InfoTech Spectrum: Iraqi Journal of Data Science (IJDS)
ISSN: 3007-5467, Vol 3 (No. 2), Online First (May)- (2026)
DOI: https://doi.org/10.51173/ijds.v3i2.74
https://ijds.mtu.edu.iq

CRYPTOGRAPHIC ALGORITHM PERFORMANCE ASSESSMENT FOR AI SYSTEMS: ARIA, ZUC, AES, CAMELLIA, AND HC-256

Rebwar Khalid Mohammed [1]*, Aso M. Aladdin [2], Shilan Abdullah Hassan [3], Hemin Sardar Abdulla [2], Mohammed Abubakr Ahmed [4], Shaida Jumaah Sayda [5]

[1] *Network Department, Computer Science Institute, Sulaimani Polytechnic University, Sulaimani 46001, KR, Iraq*
[2] *Computer Science Department, College of Science, Charmo University, Sulaimani, Chamchamal 46023, KR, Iraq*
[3] *School of Computer Science and Engineering, Constructor University, Bremen, Germany*
[4] *Information Technology Department, Tishk International University, Sulaimani, Iraq*
[5] *Kirkuk Education Department of Kurdish Studies, Ministry of Education, Kirkuk, Iraq*

** Corresponding author E-mail: rebwar.khalid@spu.edu.iq (Rebwar Khalid Mohammed)*

RESEARCH ARTICLE

ARTICLE INFORMATION

SUBMISSION HISTORY:
Received: 23 February 2026
Revised: 14 May 2026
Accepted: 29 May 2026
Online First (May 2026)

KEYWORDS:
ARIA;
ZUC;
AES;
Camellia;
HC-256.

ABSTRACT

With the expansion of AI applications into new horizons, security issues have become a top priority. Traditional AI systems are not very secure and are vulnerable to attacks and data breaches. The interconnected world we live in requires data to be kept private, protected, and verified, which can be achieved through modern security measures that incorporate cryptographic algorithms. This study evaluates the efficacy of five cryptographic algorithms, ARIA, ZUC, Advanced Encryption Standard (AES), Camellia, and HC-256, regarding key generation duration, encryption and decryption speeds, and memory consumption to identify the optimal solution for practical implementations. Results show HC-256 achieving the fastest encryption and decryption times of 105.6 ms and 75.6 ms, respectively, and the lowest memory usage of 1.28 MB for encryption and 1.38 MB for decryption, proving it is the most efficient in performance-sensitive settings. AES demonstrated balanced performance with moderate encryption and decryption times (116.6 ms and 115.6 ms, respectively) and relatively low memory consumption (6.36 MB), while maintaining strong cryptographic robustness. Camellia demonstrated strong cryptographic robustness according to previous literature; however, it exhibited the lowest computational efficiency in the present experimental evaluation, with the highest encryption and decryption times among the tested algorithms, but was found to yield the lowest efficiency for real-time applications, with the greatest encryption and decryption times of 1886.8 ms and 1937.0 ms, respectively. ARIA and ZUC also exhibit different trade-offs, with ZUC showing moderate efficiency and ARIA requiring greater resources.

1. INTRODUCTION

The development of artificial intelligence (AI) has advanced significantly in recent years, yet little discussion of system security has occurred [1]. The transfer of information poses serious security issues in today's interconnected world due to advanced cyber threats [2]. In data security frameworks, cryptography is an essential component that addresses urgent security issues. Cryptography protects the confidentiality and integrity of authorized users by converting data into an incomprehensible format for unauthorized individuals [3], [4].

There are different cryptographic algorithms available, each with unique trade-offs and performance characteristics. Users must, however, select a technique that satisfies their unique needs and offers robust security. Cryptography, which includes techniques such as encryption and

DOI: https://doi.org/10.51173/ijds.v3i2.74

decryption, is widely used across many industries. Its uses ensure secure data transfer across a variety of fields, including digital television, video mail, military communications, and visual telephones [5], [6]. Users should employ cryptographic techniques to establish access controls and encrypt their data before transferring it to cloud storage to protect sensitive information. Data can be secured using a variety of methods, with cryptographic algorithms being especially useful [7]. There are two types of cryptographic systems: symmetric and asymmetric. Asymmetric encryption offers greater security and flexibility by using a key pair, one for encryption and another for decryption. In contrast, symmetric encryption uses the same key for both encryption and decryption. Because it prevents unwanted access, data encryption is essential for protecting sensitive information. An essential defense mechanism in cybersecurity techniques is encryption, which makes intercepted data as hard as possible to decrypt [8], [9].

This research aims to comprehensively evaluate five cryptographic algorithms: ARIA, ZUC, AES, Camellia, and HC-256 to give insight into their efficiency and usability in a broad range of scenarios. It is intended to direct the selection of algorithms to meet specific needs based on factors such as key generation time, encryption/decryption speed, and memory consumption. The intention is to use these methods to provide useful insights into how cryptography applications can be improved through better decision-making. Today, AI systems are used to analyze vast volumes of multimodal data, including medical images, audio streams, video content, electronic documents, and data stored in the cloud. These are the use cases where the performance of the cryptography directly affects real-time AI applications, such as edge computing, IoT-enabled AI platforms, autonomous systems, or healthcare AI applications. Therefore, the evaluation of encryption algorithms using diverse files and computation performance metrics provides helpful insights into the algorithms' applicability to AI systems and data-heavy settings.

There are two types of encryption: block and stream. Block ciphers can be described as operating on different blocks of fixed length of the input message independently of each other. The encrypted blocks are combined to make the total ciphertext. Unlike block ciphers, stream ciphers encrypt the whole message once and process the input message as a stream of bits [10]. This study based on investigation of key generation time, encryption and decryption time and memory usage gives a comprehensive performance improvement analysis of five cryptographic algorithms: ARIA, ZUC, AES, Camellia and HC-256. The main contributions of this work are:

- According to the study, HC-256 is the best option for environments with limited resources because it provides the quickest encryption and decryption times while using the least amount of memory.
- HC-256 is the most resource-efficient algorithm (for systems with low memory and CPU resources). At the same time, AES is the most balanced algorithm (for applications that require strong security, memory, and CPU efficiency); it is a good general-purpose choice.
- The results show that while ARIA and ZUC exhibit different efficiency levels appropriate for particular use cases, Camellia, despite its robust security, has high computational costs.
- Researchers and practitioners can use the recommendation framework to select the most appropriate cryptographic algorithm for their applications, taking into account factors such as speed, memory usage, and security requirements.

This research contributes to the field of cryptography and decision making for cryptographic algorithm selection, with the aim of optimizing security, performance, and resource usage in various computing contexts. The present study is a multi-dimensional evaluation of the performance of cryptographic algorithms for the generation of keys, the encryption and decryption time of the keys, and the memory consumption of the algorithms, which is not done in previous comparative cryptographic studies: those studies are mainly based on individual cryptographic benchmarking or single category algorithms in different experimental conditions. Moreover, the study features a hybridized block and stream cipher algorithm and tests it with diverse real-world file formats, such as JPG, MP3, MP4, PDF, and PPTX files. An application-oriented approach allows a more realistic evaluation of the cryptographic suitability of systems for AI systems, for multimedia processing, for edge computing, and for resource-constrained systems. The analyzed algorithms are from different types of cryptography, such as block ciphers including ARIA, AES, Camellia, and stream ciphers ZUC and HC-256, but the methodology was chosen to be fair in order to have a common framework for

the analysis. The study was not intended to compare the internal structures of the algorithms. Instead, it focused on evaluating their computational performance under the same experimental setup. To ensure a consistent and application-oriented evaluation in AI-driven and data-intensive systems, all algorithms were implemented in the same software environment, hardware platform, datasets, file formats, and performance measurement procedures.

The second part reviews related works, with emphasis on recent studies and recent references. The third section describes the materials and methods used in this investigation and an intensive study of the different algorithms. The results of the system are presented in section fourth and are used to assess the performance of the system. A detailed discussion of the results is provided in section five, and they are compared to the results of previous studies. The article concludes with a summary of the key conclusions and implications of the findings.

2. LITERATURE REVIEW

Data security remains a major challenge as it is important to protect sensitive information and applications, and cybersecurity threats are becoming more frequent. Due to the increase in the number of encryption algorithms, recent works are focusing on comparing the performance of the encryption algorithms in different aspects. In this work, we will stress recent studies that offer these comparative analyses.

Mohammed et al. [11] analyzed the performance of encryption and decryption of five cryptographic algorithms: AES, Blowfish, Twofish, Salsa20, and ChaCha20. They studied the best time to execute and the throughput for the experiments of image encryption. It was found that ChaCha20 had the fastest average encryption and decryption times, and Twofish had the lowest throughput among the algorithms tested. Khadji et al. [12] analyzed lightweight cryptographic algorithms for big data security, focusing on HC-128 and HC-256. The study reported that although HC-based algorithms provide strong encryption, they require relatively high memory and processing resources. The experimental findings showed that HC-128 and HC-256 were slower than lightweight alternatives such as Rabbit and ChaCha20. Therefore, the authors concluded that HC algorithms may be less suitable for real-time big data applications, where faster and more resource-efficient algorithms are preferred.

