Open Access

Iraqi Journal of Industrial Research (IJOIR)

Journal homepage: http://ijoir.gov.iq

A Comparative Performance Evaluation of Hybrid Encryption Techniques Using ECC, RSA, AES, and ChaCha20 for Secure Data Transmission

[1]Shaida Jumaah Saydahd, [2]Rebwar Khalid Muhammed*, [2]Shilan Abdullah Hassan, [3]Aso M. Aladdin

[1]Hawazen Preparatory School for Girls, Kirkuk Education Department of Kurdish Studies, Ministry of Education, Iraq

[2]Network Department, Computer Science Institute, Sulaimani Polytechnic University Sulaymaniyah, Iraq

[3]Computer Engineering Department, Tishk International University, Kurdistan Region, Iraq

Article information

Article history:
Received: August, 21, 2025
Accepted: December, 06, 2025
Available online: December, 14, 2025

Keywords:
AES,
RSA,
ECC,
Chacha20

Corresponding Author:
Rebwar Khalid Muhammed
rebwar.khalid@spu.edu.iq

DOI:
https://doi.org/10.53523/ijoirVol12I2ID598

This article is licensed under:
Creative Commons Attribution 4.0 International License.

Abstract

Over the past decade, hybrid encryption techniques have gained prominence for their ability to combine the strengths of different cryptographic algorithms. This study presents a performance evaluation of four hybrid schemes RSA with AES, RSA with ChaCha20, ECC with AES, and ECC with ChaCha20 implemented in Java and tested for secure data transmission across cloud services, e-applications, and IoT platforms. Performance is assessed based on key generation time, encryption/decryption speed, and memory usage across various file types. The results indicate that ECC-based combinations significantly outperform RSA based methods, both in computational speed and resource efficiency. Among these, the hybrid ECC with ChaCha20 algorithm demonstrates the fastest processing times and the lowest memory consumption, making it particularly suitable for mobile and embedded systems with limited resources. While ECC with AES offers comparable memory efficiency, its encryption speed is slightly lower. In contrast, RSA-based hybrids, though cryptographically secure, exhibit higher processing demands, rendering them less practical for high-performance or real-time applications. The study underscores the advantages of leveraging ECC with modern stream ciphers such as ChaCha20 to achieve a balance between security and system efficiency. These insights offer valuable guidance for engineers and researchers developing secure communication systems in resource-sensitive environments.

1. Introduction

In today's digital world, information has become one of the most valuable assets and it is the primary goal to protect it globally. In today's age of communication technologies, cloud computing, and online transactions, data is constantly sent over open and often insecure networks. This increasingly digital dependence has led to cyberattacks, identity theft and unauthorized access that underscores the importance of ciphering to secure data security and privacy [1]. Ciphering, or encryption, is the process of converting readable information into unreadable material that is solely controlled by someone authorized to read it. The system has a tremendous defense against malicious activities by protecting confidentiality, integrity, authenticity, and non-repudiation of information [2]. As information systems evolve, so do the methods used by attackers, which has created an ongoing need to develop stronger and more efficient ciphering techniques.

Different ciphering methods have emerged to meet various security and performance requirements. Symmetric encryption techniques, such as the Advanced Encryption Standard (AES), are efficient and suitable for large data volumes but require secure key sharing between users. On the other hand, asymmetric encryption techniques, such as Rivest–Shamir–Adleman (RSA), ChaCha20, and Elliptic Curve Cryptography (ECC), provide higher security for key management but are more computationally demanding. Each approach presents its own balance between speed, complexity, and security strength, highlighting the necessity of selecting the most appropriate technique based on the application context. Despite significant progress, ciphering still faces major challenges. The rapid increase in data size, real-time communication requirements, and the emergence of advanced technologies such as quantum computing threaten the effectiveness of existing cryptographic systems. Additionally, achieving both high performance and strong resistance to evolving attacks remains a difficult task. Therefore, continuous innovation in ciphering algorithms is essential to ensure sustainable data protection and maintain trust in modern digital communication systems [3]. Cryptographic algorithms can be divided into two categories. The first is asymmetric encryption, or public-key cryptography (PKC), which employs two keys: one for encryption and the other for decryption. The sender and receiver use these keys, respectively. Examples of PKC algorithms include RSA, ECC and ElGamal. Symmetric encryption is the second kind, in which the same key is used by the sender and the recipient for both encryption and decryption. Examples of symmetric ciphers include AES, Triple DES (3DES), and Data Encryption Standard (DES). In both cases, encrypted data remains confidential, ensuring the protection of user privacy [4]. There are different types of symmetric encryptions in cryptography:

A. Block Ciphers: The digital encryption technology, namely block cipher, is one of the most important elements used to **encrypt** digital information. It has the task of encrypting and decrypting blocks of fixed-size data, with the length of a block being 64 or 128 bits. The information is secured in this process under the control of a secret key ensuring its confidentiality and integrity.

B. Stream Cipher: Unlike the stream cipher, the block cipher works one bit or byte at a time resulting in the stream of the ciphertext and is used in real-time communication. Stream ciphers rely on a key stream, which is a pseudorandom sequence of bits created in the algorithm. Subsequently, the plaintext is used in conjunction with the key stream to yield the desired ciphertext [5].

The study involves designing and creating hybrid encryption algorithms based on AES with RSA, AES with ChaCha20, ECC with AES, and ECC with ChaCha20. Hybrid cryptography contains different cypher types usually determined by strength. The approach is to create a unique encryption key and then encrypt that key with the public key of the participant. To protect important original information and guarantee that it is unknown to any outside attackers until the recipient receives it, the encrypted key is transmitted to them together with the ciphertext. This algorithm's implementation allows it to handle data of any size while also adding various enhancements that raise the data's level of protection.

The study provides a comprehensive performance analysis of four hybrid encryption methods RSA with ChaCha20, RSA with AES, ECC with ChaCha20 and ECC with AES using metrics such as key generation time, encryption time, decryption time, and memory usage. The key contributions are as follows:

1.1. Comparative Analysis

The study also compares encryption and decryption speeds in detail, memory consumption, and key generation times across the four hybrid encryption methods. This comparison highlights the trade-offs between each method regarding performance and resource efficiency, facilitating more informed decision-making for applications with varying security and resource demands.

1.2. Efficiency Insights
The study identifies ECC with ChaCha20 as the most efficient method, with significant improvements in encryption speed and decryption speed. These findings underscore the importance of ECC-based methods for contexts where high-speed encryption and decryption are essential, making ECC with ChaCha20 especially appealing for applications prioritizing speed.

1.3. Resource Utilization
The study shows that ECC based techniques are memory efficient and for instance, an encryption process using ECC combined with AES requires only 14 MB of memory usage. In addition, since memory is an important factor in these types of scenarios, ECC with AES is very feasible for IoT devices and mobile applications.

