NOTTINGHAM TRENT UNIVERSITY

ENHANCING THE PERFORMANCE OF CRYPTOGRAPHIC ALGORITHMS FOR SECURED DATA TRANSMISSION

By

KWAME ASSA-AGYEI
(N0950209)

A DISSERTATION

Submitted to

The Doctoral School, Nottingham Trent University, United Kingdom in partial fulfillment of the Requirement for the Degree of

DOCTOR OF PHILOSOPHY in COMPUTER SCIENCE (EMERGENT TECHNOLOGY)

SPECIALIZATION: CRYPTOGRAPHY

Supervisory Team:

Director of Studies: Dr. Kayode Owa

Co-Supervisor: Dr. Tawfik Al-Hadhrami

Co-Supervisor: Dr. Funminiyi Olajide

JULY, 2024

Declaration of Authorship

I solemnly declare that the dissertation submitted is the result of my independent research work under the guidance of my Supervisory Team. In addition to the content cited in the article, this article does not contain any of the works published or written by any other individual or group, nor does it contain materials used to obtain degrees or certificates from the Nottingham Trent University or other educational institutions. The individuals and collectives that have made important contributions to this study have been clearly identified in the text. I am fully aware that I am obligated to undertake the legal consequence of the statement.

Signature of the Candidate: [illegible]

Year-Month Date: 15 July 2024

DEDICATION

I would like to dedicate this dissertation to my Late Dad (Kwame Assa-Agyei), Mum (Sarah Konadu-Minkah), spouse (Rosemary Boateng), children, Siblings (Rita Assa-Agyei, Maame Yaa Agyeiwaa, Millicent Abena Konadu and Franklyn Opoku Agyemang), Apostle Emmanuel Tetteh and Late Mrs. Dorcas Tandoh for their constant prayers, love, inspiration, and understanding which has made it possible for me to complete this Ph.D. program at Nottingham Trent University, United Kingdom.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Dr. Kayode Owa, my Director of Studies, and Dr. Funminyi Olajide, my former Director of Studies, who later assumed the role of my external supervisor. Additionally, I am grateful to Dr. Tawfik Al-Hadhrami, my Co-Supervisor, and Dr. Temitope Alade (Former Co-Supervisor) at the School of Science and Technology, Nottingham Trent University, United Kingdom, for their unwavering guidance and support throughout the duration of this project. Your influence on my academic pursuit has been exceptional. I wish to thank you sincerely for your constructive comments, suggestions, and encouragement during my period of study at Nottingham Trent University. More specifically, I am grateful you believed in my academic and research capabilities and provided the needed environment and conditions that made my Ph.D. study at Nottingham Trent University a remarkable one. Your dedication and passion have made it possible for me to gain in-depth experience to effectively write journal and conference papers. I am heavily indebted to all, and l will forever cherish the valuable experience gained under your guardianship.

I would like to express my profound gratitude to the members of the SRDC, chaired by Dr. Dmitry Volodkin, for their invaluable comments which have shaped me and significantly improved the quality of my dissertation. I am also deeply grateful to Prof. Ahmad Lotfi (Head of the Department of Computer Science) and Dr. Jo Hartley (Line Manageress) for appointing me as an Hourly Paid Lecturer in the Department of Computer Science.

My appreciation also goes to Vice Admiral Seth Amoama, R/Adm Moses Baffour Beick (Rtd), and Staff of the Ghana Armed Forces Command and Staff College, Dr. George Ofori-Dwumfuo, Air Vice Marshall IS Kadri (Rtd), Colonel M Mantey, Colonel BB Pantoah, Colonel Komblaga, Colonel E Annor, Colonel EO Seppey, Lt Colonel N Aboagye-Bonsu, Dr. Patrick Kudjo, Dr. William Acquaye-Brown, Dr. Lempogo Forgor, Dr. Emmanuel Acquah-Sam, Dr. Emmmanuel Sam, Mr. Fred Charles Anson (Former Commissioner, Support Service (GRA)), Mr. Seth Kwame Asafo (NITA), Mr. Leonard Kyei, Dr. Salifu Nanga, Paa Kobina Todd, Mr. Paul Arhin, Mr. Evans King, Mr. Frederick Broni, Mrs. Nora Agyei-Ababio, Mr. Francis Okah, Mr. Peter Kofi Bani, Mr. Victor Nmese, Mr. Harrison Agbottah Mawuli, Mr. Frank Nii Teiko Taoe, Mr. Nathan Nyarko Sackey (Stegor), Mr. Maxwell Solomon, Miss Selina Amoah, Mr. Nana-Yao Nsowah, Mr. Johnson Amenadam, Mr. Andrews Ekow-Fenyi Annan, Miss Joyceline Nutakor, Augustus Ofosu Amoa, Mr. Daniel Okechukwu-Tonte Ihenacho, and Jacob Lokko for your candid advice. I thank you for your support and encouragement.

To the Registrar, the Board and Management of Ghana Scholarship Secretariat and Naa Dedei Tetteh (Head/Education & Recruitment- Ghana High Commission) for their encouragement and financial assistance throughout my period of stay in United Kingdom.

My profound heartfelt gratitude also goes to Mr. Bruce Svondo and Lady Sue Lathall (Manageress), Sarah Tannis and Staff of Cherry Trees Resource Centre for your support and encouragement. Finally, to anyone who has contributed in one way or the way towards the completion of my Ph.D. programme, I say thank you.

ABSTRACT

Cryptography is crucial in the digital age for protecting data integrity, verifying users and entities, and maintaining the secrecy of sensitive information. This is accomplished by utilizing various cryptographic techniques such as symmetric-key and asymmetric-key encryption, digital signatures, and secure communication protocols. These strategies collectively create a thorough defense strategy to prevent unauthorized access, enhancing the overall security of digital data in today's environment. This study has pinpointed various research gaps linked to the two categories of cryptographic algorithms and proposed innovative approaches to address these gaps. The study not only identified areas where further research is needed but also recommended novel strategies to tackle these gaps effectively. This dissertation's primary contributions are:

(1) This study empirically evaluates the performance of widely utilized symmetric algorithms, including AES, Blowfish, 3DES, and Twofish. The assessment includes important measurements like encryption and decryption times, throughput, and memory utilization. The objective is to conduct a thorough investigation of the efficiency and usefulness of these cryptographic methods in real-world situations. The findings reveal that the AES algorithm with a 256-bit key size exhibits the highest encryption time compared to AES with 128-bit and 192-bit key sizes. The specific average encryption times are as follows: (1) AES-128: Average encryption time of 0.057 seconds (2) AES-192: Average encryption time of 0.049 seconds and (3) AES-256: Average encryption time of 0.038 seconds. The experimental results demonstrate that AES excels over alternative symmetric encryption techniques in terms of both encryption and decryption speeds, along with overall throughput. Furthermore, the findings establish that the Blowfish algorithm can compete with AES in terms of encryption and decryption speed.

(2) This study conducts a practical evaluation of commonly used asymmetric algorithms, specifically RSA and ECC, focusing on key parameters such as key exchange time, encryption and decryption times, and signature generation and verification times. The investigation uses secure email communication as a case study to analyze the real-world performance of these cryptographic algorithms. Additionally, the research proposes a hybrid cryptography algorithm that combines both RSA and ECC to enhance security and confidentiality in secure email communication. The proposed hybrid algorithm demonstrated an average key exchange time of 0.064191 seconds, which is faster compared to ECC and RSA. For a larger file size of 500 MB, the proposed hybrid algorithm achieved average encryption and decryption times of 0.832917 seconds and 0.636395 seconds, respectively. In comparison, the ECC algorithm recorded average

encryption and decryption times of 0.866455 seconds and 0.753799 seconds, respectively with a minimal disparity in runtime compared to ECC, indicating improved efficiency. Experimental results also highlight ECC's advantages in Key Exchange Time, making it a preferable choice for establishing secure email channels, especially for larger file sizes. While RSA shows a slight edge in efficiency for smaller files, the hybrid encryption algorithm optimizes key exchange times, encryption efficiency, and signature generation and verification times.

