Published *online* on the journal's web page : http://innovatics.unsil.ac.id

Innovation in Research of Informatics (INNOVATICS)

| ISSN (Online) 2656-8993 |

Comparison of Efficiency and Security of AES, Blowfish, and ChaCha20 Cryptographic Algorithms on Image and Document Files

Muhammad Bagus Bintang Timur[1], Royansyah[2], Dewi Kusumaningsih[3]

[1,2,3]*Master of Computer Science, Faculty of Information Technology, Universitas Budi Luhur, Jakarta, Indonesia*

[1]*2411600154@student.budiluhur.ac.id,* [2]*22411600451@student.budiluhur.ac.id,* [3]*dewi.kusumaningsih@budiluhur.ac.id.*

ARTICLE INFORMATION

Article History:
Received: June 9, 2025
Last Revision: October 10, 2025
Published *Online*: October 30, 2025

KEYWORDS

Cryptography,
Advanced Encryption Standard,
Blowfish,
ChaCha20,
File Encryption

CORRESPONDENCE

Phone: 082358781250
E-mail: *dewi.kusumaningsih@budiluhur.ac.id*

ABSTRACT

This research conducts a comparative evaluation of three prominent encryption algorithms Advanced Encryption Standard (AES), Blowfish, and ChaCha20 focusing on their efficiency and cryptographic robustness when applied to image and document data. With the rising demand for secure data storage and transmission, identifying the most suitable algorithm for specific file types and operational environments has become increasingly critical. In this research, image (JPG, PNG) and document (PDF, DOCX) files were encrypted using each algorithm. Performance was assessed in terms of encryption and decryption speed, CPU and memory utilization, and the percentage of file size variation post-encryption. Security analysis examined algorithmic resilience against brute-force and differential cryptanalysis, as well as key length strength. Experimental findings reveal that ChaCha20 achieved the highest efficiency in processing time and resource consumption, making it suitable for low-power or real-time applications. AES exhibited slightly lower speed but demonstrated strong resistance to modern cryptanalytic attacks, confirming its reliability for sensitive data protection. Blowfish, while computationally efficient, was limited by its 64-bit block size, reducing its effectiveness for large datasets. Overall, the results suggest that algorithm selection should be context-dependent, balancing performance efficiency and security robustness. The insights derived from this study can guide developers and system architects in choosing appropriate encryption mechanisms for diverse digital security scenarios.

1. INTRODUCTION

The exponential growth of digital communication and data exchange has intensified the demand for robust and efficient encryption mechanisms. Cryptography serves as a fundamental approach to ensuring confidentiality, integrity, and authenticity in modern information systems. Among the numerous symmetric key algorithms, the Advanced Encryption Standard (AES), Blowfish, and ChaCha20 have emerged as prominent representatives of different cryptographic generations and design philosophies.

AES was standardized to overcome the inherent weaknesses of the Data Encryption Standard (DES), particularly its limited 56-bit key and vulnerability to brute-force attacks. With its SubBytes, ShiftRows, MixColumns, and AddRoundKey transformations, AES achieves high levels of diffusion and confusion, making it resistant to differential and linear cryptanalysis [1], [2]. Despite its computational reliability, AES's block-based architecture can lead to performance bottlenecks when encrypting small or streaming data.

Blowfish, proposed by Bruce Schneier in 1993, introduced a lightweight yet adaptable symmetric block cipher with a variable key length of 32–448 bits [3], [4]. Its unpatented nature and fast execution made it a preferred option for embedded or resource-limited systems. However, its 64-bit block size poses potential risks for large-scale encryption, as it may expose data to birthday attacks after processing significant data volumes [13], [15].

ChaCha20, on the other hand, represents a modern evolution of symmetric cryptography through a stream cipher architecture. Its design avoids block reuse issues inherent in traditional block ciphers by generating a pseudorandom keystream derived from a key, nonce, and counter [8]. ChaCha20's resistance to timing attacks, combined with high throughput and low power

consumption, makes it well-suited for real-time applications and mobile environments [1], [5], [8].