Song et al. [13] proposed an image compression encryption algorithm that integrates the ZUC stream cipher with chaotic maps and compressed sensing. The method improves image encryption security and computational efficiency while enabling simultaneous data compression. Their simulation results showed strong robustness, fast processing performance, and high reconstruction quality. However, the study focused mainly on image encryption. Lee et al. [14] revisited the quantum rebound attack on the double-block-length hash functions of AI-Hirose, which is based on round-reduced AES-256 and ARIA-256. They were able to find the weaknesses in the existing research, which were based on wrong differential equations of the S-boxes, and recalculated the real attack complexities. The authors suggested two novel quantum rebound attacks: quantum state preparation and nested quantum amplitude amplification, to decrease the computational complexity and boost the efficiency of the attacks. Their study, however, was rather theoretical quantum cryptanalysis and did not actually compare and benchmark the performance of cryptographic algorithms using a common evaluation framework.

Premalatha et al.[15]analyzed the efficiency of the implemented cryptographic algorithms in blockchains. They analyzed multiple performance metrics, including speed, security, resource consumption, and scalability. It also reflected cryptographic operations related to blockchain, like key generation, digital signatures, and encryption. The obtained results offer helpful information to choose appropriate cryptographic algorithms based on the application needs. The study, however, was largely centered on blockchain environments, and it failed to offer a general performance comparison of ARIA, ZUC, AES, Camellia, and HC-256. Liu et al. [16] proposed advanced cryptanalysis techniques for the ZUC-256 stream cipher. In their study, they focused on the initialization phase. They were able to distinguish on 31 rounds and to partially recover the key under weak-key assumptions using modular, signed, and XOR differences. The work primarily explored theoretical aspects of cryptanalysis and did not compare ZUC-256 to other algorithms

under a common benchmarking framework from a practical standpoint. Saydahd et al. [17]evaluated four hybrid encryption schemes (RSA–AES, RSA–ChaCha20, ECC–AES, and ECC–ChaCha20) for secure data transmission in cloud, e-application, and IoT environments. Their Java-based experiments assessed key generation time, encryption/decryption speed, and memory usage across various file types. Results indicate that ECC-based hybrid algorithms outperform RSA-based methods in terms of computational speed and resource consumption, with ECC–ChaCha20 achieving the fastest processing time and the least memory use. Swathimuttu et al. [18] A comparison of the security and performance of AES, 3DES, Blowfish, RSA, and ECC, with particular focus on encryption rate, key size, and suitability for applications. AES was very efficient for bulk data encryption, and Blowfish performed well for smaller-scale applications. The research also highlighted the importance of RSA and ECC for secure key exchange and cryptographic key management. Table 1 shows differences between selected symmetric and asymmetric cryptography algorithms used in previous works.

Table 1. Differences in selected symmetric and asymmetric cryptography algorithms used in previous works

Description	Cipher Algorithm Name	Varieties of Data Used	Year	Ref.
The study compares AES, Blowfish, Twofish, Salsa20, and ChaCha20 algorithms for image encryption, emphasizing ChaCha20's superior speed and throughput.	AES, Blowfish, Twofish, Salsa20, and ChaCha20	Image	2024	[11]
AES and Chacha20 ensure strong data confidentiality, with Chacha20 excelling in speed for small files. HC-128 and HC-256 have higher overhead due to large secret tables and are thus not very suitable for big data.	AES and Chacha20, HC128, HC-256	Big Data	2023	[12]
A Chaos and ZUC Stream Cipher-Based Image Compression Encryption Algorithm	Chaos, ZUC	Image	2022	[13]
Quantum rebound attacks and cryptanalysis of round-reduced hash constructions	AES-256 and ARIA-256	Hash-function structures	2024	[14]
Performance evaluation of cryptographic algorithms for blockchain technologies, focusing on efficiency, security strength, hardware overhead, scalability, key generation, digital signatures, and encryption operations	Blockchain cryptographic algorithms	Data	2024	[15]
The study introduces advanced cryptanalysis techniques for ZUC-256, achieving distinguishing attacks on 31 rounds and partial key recovery in weak-key settings. It leverages modular, signed, and XOR differences to effectively analyze the cipher's initialization phase.	ZUC-256	Data	2022	[16]
Comparative performance evaluation of hybrid encryption techniques for secure data transmission	RSA–AES, RSA–ChaCha20, ECC–AES, ECC–ChaCha20	Various file types	2025	[17]
Comparison of performance and security of symmetric and asymmetric encryption algorithms by speed, key size, and application. AES was found to be very efficient for bulk data encryption, Blowfish was suitable for smaller-scale applications, and RSA and ECC were secure for key exchange and key management.	AES, 3DES, Blowfish, RSA, ECC	data	2025	[18]

Although some researchers have examined selected cryptographic algorithms, most have investigated block ciphers or stream ciphers separately, with limited sets of evaluation measures or specific data types and applications. A few have also looked at ARIA, ZUC, AES, Camellia, and HC-256 in one environment. Therefore, such studies need to be applied to better understand the trade-offs between security effectiveness, computation cost, and usage in modern AIs.

Although the algorithms tested are block ciphers (ARIA, AES, and Camellia) and stream ciphers (ZUC and HC-256), the benchmarking procedure used identical data sets, environments, and performance evaluation procedures. The comparison was not strictly based on computational efficiency but rather on structural equivalence.

Also, Table 2 compares five cryptographic algorithms, ARIA, ZUC, AES, Camellia, and HC-256 based on key size, block size, efficiency, security, applications, and performance. AES is widely used and very secure; HC-256 is fast but not well standardized. ZUC is designed for mobile networks, and ARIA and Camellia are strong and perform well.

Table 2: Comparison of Cryptographic Algorithms: ARIA, ZUC, AES, Camellia, and HC-256

Factor	ARIA	ZUC	AES	Camellia	HC-256
Type	Symmetric block cipher [19]	Stream cipher [16],[20]	Symmetric block cipher [11]	Symmetric block cipher [21]	Stream cipher [8]
Key Size	128, 192, 256 bits	128 bits	128, 192, 256 bits	128, 192, 256 bits	256 bits
Block Size	128 bits	N/A (stream cipher)	128 bits	128 bits	N/A (stream cipher)
Efficiency	High, optimized for hardware and software	Optimized for mobile networks	High, widely implemented in hardware	High, optimized for both hardware/software	Very high, designed for software
Security	Resistant to known attacks	Secure for LTE/5G systems	Extensively analyzed and robust	Similar security to AES	Secure but less scrutinized
Applications	E-government, financial systems in Korea	LTE/5G communication encryption	Global standard for data encryption	Embedded systems, secure communications	High-speed encryption in software
Performance	Slightly slower than AES	High throughput for communication	Excellent performance in hardware	Comparable to AES	Very fast for large data streams
Standardization	Korean standard (KS X 1213)	3GPP LTE/5G standard	NIST FIPS PUB 197	ISO/IEC 18033-3	Not standardized by major organizations
Strengths	Strong SPN structure	Lightweight, low latency	Robust, efficient, well-supported	High flexibility and security	High speed, simple design
Weaknesses	Limited global adoption	Limited to telecom use cases	Target of frequent cryptanalysis	Less popular than AES	Less analyzed than AES

Recent studies comparing cryptographic performance have largely focused on individual encryption algorithms or comparisons within the same cryptographic category, in particular, block ciphers. Although several studies have examined encryption speed, computational overhead, throughput, and security efficiency, they were limited in scale or focused on individual data sets and security-related aspects. Comparative analyses of block and stream ciphers within a unified benchmarking methodology are still rare for AI-based and heterogeneous environments. Also, there has been little attention paid to evaluating modern cryptographic algorithms using multimedia and document-based formats prevalent in the contemporary AI ecosystem. Thus, this study attempts to evaluate ARIA, AES, Camellia, ZUC, and HC-256 against a unified Java-based implementation and several real-world file formats such as JPG, MP3, MP4, PDF, and PPTX. Key generation time, decryption time, and memory use are studied to provide an understanding of whether these algorithms are suitable for AI-oriented, data-intensive, and resource-constrained computing.

3. MATERIALS AND METHODS

This research employs an analytical methodology to meet the goals of the study. It involves preparation of data, collection and analysis of data, evaluation of performance, and implementation of encryption algorithms [22], [23]. The encryption algorithms are ARIA, ZUC, AES, Camellia, and HC-256 that are implemented and simulated in a Java environment. For data files, JPG, MP3, MP4, PDF, and PPTX, different data formats were used as input for the simulations. The files were designed to represent a variety of common multimodal data types common in AI systems, such as image-based AI analysis, speech and audio processing, video analysis, digital documentation, and knowledge-based presentation. Such formats allow for more realistic simulations of AI operations rather than the standard encryption tests used for single-signal computations. Performance was evaluated on key generation time, encryption time, memory usage during encryption, decryption time, memory consumption during decryption, and data size. As shown in Fig. 1.