(3) This research introduces novel strategies to enhance the performance of existing cryptographic algorithms, specifically AES and RSA, by addressing identified research gaps. To achieve this, the study incorporates performance optimization techniques and numeric functions aimed at improving both the efficiency and security of data transmission. Specifically, Multi-Chaotic AES demonstrated notable improvements in performance compared to standard AES. It achieved faster encryption times with reductions ranging from approximately 50% to over 70%. Additionally, Multi-Chaotic AES exhibited enhanced decryption efficiency, showing time reductions of about 40% to over 70%. These results underscore the substantial performance gains of Multi-Chaotic AES in both encryption and decryption processes. Similarly, for the Modified RSA algorithm, notable improvements were observed. For instance, at a key size of 1024 bits, traditional RSA requires an average key generation time of 111 milliseconds. In contrast, the NGOA-DE-RSA approach significantly reduces this time to 55 milliseconds. This trend of improved performance extends to other metrics as well, with NGOA-DE-RSA consistently outperforming the standard RSA algorithm.

Finally, this study proposes a hybrid approach incorporating both modified AES and RSA and assesses its performance relative to existing hybrid techniques. Remarkably, the proposed algorithms exhibit significantly improved prediction accuracy across various metrics, including process times, throughput, memory utilization, and security.

In summary, this thesis makes significant contributions to applied cryptography by introducing innovative techniques designed to enhance data transmission security. The research presented in this thesis offers valuable insights and novel approaches that contribute to the advancement of cryptographic practices in practical applications, particularly in the realm of secure data transmission.

TABLE OF CONTENT

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Nomenclature

Acronyms

AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
ECC	Elliptic Curve Cryptography
CPU	Central Processing Unit
GPU	Graphic Processing Unit
DES	Data Encryption Standard
XOR	Exclusive -OR operation
GF	Galois Fields
3DES	Triple Data Encryption Standard
IDEA	International Data Encryption Algorithm
CAST	Carlisle Adams and Stafford Tavares
TEA	Tiny Encryption Algorithm
DSA	Digital Signature Algorithm
RAM	Random Access Memory
ECDLP	Elliptic Curve Discrete Logarithm Problem
SSL	Secure Sockets Layer
TLS	Transport Layer Security
PRNG	Pseudo-random Number Generation
PKI	Public-Key Infrastructure
LAN	Local Area Network
KET	Key Exchange Time
CUDA	Compute Unified Device Architecture
GCD	Greatest Common Denominator
MM-AES	Modified MixColumn Advanced Encryption Standard
EEF-AES	Enhanced Encryption Framework Advanced Encryption Standard
LZMA	Lempel-Ziv-Markov chain algorithm
FMT	Faber-Schauder Multi-scale Transform
IV	Initialization Vector

List of Publications

Published papers

1. Agyei-Ababio, N., Ansong, E., & **Assa-Agyei, K.** (2023). Digitalization of revenue mobilization in an emerging economy: the new Institutional Theory perspective. *International Journal of Information Systems and Project Management*, *11*(2), 5–22. https://doi.org/10.12821/ijispm110201

2. **Assa-Agyei, K**., & Olajide, F. (2023a). A Comparative Study of Twofish, Blowfish, and Advanced Encryption Standard for Secured Data Transmission. *International Journal of Advanced Computer Science and Applications*, *14*(3), 393–398. https://doi.org/10.14569/IJACSA.2023.0140344

3. **Assa-Agyei, K.**, & Olajide, F. (2023b). A Comprehensive Evaluation of the Rivest-Shamir-Adleman (RSA) Algorithm Performance on Operating Systems using Different Key Bit Sizes. *International Journal of Computer Applications*, *185*(19), 14–20. https://doi.org/10.5120/ijca2023922884

4. **Assa-Agyei, K.**, Olajide, F., & Lotfi, A. (2022). Security and Privacy Issues in IoT Healthcare Application for Disabled Users in Developing Economies. *Journal of Internet Technology and Secured Transactions*, *10*(1), 770–779. https://doi.org/10.20533/jitst.2046.3723.2022.0095

5. **Assa-Agyei, K.**, Owa, K., Al-Hadhrami, T., & Olajide, F. (2024). Hybrid Algorithm using Rivest-Shamir-Adleman and Elliptic Curve Cryptography for Secure Email Communication. *International Journal of Advanced Computer Science and Applications*, *15*(4), 1037–1047. https://doi.org/10.14569/IJACSA.2024.01504105

6. **Assa-Agyei, K.**, Owa, K., Olajide, F., & Al-Hadhrami, T. (2024). A Multi-Chaotic Key Expansion for Advanced Encryption Standard (AES) Algorithm. *2024 International Conference on Computing, Networking and Communications, ICNC 2024*, 711–717. https://doi.org/10.1109/ICNC59896.2024.10556263

6. Olajide, F., **Assa-Agyei, K.**, & Edo, C. (2023). An Empirical Evaluation of Encryption and Decryption Times on Block Cipher Techniques. *Proceedings - 2023 Congress in Computer Science, Computer Engineering, and Applied Computing, CSCE 2023*, 2385–2390. https://doi.org/10.1109/CSCE60160.2023.00386

Accepted papers

1. Conference Paper: A comparative study of the various key bit sizes on RSA algorithm with CUDA (Accepted)
   Author(s): **Kwame Assa-Agyei**, Funminiyi Olajide, Temitope Alade and Ahmad Lotfi International Conference on Security & Management (SAM'22)

2. Conference Paper: Analysis and Improvement of MixColumn Operations in the Advanced Encryption Standard Algorithm (Accepted)
   Author(s): **Kwame Assa-Agyei**, Kayode Owa, Tawfik Al-Hadhrami, International Conference on Security & Management (SAM'24)

3. Conference Paper: Analysis and Improvement of MixColumn Operations in the Advanced Encryption Standard Algorithm (Accepted)

Author(s): **Kwame Assa-Agyei**, Kayode Owa, Tawfik Al-Hadhrami,
International Conference on Security & Management (SAM'24)

4. Conference Paper: An Efficient Generation of Prime Numbers for RSA Encryption Scheme (Accepted)

Author(s): **Kwame Assa-Agyei**, Kayode Owa, Tawfik Al-Hadhrami
International Conference on Security & Management (SAM'24)

5. Conference Paper: Blockchain Technology Readiness in Engineering Project Management: Industry Experts' Perspectives (Accepted)

Author(s): Nana-Yao Nsowah, **Kwame Assa-Agyei**, Frederick Edem Junior Broni, Selina Amoah and Daniel O.T. Ihenacho
International Conference on Security & Management (SAM'24)

6. Journal Article: Optimizing Data Security with Advanced Encryption Standard and Lempel-Ziv-Markov Chain algorithm (Revision Submitted)

Author(s): Funminiyi Olajide, **Kwame Assa-Agyei,** Tawfik Al-Hadhrami
PLOSOne

Chapter 1 Introduction

This chapter introduces the background of the study, problem statement, objectives of the study, research question, scope of the study, significance of the study, contributions, and a general overview of the dissertation. Finally, the contributions and research framework are presented in this chapter.