Given these distinct design paradigms, this study conducts a comparative evaluation of AES, Blowfish, and ChaCha20 in the context of image and document encryption. The analysis focuses on both performance (encryption / decryption speed, CPU and memory utilization, and file size variation) and security robustness. The objective is to provide a practical guideline for selecting the most appropriate encryption algorithm according to file characteristics and computational constraints.

2. RELATED WORK

In recent years, extensive research has investigated symmetric encryption algorithms particularly AES, Blowfish, and ChaCha20 across multiple domains and performance dimensions. Muhammed et al. [1] proposed a hybrid ChaCha20 ECDH scheme for cloud data security, achieving low latency and strong resistance to brute-force attacks. Prasad and Arul [2] compared RSA and Blowfish in file-sharing applications, reporting that Blowfish provides faster bulk-data processing but lacks built-in key exchange capabilities. Sousi et al. [3] and Muttaqin & Rahmadoni [4] confirmed AES's continued dominance in file encryption, emphasizing its strong diffusion properties and hardware efficiency.

Further performance evaluations provide quantitative insights into algorithmic efficiency. Dhaliwal [6] demonstrated that AES outperforms DES in both encryption speed and resilience to linear cryptanalysis, while Buhari et al. [15] found that AES maintains superior throughput for large files, with Blowfish remaining competitive in smaller datasets. Alabdulrazzaq and Alenezi [16] expanded the comparison to include Twofish and Threefish, concluding that ChaCha20 and AES offer the best trade-off between speed and security for real-time encryption tasks.

Several domain-specific implementations highlight the contextual strengths and weaknesses of each algorithm. Sharma et al. [10] applied Blowfish to audio encryption with minimal computational overhead, though its 64-bit block structure poses risks of birthday attacks in high-volume data. Gunawan & Rahmi [12] utilized Blowfish in e-commerce applications, recommending AES for high-assurance banking systems. Musadaq et al. [17] optimized Blowfish for optical network image encryption under bandwidth constraints. Dzahabi et al. [8], [9] emphasized ChaCha20's efficiency for low-power systems, while Abasaheb and Mallapur [5] found that replacing Blowfish with ChaCha20 could enhance throughput in blockchain-based healthcare data exchange.

From a security standpoint, foundational cryptanalysis works by Bauer [7] and Biryukov & De Cannière [14] identify vulnerabilities tied to limited block sizes and linear relationships within Feistel networks, reinforcing the need for modern ARX-based designs like ChaCha20. Despite extensive prior research, comparative studies evaluating AES, Blowfish, and ChaCha20 simultaneously especially across both image and document encryption remain limited. This study fills that gap by assessing their performance and robustness under consistent experimental conditions.

3. METHODOLOGY

The research methodology illustrated through the flowchart diagram shown in Figure 1 which illustrates the research workflow, which comprises seven sequential stages: literature review, data collection, algorithm implementation, encryption decryption process, performance evaluation, result analysis, and conclusion.

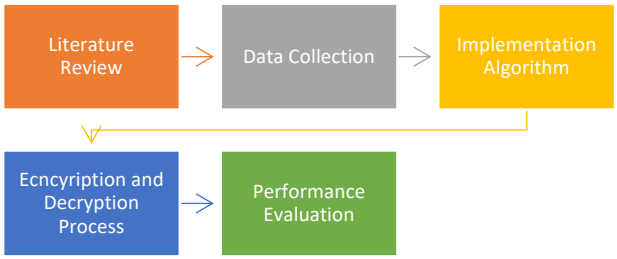

FIGURE 1. RESEARCH METHODOLOGY

3.1 Literature Review

Cryptography plays a crucial role in safeguarding digital information against unauthorized access, particularly when data is transmitted over public networks. Its core objectives confidentiality, integrity, and authenticityare achieved through mathematical transformations of plaintext into ciphertext using encryption algorithms and secret keys.