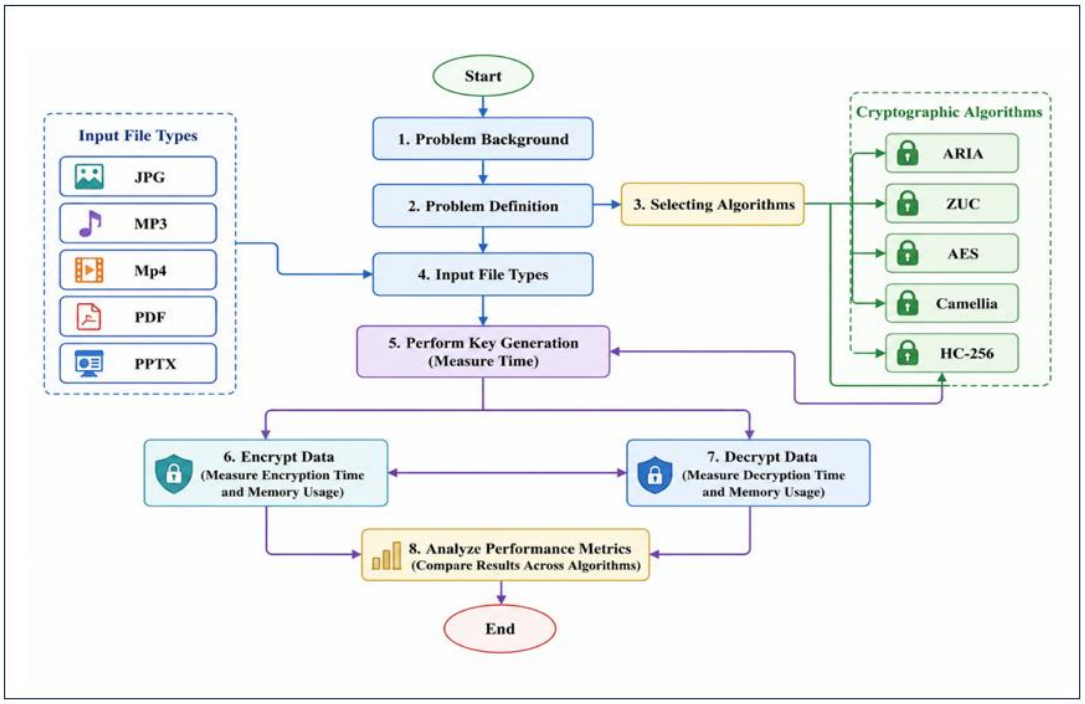


Figure 1: Performance Evaluation of Cryptographic Algorithms

The experiments were conducted on Windows 11 using Java JDK 8 in the NetBeans integrated development environment. Cryptographic operations were implemented using standard Java security APIs and the Bouncy Castle cryptographic library, particularly for algorithms not natively supported by the Java Cryptography Architecture (JCA). Random cryptographic keys and initialization vectors (IVs) were securely generated using the SecureRandom class. In the HC-256 implementation, 256-bit secret keys and 256-bit IVs were employed throughout the experimental evaluation. Performance benchmarking was performed by measuring key generation time, encryption time, decryption time, and heap memory consumption under identical experimental conditions. Execution times were recorded in milliseconds using System.currentTimeMillis() before and after each cryptographic operation. At the same time, memory utilization was monitored using Java's MemoryMXBean by measuring heap memory usage before and after execution. To ensure methodological consistency and fairness, all evaluated algorithms were tested using the same benchmarking workflow, identical datasets, and equivalent file-processing procedures. Since block ciphers and stream ciphers work differently, only the measurement of external parameters (such as computational speed) was addressed. The benchmarking process thus focused on implementation efficiency, such as the latency of processing and the use of memory, to enable a fair comparison of the suitability of algorithms for real-world AI applications, for the processing of multimedia data, and for computing devices with limited resources.

The experimental dataset included heterogeneous multimedia and document-based file formats, namely JPG, MP3, MP4, PDF, and PPTX files, selected to simulate practical workloads

commonly encountered in modern AI-oriented and data-intensive computing environments. It proved to be the most balanced, with a good compromise between speed and resource usage, and, based on this evaluation, it was considered the best overall performer across all metrics. The flowchart displays the assessment of cryptographic algorithms with respect to encryption and decryption time and memory consumption for different file types. It has key generation, data encryption/decryption, and performance analysis to compare the algorithms as depicted in Fig. 1.

3.1. ARIA

ARIA is a symmetric-key cipher developed in 2003 by the Korean government and is a standard in the Korean Cryptographic Module Validation Program (KCMVP) and the TLS/SSL protocol. It is based on the so-called Substitution-Permutation Network (SPN) structure, which offers excellent security and is resistant to linear and differential cryptanalysis. ARIA operates with 128-bit input and output blocks and supports keys of 128, 192, and 256 bits, corresponding to 12, 14, and 16 encryption rounds [19], [24].

ARIA consists of three major layers for each round of encryption:

1. **Round Key Addition Layer**: This layer performs a bitwise XOR operation between the 128-bit plaintext or intermediate state and the round-specific key.
2. **Substitution Layer**: This layer substitutes each state byte using four pre-computed S-boxes (S1, S2, and their inverses) to provide non-linearity and increase resistance against cryptographic attacks.
3. **Diffusion Layer**: A linear transformation is applied to ensure that each input bit affects the entire output, thereby imparting strong diffusion properties.

ARIA implementation is optimized for hardware and software and is versatile and efficient. For its high performance and compact design, it is widely used in various applications, such as secure communication and data encryption, and complies with international cryptography standards [25].

3.2. ZUC

ZUC-256 is an algorithm developed by China's Zu Chongzhi, a stream cipher to make 5G communications more secure. It offers 256-bit of security and is an upgraded version of the previous ZUC-128. It adds optimizations to the initialization and authentication of messages, aiming to provide higher performance and lower resource overhead. The ZUC algorithm, which is widely adopted in LTE for confidentiality and integrity in 4G communication, provides a building block for ZUC-256, a strong alternative to existing algorithms. Its architecture has added modules, XOR operations, S-box transformation, and linear transformation in different fields, which makes the cryptanalysis difficult. Although it is an improved system, recent studies have shown that it may have some weaknesses in the reduced rounds, and further research is underway to determine the strength of this system and to develop new systems that would perform better and be more secure [16], [20].

3.3. Advanced Encryption Standard

AES encryption is a symmetric block cipher that is intended to protect 128-bit data blocks. AES is the latest U.S. government encryption standard, which was adopted in 2000 following a five-year public competition by the National Institute of Standards and Technology (NIST). AES comes in three key sizes – 128 bits, 192 bits, and 256 bits. For a 128-bit key, there are 10 rounds of encryption, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key, depending upon the key size. AES is widely acknowledged as one of the strongest encryption algorithms on the market and is essential to protecting sensitive data in a wide range of applications, including financial transactions, government communications, and electronic medical records. Fig. 2 describes the encryption procedure and explains the AES method [9], [11].

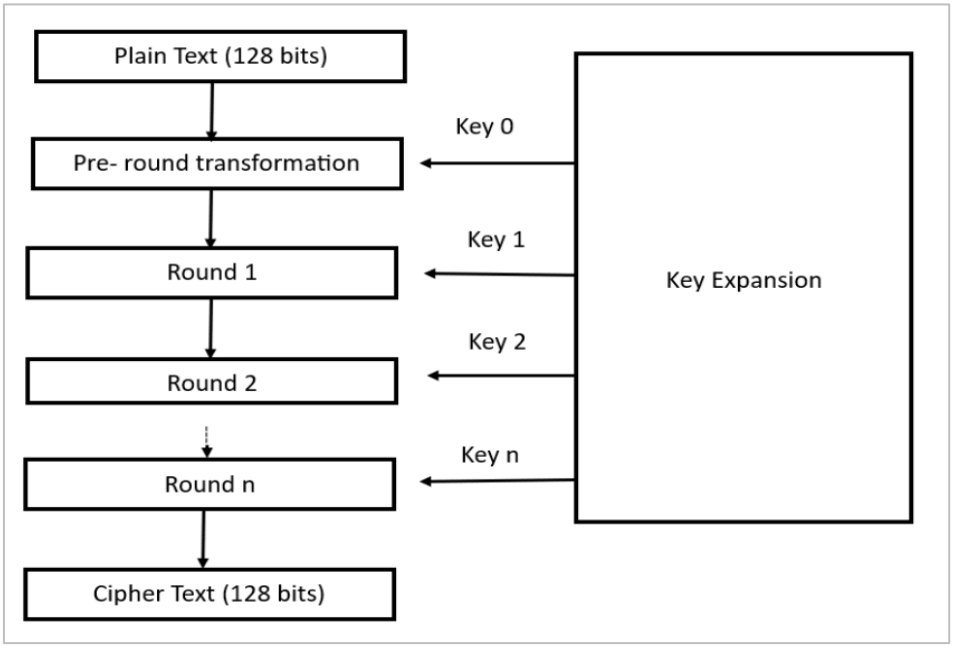


Figure 2: Structure of AES Algorithm

3.4. Camellia

Established in 2000 by Nippon Telegraph and Telephone (NTT) Corp. and Mitsubishi Electric Corporation (MEC), a secret-key, block-cipher crypto algorithm. Camellia shares a few characteristics with AES: a 128-bit block size, support for 128, 192, and 256-bit key lengths, and appropriateness for both software and hardware implementations on common 32-bit processors as well as 8-bit processors (e.g., smart cards, cryptographic hardware, and embedded systems). The same interface as the AES, Camellia defines the 128-bit block size and 128, 192, and 256-bit key sizes. Camellia is noted for its high degree of security as well as its appropriateness for both software and hardware implementations. Practically speaking, it is meant to allow flexibility in software and hardware implementations on 32-bit processors often used over the Internet and many applications, 8-bit processors used in smart cards, cryptographic hardware, embedded systems, and so on. Its main setup time is also rather good, and its key agility beats that of AES. Camellia has been studied by the wider cryptographic community through numerous programs for reviewing crypto algorithms. Camellia was chosen as a suggested cryptographic primitive by the EU NESSIE (New European Schemes for Signatures, Integrity and Encryption) and included in the list of cryptographic techniques for Japanese e-Government systems, which were selected by the Japan CRYPTREC (Cryptography Research and Evaluation Committees) [21], [26]