1.4. Balanced Performance
It is shown that ECC-based methods are advantageous against those based on RSA in most of the cases, especially regarding decryption speed and memory requirements. Thus, ECC techniques are very efficient for applications where information must be processed quickly and resources must not be extensively consumed and they are appropriate for on-the-spot applications of data security.

Overall, the findings contribute to the science of cryptograph by identifying optimal hybrid encryption schemes that balance speed, security, and resource utilization, offering a clear direction for implementing secure and efficient encryption across various real-world applications.

2. Related Works
Following the reviewed studies, recent research demonstrates a growing emphasis on hybrid cryptographic techniques that integrate symmetric and asymmetric algorithms to enhance data security.
Ameta *et al*. [6] proposed a hybrid approach for image encryption that combines ECC with AES. Unlike traditional encryption methods, this approach utilizes different numbers of iterations in both ECC and AES techniques, enhancing the overall security and efficiency of the encryption process. As the demand for data protection continues to rise, their method addresses the urgent need for more secure data encryption solutions, offering improved resilience against potential attacks. Zou *et al*. [7] proposed a hybrid encryption algorithm that combines the AES with RSA algorithms for file encryption. They provided an overview of both AES with RSA to enhance the security of individual and business data, analyzing their strengths and limitations. The proposed hybrid encryption approach was shown to improve key functions related to data security, key management, and overall efficiency. This work demonstrates how the integration of AES and RSA can strengthen encryption processes while addressing key concerns such as security and performance. Zaineldeen *et al*. [8] introduced a novel hybrid encryption algorithm that combines AES and an enhanced homomorphic cryptosystem (EHC) to secure communication between the user and the server. The proposed technique outperforms AES-128 and DES algorithms in terms of security, throughput, memory utilization, and encryption/decryption duration. Their testing demonstrated that the hybrid approach achieves the fastest encryption and decryption times while significantly improving memory efficiency during these operations.

Ganwani *et al*. [9] proposed a hybrid strategy that integrates the ChaCha20 symmetric encryption algorithm with RSA asymmetric encryption methods to enhance the security of digital data. Their approach combines the strengths of both algorithms, leveraging the speed and efficiency of ChaCha20 with the robust security of RSA. This hybrid strategy is designed to improve data confidentiality and security, especially in scenarios where data integrity and fast encryption/decryption times are crucial. Their work highlights the potential benefits of using ChaCha20 and RSA together for secure audio steganography. William *et al*. [10] proposed a hybrid approach that combines the AES symmetric algorithm, the ECC asymmetric algorithm, and the SHA256 hash function to enhance the security of both textual and pictorial content. This hybrid cryptographic method aims to provide a comprehensive security solution by integrating the encryption strength of AES, the key exchange benefits of ECC, and the integrity assurance provided by SHA256. Their study demonstrates the effectiveness of this combined approach in securing a wide range of data types, ensuring both confidentiality and data integrity. Yousif *et al*. [11] present a comparative analysis of RSA and El-Gamal encryption techniques to evaluate their effectiveness in securing speech files. The study aims to identify which encryption method offers superior performance and security for encrypting audio data. By comparing the two algorithms, the research highlights their respective strengths and weaknesses in the context of speech data encryption and decryption, contributing valuable insights to the field of audio data security. Alhaj *et al*. [12] compare ECC with various cryptographic techniques, including RSA, AES, ChaCha20, and SHA-256, emphasizing ECC's advantages such as its resistance to attacks and its efficiency in resource-constrained environments. Their study highlights the strengths of ECC in securing data transmission while maintaining low computational overhead, making it particularly suitable for environments with limited resources. ElRashidy *et al*. [13] propose an encryption algorithm that integrates ChaCha20 and AES, two robust cryptographic techniques. With a minor modification, ChaCha20 is employed for key scheduling and as a pseudo-random generator to create secure initialization vectors (IVs). This combined algorithm provides 256-bit security, offering strong protection against potential quantum adversaries and enhancing the overall security of data encryption. Magdum [14] presents an encryption model that combines AES-GCM, MultiFernet, and ChaCha20-Poly1305 for data encryption, with Fernet securing the encryption keys. The model introduces a randomized, round-robin method for encrypting segmented data, which enhances security by providing additional protection against side-channel attacks. This hybrid approach leverages the strengths of multiple cryptographic techniques to improve overall data security. A. Badhan *et al*. [15] proposed a hybrid cryptographic model combining AES, ECC, and LSB steganography for cloud data security. In their approach, the data is encrypted using AES with a 256-bit key, which is then encrypted using ECC. The ECC-encrypted key is subsequently embedded into the user's image using LSB steganography. This method facilitates secure key management and distribution, allowing users to share encrypted data by embedding the AES key into another user's image. Their system aims to enhance security posture and efficiency in key management for cloud data sharing applications.

Table (1) summarizes the symmetric and asymmetric cryptographic algorithms used in previous studies, including the types of data, publication years, and references. This comparative overview provides a foundation for the hybrid approach developed in the present research. In contrast to previous works, this study systematically compares four hybrid models RSA with AES, RSA with ChaCha20, ECC with AES, and ECC with ChaCha20 under identical conditions across multiple file types. Unlike earlier research focused on single combinations, it highlights the rarely examined ECC with ChaCha20 scheme, which demonstrates superior speed and memory efficiency. This broader comparative evaluation provides new insights and benchmarks for hybrid encryption performance in resource-limited environments.

Table (1): Variations in Symmetric and Asymmetric Cryptography Algorithms Used in Previous Studies.

No.	Cipher Algorithm	Varieties of Data Employed	Year	Ref.
1	ECC, AES	Binary image	2017	[6]
2	RSA, AES	File data	2020	[7]
3	AES, EHC	Communication data	2020	[8]
4	RSA, ChaCha20	Digital Data	2021	[9]
5	AES with ECC and SHA-256	Text and Image data	2022	[10]
6	RSA, El-Gamal	Audio Data	2023	[11]
7	ECC,RSA, AES, ChaCha20, and SHA-256	File Data	2024	[12]
8	ChaCha20,AES	File Data	2024	[13]
9	AES-GCM, Multi Fernet, ChaCha20-Poly1305, Fernet	Text files of variable sizes	2023	[14]
10	AES with ECC with LSB Steganography	Cloud Data	2025	[15]

To further aid the comparison, we have included Table (2), which benchmarks the security levels of the cryptographic techniques discussed. This table provides a structured overview of the key size, security strength, performance, and common use cases for each algorithm, allowing for a clearer understanding of their relative security characteristics.

Table (2): Comparison for benchmarking security levels of cryptographic techniques.