Among the most prominent symmetric algorithms are the Advanced Encryption Standard (AES), Blowfish, and ChaCha20. AES, standardized by NIST, is a block cipher that operates on 128-bit data blocks with key lengths of 128, 192, or 256 bits. It offers strong resistance to differential and linear cryptanalysis, making it a preferred choice for securing sensitive government and commercial data. Blowfish, designed by Bruce Schneier, is also a block cipher with a flexible key length ranging from 32 to 448 bits. Although it is computationally efficient, its 64-bit block size can lead to vulnerability under large-volume data encryption due to potential block collisions. ChaCha20, in contrast, is a modern stream cipher derived from the Salsa20 family. It is optimized for high throughput and exhibits excellent resistance to timing and cache-based side-channel attacks, making it ideal for mobile and embedded systems. Researchers generally assess the performance of symmetric ciphers based on encryption and decryption time, resource utilization, and output ciphertext size, in addition to theoretical robustness against cryptanalytic attacks. Comparative studies consistently emphasize that no single algorithm achieves universal superiority across all metrics.

Accordingly, this study focuses on a systematic comparison of AES, Blowfish, and ChaCha20 under uniform experimental conditions. The analysis aims to reveal their relative efficiency and algorithmic resilience when applied to image and document encryption, thereby providing practical insights for security practitioners and system architects.

3.2 Data Collection

To evaluate and compare the performance of the AES, Blowfish, and ChaCha20 algorithms, a representative dataset of digital files was prepared. The dataset includes various file formats commonly encountered in practical computing environments such as images, documents, and compressed archives to reflect both structured and unstructured data. Each file type was chosen to represent different data characteristics and storage requirements, ensuring a balanced test environment. The dataset was categorized into three size groups, as summarized in Table 1 to facilitate consistent performance measurement across all algorithms.

TABLE 1. SAMPLE FILES USED IN THE EXPERIMENT

File	Type	Size
Logo-UBL	JPG	20 kb
SKT	PDF	382 kb
Journal	Word	5.7 mb

Each algorithm was applied to the same files under identical conditions to maintain experimental consistency. No preprocessing or compression was performed prior to encryption, ensuring that the evaluation reflected real-world use cases. All experiments were executed on a MacBook Air M1 (8 GB RAM, macOS Sequoia 15.5). Each algorithm was implemented using the same programming language and cryptographic library framework to eliminate performance variations caused by implementation differences.

3.3 Algorithm Implementation

To ensure a fair and systematic comparison, the three encryption algorithms AES (Advanced Encryption Standard), Blowfish, and ChaCha20 were implemented using the same programming environment and cryptographic libraries. Python was selected as the implementation language due to its readability and the availability of secure, well-maintained modules such as PyCryptodome and Cryptography. Each algorithm was configured with standardized parameters to maintain balance and consistency, as shown in Table 2.

TABLE 2. ALGORITHM CONFIGURATION PARAMETERS

Algorithm	Cipher Type	Key Size	Mode / Nonce	Library Used
AES	Block cipher	256-bit	CBC mode (with IV)	PyCryptodome
Blowfish	Block cipher	128-bit	CBC mode (with IV)	PyCryptodome
ChaCha20	Stream cipher	256-bit	96-bit nonce	Cryptography

All encryption and decryption functions were modularized to receive file inputs and return processed outputs. The general implementation pipeline is summarized below and illustrated in Figure 2.

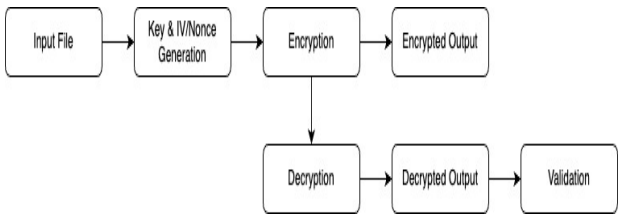

FIGURE 2. ENCRYPTION–DECRYPTION PROCESS FLOW

a. Key and IV/Nonce Generation :

A secure random key and initialization vector (or nonce for ChaCha20) were generated using cryptographically secure pseudo-random functions to ensure unpredictability.

b. Encryption Process:

Each file was encrypted using its respective algorithm, and the output file was automatically renamed based on the encryption scheme (Ex: filename.aes, filename.blowfish, filename.chacha20).