3.5. HC-256

HC-128 is a lightweight and efficient stream cipher for situations requiring fast, lightweight encryption in memory-constrained devices like embedded systems and Internet of Things devices. It starts with a single 1,024-word table of keystream from a 128-bit secret key and a 128-bit initialization vector (IV). The low computational burden and small size make the cipher ideal for real-time applications such as secure video streaming, messaging, and mobile communications. Strong defense against cryptographic attacks, such as differential and linear cryptanalysis, is provided by HC-128. It can be easily implemented on a variety of hardware architectures thanks to its cross-platform compatibility. HC-256, on the other hand, is a high-performance stream cipher designed for applications requiring high scalability and security. Initializes two tables: P, Q, of 1024 words to be used for the generation of the keystream, corresponding to a 256-bit secret key and 256-bit IV. To guarantee excellent randomness and attack resistance, these tables are updated rapidly. The HC-256 is designed to encrypt data at high speeds, making it suitable for various applications such as cloud security, media streaming, and secure data transfer. It is designed to be compatible with modern CPUs and platforms and features a dual-table architecture and a large state space to enhance security [8], [27].

3.6. Performance Metrics

A set of well-defined performance metrics was used for the evaluation of the selected encryption and decryption algorithms as described below:

A. Encryption Time: Encryption time refers to the length necessary to transform plaintext into ciphertext and acts as a vital measure of the algorithm's performance. It was calculated using Equation (1).

$$T = \frac{DS}{S} \quad \ldots (1)$$

Where:
T = Time (in seconds)
DS = Data Size (in MB or bytes)
S = Processing Speed (in MB/s)

B. Decryption Time: The time taken to decrypt the ciphertext to form plaintext is called the decryption time. It was calculated based on the same formula as Equation (1), assuming that the data size and speed of decryption are the same as those of the encryption process.

C. Memory Consumption During Encryption (MB): This metric measures the amount of memory the algorithm uses while encrypting data.
It is expressed as:

$$Me = \frac{DS}{Se} \quad \ldots (2)$$

Where:
Me = Memory used for encryption (in MB)

DS = Data Size

Se = Efficiency or speed of the encryption process (in MB/unit time)

D. Memory Consumption During Decryption (MB): Additionally, the same formula as Equation (2) is used to determine the amount of memory used during the decryption process, with the speed parameter representing the decryption efficiency.

E. Standard Deviation: Standard deviation (SD) was used to measure the variability and dispersion of the experimental results for each performance metric across different file types. It was calculated using Equation (3):

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad \ldots (3)$$

Where:
SD= Standard deviation
x_i= Individual observed value
$\bar{x}$= Mean value
n= Number of observations

F. Confidence Interval (CI): The 95% confidence interval was calculated to estimate the reliability and precision of the mean values obtained from the experimental measurements. It was calculated using Equation (4):

$$CI = \bar{x} \pm t\left(\frac{SD}{\sqrt{n}}\right) \quad \ldots (4)$$

Where:
CI= Confidence interval
$\bar{x}$= Mean value
t= t-score at 95% confidence level
SD= Standard deviation
n= Number of observations

4. RESULTS

Significant variations between the ARIA, ZUC, AES, Camellia, and HC-256 algorithms were found in a Java-based performance analysis carried out in NetBeans. The differences were examined based on key generation time, encryption time, memory usage during encryption, decryption time, and memory usage during decryption. The findings demonstrate variances in efficiency and resource usage across the tested encryption techniques.

Table 3. Performance Evaluation of the ARIA Algorithm for Various File Types

Types	Size of File (KB)	Key Generation Time (ms)	Encryption Time (ms)	Memory Used for Encryption (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
JPG	9328	1345	339	30	330	30
MP3	7240	1243	309	29	254	29
MP4	4555	1237	166	28	150	29
PDF	1136	1303	59	30	41	28
PPTX	1743	1361	74	30	95	30
Average		1297.8	189.4	29.4	174.0	29.2
SD		56.8	131.6	0.89	116.4	0.84
Confidence Interval		1297.8 ± 70.5	189.4 ± 163.4	29.4 ± 1.1	174.0 ± 144.5	29.2 ± 1.0

The performance results of ARIA for various file types are shown in Table 3, including key generation time, encryption time, memory usage during encryption and decryption, and decryption time. The results showed no significant differences among the various file formats. The average key generation time was 1297.8 ms (SD = 56.8 ms), while the average encryption and decryption times were 189.4 ms (SD = 131.6 ms) and 174.0 ms (SD = 116.4 ms), respectively. File size and file type were the prime factors that affected the encryption and decryption times. Conversely, the memory used was very similar, with an average of 29.4 ± 1.1 MB for encryption and 29.2 ± 1.0 MB for decryption. The evaluated file formats performed best in the MP4 file format, with the smaller files, PDF and PPTX, exhibited lower encryption and decryption times than the larger files, all having lower encryption and decryption times than larger files. JPG files had the longest key generation and encryption times due to their large size. As a whole, the results indicate that the ARIA algorithm is consistent in terms of memory usage and is stable when operating with different multimedia and document file formats.

Table 4. Performance Evaluation of the ZUC Algorithm for Various File Types

Types	Size of File (KB)	Key Generation Time	Encryption Time	Memory Used for Encryption (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
JPG	9328	413	316	14	274	14
MP3	7240	269	269	12	209	14
MP4	4555	301	157	10	205	13
PDF	1136	280	47	6	47	9
PPTX	1743	344	127	8	79	10
Average		321	183.2	10	162.8	12
SD		60.7	111.9	3.16	95.7	2.35
Confidence Interval		321.4 ± 75.4	183.2 ± 138.9	10.0 ± 3.9	162.8 ± 118.8	12.0 ± 2.9

The performance of the ZUC algorithm was evaluated by measuring the key generation time, encryption time, decryption time, and memory consumption for various file formats, as shown in Table 4. The tests showed efficient overall performance, with a moderate difference between file types. The mean key generation time was 321.4 ms, SD = 60.7 ms, and CI = ±75.4 ms. This shows that, depending on the file size and type, the encryption and decryption times are: 183.2 ± 111.9 ms (CI: ±138.9 ms) and 162.8 ± 95.7 ms (CI: ±118.8 ms), respectively. The time to encrypt and decrypt larger multimedia files, such as JPG and MP3, was longer than for smaller files, such as PDF and

PPTX. The memory usage during encryption did not vary greatly, ranging from 10.0 ± 3.16 MB (CI: ±3.9 MB), whereas that during decryption was 12.0 ± 2.35 MB (CI: ±2.9 MB). PDF files had the lowest processing times and the least memory usage, whereas JPG files had the highest processing times due to their larger file size. Overall, the results indicate that the ZUC algorithm offers efficient cryptographic performance while maintaining relatively constant memory usage, making it appropriate for securing multimedia and document-based data. The performance analysis of the AES algorithm for different file types is summarized in Table 5, which includes the time required to generate the key, encryption time, decryption time, and the memory used for encryption and decryption for each file type. The results showed that the AES algorithm performed steadily and efficiently across different file formats. The average key generation time was 738.0 ms, with a SD of 48.3 ms and a CI of ±60.0 ms, indicating that the key generation times for the tested files were consistent. Encryption and decryption took an average of 116.6 ± 42.5 ms and 115.6 ± 53.4 ms, respectively, with confidence intervals of ±52.8 ms and ±66.3 ms, for larger files (JPG and MP3 formats) and smaller files (PDF and PPTX formats), respectively.

Memory usage for encryption and decryption remained relatively small and consistent across all file types tested. Average memory use during encryption was 6.36 ± 0.88 MB (CI: ±1.09 MB) and during decryption was 6.10 ± 0.92 MB (CI: ±1.14 MB); both had low variability in memory use. Based on the results from the evaluated file formats, it can be concluded that the JPG file had the longest encryption and decryption times due to its large file size; however, the PDF and PPTX files had the fastest processing speeds and required less memory. In conclusion, the results suggest that AES can be used as a secure cryptographic tool with good memory efficiency and performance across various file formats.