1.1. Introduction and Motivation

Cryptographic algorithms have emerged as the predominant method for ensuring security in safeguarding sensitive information. Among the key objectives of protection, confidentiality is a critical aspect in implementing and integrating cryptographic algorithms in modern communication system. This objective of confidentiality is given significant consideration by the hardware involved in the process. In today's communication systems, the reliance on cryptographic algorithms has grown substantially due to their ability to provide a high level of protection for vital information. By employing complex mathematical computations and encryption techniques, cryptographic algorithms encode data in a manner that renders it unintelligible to anyone without the appropriate decryption key [1]. The integration of cryptographic algorithms into communication systems involves careful consideration of various factors, including hardware components. These components, such as processors and cryptographic modules, are responsible for executing the algorithms and managing the encryption and decryption processes. During implementation, special attention is given to the confidentiality objective, ensuring that the hardware effectively maintains the secrecy and integrity of sensitive data. Confidentiality as an essential element in cryptographic algorithm implementation involves the protection of information from unauthorized disclosure. It encompasses measures taken to prevent eavesdropping, data breaches, or unauthorized access to encrypted data. The hardware involved in this process is designed to handle encryption and decryption operations securely, maintaining the confidentiality of the data throughout the communication system [2]. Cryptography finds applications in various domains to ensure secure communication and protect sensitive information [3] [4]:

a. Secure Communication: Cryptography forms the foundation of secure communication protocols such as SSL/TLS, which are used to establish secure connections between web browsers and servers. It enables the encryption of data transmitted over the internet, preventing eavesdropping and unauthorized access.

b. Data Protection: Cryptography is used to protect sensitive data at rest, such as stored passwords, financial records, and medical records. Encryption algorithms safeguard this information, ensuring that even if the data is compromised, it remains unreadable without the appropriate decryption key.

c. Digital Signatures: Cryptographic algorithms enable the creation of digital signatures, which verify the authenticity and integrity of digital documents. Digital signatures provide non-repudiation, ensuring that the signer cannot deny their involvement in the document.

d. Secure Authentication: Cryptography plays a crucial role in user authentication mechanisms, such as password-based authentication, two-factor authentication, and biometric authentication. It ensures that only authorized individuals can access protected systems or sensitive information.

As a result of the diverse requirements and security considerations in various applications, there is a wide range of cryptographic algorithms available. These algorithms differ in terms of their underlying mathematical principles, key sizes, security features, and performance characteristics [5]. The availability of multiple cryptographic algorithms provides options for selecting the most suitable algorithm based on specific needs and considerations. For symmetric key encryption, there are several well-known algorithms such as Advanced Encryption Standard (AES), Data Encryption Standard (DES), Triple Data Encryption Standard (3DES), and Blowfish. Each algorithm has its own strengths and weaknesses in terms of security, speed, and suitability for different use cases. In the realm of asymmetric key algorithms, the most widely used ones include RSA, Diffie-Hellman, Elliptic Curve Cryptography (ECC), and ElGamal [6] [4]. These algorithms offer different levels of security and efficiency, and they are often used for tasks like key exchange, digital signatures, and secure communication. Furthermore, there are hash functions like SHA-256, MD5, and SHA-3, which are used for data integrity verification and password storage. Hash functions play a crucial role in ensuring the integrity of data by generating fixed-length representations (hash values) from variable-length input data. In addition to the commonly used cryptographic algorithms, there are also specialized algorithms designed for specific purposes [7]. These include algorithms for homomorphic encryption, zero-knowledge proofs, post-quantum cryptography, and more. These specialized algorithms address unique requirements, such as secure computation on encrypted data, proving knowledge of information without revealing it, and resistance against attacks by quantum computers [8].

This study empirically investigates the performance of the most commonly used asymmetric and symmetric cryptographic algorithms. Furthermore, this study examines the performance based on encryption and decryption times, throughput (speed), memory usage (space complexity) and power consumption. In addition to the availability of various cryptographic algorithms, this thesis proposes techniques that focus on improving the performance of both symmetric and asymmetric algorithms through low-level programming. Low-level programming involves working at a lower level of abstraction, closer to the hardware and processor architecture, to optimize code execution and maximize performance. By leveraging low-level programming techniques, the thesis aims to enhance the efficiency and speed of cryptographic algorithms. For symmetric key algorithms, such as AES, the thesis explores low-level programming improvement that can be applied to the encryption and decryption processes. This may involve utilizing specific processor instructions or optimizing memory access patterns to minimize computational overhead and improve overall performance. Similarly, for asymmetric key algorithms like RSA or Elliptic Curve Cryptography (ECC), the thesis proposes techniques to enhance key generation, encryption, and decryption operations. Low-level programming can be used to leverage processor capabilities, parallelization, or specialized instructions to speed up the mathematical computations involved in these algorithms.

The goal of these proposed techniques is to reduce the computational complexity and runtime of cryptographic operations without compromising security. By fine-tuning the implementation at a low level, the thesis aims to achieve significant performance improvements, making cryptographic algorithms more efficient and practical for real-world applications. The thesis delves into these technical details to demonstrate the effectiveness of the proposed techniques and their impact on the performance of both symmetric and asymmetric algorithms.

Overall, by incorporating low-level programming techniques, the thesis aims to contribute to the field of cryptography by enhancing the performance and efficiency of cryptographic algorithms, making them more viable for resource-constrained environments or applications that require high-speed cryptographic operations.

1.2. Problem statement

According to Bruce Schneier, a renowned cryptographer makes it clear and obvious that cryptography can be strong or weak [9]. Cryptographic algorithms are evaluated based on the time and resources required to decrypt the ciphertext and their susceptibility to various types of attacks.

Deciphering a robust encryption without the necessary decoding equipment is extremely challenging and arduous. Many cryptographic algorithms exist, and various aspects support the selection of a certain method, including its capacity to secure data against assaults, time complexity (speed), space complexity (memory), latency, and efficiency. These algorithms depend on complex mathematical computational problems that are heavily influenced by parameter selection, causing them to operate slowly during transactions. The performance of an algorithm, including factors like running time and memory usage, is crucial and can significantly impact the overall efficiency of the method. [10].

Previous studies identified a number of research gaps:

1. The Advanced Encryption Standard (AES) is recognized as one of the most secure and efficient symmetric cryptosystems for encryption. It utilizes the Rijndael Algorithm for the generation and expansion of keys, supporting key sizes of 128, 192, and 256 bits. Traditional methods for key expansion in AES employ fixed approaches, where the same expansion mode is consistently applied throughout the encryption process [1][11]. The AES key expansion algorithm is a crucial component of the AES encryption and decryption procedures. It takes the initial secret key and generates a series of round keys used in various rounds of AES. Despite its efficiency, the AES key expansion algorithm has a significant vulnerability. In the event that an adversary gains knowledge of any round key, they can deduce all other round keys, a weakness known as the "related-key attack." This vulnerability poses a substantial threat to the overall security of AES [12].

2. In contemporary digital environments, ensuring the security of data during transmission is crucial, often achieved through the application of encryption techniques such as the Advanced Encryption Standard (AES). Many methods that incorporate encryption and compression face the challenge of minimizing data size while simultaneously maintaining the security of the algorithm [13][14]. This means finding a balance between optimizing data storage and safeguarding the integrity and confidentiality of the information.

3. Advanced Encryption Standard (AES) has become widely used in the information security industry because of its superior security and effectiveness. In 2001, the National Institute of Standards and Technology (NIST) released AES as Federal Information Processing Standard 197 (FIPS 197) [15]. Implementing AES encryption resolves the aging issues associated with the Data

Encryption Standard (DES). The Rijndael (AES) symmetric block cipher standard version is capable of encrypting and decrypting plaintext in 128-bit blocks using a key of 128-bit, 192-bit, or 256-bit size [16]. AES follows a precise sequence of four distinct transformations—Sub Bytes, ShiftRows, MixColumns, and AddRoundKey—in that particular order. Each transformation involves mapping a 128-bit input state to a corresponding 128-bit output state. The number of rounds needed to produce the cipher text is determined by the size of the cipher key and the iterations in a loop, Nr, which can be set to 10, 12, or 14 [17]. Previous studies into the MixColumn operation have highlighted that the MixColumn transformation within the AES encryption process is resource-intensive, particularly in terms of delay and throughput. The multiplication operation inherent in MixColumn is slow and can have a substantial impact on the overall speed of encryption [18][19].

