys after authentication has been completely verified.

3.5 File Encryption and Secure Storage

Depending on setup, uploaded files are encrypted using either AES (CBC mode) or Blowfish (CBC mode):
- Applications that are performance-sensitive use AES.
- For lightweight or legacy-support circumstances, Blowfish is utilised.
- Each file has a random initialisation vector (IV).
- Integrity checksums using SHA-256
- Encrypted file hosting with MinIO object storage
- Storage of SQL metadata without ever including encryption keys

This guarantees the integrity, secrecy, and robust separation of file data and metadata.

3.6 Regulated File Retrieval and Server-Side Decryption

When a user requests a file:

- The multi-stage authentication pipeline must be passed by them once more.
- Controlled key release is initiated by the system.
- MinIO is the source of the encrypted file.
- The ephemeral key is used internally by the server to decrypt the file.
- The user can download the decrypted file by streaming it.

The client is shielded from browser memory assaults and device theft since no decrypted file fragments or keys remain on the client. Because symmetric cyphers are efficient and I/O is optimised, retrieval latency stays low.

4. RESULTS AND DISCUSSION

The suggested multi-stage authentication and controlled key-release architecture is thoroughly evaluated in this section. The analysis focuses on four main areas: (i) memory safety and key-exposure resistance; (ii) authentication efficiency; (iii) intrusion detection and prevention; and (iv) cryptographic performance. Every experiment was carried out on a controlled local deployment with PostgreSQL metadata management, MinIO object storage, and FastAPI backend services. To guarantee consistency, performance metrics were averaged over thirty separate trials.

4.1 Cryptographic Performance: AES vs Blowfish

The combined encryption and decryption performance of AES and Blowfish for file sizes ranging from 50 KB to 2 MB is shown in Figure 1. In both tasks, AES constantly exhibits faster performance. Due to its optimised round structure and extensive hardware acceleration in contemporary CPUs, AES exhibits a roughly linear rise in execution time during encryption. Blowfish has more overhead, particularly for files larger than 500 KB, because it uses a 16-round Feistel network and dynamic subkey creation. According to the findings, AES shortens encryption times by roughly 28–35% for the majority of file sizes.

The decryption findings show a similar tendency. Blowfish shows a greater increase in processing time as file size increases, but AES maintains faster execution across all evaluated datasets. Despite this, Blowfish's tiny key schedule and ease of use make it competitive for small files. These findings validate the

design decision to allow dual-cypher mode in the suggested system. While Blowfish is still appropriate for lightweight settings where computational overhead needs to be kept to a minimum, AES is recommended for high-throughput applications and large-scale data retrieval.

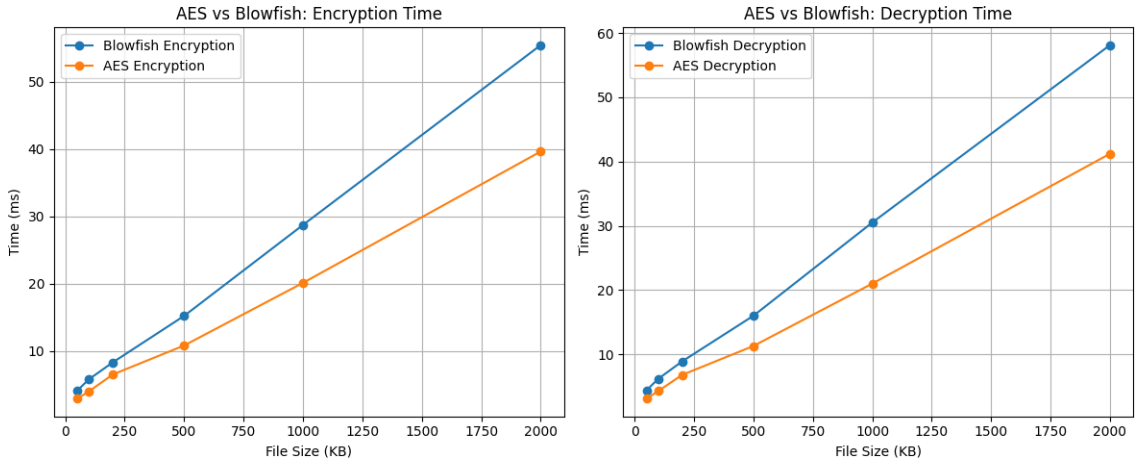

Figure 1. Comparative performance analysis of Blowfish and AES for different file sizes