Table 5. Performance Evaluation of the AES Algorithm for Various File Types

Types	Size of File (KB)	Key Generation Time	Encryption Time	Encryption Memory Used (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
JPG	9328	679	159	7	180	6.7
MP3	7240	756	151	6.9	161	6.7
MP4	4555	804	134	6.9	114	6.6
PDF	1136	747	74	5	58	4.5
PPTX	1743	704	65	6	65	6
Average		738	116.6	6.36	115.6	6.1
SD		48.3	42.5	0.88	53.4	0.92
Confidence Interval		738.0 ± 60.0	116.6 ± 52.8	6.36 ± 1.09	115.6 ± 66.3	6.10 ± 1.14

For the performance evaluation of the Camellia algorithm, key generation time, encryption time, decryption time, and memory consumption during encryption and decryption were measured for various file types, as shown in Table 6. The results showed that the Camellia algorithm worked well across various file formats, and computation time increased significantly with file size. Results showed that the average time to generate the keys was 1911.2 ms, with SD = 1290.6 and CI = ±1602.0, indicating a wide range across the file types tested. The times for encryption and decryption were 1886.8 ± 1285.8 and 1937.0 ± 1286.2 milliseconds, with high-confidence intervals of ±1596.0 and ±1596.5 milliseconds, respectively. Larger files, especially JPG and MP3, took significantly longer to encrypt and decrypt than smaller file types such as PDF and PPTX. For instance, the 9328 KB JPG file took 3658 ms to encrypt and 3728 ms to decrypt, while the PDF file took only 513 ms and 416 ms, respectively. Memory usage was consistent at both encryption and decryption. These average memory usage figures for encryption were 10.8 ± 1.30 MB (CI: ±1.61 MB) and for decryption were 11.8 ± 1.48 MB (CI: ±1.84 MB). The memory usage for processing most file types was about 10-13 MB, except for PPTX, which had the highest memory usage during encryption, and PDF, which had the highest during decryption. The results indicate that the Camellia algorithm is a secure cryptographic algorithm across various file formats, but larger file sizes lead to higher computational costs and longer run times.

Table 6. Performance Evaluation of the Camellia Algorithm for Various File Types

Types	Size of File (KB)	Key Generation Time (ms)	Encryption Time (ms)	Memory Used for Encryption (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
JPG	9328	3680	3658	10	3728	11
MP3	7240	2788	2766	10	2901	12
MP4	4555	1773	1737	10	1953	10
PDF	1136	537	513	11	416	14
PPTX	1743	778	760	13	687	12
Average		1911.2	1886.8	10.8	1937.0	11.8
SD		1290.6	1285.8	1.30	1286.2	1.48
Confidence Interval		1911.2 ± 1602.0	1886.8 ± 1596.0	10.8 ± 1.61	1937.0 ± 1596.5	11.8 ± 1.84

The performance of the HC-256 algorithm when evaluating different file types included key generation time, encryption time, decryption time, and memory usage during encryption and decryption (as presented in Table 7). It was shown that the HC-256 algorithm maintained overall efficiency across all considered file formats. The generated keys had an average time of 295.0 ms, a SD of 20.45 ms, and a CI of ±25.38 ms, indicating low variation across the set of files tested. The average encryption time was 105.6 ± 57.69 ms (CI: ±71.62 ms), while the average decryption time was 75.6 ± 53.51 ms (CI: ±66.44 ms). The encryption and decryption times for larger file formats like JPG and MP3 were longer than for smaller formats like PDF and PPTX. For the JPG file (9328 KB), the maximum encryption and decryption times were 179 ms and 137 ms, respectively, whereas the PDF file required only 43 ms for encryption and 21 ms for decryption. For all the files tested, the memory usage during encryption and decryption was small. Average memory usage during encryption was 1.28 ± 0.48 MB (CI: ±0.59 MB), and memory usage during decryption was 1.38 ± 1.03 MB (CI: ±1.27 MB). The tested file formats were PDF, PPTX, MP4, etc., where PDF and PPTX files required the most memory during encryption, and MP4 files required the least during decryption. In conclusion, the results show that the HC-256 algorithm is fast for cryptographic processing and has low memory consumption, making it well-suited for encrypting and decrypting multimedia and document file formats.

Table 7. Performance Evaluation of the HC-256 Algorithm for Various File Types

Types	Size of File (KB)	Key Generation Time (ms)	Encryption Time (ms)	Memory Used for Encryption (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
JPG	9328	318	179	1.4	137	0.6
MP3	7240	268	142	0.6	125	1.5
MP4	4555	293	110	1.0	66	0.2
PDF	1136	312	43	1.7	21	2.8
PPTX	1743	284	54	1.7	29	1.8
Average		295.0	105.6	1.28	75.6	1.38
SD		20.45	57.69	0.48	53.51	1.03
Confidence Interval		295.0 ± 25.38	105.6 ± 71.62	1.28 ± 0.59	75.6 ± 66.44	1.38 ± 1.27

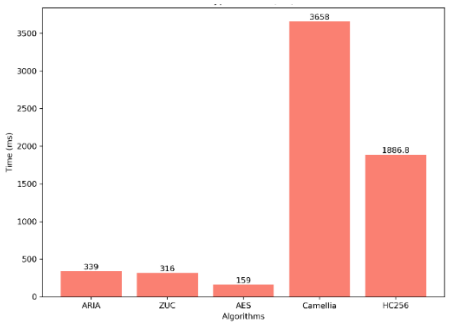
The comparative performance of the different cryptographic techniques was evaluated in terms of efficiency, speed, and resource consumption. The results highlight the importance of selecting an appropriate cryptographic algorithm based on application requirements, security needs, and performance considerations. As shown in Table 8, HC-256 demonstrated the best overall performance, ranking first across all evaluated metrics. It required the least memory and achieved the fastest encryption and decryption times (105.6 ms and 75.6 ms, respectively). AES also performed well and ranked second in most categories because of its low memory consumption and balanced encryption and decryption times (116.6 ms and 115.6 ms, respectively). ZUC exhibited moderate efficiency with reasonable computational and memory requirements. In contrast, Camellia incurred the highest computational cost and recorded the longest encryption and

decryption times (1886.8 ms and 1937.0 ms, respectively), despite providing strong cryptographic security. Overall, HC-256 appears to be the most suitable choice for resource-constrained environments where memory efficiency and processing speed are critical.

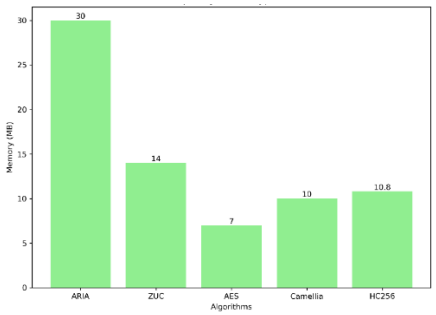
Table 8. Performance Evaluation of Cryptographic Algorithms

Algorithm	Metrics	Key Generation Time	Encryption Time	Memory Used for Encryption (MB)	Decryption Time (ms)	Memory Used for Decryption (MB)
ARIA	Average	1297.8	189.4	29.4	174	29.2
	Ranking	4	4	5	4	5
ZUC	Average	321	183.2	10	162.8	12
	Ranking	2	3	3	3	4
AES	Average	738	116.6	6.36	115.6	6.1
	Ranking	3	2	2	2	2
Camellia	Average	1911.2	1886.8	10.8	1937.0	11.8
	Ranking	5	5	4	5	3
HC-256	Average	295	105.6	1.28	75.6	1.38
	Ranking	1	1	1	1	1

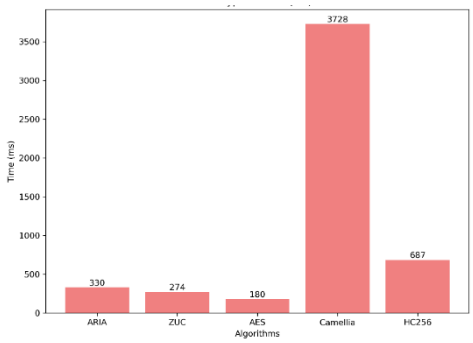
The performance of some cryptographic algorithms used in image processing is shown in Fig. 3, focusing on encryption/decryption efficiency. The encryption time (ms) is shown in Fig. 3(A), with HC-256 proving the most efficient and Camellia the slowest. The performance of AES and ZUC is relatively good when compared to ARIA and Camellia as well. Fig. 3(B) shows memory consumption during encryption, with HC-256 consuming the least (1.4 MB) and ARIA consuming the most (30 MB). As shown in Fig. 3(C), the decryption time (ms) remains the shortest for HC-256, whereas Camellia has the slowest performance. For decryption, it can be observed that HC-256 has the lowest memory consumption, whereas ARIA has the highest among the algorithms, as shown in Fig. 3(D). The findings demonstrate the trade-offs between speed and memory use of the evaluated cryptographic algorithms.