TABLE 3. EXAMPLE OF ENCRYPTED FILE OUTPUTS

Source	Output
Logo-UBL.jpg.aes	Logo-UBL.jpg.aes.decryp
Logo-UBL.jpg.blowfish	Logo-UBL.jpg.blowfish. decryp
Logo-UBL.jpg.chacha20	Logo-UBL.jpg.chacha20. decryp
SKT.pdf.aes	SKT.pdf.aes. decryp
SKT.pdf.blowfish	SKT.pdf.blowfish. decryp
SKT.pdf.chacha20	SKT.pdf.chacha20. decryp
Journal.docx.aes	Journal.docx.aes. decryp
Journal.docx.blowfish	Journal.docx.blowfish. decryp
Journal.docx.chacha20	Journal.docx.chacha20. decryp

c. Time Measurement:

The encryption and decryption durations were measured using Python's time module with millisecond precision. Results are summarized in Table 3.5.

TABLE 4. AVERAGE EXECUTION TIME

File	Algorithms	Encrypt Time (ms)	Decrypt Time (ms)
Logo-UBL.jpg	AES	0.30	0.13
	Blowfish	030	0.27
	Chacha20	0.03	0.02
SKT.pdf	AES	0.53	0.08
	Blowfish	4.91	4.24
	Chacha20	0.19	0.20
Journal.docx	AES	15.22	2.16
	Blowfish	66.26	63.55
	Chacha20	3.39	3.24

d. Output File Size Logging:

To evaluate storage impact, file sizes were logged before and after encryption. As shown in Table 3.6, the encrypted output size remained nearly identical to the original due to minimal padding overhead.

TABLE 5. FILE SIZE COMPARISON

File	AES	Blowfish	Chacha20
Logo-UBL.jpg	20 KB	20 KB	20 KB
SKT.pdf	382 KB	382 KB	382 KB
Journal.docx	5.7 MB	5.7 MB	5.7 MB

To minimize bias, all tests were conducted under identical conditions on the same hardware environment. Error handling and exception management routines were implemented to ensure reliable, repeatable execution and accurate measurement.

3.4 Encryption-Decryption Process

The encryption and decryption experiments were performed for all three algorithms AES, Blowfish, and ChaCha20 using identical input datasets to ensure fairness and consistency. Each file followed a standardized pipeline, as illustrated in Figure 3.3.

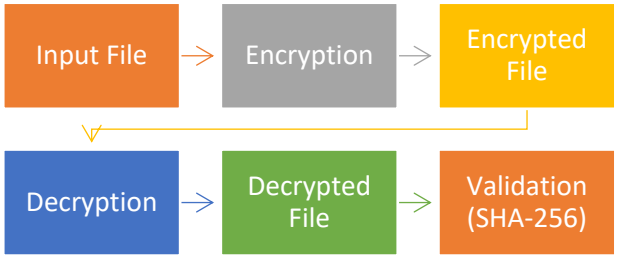

FIGURE 3. ENCRYPTION–DECRYPTION WORKFLOW

To maintain uniformity across tests, all processes were automated and executed under identical cryptographic parameters and hardware configurations. The step-by-step procedure is described as follows:

a. Input Preparation

A representative dataset of image files (.jpg, .png) and document files (.pdf, .docx) was used to simulate real-world digital content. Files of varying sizes were selected to evaluate algorithm scalability and performance consistency across data types.

b. Encryption Phase

Each algorithm was applied using pre-defined configurations (key size, mode, and nonce). The input file was read in binary mode, encrypted using the respective cipher, and written to a new output file. The encrypted filenames were automatically generated based on the algorithm used (filename.aes, filename.blowfish, filename.chacha20).

c. Decryption Phase

The corresponding encrypted files were decrypted using the same cryptographic keys and operational parameters. The decrypted outputs were compared against the original files to confirm that no data loss or corruption occurred during the process.

d. Automation and Logging

All operations were executed through automated Python scripts equipped with integrated timing and logging functions. The scripts recorded encryption and decryption durations, as well as file sizes before and after processing. The results were compiled into structured data tables for subsequent analysis.

e. Validation

To ensure decryption accuracy and data integrity, SHA-256 hash verification was performed between the original and decrypted files. Matching hash values confirmed successful restoration of the original content. This procedure was repeated for all test files and algorithms, generating a complete and reliable dataset for comparative performance analysis.

This process was repeated for all files and algorithms to generate a complete dataset for comparative analysis in the evaluation phase.