4. The RSA cryptographic algorithm places a significant reliance on the secure generation of substantial prime numbers during its initialization process [20]. However, challenges emerge concerning the speed of prime number selection and the imperative need for larger primes to enhance security [21][22].

5. Research papers examining symmetric cryptographic algorithms highlight the presence of experimental gaps related to process times, throughput, space complexity, and power consumption [23][24][25][26]. These gaps arise due to limited methodological descriptions and flawed comparisons caused by variations in key bit sizes and fixed block sizes among different algorithms. The observed experimental gaps underscore areas that require further investigation and improvement. One contributing factor to these gaps is the inadequate level of detail provided in the methodology descriptions of certain research papers. Thorough descriptions of the experimental setup, input data, computational environment, and measurement techniques are vital for ensuring reproducibility and reliable results. Without comprehensive methodology descriptions, accurate comparisons and evaluations of cryptographic algorithm performance become challenging. Another factor leading to flawed comparisons is the presence of different key bit sizes and fixed block sizes among the cryptographic algorithms being assessed. Key bit size refers to the length of the cryptographic keys used in the algorithms, while fixed block size refers to the fixed-length blocks of data processed by the algorithms [27]. These variations can significantly impact the performance metrics and make direct comparisons problematic. To obtain

meaningful and valid comparisons, it is essential to ensure that the key bit sizes and fixed block sizes are consistent across the evaluated algorithms.

6. Research papers focused on the analysis of asymmetric algorithms have consistently demonstrated that Elliptic Curve Cryptography (ECC) outperforms all other asymmetric algorithms in terms of speed and efficiency. Nevertheless, it is crucial to note that existing studies have primarily concentrated on ECC's comparative advantage in specific scenario [28][29].

However, there remains a significant gap in the literature as no comprehensive investigation has been conducted to evaluate the performance of asymmetric algorithms across a diverse range of applications. A thorough exploration of such algorithms, considering various use cases, is necessary to determine their respective process times, throughput, and other relevant metrics. This broader analysis would provide a more comprehensive understanding of how different asymmetric algorithms perform under different conditions and could potentially unveil novel insights into their practical applicability and optimization opportunities.

Thus, this study envisions that the introduced techniques can improve the performance of cryptographic algorithms for secured online data transmission in terms of security, process times, space complexity, and throughput and power consumption.

1.3. Aim

This research aims to introduce techniques to enhance the existing cryptographic algorithms, ensuring faster and more secure data transmission and transactional operations. This will be achieved by addressing various crucial aspects such as encryption and decryption times, throughput, memory usage, power consumption, and security of the proposed algorithms.

1.4. Objectives of the study

The primary objective of this research is to put forward advanced cryptographic algorithms that concentrate on securing data communication and transactions. The specific objectives are as follows:

1. To conduct a performance analysis of the most commonly used symmetric algorithms
2. To evaluate the performance of the most commonly used asymmetric algorithms
3. To create and introduce an improved asymmetric algorithm that specifically targets efficient data communication and transaction processing

4. To develop enhanced symmetric algorithms focusing on data communication and transaction

1.5. Research Questions

1. Which symmetric algorithm exhibits optimal performance in terms of processing times, throughput, memory usage, and power consumption?

2. Which asymmetric algorithm shows the best performance in terms of key exchange time, encryption and decryption durations, signature generation, and verification times?

3. What research investigations can be conducted to develop and introduce an advanced asymmetric algorithm that focuses on maximizing the efficiency of data communication and transaction processing?

4. What innovative approaches can be developed to enhance symmetric algorithms specifically tailored for improving data communication and transaction processes?

1.6. Significance of the study

This study holds significant importance in terms of research, policy, and practice. It adds valuable insights to the existing body of literature on Cryptography and addresses the research gap by exploring areas that lack sufficient scholarly work. Moreover, this research serves as a comprehensive reference for students and researchers interested in further exploring the utilization of Cryptography.

Furthermore, the findings of this study have practical implications, as they contribute to the development and implementation of effective cryptographic algorithms to enhance communication and strengthen cryptographic security. Policymakers can rely on the outcomes of this study to make informed decisions regarding the application and usage of cryptosystems.

1.7. Contributions to Knowledge

This section outlines and describes the key research contributions of this dissertation:

- This study conducts both theoretical and empirical analyzes to evaluate the performance of AES, Blowfish, 3DES, and Twofish in terms of encryption times, decryption times and throughput (speed).The evaluation is carried out using comparable key bit sizes and their respective fixed block sizes. Based on the identified research gaps, the study formulated and successfully executed two primary tasks. These accomplishments have been thoroughly documented in a peer-reviewed journal article (Q3) and presented in a conference paper published in the IEEE proceedings. The respective publications can be accessed via the following links: [Journal Article (Q3)] and [Conference Paper (IEEE Proceedings)].

1. https://doi.org/10.14569/IJACSA.2023.0140344
2. https://doi.org/10.1109/CSCE60160.2023.00386

- In this research study, a thorough performance evaluation is carried out on the most commonly used asymmetric algorithms. The main objective of this evaluation is to assess the algorithms' effectiveness and efficiency across various real-world use case (secure email communication). The study aims to determine essential metrics such as process times, throughput, and other relevant factors that directly impact the algorithms' practical applicability. By examining the algorithms under diverse scenarios, this research seeks to provide a comprehensive understanding of how they perform in different situations. The findings of this evaluation will not only shed light on the strengths and weaknesses of each asymmetric algorithm but also offer valuable insights into their suitability for specific applications. Through this investigation, researchers hope to contribute to the optimization of asymmetric algorithms and aid in the decision-making process for selecting the most suitable algorithm for the use case. Furthermore, the study's results may serve as a valuable resource for developers, cybersecurity experts, and other professionals involved in cryptographic systems, helping them make informed choices to enhance the security and efficiency of their applications.

This task was also documented in a peer-reviewed journal article classified as Q3. Below is the link: https://doi.org/10.14569/IJACSA.2024.01504105

- The final contribution mainly seeks to make a significant contribution by developing novel cryptographic algorithms that are specifically designed to enhance the efficiency of data communication and transaction processing. These algorithms will be tailored to address the challenges and requirements associated with secure and efficient data exchange and transactional activities. By introducing these innovative cryptographic algorithms, this study intends to provide practical solutions that can optimize the security and effectiveness of data communication and transaction processing systems. The development of such algorithms has the potential to revolutionize the field of cryptography and significantly impact various domains that rely on secure and efficient data transmission and transactional operations.

Based on the gaps identified in existing AES and RSA frameworks and their modifications, this study has made several significant contributions to the body of knowledge. Below is the list of published and accepted papers:

1. https://doi.org/10.1109/ICNC59896.2024.10556263

Accepted Journal and Conference papers

1. Analysis and Improvement of MixColumn Operations in the Advanced Encryption Standard Algorithm
2. An Efficient Generation of Prime Numbers for RSA Encryption Scheme
3. Optimizing Data Security with Advanced Encryption Standard and Lempel-Ziv-Markov Chain algorithm

In summary, the unique contribution to knowledge is the creation of algorithms that will improve processor efficiency and security. Based on the research gaps, the direction is to develop new algorithms which will be compared to existing cryptographic algorithms to measure the following indicators: throughput, execution times, memory usage, and security.

1.8. Sources of information and resources

This research with get information from the following resources;

- The secondary source is the NTU library sources
- The internet such as Google Scholar, IEEE Xplore, ISI Web of Science and many more
- Subsequently, the primary research phase will be initiated, involving simulations and experiments to effectively address the identified research problem.