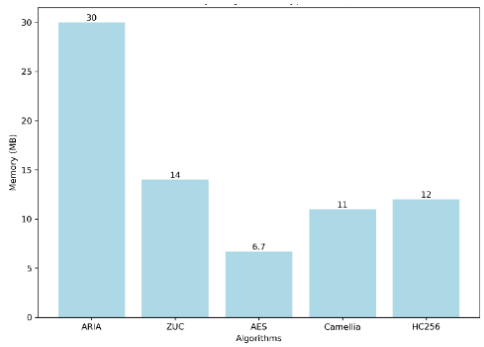
A. Encryption Time (ms)



B. Memory Usage for Encryption (MB)



C. Decryption Time (ms)

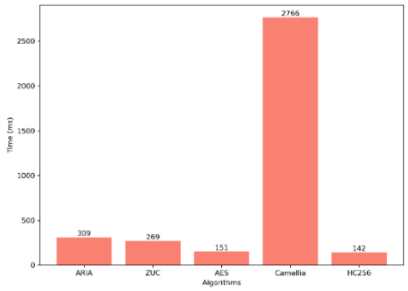


D. Memory Usage for Decryption (MB)

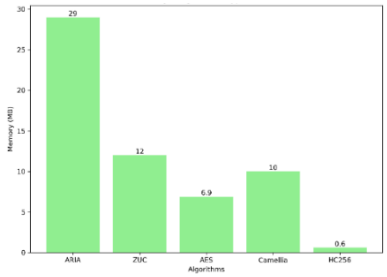
Figure 3: Performance Metrics of Cryptographic Algorithms for Image Processing

The performance of some cryptographic algorithms for MP3 processing is presented in Fig. 4, which measures memory usage and encryption and decryption time. Based on the encryption time

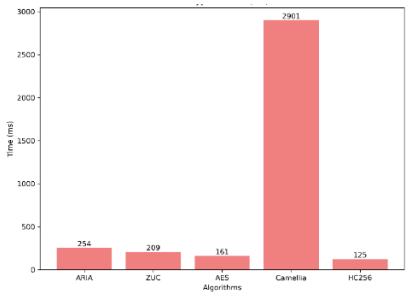
(A) and decryption time (C), Camellia has the longest processing time, while HC-256 is the fastest. Memory usage during encryption (B) and decryption (D) shows that HC-256 has the lowest memory usage and ARIA has the highest among all the ciphers. AES is a very well-balanced trade-off between memory efficiency and speed. The results emphasize the cost/benefit aspects of computational and resource requirements, as there are significant differences in algorithm performance.



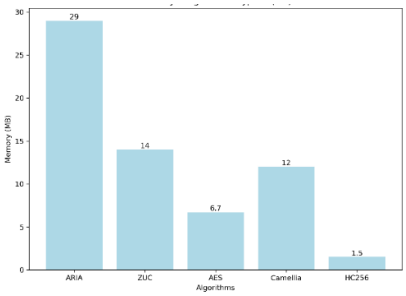
A. Encryption Time (ms)



B. Memory Usage for Encryption (MB)



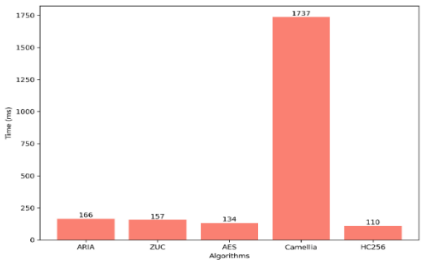
C. Decryption Time (ms)



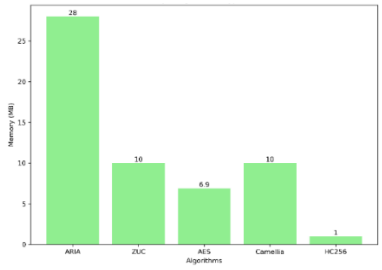
D. Memory Usage for Decryption (MB)

Figure 4: Performance Metrics of Cryptographic Algorithms for MP3 Processing

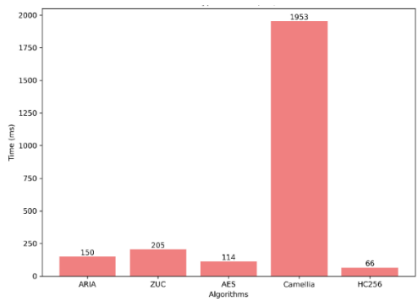
In the following, the performance of a few cryptographic algorithms for MP4 processing is depicted in Fig. 5. The encryption time (A) is longest for Camellia (1737 ms), and shortest for HC-256 (110 ms). Memory usage for encryption (B) shows that HC-256 is the most effective (1 MB), while ARIA uses the most memory (28 MB). The same holds for the decryption time (C), with HC-256 being the fastest (66 ms) and Camellia the slowest (1953 ms). The memory required for decryption (D) indicates that HC-256 is still the most efficient (0.2 MB), while ARIA is the most memory-intensive (29 MB). Overall, HC-256 is the fastest and most memory-efficient, and Camellia is the worst. The results provide valuable data for selecting cryptographic algorithms based on performance criteria.



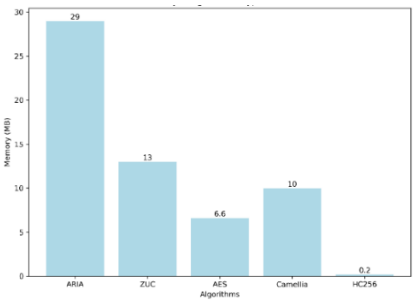
A. Encryption Time (ms)



B. Memory Usage for Encryption (MB)



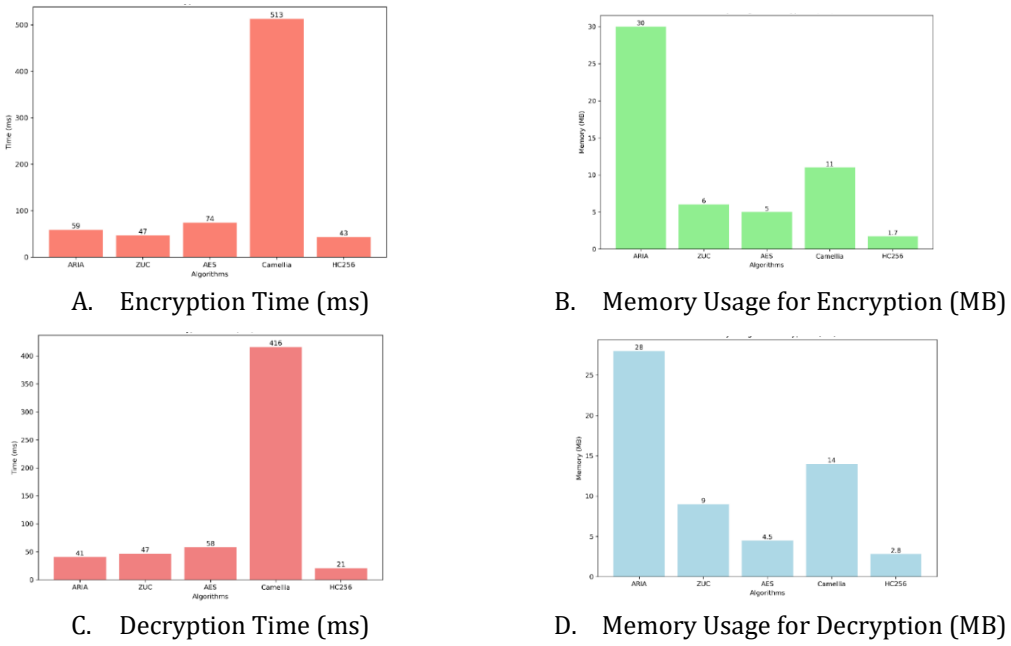
C. Decryption Time (ms)



D. Memory Usage for Decryption (MB)

Figure 5. Performance Metrics of Cryptographic Algorithms for MP4 Processing

The performance of some cryptographic algorithms applied to PDF files, such as ARIA, ZUC, AES, Camellia, and HC-256, is illustrated in Fig. 6. The time required for encryption is shown in (A): Camellia is the slowest, taking 513 ms, while HC-256 is the quickest, taking 43 ms. The memory consumption for encryption is shown in (B): ARIA requires the most memory (30 MB) while HC-256 requires the least (1.7 MB). The decryption time is displayed on (C), and the Camellia is the slowest at 416 ms, while HC-256 is the fastest (21 ms). The memory consumption during decryption is finally displayed in (D), with HC-256 consuming the least (2.8 MB) and ARIA consuming the most (28 MB). These metrics show the resource usage vs computational efficiency of the algorithms.

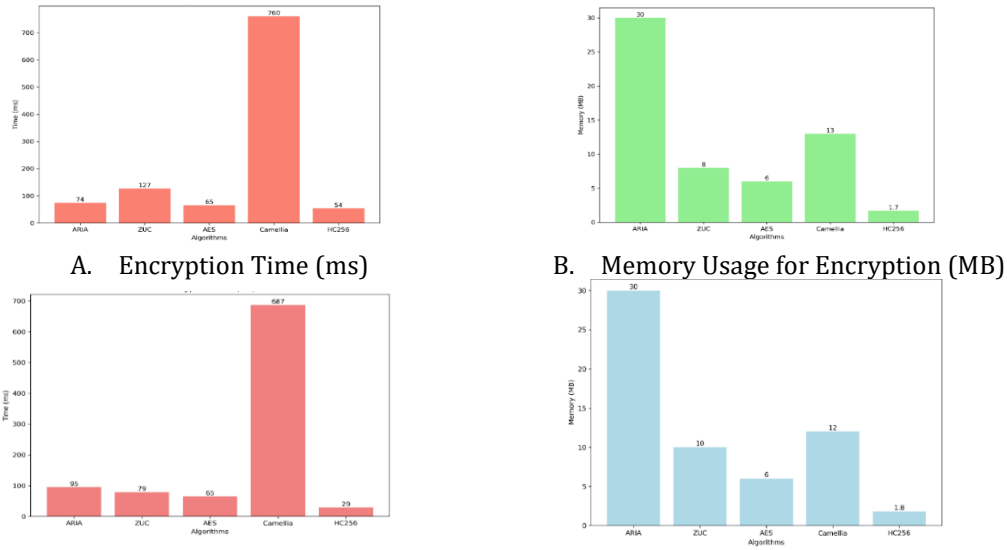


A. Encryption Time (ms) B. Memory Usage for Encryption (MB)

C. Decryption Time (ms) D. Memory Usage for Decryption (MB)

Figure 6. Performance Metrics of Cryptographic Algorithms for PDF File Processing

The encryption and decryption times, along with memory utilization, of the cryptographic algorithms used to process PPTX files are presented in Fig. 7. HC-256 achieved the fastest encryption and decryption times (54 ms and 29 ms, respectively), whereas Camellia recorded the slowest encryption and decryption times (760 ms and 687 ms, respectively). During encryption, ARIA required the highest memory consumption (30 MB), while HC-256 required the least (1.7 MB). Similarly, during decryption, HC-256 exhibited the lowest memory usage (1.8 MB), whereas ARIA consumed the most memory (30 MB). These results demonstrate significant differences in computational efficiency and resource utilization among the evaluated algorithms, indicating that algorithm selection should be guided by the specific requirements of the target application.