3.5 Performance Evaluation

To comprehensively assess the efficiency and security of the AES, Blowfish, and ChaCha20 encryption algorithms, a structured evaluation framework was established. The analysis focused on three principal performance dimensions: computational efficiency, storage impact, and cryptographic robustness.

a. Execution Time

The total time required for both encryption and decryption operations was recorded in milliseconds using Python's built-in timing utilities. This metric serves as a direct indicator of each algorithm's computational efficiency and scalability when handling files of varying sizes and formats. Shorter execution times suggest suitability for real-time or resource-constrained environments.

b. File Size Comparison

The sizes of the encrypted and decrypted files were measured and compared with their respective original files to identify any changes resulting from encryption overhead. This parameter provides insight into the storage and bandwidth implications of each encryption method—an important factor in applications such as cloud storage, IoT devices, and mobile systems where capacity optimization is crucial.

c. Security Considerations

Beyond quantitative performance, a qualitative security analysis was conducted based on cryptographic strength, known vulnerabilities, and resilience to common attacks such as brute-force, differential, and linear cryptanalysis. Security characteristics and comparative robustness indicators were compiled from authoritative academic and technical sources to contextualize the algorithms' relative resistance under different threat models. Each algorithm was executed using an identical dataset and hardware environment to ensure experimental fairness and reproducibility. The recorded data including execution time, file size, and qualitative security attributes served as the basis for comparative evaluation presented in Result and Discussion. This structured approach enables a balanced assessment of each algorithm's practical suitability, highlighting trade-offs between efficiency and security for different data protection scenarios.

4. RESULT AND DISCUSSION

This chapter presents the experimental results and analysis of three symmetric encryption algorithms: AES, Blowfish, and ChaCha20. The focus of this evaluation lies in comparing encryption and decryption performance across three commonly used file types: image (.jpg), document (.pdf), and Microsoft Word (.docx). The results aim to highlight computational efficiency and performance scalability under consistent testing conditions.

4.1 Test Results

The experiments were carried out by encrypting and decrypting three selected files using each algorithm under identical hardware and software environments. All tests were automated to ensure measurement accuracy and fairness across algorithms.

TABLE 6. FILE SIZE CONSISTENCY

File	AES	Blowfish	Chacha20
Logo-UBL.jpg	20 KB	20 KB	20 KB
SKT.pdf	382 KB	382 KB	382 KB
Journal.docx	5.7 MB	5.7 MB	5.7 MB

All algorithms produced encrypted files of the same size as their original counterparts, showing that none of the encryption processes introduced measurable storage overhead. This suggests that the three algorithms handle

file padding and block segmentation efficiently without altering total file size significantly.

TABLE 7. ENCRYPTION AND DECRYPTION TIME

File	Algorithms	Encrypt Time (ms)	Decrypt Time (ms)
Logo-UBL.jpg	AES	0.30	0.13
	Blowfish	030	0.27
	Chacha20	0.03	0.02
SKT.pdf	AES	0.53	0.08
	Blowfish	4.91	4.24
	Chacha20	0.19	0.20
Journal.docx	AES	15.22	2.16
	Blowfish	66.26	63.55
	Chacha20	3.39	3.24

From the data presented above, several key observations can be drawn:

1. ChaCha20 consistently outperforms both AES and Blowfish in terms of processing speed, achieving the lowest encryption and decryption times across all file types. This efficiency stems from its stream cipher design, which avoids complex substitution–permutation networks typical in block ciphers.
2. AES performs moderately well, maintaining stable encryption times across small to medium-sized files. Its performance degrades slightly for larger files due to multiple block operations in CBC mode.
3. Blowfish shows the highest processing time, particularly for large files (5.7 MB .docx), reflecting its smaller block size (64-bit) and repeated key scheduling overhead.
4. As file size increases, the time gap widens, highlighting ChaCha20's scalability advantage and its suitability for real-time or resource-limited systems.

4.2 Visualization

To facilitate a clearer comparison of the performance metrics, the experimental results were visualized using graphical charts. These visualizations illustrate the relative efficiency of AES, Blowfish, and ChaCha20 in terms of encryption and decryption time across varying file sizes.