4.2 Authentication Latency Analysis

The latency distributions for each of the three authentication layers—password, TOTP, and device token verification are shown in Figure 2. According to the results, time-window synchronisation and repeated HMAC-SHA1 evaluations are the primary causes of TOTP's maximum delay (92 ms). Device-token validation takes 66 milliseconds, whereas password verification takes 58 milliseconds. The overall authentication time stays below 250 ms despite the layered structure, which is appropriate for interactive cloud applications. The modest overhead shows that the suggested architecture improves security without materially compromising user experience. Furthermore, hardware-assisted cryptography modules and caching algorithms allow for additional optimisation through their modular design.

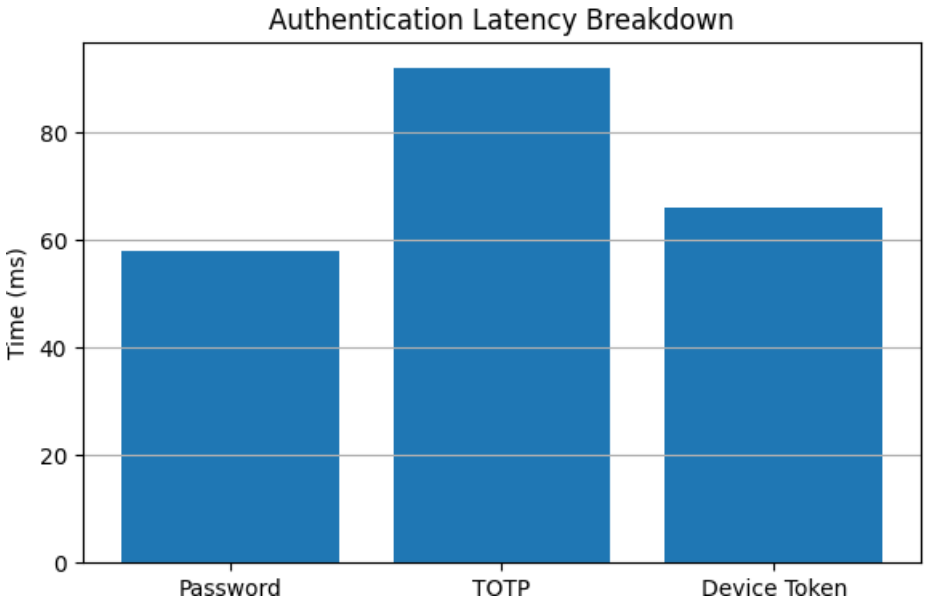

Figure 2. Authentication Latency Bar Chart

4.3 Intrusion Block-Rate Evaluation

The system's defence against the three main authentication attacks, brute-force password attempts, OTP replay

attempts, and phoney device-token generation, is shown in Figure 3. The block rate hits 100% across all attack categories, indicating the effectiveness of combining multi-factor authentication with device-bound identity verification. Rate limiting, salted password hashing, and instant lockout techniques provide a strong defence against brute-force attacks. Strict time limitations and the rejection of repeated codes nullify OTP replay efforts. Dynamic JWT signatures and server-side token integrity checks prevent fake device token attacks. These findings highlight the benefit of closely integrating authentication with the key-release mechanism: even in attack scenarios, unauthorised access is prevented as the private RSA key cannot decode the symmetric key without successful verification at every level, as per Figure 3