A. Encryption Time (ms) B. Memory Usage for Encryption (MB)

C. Decryption Time (ms) D. Memory Usage for Decryption (MB)

Figure 7: Performance Metrics of Cryptographic Algorithms for PPTX File Processing

5. DISCUSSION

The present study is a thorough and application-oriented performance analysis of the 5 most popular cryptographic algorithms, ARIA, ZUC, AES, Camellia, and HC-256, under the same application, real-world, random-access conditions. The algorithms were tested across various parameters, including the time required to generate keys, latencies for encrypting and decrypting different file types, memory consumption, etc. This multi-dimensional approach enables the evaluation of the applicability of these cryptographic algorithms to the current AI and big data landscape. When it comes to AI systems, efficient encryption is strongly correlated with responsiveness, particularly in systems that involve ongoing data sharing, real-time inference, edge computing, federated learning, and cloud-based ML services. Implementing algorithms that are less intensive in terms of latency and memory usage can also significantly reduce the computational burden required for secure data transmission and storage, which is crucial for the scalability and efficiency of today's AI systems.

In AI systems, encryption efficiency is closely related to overall system responsiveness, particularly when continuous data exchange and inference are performed in real time via edge computing, federated learning, or cloud-based machine learning services. Faster algorithms can be deployed to reduce the computational overhead of data transmission and storage, enabling modern AI infrastructure to scale and operate efficiently. As shown in Table 8, HC-256 performs best across most criteria, with the lowest encryption time (105.6 ms), decryption time (75.6 ms), and memory consumption during encryption (1.28 MB) and decryption (1.38 MB). Its superior efficiency is due to the stream cipher's structure and the optimized mechanism for generating the keystream, which has lower computational complexity than that of block cipher structures. Typically, stream ciphers operate on data one at a time and do not require repeated block transformations, unlike some implementations of block ciphers, potentially leading to reduced latency and memory usage in real-time applications. The results are similar to those of other studies that highlight the high throughput of stream ciphers; however, they are the opposite of what is reported in other studies that indicate that HC-based algorithms would have a higher computational overhead in large-scale systems [12]. This difference highlights that the algorithm's performance is very sensitive to implementation context, system configuration, and the nature of the dataset.

Although AES is not as efficient as HC-256, it demonstrates the most balanced and consistent performance profile. The results in Table 8 and Figures 3-7 indicate that AES can achieve moderate encryption and decryption speeds (116.6ms and 115.6ms) and relatively low memory consumption (~6 MB). This sets a widely accepted standard that strikes a good balance between performance, reliability, and security. Further, AES is available with hardware acceleration (e.g., AES-NI) and is considered a global standard; it was not modeled in this study, but would likely further improve performance in real-world deployments. ZUC demonstrates intermediate performance, particularly in continuous data-stream scenarios. It has a relatively fast key generation and moderate resource consumption, which reflects its optimization for LTE/5G communication systems. However, its limited adoption for general-purpose use and emerging cryptanalytic concerns [16] may limit its scope outside specialized communication settings. ARIA and Camellia, on the other hand, have much higher computation overheads. ARIA has a good cryptographic design, but it is not the best choice if you are resource-limited because it uses a lot of memory (~29 MB) and is slower. Among all algorithms analyzed, Camellia has the highest encryption and decryption times (1886.8 ms and 1937.0 ms, respectively). Camellia is known for its high security and flexibility. These results demonstrate an important trade-off: strong theoretical security does not always translate to practical efficiency, especially for latency-sensitive applications such as AI systems, edge computing, and real-time analytics.

From an application perspective, the findings emphasize that cryptographic algorithm selection should be driven by system-specific requirements rather than theoretical strength alone. HC-256 is highly suitable for real-time AI systems, IoT devices, and edge computing due to its low latency and minimal memory footprint. AES remains the preferred choice for enterprise systems, cloud computing, and general-purpose applications due to its balanced performance, standardization, and widespread support. ZUC is most appropriate for telecommunications and

mobile network encryption, while ARIA and Camellia may be reserved for high-security scenarios where computational cost is less critical.

Another important observation is the influence of file type and data size on performance variability. While overall trends remain consistent, variations across JPG, MP3, MP4, PDF, and PPTX files suggest that input characteristics and I/O behavior can affect encryption efficiency. This factor is often overlooked in comparative studies but is particularly relevant for AI systems handling multimodal data. Despite its contributions, this study has several limitations. The experiments were conducted in a controlled Java environment, which may not fully reflect optimized implementations in lower-level languages or on hardware-accelerated platforms. Additionally, the study focuses primarily on performance metrics and does not include a comprehensive evaluation of cryptographic strength, resistance to modern attack vectors, or post-quantum security considerations. Although HC-256 demonstrates superior efficiency, it lacks the extensive standardization and cryptanalytic validation of AES, which may affect its adoption in security-critical systems.

Future research should address these limitations by incorporating security-focused metrics such as entropy, avalanche effect, and resistance to differential and linear cryptanalysis. Evaluating hybrid cryptographic approaches that combine the advantages of stream and block ciphers may further enhance performance and security. Moreover, extending the analysis to distributed systems, real-time AI pipelines, and hardware-accelerated environments would improve the generalizability of the findings. Finally, assessing algorithm resilience against emerging threats, including quantum computing, is essential for ensuring long-term security. Therefore, the present study provides not only quantitative benchmarking results but also a practical decision-support perspective for selecting cryptographic algorithms based on computational constraints, data characteristics, and real-world AI application requirements. In conclusion, HC-256 emerges as the most efficient algorithm under the evaluated conditions, while AES remains the most practical and widely applicable solution. This study contributes a decision-oriented framework that links performance metrics to real-world deployment scenarios, thereby supporting informed selection of cryptographic algorithms in modern AI and data-driven systems.

6. CONCLUSION

The studied algorithms differ in terms of key generation, decryption time, and memory requirements. HC-256 is the fastest algorithm, achieving encryption and decryption times of 105.6 ms and 75.6 ms, respectively, while requiring the lowest memory consumption (1.28 MB during encryption and 1.38 MB during decryption). AES demonstrated balanced performance, with encryption and decryption times of 116.6 ms and 115.6 ms, respectively, while maintaining a moderate memory footprint of 6.36 MB during encryption and 6.1 MB during decryption, making it a reliable option for a broad range of applications, making it a reliable option for a broad range of applications. Because of its memory and computational requirements, ARIA and Camellia are not as well-suited for low-resource environments as ZUC. Depending on the application, the algorithm should be selected based on its requirements: If efficiency and resource utilization are both most important for a good solution, the best choices are HC-256 and AES. For speed and low resource usage, HC-256 and AES would be best. Those results might vary across different architectures, operating systems, and processing environments; this study was conducted on a single hardware configuration.

Also, no tests were conducted to verify the strength of each algorithm against new adversaries, such as quantum computing attacks, although the researchers focused on key generation time, encryption and decryption speeds, and memory usage. Finally, the performance of the optimization algorithms was not evaluated, nor did it provide information on their weaknesses or on which ones would be safest [28], [29]. The study did not use any large-scale, network-based, or real-time applications, which could affect the actual performance. The results show that the choice of cryptographic algorithms can have a significant impact on the efficiency, agility, and responsiveness of modern AI-driven systems, especially for multimodal data processing and real-time secure communication. This study can be extended, and some further research directions can be suggested. To our dataset, more file types can be added, as well as real-time streaming data and large encrypted

transactions, to obtain a more detailed understanding of the algorithm's performance. Performance tests across different hardware platforms, such as cloud, embedded systems, and Internet of Things devices, can assess scalability and adaptivity [30], [31]. The analysis of hybrid cryptography, which combines the benefits of multiple algorithms, may help improve efficiency and security. Future safety can be ensured by assessing these algorithms' resistance to threats posed by quantum computing and by investigating alternative post-quantum cryptography techniques. Real-time encryption/decryption scenarios can be implemented in network security applications to measure the actual execution time and resource consumption.

CONFLICT OF INTEREST

The authors declare that there is *no* conflict *of interest* regarding the publication of this paper.