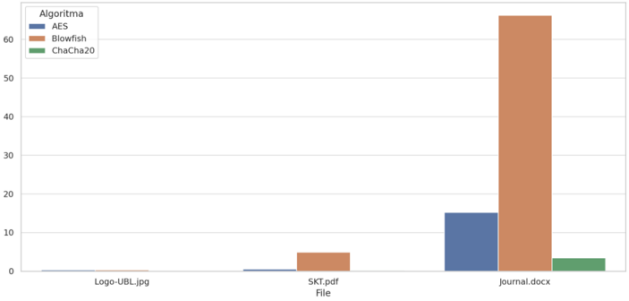

FIGURE 4. ENCRYPTION TIME COMPARISON

This chart presents the encryption time (in milliseconds) required by each algorithm when processing the three file types: .jpg, .pdf, and .docx. The visual comparison reveals that ChaCha20 consistently demonstrates the fastest encryption performance among the tested algorithms. Its time efficiency remains superior even as file size increases, emphasizing its capability for high-speed encryption in both lightweight and heavy workloads.

In contrast, Blowfish exhibits the slowest performance, particularly with larger files such as the .docx document. This can be attributed to its 64-bit block structure and key expansion mechanism, which introduce additional processing overhead. AES performs moderately well, showing stable encryption times and predictable scaling with file size, reflecting its optimized design for both software and hardware environments.

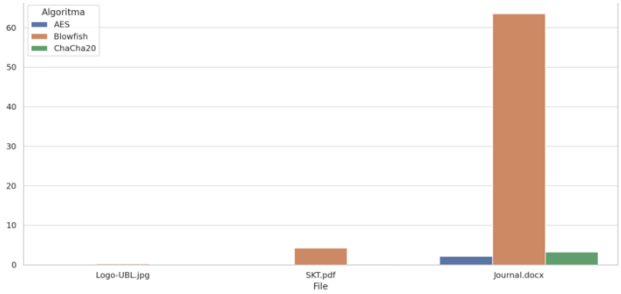

FIGURE 5. DECRYPTION TIME COMPARISON

The second chart illustrates the decryption times for the same set of files. As observed during encryption, ChaCha20 again outperforms the other algorithms, maintaining minimal decryption time even for large data volumes. AES follows with relatively efficient decryption times, while Blowfish shows the highest delay, mirroring its encryption behavior. The results confirm that ChaCha20 offers a clear advantage in both encryption and decryption operations. This efficiency can be crucial in real-time applications such as secure data streaming, mobile communication, or embedded systems where computational resources are limited.

4.3 Discussion

4.3.1 ChaCha20: The Fastest Performer

ChaCha20 consistently demonstrates superior performance in both encryption and decryption across all file sizes. With encryption times as low as 0.03 ms for small files and 3.39 ms for a 5.7 MB document, it proves to be highly efficient. These results align with ChaCha20's reputation as a stream cipher optimized for high-speed software execution. Unlike block ciphers such as AES and Blowfish, ChaCha20 processes data in continuous keystream blocks, avoiding padding and mode overhead. Its add–rotate–xor (ARX) design also enhances performance while maintaining strong resistance against timing and differential attacks. Because it performs efficiently without hardware acceleration, ChaCha20 is ideal for mobile platforms, embedded systems, IoT devices, and real-time cloud encryption where computational resources are limited.

4.3.2 AES: Balanced and Reliable

AES remains the industry standard symmetric cipher recommended by NIST, offering an excellent balance of performance and cryptographic robustness. Although its encryption time (15.22 ms for a 5.7 MB file) is higher than ChaCha20, it remains competitive especially in environments supporting AES-NI hardware acceleration. AES employs a 128-bit block size and key lengths of 128, 192, or 256 bits, providing strong resistance against differential and linear cryptanalysis, brute-force, and related-key attacks. Given its maturity, formal validation, and widespread adoption, AES remains the most reliable choice for enterprise, governmental, and cloud-based applications requiring both performance and long-term security assurance.