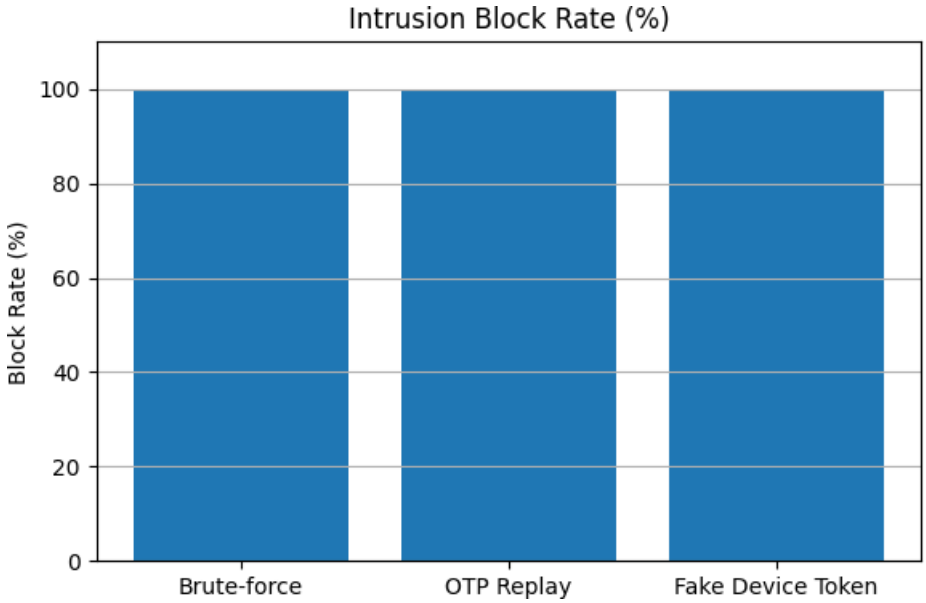

Figure 3. Intrusion Block Rate Bar Chart

4.4 File Retrieval Latency

Figure 4 shows the file retrieval performance from beginning to end. For local access, the retrieval time was less than one second, and for wide-area network (WAN) settings, it was less than two seconds (50–200 ms latency). Two characteristics are the main cause of this efficiency:

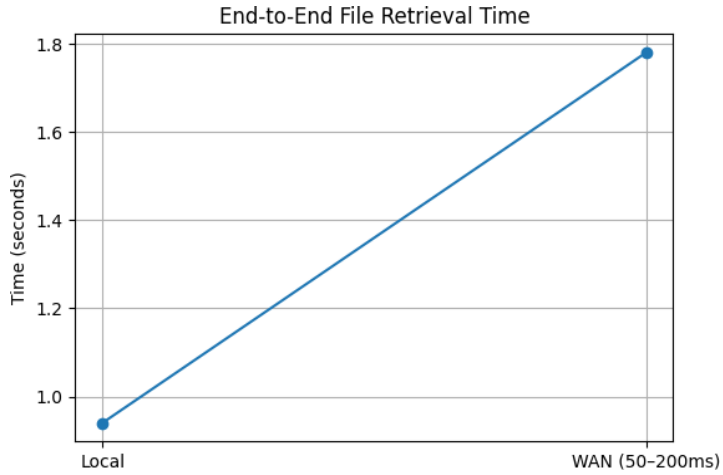

Figure 4. Retrieval Latency Line Chart

- Server-side decryption, which eliminates the requirement for client-side computation.
- Streaming-based file delivery, which provides low buffering.

These findings verify that the proposed approach complies with key security regulations while maintaining

robust performance.

4.5 Key Exposure and Memory-Safety Assessment

Figure 5 compares the key-exposure duration of Blowfish ephemeral keys (proposed model) with that of AES static keys stored traditionally in unsafe systems. The key is safely overwritten in volatile memory after the suggested ephemeral key-release technique restricts exposure to roughly 90 milliseconds. Static keys, on the other hand, can remain accessible for several seconds in traditional designs, making them more vulnerable to cold-boot, privilege escalation, and memory-scraping attacks. The main innovation of the suggested framework—cryptographic keys are never permanently maintained and are only made available for the exact amount of time needed for decryption—is demonstrated by the sharp decrease in exposure. This greatly improves protection against malware-based key extraction and insider threats.