REFERENCES

[1] H. Taherdoost, T.-V. Le, and K. Slimani, "Cryptographic Techniques in Artificial Intelligence Security: A Bibliometric Review," *Cryptography*, vol. 9, no. 1, p. 17, Mar. 2025, doi: 10.3390/cryptography9010017.

[2] I. K. Ogundoyin, "Comparative Analysis and Performance Evaluation of Cryptographic Algorithms," *UNIOSUN Journal of Engineering and Environmental Sciences*, vol. 4, no. 1, Mar. 2022, doi: 10.36108/ujees/2202.40.0140.

[3] M. P. R, P. Dharshini, and S. P. K, "EnConvo: Secure End-to-End Encrypted Messaging Application," in *2025 International Conference on Electronics and Renewable Systems (ICEARS)*, IEEE, Feb. 2025, pp. 995–1002. doi: 10.1109/ICEARS64219.2025.10940216.

[4] B. E. H. H. Hamouda, "Comparative Study of Different Cryptographic Algorithms," *Journal of Information Security*, vol. 11, no. 03, pp. 138–148, 2020, doi: 10.4236/jis.2020.113009.

[5] I. Afrianto, A. Heryandi, and A. Finandhita, "E-document autentification with digital signature model for smart city in Indonesia," *Journal of Engineering Science and Technology*, vol. 14, 2020.

[6] H. S. Abdulla and A. M. Aladdin, "Enhancing Design and Authentication Performance Model: A Multilevel Secure Database Management System," *Future Internet*, vol. 17, no. 2, p. 74, Feb. 2025, doi: 10.3390/fi17020074.

[7] 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," *Kurdistan Journal of Applied Research*, vol. 10, no. 1, pp. 66–82, Mar. 2025, doi: 10.24017/science.2025.1.5.

[8] R. K. Muhammed, K. H. Ali Faraj, J. F. G. Mohammed, T. N. Ahmad Al Attar, S. J. Saydah, and D. A. Rashid, "Automated Performance analysis E-services by AES-Based Hybrid Cryptosystems with RSA, ElGamal, and ECC," *Advances in Science, Technology and Engineering Systems Journal*, vol. 9, no. 3, pp. 84–91, Jul. 2024, doi: 10.25046/aj090308.

[9] S. Fatima, T. Rehman, M. Fatima, S. Khan, and M. A. Ali, "Comparative Analysis of AES and RSA Algorithms for Data Security in Cloud Computing," in *The 7th International Electrical Engineering Conference*, Basel Switzerland: MDPI, Jul. 2022, p. 14. doi: 10.3390/engproc2022020014.

[10] S. D. Jonathan, J. S. Paul, N. C. Gowda, B. J. Ambika, and K. K. PN, "Comparative Analysis of Cryptographic Algorithms," *International Journal of Human Computations & Intelligence*, vol. 2, no. 5, pp. 212–219, 2023.

[11] R. K. Muhammed *et al.*, "Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption," *Kurdistan Journal of Applied Research*, vol. 9, no. 1, pp. 52–65, May 2024, doi: 10.24017/science.2024.1.5.

[12] M. Khadji, S. Khoulji, and M. L. Kerkeb, "EFFICIENT BIG DATA SECURITY: EVALUATING THE PERFORMANCE OF A PROPOSED HYBRID KEY MANAGEMENT ALGORITHM USING LIGHTWEIGHT CRYPTOGRAPHY," *J. Theor. Appl. Inf. Technol.*, vol. 101, no. 13, 2023.

[13] X. Song, M. Shi, Y. Zhou, and E. Wang, "An Image Compression Encryption Algorithm Based on Chaos and ZUC Stream Cipher," *Entropy*, vol. 24, no. 5, p. 742, May 2022, doi: 10.3390/e24050742.

[14] D. Lee and S. Hong, "Improved Quantum Rebound Attacks on Double Block Length Hashing with Round-Reduced AES-256 and ARIA-256," *IACR Transactions on Symmetric Cryptology*, vol. 2024, no. 3, pp. 238–265, Sep. 2024, doi: 10.46586/tosc.v2024.i3.238-265.

[15] K. P. Premalatha, K. K. Sahu, and M. M. Gour, "Examining the Efficiency of Implemented Cryptographic Algorithms for Blockchain Technologies," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jun. 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10725567.

[16] F. Liu, W. Meier, S. Sarkar, G. Wang, R. Ito, and T. Isobe, "New Cryptanalysis of ZUC-256 Initialization Using Modular Differences," *IACR Transactions on Symmetric Cryptology*, pp. 152–190, Sep. 2022, doi: 10.46586/tosc.v2022.i3.152-190.

[17] S. J. Saydahd, R. K. Muhammed, S. A. Hassan, and A. M. Aladdin, "A Comparative Performance Evaluation of Hybrid Encryption Techniques Using ECC, RSA, AES, and ChaCha20 for Secure Data Transmission," *Iraqi Journal of Industrial Research*, vol. 12, no. 2, pp. 157–172, Dec. 2025, doi: 10.53523/ijoirVol12I2ID598.

[18] S. SR, U. N, C. R, and A. CM, "Comparison Between Encryption Algorithms: A Performance and Security Perspective," *International Journal on Science and Technology*, vol. 16, no. 3, Sep. 2025, doi: 10.71097/IJSAT.v16.i3.7986.

[19] Y. Oh, K. Jang, Y. Yang, and H. Seo, "Quantum Implementation and Analysis of ARIA," in *2024 Silicon Valley Cybersecurity Conference (SVCC)*, IEEE, Jun. 2024, pp. 1–7. doi: 10.1109/SVCC61185.2024.10637311.

[20] S. Wang, R. Zhao, Z. Yu, and L. Wang, "Optimized implementations of stream cipher ZUC-256 algorithm," in *2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, IEEE, Dec. 2022, pp. 952–956. doi: 10.1109/IAECST57965.2022.10061968.

[21] L. Liu and X. Liu, "Research on the New Camellia Variety' Four Seasons Treasure' and Its Applications," *International Journal of Agriculture and Food Sciences Research*, vol. 3, no. 3, pp. 18–22, Oct. 2025, doi: 10.62051/ijafsr.v3n3.04.

[22] D. O. Hasan and A. M. Aladdin, "Sleep-Related Consequences of the COVID-19 Pandemic: A Survey Study on Insomnia and Sleep Apnea Among Affected Individuals," *Insights in Public Health Journal*, vol. 5, no. 2, Jan. 2025, doi: 10.20884/1.iphj.2024.5.2.12972.

[23] D. O. Hasan *et al.*, "Perspectives on the Impact of E-Learning Pre- and Post-COVID-19 Pandemic—The Case of the Kurdistan Region of Iraq," *Sustainability*, vol. 15, no. 5, p. 4400, Mar. 2023, doi: 10.3390/su15054400.

[24] T.-H. Yoo, J. Kivilinna, and C.-H. Cho, "AVX-Based Acceleration of ARIA Block Cipher Algorithm," *IEEE Access*, vol. 11, pp. 77403–77415, 2023, doi: 10.1109/ACCESS.2023.3298026.

[25] S. Eum, H. Kim, H. Kwon, M. Sim, G. Song, and H. Seo, "Parallel Implementations of ARIA on ARM Processors and Graphics Processing Unit," *Applied Sciences*, vol. 12, no. 23, p. 12246, Nov. 2022, doi: 10.3390/app122312246.

[26] Y. Y. ZHU *et al.*, "COMPARATIVE TRANSCRIPTOMICS ANALYSIS OF HIGH AND LOW YIELD CAMELLIA OLEIFERA (OIL-CAMELLIA) CLONES," *Appl. Ecol. Environ. Res.*, vol. 23, no. 1, pp. 475–487, 2025, doi: 10.15666/aeer/2301_475487.

[27] E. J. Madarro-Capó, E. C. Ramos Piñón, G. Sosa-Gómez, and O. Rojas, "Practical Improvement in the Implementation of Two Avalanche Tests to Measure Statistical Independence in Stream Ciphers," *Computation*, vol. 12, no. 3, p. 60, Mar. 2024, doi: 10.3390/computation12030060.

[28] M. S. Abdulkarim, A. Ibrahim Mustafa, S. R. Mohammed-Taha, A. M. Aladdin, D. O. Hasan, and T. A. Rashid, "Multi-objective Optimization Vectors," in *Multi-objective Optimization Techniques*, Boca Raton: CRC Press, 2025, pp. 242–272. doi: 10.1201/9781003601555-13.

[29] A. I. Mustafa, A. M. Aladdin, S. R. Mohammed-Taha, D. O. Hasan, and T. A. Rashid, "Is PSO the Ultimate Algorithm or Just Hype?," Jan. 05, 2025. doi: 10.36227/techrxiv.173609965.50383071/v1.

[30] A. M. Aladdin *et al.*, "Fitness-Dependent Optimizer for IoT Healthcare Using Adapted Parameters," in *Practical Artificial Intelligence for Internet of Medical Things*, Boca Raton: CRC Press, 2023, pp. 45–61. doi: 10.1201/9781003315476-3.

[31] A. M. Aladdin and T. A. Rashid, "LEO: Lagrange elementary optimization," *Neural Comput. Appl.*, May 2025, doi: 10.1007/s00521-025-11225-2.