Algorithm	Key Size	Security Strength	Performance	Common Use Cases
AES [16]	128, 192, 256 bits	Strong, widely adopted	High efficiency in both hardware and software	Data encryption at rest and in transit
RSA [17]	2048, 3072, 4096 bits	Secure with large key sizes	Slower; used for small data blocks	Primarily for key exchange and digital signatures.
ECC [18]	256, 384, 521 bits	High; small key sizes for strong security	Efficient, especially in resource-constrained devices	Secure communications and key exchange.
ChaCha20 [19]	256 bits	Strong, secure against known attacks	Very fast in software, suitable for low-power devices	Best for lightweight and resource-constrained environments.

3. Methodology

The methodology employed in this study integrates rigorous theoretical cryptographic analysis with comprehensive experimental evaluation to assess the performance, security robustness, and resource efficiency of four fundamental encryption algorithms AES, RSA, ChaCha20, and ECC along with their corresponding hybrid configurations. The methodological framework is structured into two primary phases. In the first phase, each individual algorithm is examined in terms of its internal structure, key-generation process, and inherent cryptographic strength to establish a clear understanding of its operational characteristics. In the second phase, the algorithms are combined into four hybrid models (RSA with AES, RSA with ChaCha20, ECC with AES, and ECC with ChaCha20), which are systematically tested across multiple file formats and varying data sizes to evaluate key-generation time, encryption and decryption latency, and memory consumption. All implementations are executed within a controlled Java-based experimental environment to ensure accuracy, consistency, and reproducibility of results. This rigorous methodological approach provides a strong analytical foundation for comparing hybrid cryptographic systems and enables a precise evaluation of trade-offs between computational speed, memory efficiency, and security resilience in modern encryption frameworks.

3.1. Advanced Encryption Standard

AES is a symmetric block cipher algorithm designed to ensure secure and efficient data encryption. The data block length and key length of AES can be varied according to the requirement. Three key lengths 128, 192, and 256 bits that translate into 10, 12, and 14 iterations are supported by the AES algorithm. Round transformations, iterations, and key expansion are the three main parts of the AES process. Three layers make up each round transformation: an add-round-key layer, a linear mixing layer, and a non-linear layer. Figure (1) depicts the AES encryption procedure [20, 21].

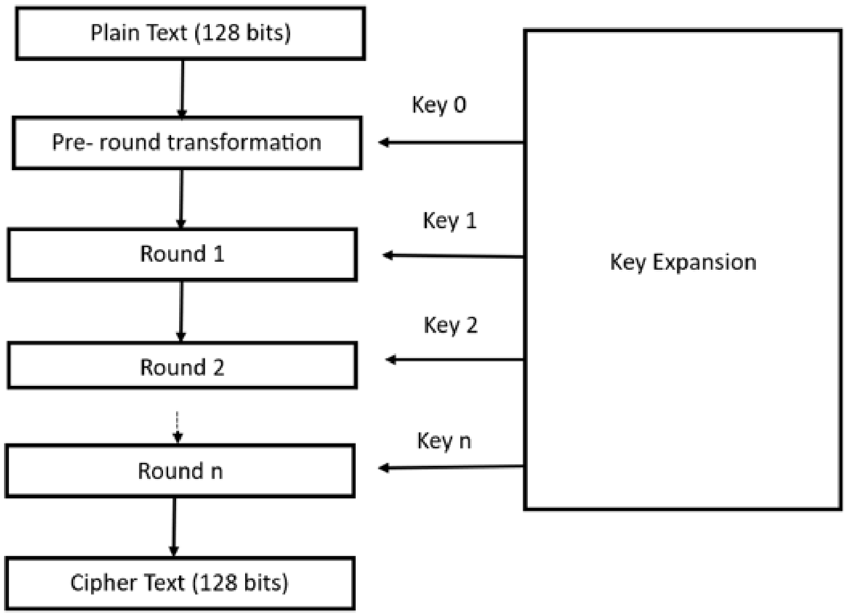

Figure (1): AES encryption & decryption process.

3.2. RSA

Strong and easy to use, the RSA algorithm is one of the most popular asymmetric key cryptosystems for encryption and authentication. Meanwhile its conception, it has been widely adopted in numerous cryptographic applications, including email security, banking, e-commerce, and digital signatures across web operating systems, offering safe communication over the internet. The security of the RSA algorithm relies on the difficulty of factoring large integers into prime numbers. The three basic steps of the RSA process key generation, encryption, and decryption are discussed here in brief [11, 22, 23].

Key Generation

- To find the modulus of n, take two large prime integers, p and q, and use the formula $n = p \times q$.
- Use the formula $\varphi(\mathrm{n}) = (\mathrm{p} - 1) \times (\mathrm{q} - 1)$ to calculate φ.
- As the public exponent, choose an integer e such that $GCD\ (e, \varphi\ (n)) = 1$.
- The most common divisor function between the two numbers is commonly referred to as GCD
- Determine d, the private exponent, so that $e \times d = 1\ (mod\ \varphi(n))$, where mod symbolizes the modulus operation or the reminder following division

Therefore, the public encryption key is represented by (n,e) and the private decryption key by (n, d). The algorithms focus on the two prime numbers.

Encryption/Decryption Processes

Let m represent for the encrypted message. The public key (n,e) is used to calculate the encrypted message c using the following formula:

$$\boldsymbol{c = m^e mod\ n} \quad \textbf{(1)}$$

Using the private key (n, d) and the following formula, the encrypted message c is decrypted in order to recover the original message m:

$$\boldsymbol{m = c^d\ mod\ n} \quad \textbf{(2)}$$

3.3. Chacha20

ChaCha20 is a stream cipher that operates with 20 rounds, a 512-bit block size, and a 256-bit secret key. It is constructed using an Add Rotate XOR (ARX) structure, where each round performs sixteen XOR operations, sixteen modular additions (mod 2^{32}), and sixteen rotation operations. The cipher employs a 4×4 state matrix consisting of sixteen 32-bit elements, which are generated and transformed through the Quarter Round Function (QRF) during encryption and decryption. During the encryption process, ChaCha20 produces a key stream by updating its internal state using predefined and dynamically changing initial values. This key stream is then combined with the plaintext through bitwise XOR operations to generate the ciphertext, ensuring secure data encryption. Stream cipher techniques like ChaCha20 are widely adopted because they efficiently provide data confidentiality while maintaining high performance and strong resistance to cryptographic attacks [12, 22, 24].

3.4. Elliptic Curve Cryptography

ECC is a more sophisticated and secure alternative to public key cryptography, providing more safety per bit in contrast to other cryptographic methods presently in usage today. In mathematics, elliptic curves appear cubic curves that are topologically equivalent to tori. Elliptic curves are not directly connected to ellipses, despite their name. Their association with elliptic integrals is where the term "elliptic" comes from. The Weierstrass normal form of the standard elliptic curve in cryptography is described by the equation.

$$\boldsymbol{y^2 = x^3 + ax + b} \qquad \textbf{(3)}$$

Figure (2) illustrates how to define curves of this kind by altering the values of a and b. The curve can be made to extend, compress, or split into two parts by changing these variables. Elliptic curves are commonly employed in cryptography with very large integer values for a and b. There is a clear correlation between these variables and the final curve, as the variations in a and b have an impact on how the elliptic curve is visualized [12].