1.8.1. Software requirements:

- NVIDIA CUDA
- Python
- Math Lab/Mini Tab

- MathType

1.8.2. Hardware requirements:

- CUDA – Enabled GPU NVIDIA GeForce GT 130M
    - 1.5 GHz with 32 Cores
    - Processor clock of 1500MHz
    - Memory Clock of 800MHz
    - Memory Interface width of 128
- CPU
    - Intel i7/i9 CPU with speed of 5.0 GHz

1.8.3. Standard Libraries/Dependencies used

- cffi
- cryptography
- numpy
- pip
- psutil
- pycparser
- pycryptodomex
- pycryptoplus
- scipy
- wheel

1.9. Organization of the Dissertation

The dissertation follows this structure: Chapter 2 explores prior research on widely used asymmetric and symmetric algorithms, the mathematics of cryptography, AES and RSA algorithms, and post-quantum cryptography. Chapter 3 investigates the effectiveness of symmetric algorithms commonly employed for data transmission, aligned with research objective (RO 1).

Chapter 4 outlines the performance evaluation of the most commonly used asymmetric algorithms, aligned with research objective (RO 2). Chapter 5 introduces innovative architectures for AES and RSA algorithms, with a focus on data communication and transmission. Chapter 6 concludes and provides recommendation by elaborating on vital points and contributions made throughout the dissertation. Lastly, future research directions are provided in the concluding section.

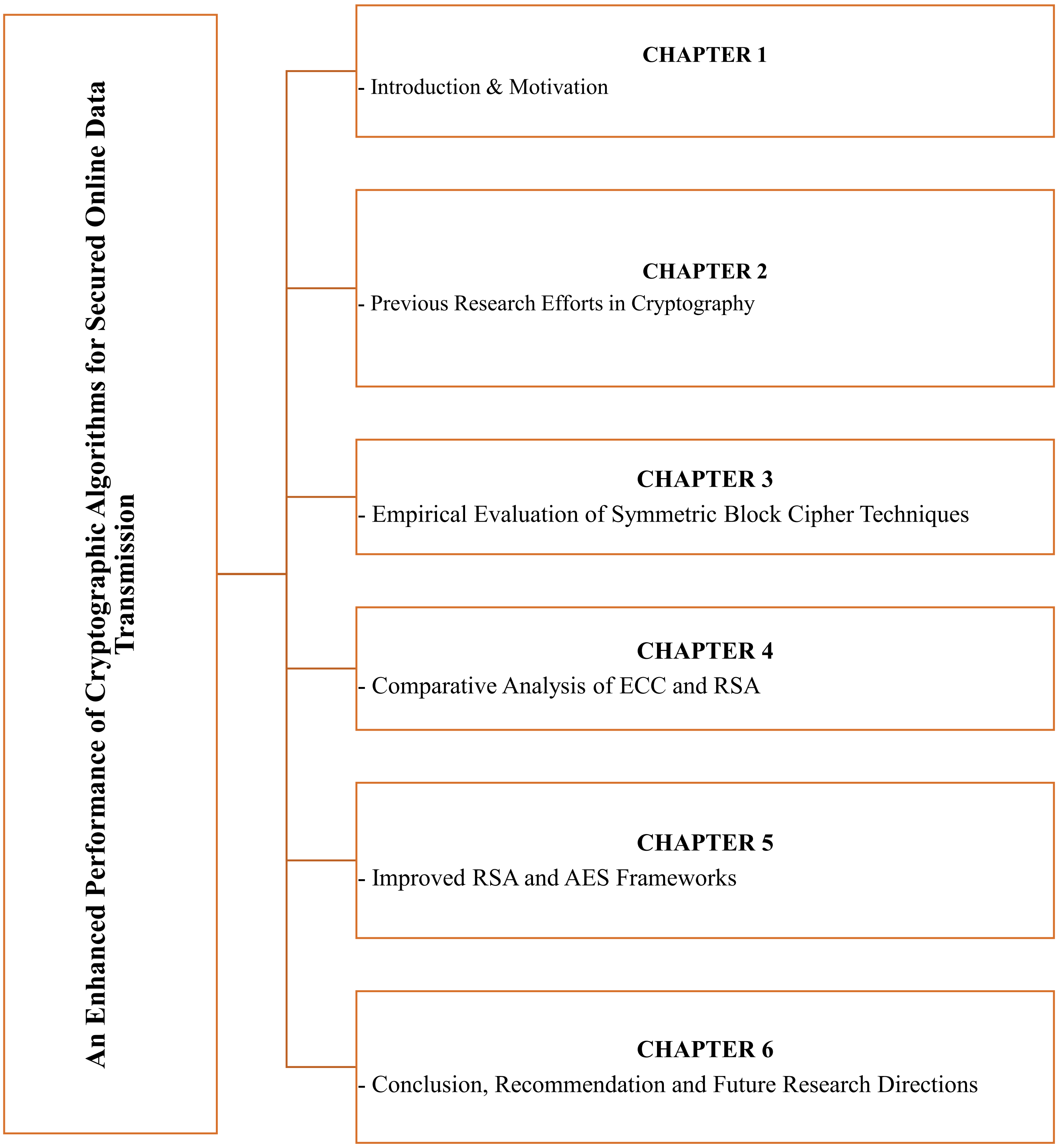

An Enhanced Performance of Cryptographic Algorithms for Secured Online Data Transmission
CHAPTER 1
- Introduction & Motivation
CHAPTER 2
- Previous Research Efforts in Cryptography
CHAPTER 3
- Empirical Evaluation of Symmetric Block Cipher Techniques
CHAPTER 4
- Comparative Analysis of ECC and RSA
CHAPTER 5
- Improved RSA and AES Frameworks
CHAPTER 6
- Conclusion, Recommendation and Future Research Directions

Figure 1.1: Organization of dissertation

Chapter 2 Literature Review

2.1. Introduction

This chapter reviews the available literature written on this topic and in other related areas. This will be made possible by the identification, collection, and review of this literature from various sources such as textbooks, journals, reports, and the internet.

In today's interconnected world, ensuring the security of both wired and remote organizations is of utmost importance [30]. The exchange of data necessitates a strong focus on organizational security. To achieve this objective, numerous innovative implementations and security measures have been developed. Rather than the knowledge transferred, the primary concern lies in the level of security offered by the communication channel during data transmission. This technology enables the secure transfer of data while ensuring confidentiality and reliable encryption. Remote organizations are susceptible to various attacks due to their open design, dynamic geographical nature, and lack of a physical perimeter. Wired and remote organizations can be targeted by different types of attacks, including ciphertext attacks, brute force attacks, known-plaintext attacks, denial of service attacks, side-channel attacks, and more. Implementing diverse cryptographic techniques is crucial to safeguarding customer information and mitigating these types of attacks. Cryptography plays a pivotal and indispensable role in achieving optimal security. Its purpose is to establish a secure, robust, and long-lasting connection while safeguarding data integrity [31].

2.2. Defining Cryptography

This research focuses on the practical implementation of Cryptography and its empirical results. Therefore, it is important to consider definitions that are practical and can be found in both scientific and non-scientific literature. Table 2.1 presents a summary of different definitions of Cryptography.

Table 2.1: Some definitions of Cryptography.

Authors	**Definitions/Conceptualization**
Bruce Schneier [9]	*"Cryptography is the science of secret communication, and its goal is to provide confidentiality, integrity, and authenticity of information".*
William Stallings [32]	*"Cryptography is the practice and study of techniques for secure communication in the presence of adversaries. It encompasses encryption, decryption, and related techniques that are used to protect*

	information from unauthorized access or modification."
Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone [33]	*"Cryptography is the mathematical science of secret writing. It involves transforming messages to make them secure and immune to attacks. Cryptographic techniques can be used to ensure privacy, integrity, authentication, and non-repudiation in various applications*."
Oded Goldreich [34]	*"Cryptography deals with the secure transmission of information in the presence of adversaries. It encompasses the design, analysis, and implementation of methods and protocols for secure communication."*
John F. Dooley [35]	*"Cryptography is the science of encoding and decoding messages so that they remain secure during transmission and storage. It involves the use of mathematical algorithms and keys to transform plaintext into ciphertext and vice versa."*

Source: Author's Table

These are just a few examples, and different authors may have slightly different perspectives or emphasize different aspects of cryptography. However, they all generally recognize cryptography as a field that focuses on secure communication, protection of information, and the use of mathematical techniques to achieve confidentiality, integrity, and authenticity.