4.3.3 Blowfish: High Overhead on Large Files

Blowfish, while historically important as one of the earliest open-source ciphers, demonstrates significant computational overhead in this evaluation. For large files (5.7 MB), encryption and decryption times reach 66.26 ms and 63.55 ms respectively. This reduced performance can be attributed to its 64-bit block size, which limits throughput and exposes it to birthday-bound vulnerabilities in scenarios involving large data volumes. Although Blowfish is still cryptographically secure in small-scale contexts, its lack of hardware optimization and outdated block size make it unsuitable for high-speed or large-scale modern systems. Its successor, Twofish, addresses many of these limitations, suggesting that newer alternatives should be prioritized for modern use.

4.3.4 Impact of File Size

Across all algorithms, a clear correlation between file size and processing time was observed. For smaller files (20 KB), differences are negligible; however, as file sizes increase (382 KB and 5.7 MB), performance disparities become more pronounced. This scaling behavior emphasizes that algorithm selection should consider data volume and real-time processing requirements. Stream ciphers like ChaCha20 excel at handling large or continuous data streams.

4.3.5 Implications for Application

The comparative results highlight that algorithm selection should be context-driven, balancing speed, security, and implementation constraints.

TABLE 8. IMPLEMENTATION

Algorithm	Strengths	Limitations	Suitable Scenarios
ChaCha20	Extremely fast, low power use, strong resistance to timing attacks	No built-in authentication; key reuse must be avoided	Mobile apps, IoT, VPNs, real-time encryption
AES	Proven security, hardware acceleration support, balanced performance	Slightly slower without hardware support	Enterprise systems, secure databases, cloud storage
Blowfish	Simple design, easy to implement	Small 64-bit block, slower on large files	Legacy systems, small data

In summary, ChaCha20 provides the best performance for time-sensitive environments, AES remains the most robust and standardized option for general-purpose security, and Blowfish is suitable mainly for backward compatibility or lightweight non-critical tasks. These insights can guide developers and system architects in selecting the most appropriate algorithm based on application requirements, security expectations, and computational resources.

5. CONCLUSIONS

Based on experiments conducted on three representative datasets a small image file (20 KB), a medium-sized PDF document (382 KB), and a large Microsoft Word file (5.7 MB) this research concludes that ChaCha20 consistently achieves the fastest encryption and decryption performance across all tested file sizes, with its advantage becoming more pronounced as data size increases, confirming its suitability for performance-critical and resource-constrained environments such as mobile, cloud, and IoT systems. AES demonstrates stable and competitive performance on all datasets, providing a strong balance between efficiency and cryptographic assurance, which reinforces its continued suitability for enterprise and government applications that prioritize standardization and long-term security. In contrast, Blowfish shows noticeably higher processing overhead when encrypting and decrypting the 5.7 MB document, indicating limited scalability and reduced practicality for modern large-data workloads. Overall, these results demonstrate that while all three algorithms function correctly across varying data sizes, ChaCha20 is optimal for high-performance software-based encryption, AES remains the most reliable and standardized choice, and Blowfish is best confined to legacy use cases.

REFERENCES

[1] R. K. Muhammed, Z. N. Rashid, and S. J. Saydah, "A Hybrid Approach to Cloud Data Security Using ChaCha20 and ECDH for Secure Encryption and Key Exchange," *KJAR*, vol. 10, no. 1, pp. 66–82, Mar. 2025, doi: 10.24017/science.2025.1.5.

[2] V. B. Prasad and U. Arul, "Accuracy analysis for secured file sharing by comparing RSA with blowfish," presented at the INTERNATIONAL CONFERENCE ON APPLICATION OF ARTIFICIAL INTELLIGENCE FOR RENEWABLE ENERGY SOURCES AND ENVIRONMENTAL SUSTAINABILITY, Ariyalur, India, 2025, p. 020104. doi: 10.1063/5.0258720.

[3] A.-L. Sousi, D. Yehya, and M. Joudi, "AES Encryption: Study & Evaluation".

[4] K. Muttaqin and J. Rahmadoni, "Analysis And Design of File Security System AES (Advanced Encryption Standard) Cryptography Based," *jaets*, vol. 1, no. 2, pp. 113–123, May 2020, doi: 10.37385/jaets.v1i2.78.

[5] J. P. Abasaheb and S. V. Mallapur, "Blockchain-Integrated Secure Healthcare Information Sharing via Advanced Blowfish Encryption Standard With Optimal Key Generation," *Trans Emerging Tel Tech*, vol. 36, no. 3, p. e70077, Mar. 2025, doi: 10.1002/ett.70077.