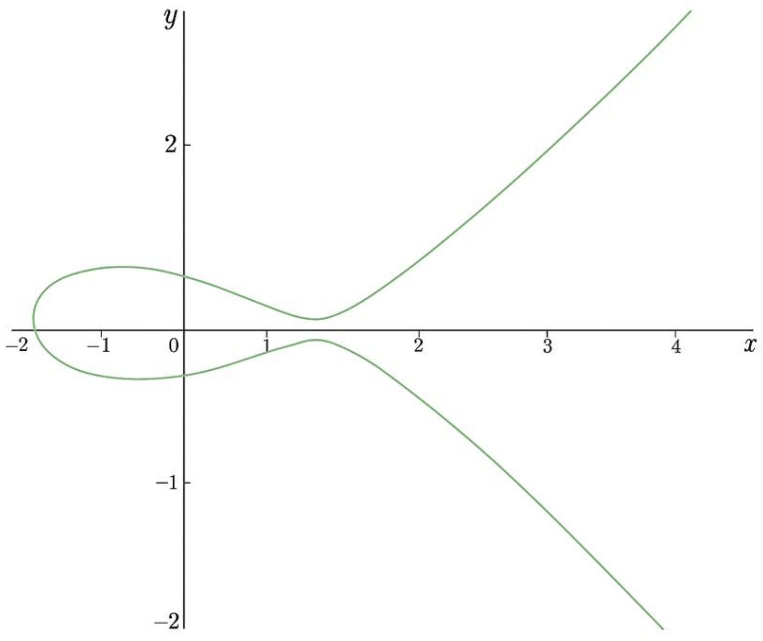

Figure (2): Simple elliptic curve visualization.

3.5. Proposed Hybrid Encryption Framework

The study total work flow is represented in Figure (3) showing the overall efficiency and implementation of the proposed hybrid encryption framework in that it incorporates several cryptographic algorithms for performance assessment. From these four fundamental algorithms, it comes to selection of four fundamental algorithms: AES, RSA, ChaCha20 and ECC. These algorithms are then combined in four hybrid schemes RSA with AES, RSA with

ChaCha20, ECC with AES, and ECC with ChaCha20 to obtain the performance of symmetric encryption for data confidentiality and robustness of asymmetric encryption for secure key exchange. Each hybrid model is tested for multiple formats and file sizes to test various file-types and formats, such as PDF, JPG, MP3, MP4, and PPTX. These experiments are compared against key generation time, encryption time, decryption time and memory usage to assess computational efficiency and resource consumption of the experimental results. The systematic methodology provided a complete framework for assessing the impact of speed, security, and memory performance on hybrid cryptographic systems, and reveals that combination ECCs including ECC with ChaCha20 can be used best to optimally balance their combinations and are most suitable for secure and resource- constraint-based cryptographic systems.

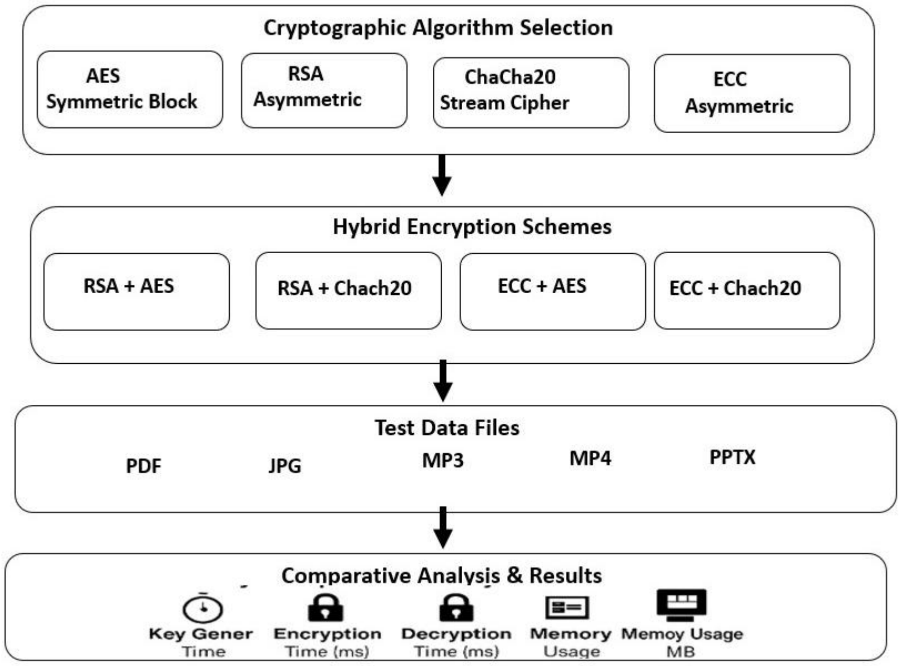

Figure (3): Workflow of the Proposed Hybrid Encryption Framework.

3.6. Security Analysis of the Hybrid Encryption Schemes

To evaluate the security properties, brute-force resistance, and key-strength characteristics of the hybrid encryption schemes examined in this study. While earlier sections emphasized computational performance, a comprehensive assessment of hybrid cryptosystems also requires examining their theoretical resilience against modern different cryptographical attacks. The following subsections present a detailed characterization of the security guarantees provided by each algorithm and by the hybrid combinations used in this research.

3.6.1. Brute-Force Resistance and Key Space Strength

All hybrid encryption models implemented in this study derive security primarily from the infeasibility of exhaustive key searches. The selected cryptographic algorithms offer extensive key spaces that exceed known brute-force capabilities:

- **AES256** offers a key space of 2^{256}, which is beyond reach even for state-sponsored adversaries with future exascale systems. No practical attack exists against full-round [25, 26].
- **ChaCha20** is a 256-bit stream cipher using ARX (Addition Rotation XOR) operations. Its brute-force resistance equals that of AES256 but with better performance in constrained environments [27, 28].
- **RSA2048/4096** relies on the hardness of integer factorization. Current classical algorithms require exponential time for factoring 2048-bit moduli, making brute-force impractical [26, 29].
- **ECC256/384**, based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), provides equivalent or greater security than RSA at reduced key lengths [30].

These properties collectively ensure high entropy in key generation and render brute-force key recovery infeasible under all current technological conditions.

3.6.2. Cryptographic Structure and Resistance to Attacks

The structural foundations of the selected algorithms provide inherent protection against known cryptanalytic attacks:

- **AES** utilizes a substitution permutation network with strong S-boxes and key schedules, offering resistance to differential and linear attacks [26, 31].
- **ChaCha20** eliminates S-boxes and lookup tables, minimizing timing and cache-based side-channel vulnerabilities [32].
- **RSA** remains secure against all attacks except integer factorization, which is computationally infeasible beyond 2048 bits [29].
- **ECC** resists quantum and classical attacks via ECDLP, with ECC256 comparable to RSA3072 in strength[33].