2.3. Objectives of Cryptography

The objectives of cryptography revolve around ensuring the secure transmission and storage of information. Here are the main objectives of cryptography [36][37]:

- Confidentiality: One of the primary objectives of cryptography is to provide confidentiality or privacy. It ensures that information remains secret and can only be accessed by authorized individuals. By encrypting data, cryptography transforms it into a format that is unintelligible to unauthorized parties. Only those with the correct decryption key can decipher the encrypted message and access the original information.

- Integrity: Cryptography aims to maintain the integrity of data, ensuring that it remains unaltered during transmission or storage. By using techniques such as digital signatures or message authentication codes (MACs), cryptography can detect any unauthorized modifications or tampering with the data. If any changes are detected, it indicates that the data may have been compromised or corrupted.

- Authentication: Cryptography provides mechanisms for authentication, which verify the identity of the communicating parties. Digital signatures, based on public key cryptography, allow the recipient to verify the sender's identity and ensure that the message has not been tampered with during transmission. Authentication ensures that the information received is from a trusted source and has not been modified by unauthorized entities.

- Non-repudiation: Non-repudiation is the ability to prevent the sender of a message from denying their involvement in the communication. Cryptographic techniques, such as digital signatures, provide a means to establish non-repudiation. A digital signature provides proof of the sender's identity and ensures that they cannot later deny sending the message.

- Key Management: Effective key management is a crucial objective of cryptography. Cryptographic algorithms rely on the use of keys for encryption and decryption. Secure generation, distribution, storage, and revocation of cryptographic keys are essential to maintain the overall security of cryptographic systems.

- Accessibility: Cryptography also aims to ensure that authorized individuals can access the encrypted information efficiently. It provides mechanisms for secure key exchange or sharing between communicating parties, allowing them to encrypt and decrypt data as needed.

Overall, the objectives of cryptography are to protect the confidentiality, integrity, and authenticity of information, establish trust and authenticity between communicating parties, prevent unauthorized access or modifications, and provide reliable mechanisms for secure communication and data storage. By achieving these objectives, cryptography plays a crucial role in ensuring information security in various domains such as communication networks, e-commerce, financial transactions, and data protection.

2.4. Classification of encryption

In Figure 2.1, the classification methods of encryption for ciphers are depicted. The classification is broadly divided into two main categories: Classical Encryption and Modern Encryption. Within these categories, there are two subcategories. Classical Encryption is comprised of Substitution and Transposition techniques, while Modern Encryption is categorized based on the usage of the key, namely Secret Key (Symmetric) and Public Key (Asymmetric). Substitution is further sub-divided into Mono-alphabetic and Polyalphabetic methods [38].

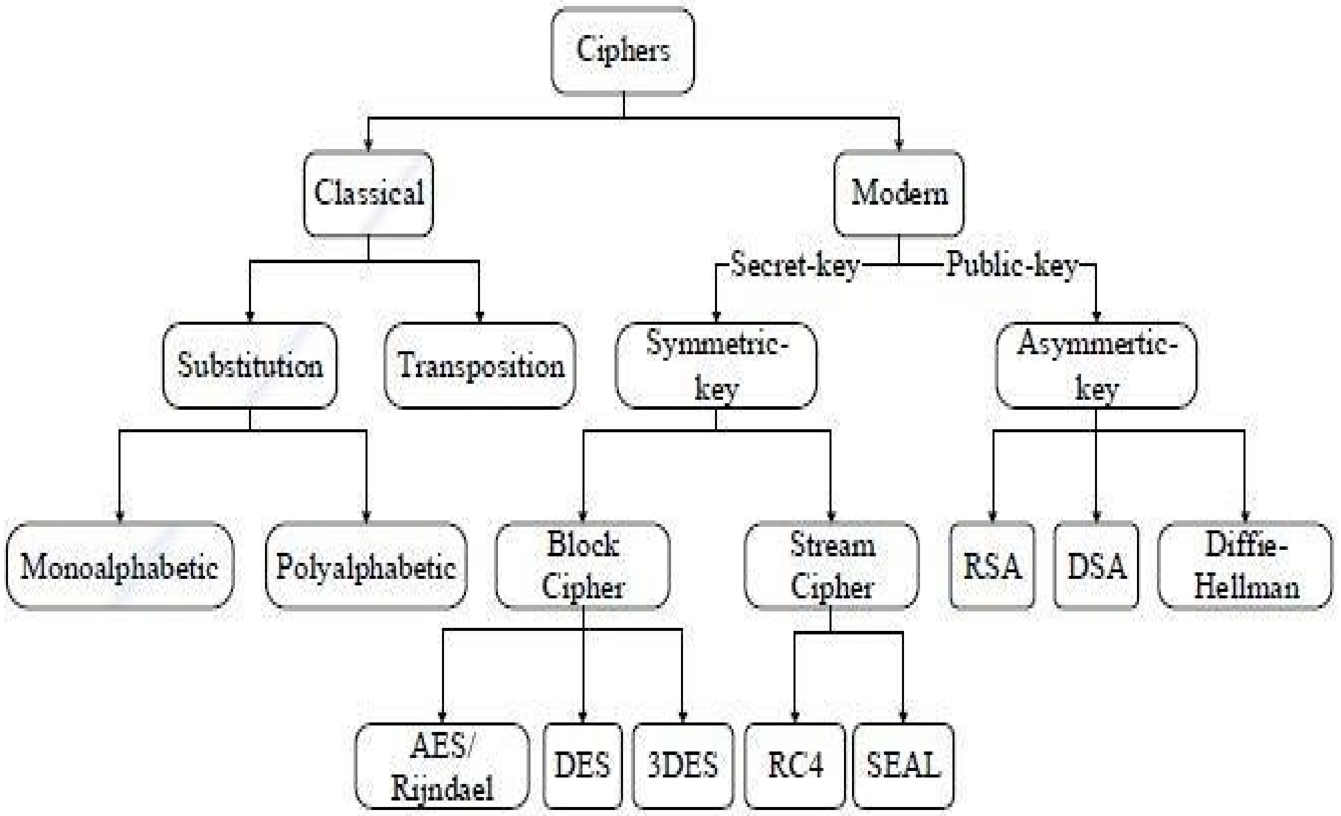


Figure 2.1: Classification of Encryption Methods.

2.5. Classical ciphers

Classical ciphers refer to encryption techniques that were developed and used before the advent of modern cryptographic methods. These ciphers have a long history and were widely used for secure communication in various contexts. Here are some of the most common classical ciphers [35]:

Caesar Cipher: The Caesar cipher is one of the simplest and earliest known substitution ciphers. It involves shifting the letters of the alphabet by a fixed number of positions. For example, with a shift of 3, 'A' would be encrypted as 'D', 'B' as 'E', and so on. It is a type of mono-alphabetic substitution cipher as each letter is substituted with a different letter of the alphabet [39].

Vigenère Cipher: The Vigenère cipher is a polyalphabetic substitution cipher that builds upon the Caesar cipher. It uses a keyword to determine the amount of shift applied to each letter. The keyword is repeated until it matches the length of the plaintext. This makes it more resistant to frequency analysis attacks compared to mono-alphabetic cipher [40].

Playfair Cipher: The Playfair cipher is a digraph substitution cipher that uses a 5x5 matrix of letters. It operates on pairs of letters (digraphs) in the plaintext, replacing them with corresponding digraphs from the matrix. The key is used to determine the positions of the letters in the matrix. The Playfair cipher offers stronger encryption than simple substitution ciphers [35].