[6] G. Dhaliwal, "Comparative analysis of DES and AES implementations in CyberSecurity applications," Jan. 05, 2025, *Preprints*. doi: 10.36227/techrxiv.173611698.82877210/v1.

[7] F. L. Bauer, "Cryptanalysis," in *Encyclopedia of Cryptography, Security and Privacy*, S. Jajodia, P. Samarati, and M. Yung, Eds., Cham: Springer Nature Switzerland, 2025, pp. 468–471. doi: 10.1007/978-3-030-71522-9_164.

[8] Z. Y. Dzahabi, N. Hayaty, and M. Bettiza, "CRYPTOGRAPHY OF CHACHA20 and RSA ALGORITHMS for TEXT SECURITY," *CNAHPC*, vol. 7, no. 1, pp. 290–301, Feb. 2025, doi: 10.47709/cnahpc.v7i1.5345.

[9] S. Sharma, K. N. Patel, and A. Siddhath Jha, "Cryptography Using Blowfish Algorithm," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India: IEEE, Dec. 2021, pp. 1375–1377. doi: 10.1109/ICAC3N53548.2021.9725661.

[10] J. Ayad, N. Qaddoori, and H. Maytham, "Enhanced Audio Encryption Scheme: Integrating Blowfish, HMAC-SHA256, and MD5 for Secure Communication," *Mesopotamian Journal of CyberSecurity*, vol. 5, no. 1, pp. 178–186, Feb. 2025, doi: 10.58496/MJCS/2025/012.
[11] K. V. Saravanan and G. S. Priya, "Hybrid blowfish cryptography with elliptic curve Diffie-Hellman key exchange protocol for enhancing data security and performance," *Discov Electron*, vol. 2, no. 1, p. 29, May 2025, doi: 10.1007/s44291-025-00071-0.
[12] R. Gunawan and E. Rahmi, "Implementasi Algoritma Blowfish Untuk Pengamanan Data Transaksi Dalam Aplikasi Berbasis Website E-commerce," *tc*, vol. 24, no. 2, pp. 427–438, May 2025, doi: 10.62411/tc.v24i2.12472.
[13] A. Biryukov and C. De Cannière, "Linear Cryptanalysis for Block Ciphers," in *Encyclopedia of Cryptography, Security and Privacy*, S. Jajodia, P. Samarati, and M. Yung, Eds., Cham: Springer Nature Switzerland, 2025, pp. 1422–1426. doi: 10.1007/978-3-030-71522-9_589.
[14] B. A. Buhari *et al.*, "Performance and Security Analysis of Symmetric Data Encryption Algorithms: AES, 3DES and Blowfish," *IJANA*, vol. 16, no. 04, pp. 6473–6486, 2025, doi: 10.35444/IJANA.2025.16404.
[15] H. Alabdulrazzaq and M. N. Alenezi, "Performance Evaluation of Cryptographic Algorithms: DES, 3DES, Blowfish, Twofish, and Threefish," *Int. j. commun. netw. inf. secur.*, vol. 14, no. 1, Apr. 2022, doi: 10.17762/ijcnis.v14i1.5262.
[16] R. Musadaq, S. N. Abdulwahid, N. N. Abd Alwahed, and E. N. Abdulla, "Security analysis of an image encryption algorithm based on Blowfish in GPON," *Journal of Optical Communications*, May 2025, doi: 10.1515/joc-2025-0109.

AUTHORS

Muhammad Bagus Bintang Timur
He is currently a second-semester student in the Master of Computer Science program at Universitas Budi Luhur. His academic interests include computer vision and image processing, especially in the field of corrosion detection on metal materials. He is actively involved in research projects related to practical applications of digital image processing.

Royansyah
He is currently a second-semester student in the Master of Computer Science program at Universitas Budi Luhur. His academic interests focus on computer networking and cybersecurity. He is actively involved in research related to network infrastructure and system optimization as part of his graduate studies.

Dewi Kusumaningsih
She is a lecture of Information Technology Faculty at Budi Luhur University. The fields currently being studied are computer security, cryptography, machine learning, data science, image processing.