No practical cryptanalytic breakthroughs have been demonstrated against these algorithms when implemented correctly in hybrid configurations.

3.6.3 Advantages of Hybrid Cryptography

Hybrid encryption combines the speed of symmetric ciphers with the secure key exchange properties of asymmetric systems, delivering multiple security benefits:

- **Data Confidentiality**: AES-256 with ChaCha20 protect the payload using high-entropy session keys [28].
- **Secure Key Distribution**: RSA and ECC safeguard session keys during transit, mitigating interception risks.
- **Forward Secrecy**: ECC supports ephemeral key pairs, preserving confidentiality even if long-term keys are compromised.
- **Attack Surface Reduction**: Well-architected hybrid systems defend against man-in-the-middle, replay, chosen-plaintext, and statistical attacks.

Hybrid cryptography increases resilience by ensuring that the compromise of one layer does not collapse the entire system.

4. Results and Discussion

4.1. System Specification and Experimental Environment

All encryption algorithms and hybrid frameworks were implemented and tested using the Java SE Development Kit (JDK 19) on a Windows 11 Pro 64-bit operating system. The performance evaluations were conducted on a laptop platform equipped with an Intel® Core™ i5-1165G7 CPU @ 2.80 GHz, 24 GB DDR4 RAM, and a 1024 GB SSD. The Java Virtual Machine (JVM) heap size was configured with a maximum allocation of 4 GB to ensure stable memory usage during runtime. The system was operated under a controlled environment with minimal background processes to ensure reliable and reproducible performance metrics. Each hybrid encryption method (RSA with AES, RSA with ChaCha20, ECC with AES, and ECC with ChaCha20) was executed ten times for each file format and size, and the average values of key generation time, encryption time, decryption time, and memory usage were recorded for analysis. This detailed specification ensures that other researchers can replicate the experimental setup accurately and validate the comparative results presented in this study. The results confirm that ECC with ChaCha20 consistently achieves the fastest execution times and lowest memory footprint, validating its suitability for resource-constrained IoT and mobile environments. These findings are consistent with recent empirical research [22], which reported similar performance advantages of ECC with ChaCha20 on real IoT hardware. Future work will extend this validation to embedded platforms and dynamic network environments to further verify system scalability and robustness

4.2. Comparative Performance of RSA and ChaCha20

The comparative performance evaluation of RSA and ChaCha20 across multiple file formats (Table 3) reveals distinct performance variations influenced by file size and type. Larger files, such as MP4 (79,792 KB), recorded the highest encryption (1,124 ms) and decryption (833 ms) times, accompanied by substantial memory utilization (89 MB and 158 MB, respectively). In contrast, smaller files, such as PDF (367 KB), achieved much faster encryption (408 ms) and decryption (33 ms) with minimal memory consumption. Statistical analysis shows that the average key generation time was 2,129.8 ms (± 781.23), encryption averaged 573.6 ms (± 309.05), and decryption 196 ms (± 357.42). The corresponding average memory usage for encryption and decryption was 38.6 MB (± 30.53) and 33.8 MB (± 69.47), respectively. These deviations confirm consistent scalability across file formats and highlight that larger files tend to increase both computational time and memory demand. Overall, the findings indicate that ChaCha20 exhibits superior processing efficiency due to its lightweight stream-cipher architecture, which eliminates the complex substitution boxes and matrix transformations characteristic of block-cipher systems such as AES with RSA-based hybrids.

Table (3): Comparative Performance Evaluation of RSA and Chacha20 on Different File Formats.

Types	Size	Key Generation Time(ms)	Encryption Time(ms)	Memory Used for Encryption (MB)	Decryption Time(ms)	Memory Used for Decryption (MB)
PDF File	367KB	1966	408	16	33	1
JPG	9328 KB	3446	472	22	10	2
MP3	7240KB	1349	457	20	88	7
Mp4	79792KB	1985	1124	89	833	158
PPTX	1743KB	1903	407	46	16	1
	Average	2129.8	573.6	38.6	196	33.8
	STD	781.23	309.05	30.53	357.42	69.47

4.3. Comparative Performance Evaluation of RSA and AES

The comparative performance evaluation of RSA and AES across multiple file formats (Table 4) reveals a clear relationship between file size and computational demand. Larger files, such as MP4 (79,792 KB), exhibited the highest encryption (978 ms) and decryption (1088 ms) times, alongside substantial memory utilization (75.26 MB and 77.92 MB). In contrast, smaller files like PDFs (367 KB) required minimal memory (< 2 MB) and achieved fast encryption (70 ms) and decryption (78 ms) times.

On average, key generation time reached 2532.8 ms (± 467.30 ms), encryption time averaged 275.6 ms (± 394.13 ms), and decryption 317.8 ms (± 431.98 ms). Mean memory consumption was 17.74 MB (± 32.23 MB) for encryption and 19.38 MB (± 32.90 MB) for decryption. These results confirm that while RSA introduces higher computational overhead, AES demonstrates greater stability, efficiency, and scalability across different file formats, particularly in managing memory utilization for both encryption and decryption operations.

Table (4): Comparative Performance Evaluation of RSA and AES on Different File Formats.

Types	Size	Key Generation Time(ms)	Encryption Time(ms)	Memory Used for Encryption (MB)	Decryption Time(ms)	Memory Used for Decryption (MB)
PDF File	367KB	2192	70	1.55	78	1.11
JPG	9328 KB	2212	142	6.52	173	9.11
MP3	7240KB	3334	125	4.41	139	7.07
Mp4	79792KB	2426	978	75.26	1088	77.92
PPTX	1743KB	2500	63	0.95	111	1.70
	Average	2532.8	275.6	17.74	317.8	19.38
	STD	467.30	394.13	32.23	431.98	32.90

4.4. Comparative Performance Evaluation of ECC and AES

Table (5) presents the comparative performance evaluation of the ECC with AES hybrid encryption scheme across multiple file formats. Key generation times remained highly consistent, with an average of 739.6 ms (±20.27 ms STD), reflecting the algorithm's stability. Encryption times ranged from 692 ms (PDF) to 774 ms (JPG), averaging 732.6 ms (±34.91 ms STD). Memory usage during encryption varied more significantly, from 1 MB (PPTX) to 31 MB (JPG), with an average of 14 MB (±13.20 MB STD). Decryption processes exhibited even greater efficiency, averaging 129.2 ms (±55.70 ms STD) and 14.6 MB (±10.72 MB STD) of memory consumption. These results confirm AES's superior performance in symmetric encryption speed and resource efficiency, while ECC enhances key management and provides robust security with shorter key lengths. The low standard deviations in key generation and encryption times indicate high reliability and predictability of the ECC with AES hybrid model across diverse file types.