2.6. Transposition ciphers

Transposition ciphers offer a different level of security compared to substitution ciphers. They can be effective in hiding the original message's structure and patterns, making it challenging for cryptanalysts to analyze and decipher the ciphertext. However, transposition ciphers do not change the characters themselves, which means that the frequency distribution and statistical properties of the original message may still be present in the ciphertext. This vulnerability can be exploited by skilled cryptanalysts to break the cipher [41].

There are various techniques for implementing transposition ciphers, each with its own method of rearranging the characters. Some common transposition cipher techniques include:

Columnar Transposition: This method involves writing the message in a grid with a fixed number of columns and then reading the ciphertext by following a specific column order. The order of the columns is determined by a key or permutation rule.

Rail Fence: The Rail Fence cipher arranges the characters of the message diagonally over a set number of "rails" or lines. The ciphertext is obtained by reading off the characters in a zigzag pattern along the rails.

Route Cipher: Route ciphers involve systematically moving the characters of the message through predetermined routes or paths. The order and direction of the routes dictate the rearrangement of the characters.

Scytale: The Scytale cipher is an ancient technique that involves wrapping a strip of paper around a cylinder of a specific diameter and then writing the message along the strip. The ciphertext is obtained by reading the characters in a specific pattern as the strip is unwound.

Transposition ciphers can be used in combination with other encryption techniques to enhance the security of a communication system. While they may not provide the same level of cryptographic strength as modern encryption algorithms, transposition ciphers have historical significance and serve as educational tools for understanding the principles of cryptography [35].

2.8. Modern ciphers

Modern ciphers can be classified into two categories based on various factors, such as the techniques used, the keys employed, and the intended applications [42]

1. Symmetric Key Cryptography: Also known as secret key cryptography, symmetric key cryptography uses a single secret key for both encryption and decryption. The same key is shared between the sender and the receiver. Symmetric key algorithms are generally faster and more efficient than other types of algorithms. However, the main challenge lies in securely exchanging the secret key between the communicating parties. Examples of symmetric key algorithms include the Data Encryption Standard (DES), Advanced Encryption Standard (AES), and Triple Data Encryption Standard (3DES) [27][23].

2. Asymmetric Key Cryptography: Asymmetric key cryptography, also called public key cryptography, employs a pair of keys - a public key and a private key. The public key is freely available to anyone, while the private key is kept secret by the owner. Messages encrypted with the public key can only be decrypted with the corresponding private key, and vice versa. Asymmetric key cryptography addresses the key exchange problem faced in symmetric key cryptography. It provides a solution for secure communication and digital signatures. The most widely used asymmetric key algorithm is the RSA algorithm, while others include Diffie-Hellman, Elliptic Curve Cryptography (ECC), and Digital Signature Algorithm (DSA) [43][44].

2.9. Types of Symmetric Algorithms

Symmetric encryption algorithms operate on the principle of using a single secret key for both encryption and decryption. These algorithms can be classified into two main types: block ciphers and stream ciphers [45][46].

2.9.1. Block Ciphers

Block ciphers divide the plaintext into fixed-size blocks and encrypt each block independently using the secret key. The most common block cipher is the Advanced Encryption Standard (AES). AES operates on fixed-size blocks of 128 bits and supports key sizes of 128, 192, and 256 bits. Other examples of block ciphers include Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES).

In block ciphers, the encryption and decryption processes are typically implemented using rounds of mathematical operations such as substitution, permutation, and key mixing. Each block is transformed using these operations in a series of rounds, and the final output is the encrypted ciphertext. The same process is followed in reverse during decryption to obtain the original plaintext.

Block ciphers are highly secure and provide strong encryption, making them suitable for a wide range of applications. They are particularly efficient when processing large amounts of data in a batch mode. However, block ciphers require padding when the input message length is not an exact multiple of the block size. The use of modes of operation, such as Cipher Block Chaining (CBC) or Counter (CTR), allows for the encryption of messages longer than a single block.

2.9.1.1. Advanced Encryption Standard (AES)

The AES (Advanced Encryption Standard) algorithm is a symmetric block cipher used for encrypting and decrypting sensitive data. It is widely considered to be one of the most secure encryption algorithms available today. The AES algorithm operates on fixed-size blocks of data and uses a secret key to perform the encryption and decryption processes. Here is an overview of the AES algorithm [47]:

1. Key Sizes: AES supports three key sizes: 128 bits, 192 bits, and 256 bits. The key size determines the level of security and the number of rounds performed during the encryption process.

Substitution-Permutation Network (SPN): AES uses a substitution-permutation network structure, which consists of multiple rounds of substitution and permutation operations. In each round, four different transformations are applied to the data: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

2. SubBytes Transformation: The SubBytes transformation replaces each byte in the input block with a corresponding value from an S-box lookup table. The S-box provides confusion and non-linearity to the algorithm.

3. ShiftRows Transformation: The ShiftRows transformation cyclically shifts the bytes in each row of the state matrix. This operation ensures that the data is spread out across different rows, enhancing the diffusion property of the algorithm.

4. MixColumns Transformation: The MixColumns transformation operates on the columns of the state matrix, treating each column as a polynomial over the Galois field. This transformation provides additional diffusion and strengthens the encryption algorithm.

5. AddRoundKey Transformation: The AddRoundKey transformation XORs each byte of the state matrix with a round key derived from the main encryption key. The round key is generated using a key expansion algorithm.

6. Key Expansion: The AES key expansion algorithm generates a set of round keys from the main encryption key. These round keys are used in each round of the encryption and decryption processes. The number of rounds depends on the key size: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

7. Security and Performance: The AES algorithm has been extensively analyzed by cryptographers, and no practical vulnerabilities have been discovered when used with recommended key sizes. AES provides a high level of security against various types of attacks, including brute-force attacks. It also offers good performance and can be efficiently implemented in both software and hardware.

8. Standardization: AES was selected by the U.S. National Institute of Standards and Technology (NIST) in 2001 after a public competition to replace the aging Data Encryption Standard (DES). AES has since become a global standard and is widely used in various applications and protocols requiring secure encryption [48].

The AES algorithm has been thoroughly vetted and is trusted by governments, organizations, and individuals worldwide. Its strength, efficiency, and wide adoption make it a cornerstone of modern cryptography, ensuring the confidentiality and integrity of sensitive data [49].

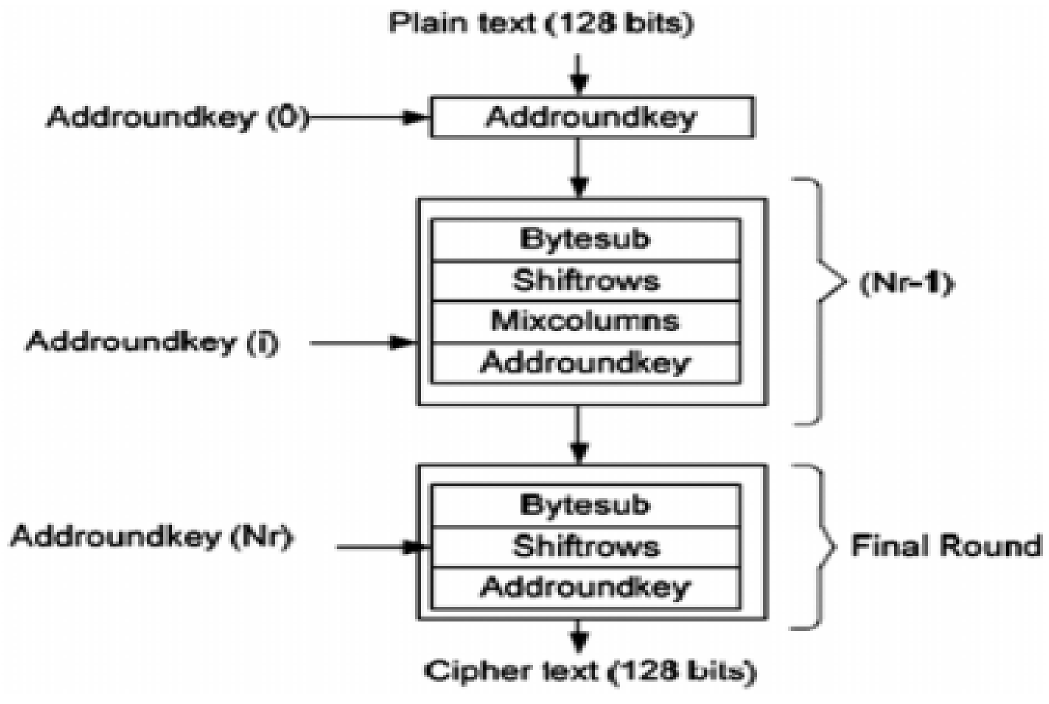


Figure 2.2: Structure of AES [61].