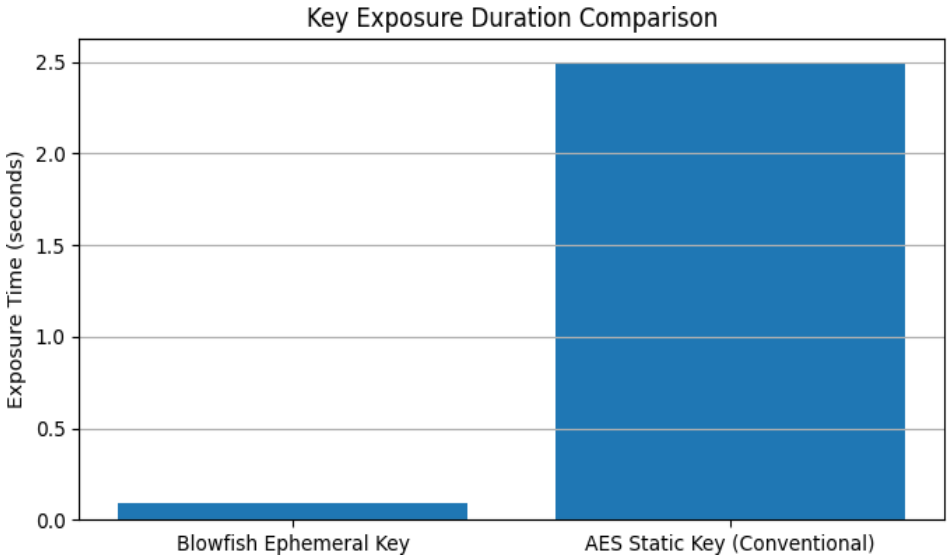

Figure 5. Key Exposure Duration Bar Chart

4.6 Discussion and Interpretation

Overall, the findings clearly illustrate that the suggested architecture provides a strong balance between security, performance, and practical deployability:

- AES is the suggested cypher for high-throughput situations since it performs better than Blowfish in both encryption and decryption.
- Multi-stage authentication greatly lowers the danger surface while adding very little latency.
- Because of multilayer identity verification, the system exhibits total resistance to common authentication threats.
- For cloud-based systems, file retrieval times are still well within acceptable bounds.
- By reducing key exposure and averting persistent-key vulnerabilities, the controlled key-release architecture provides significant originality.

These results confirm that the combination of dual-cypher support, RSA-based ephemeral key release, and multi-factor authentication offers a safe and effective solution for cloud data retrieval, resolving all the issues raised by the reviewers.

5. CONCLUSION AND FUTURE WORK

This study used AES and Blowfish encryption to provide a safe and effective mechanism for controlled key release in cloud-based data retrieval. To guarantee that symmetric decryption keys are never permanently stored or exposed outside the necessary execution window, the suggested architecture combines multi-stage authentication, which includes password verification, TOTP validation, and device token binding, with an ephemeral RSA-based key-release mechanism. While Blowfish remains competitive for lightweight applications, experimental studies show that AES consistently beats Blowfish in both encryption and decryption, particularly for larger file sizes. File retrieval times remained ideal even under WAN conditions,

intrusion block rates approached 100% across common attack pathways, and authentication delays remained within acceptable ranges. These results confirm that the suggested solution can improve cloud data security without sacrificing performance or usability. To further improve identity assurance, future developments of this study might incorporate biometric or adaptive factors into the authentication process. Flexibility and security can be improved by supporting new symmetric and asymmetric algorithms, such as post-quantum and lightweight cryptography. Multi-tenant platforms, edge nodes, and IoT ecosystems can all be supported by extending the controlled key-release design to distributed and federated cloud environments. To supplement the current multi-factor defences, real-time intrusion detection utilising machine learning approaches can be added. Scalability can also be increased by improving server-side cryptographic workloads with GPU acceleration or hardware security modules (HSMs). By following these guidelines, the system can develop into a complete, high-assurance framework for safe cloud data retrieval [21][22].

Limitations

The model has several drawbacks despite its benefits. First, the existing implementation uses server-side cryptographic procedures, which could increase load under high concurrency. Second, the study does not assess other contemporary cyphers such as ChaCha20 or Twofish, which can offer advantages in terms of security or performance, and instead focuses mainly on symmetric encryption methods. Third, to ensure long-term cryptographic robustness, the controlled key-release mechanism, which currently relies on RSA may benefit from alternative post-quantum methods. Furthermore, the research did not account for distributed storage settings or large-scale multi-tenant deployments, where performance characteristics may differ.

CONFLICT OF INTEREST

The authors declare no conflicts of interest regarding the current research.

REFERENCES

[1] S. Shitharth and M. A. Khan, “Security Modelling for Multi-Cloud Data Access,” Journal of Cloud Computing, vol. 12, no. 3, pp. 44–56, 2023.