Table (5): Comparative Performance Evaluation of ECC and AES on Different File Formats.

Types	Size	Key Generation Time(ms)	Encryption Time(ms)	Memory Used for Encryption (MB)	Decryption Time(ms)	Memory Used for Decryption (MB)
PDF File	367KB	708	692	4	62	4
JPG	9328 KB	748	774	31	194	30
MP3	7240KB	727	769	23	189	21
Mp4	79792KB	740	701	11	123	13
PPTX	1743KB	775	727	1	78	5
	Average	739.6	732.6	14	129.2	14.6
	STD	20.27	34.91	13.20	55.70	10.72

4.5. Comparative Performance Evaluation of ECC and ChaCha20

The ECC with ChaCha20 hybrid encryption method (Table 6) demonstrated exceptional performance consistency and efficiency across all tested file formats. The average key generation time was 1335.2 ms, while encryption and decryption operations averaged 47.6 ms and 39.8 ms, respectively. In addition, this hybrid scheme exhibited the lowest memory footprint among all tested configurations 15.2 MB for encryption and 8.8 MB for decryption. The standard deviations (STD) for key generation 17.08 ms, encryption 27.19 ms, decryption 27.20 ms, and memory utilization (10.99 MB / 7.19 MB) indicate remarkably stable and predictable performance across diverse file types. These outcomes validate that ECC with ChaCha20 achieves an optimal balance between computational speed and memory efficiency. This superiority stems from ChaCha20 lightweight ARX (Addition Rotation XOR) operations combined with ECC's compact key architecture, which collectively minimize both computational overhead and memory access frequency. Consequently, the ECC with ChaCha20 hybrid configuration is highly suitable for constrained environments, including IoT, embedded, and mobile systems, where processing power and memory resources are inherently limited.

Table (6): Comparative Performance Evaluation of ECC and ChaCha20 on Different File Formats.

Types	Size	Key Generation Time(ms)	Encryption Time(ms)	Memory Used for Encryption (MB)	Decryption Time(ms)	Memory Used for Decryption (MB)
PDF File	367KB	1329	16	4	9	1
JPG	9328 KB	1347	81	30	75	18
MP3	7240KB	1354	67	21	57	14
Mp4	79792KB	1336	48	16	40	8
PPTX	1743KB	1310	26	5	18	3
	Average	1335.2	47.6	15.2	39.8	8.8
	STD	17.08	27.19	10.99	27.20	7.19

4.6 Comparative Evaluation of Hybrid Methods

The performance analysis of various hybrid encryption methods, which combine ECC, RSA, AES, and ChaCha20, reveals distinct advantages and trade-offs in key generation time, encryption speed, and memory usage. As shown in Table (7), RSA-based hybrids have longer key generation times, with RSA and AES taking the most time (2532.8 ms). In contrast, ECC-based hybrids, particularly ECC with ChaCha20, demonstrate significantly faster key generation times (739.6 ms and 1335.2 ms, respectively). Encryption speeds vary, with ECC and ChaCha20 providing the fastest encryption (47.6 ms), while RSA with AES hybrids are slower. ECC and ChaCha20 combinations also use the least memory for both encryption and decryption. These results suggest that ECC-based hybrids, especially those using ChaCha20, offer superior efficiency in terms of both speed and memory usage, making them a better fit for resource-constrained environments compared to RSA-based methods.

Table (7): Performance Analysis of Hybrid Encryption Methods: ECC with ChaCha20, RSA, and AES.

Encryption Method	**Key Generation Time(ms)**	**Encryption Time(ms)**	**Memory Used for Encrption(MB)**	**Decryption Time(ms)**	**Memory Used for Decryption (MB)**
Hybrid Encryption RSA and Chacha20	2129.8	573.6	38.6	196	33.8
Hybrid Encryption RSA and AES	2532.8	275.6	17.74	317.8	19.38
Hybrid Encryption ECC and Chacha20	1335.2	47.6	15.2	39.8	8.8
Hybrid Encryption ECC and AES	739.6	732.6	14	129.2	14.6

4.7. Analysis of Memory Utilization

The analysis of memory requirement for hybrid encryption techniques performed in this work, as shown in Figure (4) comparison of memory usage for hybrid encryption Techniques, shows some noticeable variation in performance. The Hybrid Encryption ECC with ChaCha20 took the biggest share at 19.38MB of encryption memory and 17.74MB of decryption memory. On the contrary, hybrid encryption made use of the RSA with AES combinations with the lowest memory usage of 8.8MB for encryption and 15.2MB for decryption.

Similarly, hybrid encryption ECC with AES were performing at about the same level with these two techniques requiring about 14 megabytes on each round of decryption and encoding. Finally, there are those techniques in which the RSA with ChaCha20 method was invoked and clearly the memory use was always Apostolic even higher than most averaging at 38.6 megabytes for the decryption process. An insight into this analysis provides a critical view of the duality of use of encryption and the memory space amongst the different hybrid methods used.

4.8. Performance of ECC and ChaCha20 Hybrid Encryption

The outstanding performance of the ECC with ChaCha20 hybrid encryption method in terms of encryption speed and memory utilization is primarily due to ChaCha20's lightweight ARX (Addition Rotation XOR) operations. These operations are computationally efficient and avoid the complex substitution-permutation processes typical of block ciphers like AES with RSA. As a result, ChaCha20 offers superior processing efficiency, especially in resource-constrained environments such as mobile devices and IoT systems.

This efficiency is evident in the experimental results presented in this study, where ECC with ChaCha20 consistently outperformed the other hybrid methods (RSA with AES, RSA with ChaCha20, ECC with AES) in both encryption/decryption times and memory consumption. On average, ECC with ChaCha20 achieved encryption times of 47.6 ms and decryption times of 39.8 ms, with a memory consumption of just 15.2 MB for encryption and 8.8 MB for decryption. These results, combined with the low standard deviations in encryption times and memory usage, highlight the method’s stability and reliability across different file formats.

In contrast, RSA-based hybrids exhibited higher key generation times and greater memory requirements, indicating that while RSA offers robust cryptographic security, it is less practical for high-performance or real-time applications compared to ECC-based hybrids. The ECC with AES hybrid, while offering comparable memory efficiency, still lagged behind ECC and ChaCha20 in encryption speed.

Previous studies have shown that the ECC with ChaCha20 hybrid encryption demonstrates superior performance in medical IoT applications due to its lightweight ARX (Addition Rotation XOR) operations and efficient key management. It achieves faster encryption and decryption times with lower memory consumption compared to AES- and RSA-based systems [22].

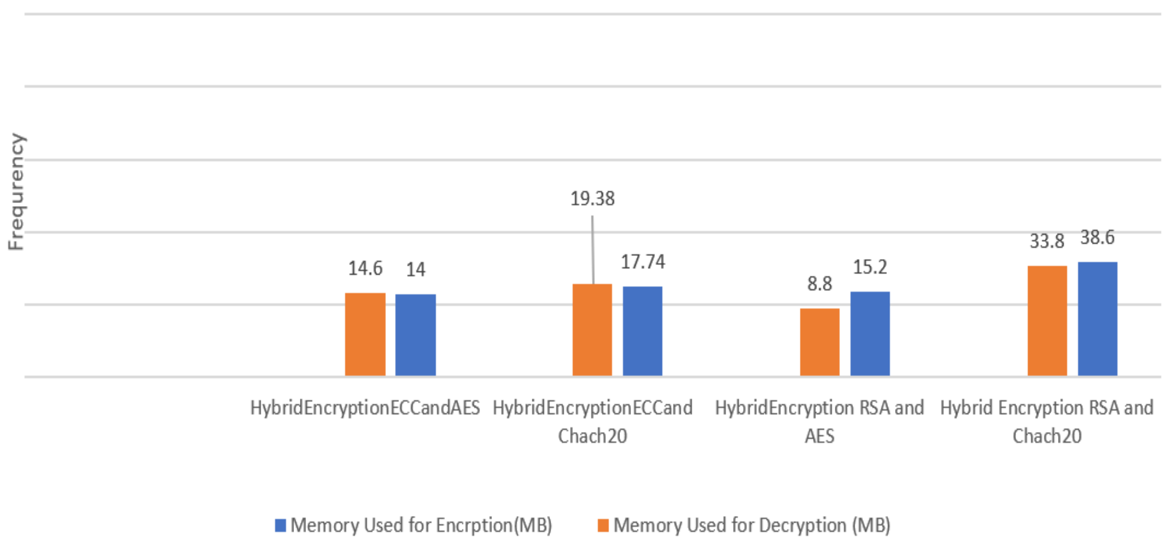

Figure (4): Comparison of Memory Usage for Hybrid Encryption Methods.

5. Conclusions

The research presented a comprehensive comparative analysis of four hybrid encryption schemes RSA with AES, RSA with ChaCha20, ECC with AES, and ECC with ChaCha20 implemented and evaluated in terms of key generation time, encryption/decryption speed, and memory utilization across multiple file types and sizes. The findings established a clear performance hierarchy among these combinations, offering valuable insights into designing secure and resource-efficient cryptographic systems.

The results indicate that ECC-based hybrid algorithms consistently outperform RSA-based ones, particularly in computational speed and memory efficiency. Among all configurations, ECC with ChaCha20 achieved the highest overall performance, with an average encryption time of 47.6 ms and decryption time of 39.8 ms, alongside the lowest memory footprint. This exceptional efficiency results from ECC's compact key structure combined with ChaCha20's lightweight arithmetic operations, making this hybrid particularly suitable for mobile, IoT, and embedded environments where computational and memory resources are limited.

Conversely, RSA with AES and RSA with ChaCha20 exhibited higher key generation times and memory demands, restricting their applicability to high-resource or legacy systems requiring RSA compatibility. The ECC and AES hybrid demonstrated balanced performance, achieving the fastest key generation (739.6 ms) and moderate encryption/decryption speeds, suggesting its practicality for systems emphasizing secure key exchange with moderate computational load.

Overall, the study demonstrates that integrating ECC with modern symmetric ciphers like ChaCha20 provides an optimal balance between security strength, speed, and resource efficiency, positioning ECC with ChaCha20 as a leading candidate for next-generation hybrid cryptosystems.

Although this study provides a comprehensive software-based experimental evaluation of hybrid encryption schemes (RSA with AES, RSA with ChaCha20, ECC with AES, and ECC and ChaCha20), further empirical validation on hardware and real-network environments is recommended. Future work will focus on implementing ECC with ChaCha20 and ECC with AES hybrids on embedded and IoT platforms to assess real-world system performance, security resilience, and energy efficiency.

Future research will focus on:

- Hardware-based implementations and real-time testing on IoT and embedded platforms to assess energy efficiency and operational stability.
- Extending the hybrid framework with post-quantum algorithms to evaluate resilience against quantum computing–based attacks.
- Conducting large-scale network simulations to analyze latency, throughput, and interoperability across diverse communication protocols.

These advancements will contribute to the development of high-performance, secure, and scalable encryption frameworks adaptable to the evolving landscape of digital communication and data protection.

Conflict of Interest: The authors declare that there are no conflicts of interest associated with this research project. We have no financial or personal relationships that could potentially bias our work or influence the interpretation of the results.

References

[1] Z. Ch. Oleiwi, W. A. Alawsi, Wisam. Ch. Alisawi, A. S. Alfoudi, and L. H. Alfarhani, “Overview and Performance Analysis of Encryption Algorithms,” *J Phys Conf Ser*, vol. 1664, no. 1, p. 012051, Nov. 2020, doi: 10.1088/1742-6596/1664/1/012051.
[2] B. A. Buhari, A. A. Obiniyi, K. Sunday, and S. Shehu, “Performance Evaluation of Symmetric Data Encryption Algorithms: AES and Blowfish,” *Saudi Journal of Engineering and Technology*, vol. 04, no. 10, pp. 407–414, Oct. 2019, doi: 10.36348/SJEAT.2019.v04i10.002.
[3] P. Chinnasamy, S. Padmavathi, R. Swathy, and S. Rakesh, “Efficient Data Security Using Hybrid Cryptography on Cloud Computing,” 2021, pp. 537–547. doi: 10.1007/978-981-15-7345-3_46.
[4] P. Banasode and S. Padmannavar, “Protecting and Securing Sensitive Data in a Big Data Using Encryption,” *EAI Endorsed Transactions on Smart Cities*, vol. 0, no. 0, p. 163991, Jul. 2018, doi: 10.4108/eai.13-7-2018.163991.
[5] U. Hasija, P. Rustagi, N. Rathore, and V. Gupta, “Cryptographic Foundations: A Comprehensive Review of Block Cipher and Stream Cipher Concepts,” in *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, Feb. 2024, pp. 1302–1306. doi: 10.23919/INDIACom61295.2024.10498260.
[6] D. Ameta and S. Upadhyay, “A Hybrid Approach for Image Encryption using Different Number Iterations in ECC and AES Techniques,” *Int J Comput Appl*, vol. 175, no. 3, pp. 10–12, Oct. 2017, doi: 10.5120/ijca2017915469.
[7] L. Zou, M. Ni, Y. Huang, W. Shi, and X. Li, “Hybrid Encryption Algorithm Based on AES and RSA in File Encryption,” 2020, pp. 541–551. doi: 10.1007/978-981-15-3250-4_68.
[8] S. Zaineldeen and A. Ate, “Improved cloud data transfer security using hybrid encryption algorithm,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 1, p. 521, Oct. 2020, doi: 10.11591/ijeecs.v20.i1.pp521-527.
[9] P. Ganwani, L. Gupta, C. Jain, R. Kulkarni, and S. Chaudhari, “LSB Based Audio Steganography using RSA and ChaCha20 Encryption,” in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jul. 2021, pp. 1–6. doi: 10.1109/ICCCNT51525.2021.9580177.
[10] P. William, A. Choubey, G. S. Chhabra, R. Bhattacharya, K. Vengatesan, and S. Choubey, “Assessment of Hybrid Cryptographic Algorithm for Secure Sharing of Textual and Pictorial Content,” in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, IEEE, Mar. 2022, pp. 918–922. doi:

10.1109/ICEARS53579.2022.9751932.