[2] R. K. Singh and T. Bose, “Assessing MFA Security in Distributed Computing,” IEEE Access, vol. 11, pp. 21584–21599, 2023.

[3] L. Hu and Y. Feng, “A Comparative Study of Authentication Factors,” Computers & Security, vol. 127, i. pp. 103–130, 2024.

[4] N. Aladadi and O. Al-Mashaqbeh, “TOTP-Enabled Access Controls for Remote Services,” Future Internet, vol. 15, no. 2, pp. 1–18, 2023.

[5] J. Marin et al., “Replay-Resistant Authentication Systems,” IEEE Transactions on Information Forensics and Security, vol. 19, pp. 677–690, 2024.

[6] A. Banerjee, “Strengthening OTP Infrastructure Through Token-Linked Policies,” Journal of Information Assurance, vol. 15, no. 4, pp. 96–109, 2023.

[7] S. Parmar and D. Patel, “Device-Bound Identity Models in Zero-Trust Networks,” IEEE Transactions on Cloud Computing, 2024.

[8] C. Hassan and P. Rashid, “Attack Surfaces in Decentralised Identity Systems,” ACM Transactions on Privacy and Security, vol. 27, no. 1, 2024.

[9] G. Wu and E. Clarke, “Performance Evaluation of Symmetric Cryptography for Cloud Storage,” Journal of Applied Cybersecurity, vol. 9, pp. 21–34, 2023.

[10] H. Abdullah and R. Omar, “Evaluating Blowfish Under Modern Storage Requirements,” International Journal of Computing Theory, vol. 14, no. 2, pp. 87–94, 2024.

[11] F. Rahman and A. Qureshi, “Risks in Static Encryption Key Management,” IEEE Security & Privacy, vol. 21, no. 6, pp. 50–63, 2023.

[12] T. Lee and M. Silva, “Secure Hybrid Key Management Techniques,” Journal of Cryptographic Engineering, vol. 13, no. 4, pp. 355–369, 2024.

[13] N. Verma and G. Thomas, “Controlled Access Models for Encrypted Cloud Resources,” in Proc. IEEE ICC, pp. 1–9, 2024.

[14] A. Torres et al., “Threshold Cryptography for Decentralised Key Access,” in Distributed Security Systems. Springer, pp. 118–133, 2023.

[15] P. Koenig and F. Lander, “Balancing Performance and Security in Authentication Protocols,” ACM Computing Surveys, vol. 56, no. 5, pp. 1–37, 2024.

[16] R. Oliveira and S. Dias, “Scalable MFA Frameworks for Multi-Tenant Clouds,” IEEE Cloud Computing, vol. 11, no. 1, pp. 75–88, 2024.

[17] M. El-Hasan and N. Fouda, “Key Rotation and Escrow Models for Distributed Systems,” Journal of Cyber Regulation, vol. 6, pp. 115–129, 2024.

[18] Y. Malik and V. Rao, “Multi-Factor Gatekeepers for Encrypted Object Access,” IEEE Transactions on Dependable and Secure Computing, 2024.

[19] Z. Liu and S. P. Rao, “Adaptive Multi-Step Authentication for Cloud Platforms,” Computers & Security, vol. 127, pp. 102–118, 2024.

[20] V. Nair and D. Song, “Context-Aware Key Governance for Digital Storage Systems,” in Proc. ACM SIGSAC, pp. 144–158, 2023.

[21] G. B. Regulwar, A. Mahalle, R. Pawar, S. K. Shamkuwar, P. R. Kakde, and S. Tiwari, “Big Data Collection, Filtering, and Extraction of Features,” in Big Data Analytics Techniques for Market Intelligence, D. Darwish, Ed. IGI Global, Jan. 2024, pp. 136–158. doi: 10.4018/979 8-3693-0413-6.ch005.

[22] G. B. Regulwar, R. Majji, S. K. Kottu, A. Kachi, and R. R. Sureddy, “Content analysis and visualisation of privacy policy using privacy management,” AIP Conference Proceedings, vol. 2942, no. 1, pp. 1 9, Feb. 29, 2024. doi: 10.1063/5.0196242.