[11] S. F. Yousif, “Performance Comparison between RSA and El-Gamal Algorithms for Speech Data Encryption and Decryption,” *Diyala Journal of Engineering Sciences*, pp. 123–137, Mar. 2023, doi: 10.24237/djes.2023.16112.

[12] A. A. Alhaj, A. Alrabea, and O. Jawabreh, “Efficient and secure data transmission: cryptography techniques using ECC,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 36, no. 1, p. 486, Oct. 2024, doi: 10.11591/ijeecs.v36.i1.pp486-492.

[13] A. M. ElRashidy, M. H. Abd AlAzeem, A. A. Abd ElHafez, and M. F. Abo Sree, “ChaCha20-AES Combined Algorithm with 512 Bits of Security,” in *2024 6th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, IEEE, Feb. 2024, pp. 1–6. doi: 10.1109/REEPE60449.2024.10479797.

[14] N. Magdum, “Hybrid Encryption using Symmetric Block and Stream Cipher,” *International Journal of Engineering and Management Research*, vol. 13, no. 1, pp. 35–39, Feb. 2023, doi: 10.31033/ijemr.13.1.4.

[15] A. Badhan and S. S. Malhi, “Enhancing Data Security and Efficiency: A Hybrid Cryptography Approach (AES + ECC) Integrated with Steganography and Compression Algortihm,” in *2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, IEEE, Feb. 2025, pp. 450–456. doi: 10.1109/IDCIOT64235.2025.10914830.

[16] B. Sarkar, A. Saha, D. Dutta, G. De Sarkar, and K. Karmakar, “A Survey on the Advanced Encryption Standard (AES): A Pillar of Modern Cryptography,” *International Journal of Computer Science and Mobile Computing*, vol. 13, no. 4, pp. 68–87, Apr. 2024, doi: 10.47760/ijcsmc.2024.v13i04.008.

[17] E. Ochoa-Jimenez, L. Rivera-Zamarripa, N. Cruz-Cortes, and F. Rodriguez-Henriquez, “Implementation of RSA Signatures on GPU and CPU Architectures,” *IEEE Access*, vol. 8, pp. 9928–9941, 2020, doi: 10.1109/ACCESS.2019.2963826.

[18] O. Popoola, M. A. Rodrigues, J. Marchang, A. Shenfield, A. Ikpehai, and J. Popoola, “An optimized hybrid encryption framework for smart home healthcare: Ensuring data confidentiality and security,” *Internet of Things*, vol. 27, p. 101314, Oct. 2024, doi: 10.1016/j.iot.2024.101314.

[19] K. F. Jasim, R. J. Ismail, A. A. Nahi Al-Rabeeah, and S. Solaimanzadeh, “Analysis the Structures of Some Symmetric Cipher Algorithms Suitable for the Security of IoT Devices,” *Cihan University-Erbil Scientific Journal*, vol. 5, no. 2, pp. 13–19, Sep. 2021, doi: 10.24086/cuesj.v5n2y2021.pp13-19.

[20] P. Patil and R. Bansode, “Performance evaluation of hybrid cryptography algorithm for secure sharing of text & images,” *International Research Journal of Engineering and Technology*, vol. 7, no. 9, pp. 3773–3778, 2020.

[21] J. M. Parenreng, S. M. Mustari, and A. Wahid, “E-mail Security System Using El-Gamal Hybrid Algorithm and AES (Advanced Encryption Standard) Algorithm,” *Internet of Things and Artificial Intelligence Journal*, vol. 2, no. 1, pp. 1–9, Feb. 2022, doi: 10.31763/iota.v2i1.510.

[22] K. Sethi *et al.*, “Lightweight DWT Steganography With ECC-ChaCha20 for Secure Medical IoT Systems,” *IEEE Access*, vol. 13, pp. 142948–142960, 2025, doi: 10.1109/ACCESS.2025.3598099.

[23] 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.

[24] 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.

[25] R. Ganesh, B. U. I. Khan, A. R. Khan, and A. Bin Kamsin, “A panoramic survey of the advanced encryption standard: from architecture to security analysis, key management, real-world applications, and post-quantum challenges,” *Int J Inf Secur*, vol. 24, no. 5, p. 216, Oct. 2025, doi: 10.1007/s10207-025-01116-x.

[26] 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.

[27] Ami M. Shah and Ashishkumar Gor, “Comprehensive Survey of Symmetric and Public-Key Cryptographic Algorithms: Foundations, Attacks, and Applications,” *International Journal Of Informative and Futuristic Research*, vol. 12, no. 10, pp. 20–38, 2025, doi: 10.64672/IJIFR/25.06.12.10.005.

[28] 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.

[29] P. R. Brandao and C. S. Silva, “Quantum Computing and Cryptography,” Aug. 07, 2025. doi: 10.20944/preprints202508.0555.v1.

[30] P. Iqlima, M. Rayyan, A. Z. A. Rambe, E. Ardina, D. F. A. Lubis, and D. Ismawati, “Implementasi Sistem Keamanan Data Menggunakan Algoritma Kriptografi Asimetris Elliptic Curve Cryptography (ECC) Berbasis Website,” *JIKUM: Jurnal Ilmu Komputer*, vol. 1, no. 1, pp. 7–11, May 2025, doi: 10.62671/jikum.v1i1.37.

[31] M. M. Ganesan and S. Selvaraj, “An improved key scheduling for advanced encryption standard with expanded round constants and non-linear property of cubic polynomials,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 15, no. 2, p. 2455, Apr. 2025, doi: 10.11591/ijece.v15i2.pp2455-2467.

[32] M. J. S. Zahid, “Optimizing Authenticated Encryption: High-Performance Implementations of ChaCha20 and Blake3 for Large-Scale Data,” Mar. 05, 2025. doi: 10.36227/techrxiv.174119272.22132921/v1.

[33] N. Janu, S. Vishnoi, S. L. Yadav, T. Chawla, and P. Vats, “Elliptic curve-based cryptanalysis techniques : A strategic approach to enhancing information security,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 28, no. 5-B, pp. 2015–2024, 2025, doi: 10.47974/JDMSC-2419.