2.9.1.2. Blowfish algorithm

The Blowfish algorithm is a symmetric key block cipher designed by Bruce Schneier in 1993. It is known for its simplicity, security, and flexibility. Blowfish is a Feistel network cipher that operates on fixed-size blocks of data and uses a variable-length key, making it suitable for various applications [50][51].

Here are some key features and characteristics of the Blowfish algorithm:

1. Key Size: Blowfish supports variable key sizes from 32 bits to 448 bits. The key length can be any multiple of 8 bits, and it is used to initialize the subkeys during the key expansion phase.

2. Subkey Generation: Blowfish employs a key expansion algorithm to generate a series of subkeys from the original key. The subkeys are derived using a complex function that combines the key material with elements of the algorithm's internal state.

3. Feistel Network: Blowfish follows a Feistel network structure, where the data is divided into two halves, and a series of rounds are performed on these halves. In each round, one half is subjected to a function that depends on the current round's subkey and the other half is XORed with the output of the function. The halves are then swapped, and the process is repeated for the specified number of rounds.

4. Substitution and XOR Operations: Blowfish employs both substitution and XOR operations to provide confusion and diffusion. It uses large substitution boxes (S-boxes) to perform byte-level substitutions. The S-boxes are initialized with a fixed set of predefined values during the key setup phase.

5. Variable Number of Rounds: Blowfish allows for a variable number of rounds, typically ranging from 16 to 32 rounds. More rounds increase the security but also increase the computational overhead.

6. Efficient Implementation: Blowfish is known for its efficiency in both software and hardware implementations. It does not require complex arithmetic operations, making it relatively fast on various platforms.

7. Security: Blowfish is considered secure and has withstood extensive analysis over the years. No practical attacks have been discovered that would compromise the algorithm's security when used with appropriate key lengths.

8. Applications: Blowfish has been widely used in various applications, including secure file storage, virtual private networks (VPNs), password hashing, and data encryption in software products. Its flexibility in supporting different key sizes makes it suitable for a range of security requirements.

While Blowfish has been widely adopted and remains a respected encryption algorithm, it has been largely superseded by more recent algorithms such as AES (Advanced Encryption Standard) due to its limited block size and the availability of more efficient alternatives.

Overall, Blowfish is a well-regarded symmetric key encryption algorithm known for its simplicity, flexibility, and security. Its legacy continues to influence the field of cryptography and serves as the foundation for further advancements in encryption techniques [52].

2.9.1.3. Data Encryption Standard

DES (Data Encryption Standard) is a symmetric key block cipher algorithm used for encryption and decryption of data. It was developed in the 1970s by IBM and later adopted as a standard by NIST (National Institute of Standards and Technology). Here is an explanation of how DES cryptography works [53]:

Key Generation: The DES algorithm uses a 64-bit key, but only 56 bits are used for encryption, with the remaining 8 bits reserved for parity checks.

The key undergoes a key scheduling process to generate 16 subkeys. Each subkey is 48 bits long and is derived from the original key.

Encryption Process: The plaintext to be encrypted is divided into 64-bit blocks. The initial permutation (IP) is applied to the plaintext block, rearranging the bits. The plaintext block is then divided into two 32-bit halves: the left half (L0) and the right half (R0). The encryption process consists of 16 rounds, where each round applies a series of operations to the data. In each round: The right half (Ri-1) is expanded to 48 bits using a permutation operation. The expanded right half is XORed with the current subkey (Ki) derived from the key scheduling process. The XORed result goes through eight substitution operations using S-boxes, which provide non-linear transformations. The output of the S-boxes is permuted using a fixed P-box permutation. The permuted output is XORed with the left half (Li-1). The left and right halves are swapped, and the process continues for the next round. After the 16 rounds, the final left and right halves are swapped.

Decryption Process: The decryption process is similar to encryption but uses the subkeys in reverse order. The ciphertext is divided into blocks of the same size (64 bits). The input block goes through the same series of rounds as in encryption, but with the subkeys applied in reverse order. After the final round, the left and right halves are swapped.

Finalization: The final ciphertext undergoes a final permutation (IP-1) to obtain the encrypted data.

The strength of DES cryptography lies in the complexity of its algorithm, which involves a combination of substitution, permutation, and XOR operations. However, advances in computing power have made DES vulnerable to brute-force attacks. To enhance security, Triple-DES (3DES) is often used, which applies the DES algorithm three times with different keys.

The DES has been largely replaced by the Advanced Encryption Standard (AES), which provides stronger security and supports larger key sizes. Nonetheless, DES is still used in certain legacy systems and applications [54][55].

2.9.1.4. Triple Data Encryption Standard (3DES)

Triple Data Encryption Standard (3DES) is a symmetric key block cipher algorithm that provides increased security by applying the Data Encryption Standard (DES) algorithm multiple times. It is also known as TDEA (Triple Data Encryption Algorithm) [56]. Here is an overview of 3DES [57]:

Key Generation: 3DES uses three 56-bit keys (168 bits in total). These keys are referred to as Key1, Key2, and Key3. The three keys undergo a key scheduling process to generate subkeys for each encryption round.

Encryption Process: The plaintext to be encrypted is divided into blocks of 64 bits. The encryption process consists of three stages: encryption with Key1, decryption with Key2, and encryption again with Key3. In each stage, the block undergoes the same process as the standard DES algorithm, which involves an initial permutation, 16 rounds of operations, and a final permutation. Each stage uses a different key and applies the DES algorithm with the key in the corresponding direction.

Decryption Process: The decryption process is the reverse of the encryption process. The ciphertext is divided into blocks of 64 bits. The decryption process consists of three stages: decryption with Key3, encryption with Key2, and decryption with Key1. Each stage uses the respective key and applies the DES algorithm in the reverse direction.

Keying Options: 3DES supports different keying options, providing flexibility in the choice and management of keys: Keying Option 1: Each of the three keys (Key1, Key2, and Key3) is independent, providing the highest security level. Keying Option 2: Key1 and Key2 are identical, while Key3 is different. This option offers backward compatibility with single DES.

Keying Option 3: All three keys (Key1, Key2, and Key3) are identical, providing compatibility with single DES while offering a longer key length.

3DES improves the security of the original DES algorithm by increasing the effective key length to 168 bits. It offers a higher level of resistance against brute-force attacks due to the larger key space. However, it is slower compared to modern encryption algorithms such as AES due to the repeated application of the DES algorithm [58].

With the advancement of encryption standards, 3DES is gradually being replaced by more efficient and secure algorithms like AES. However, 3DES is still widely used in legacy systems and applications where compatibility with older implementations is required [59].

2.9.1.5. Twofish Algorithm

Twofish is a symmetric key block cipher algorithm that was developed as a candidate for the Advanced Encryption Standard (AES) selection process. It was designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish is known for its strong security, flexibility, and efficient performance [60].

Key Features of Twofish: Key Size Flexibility: Twofish supports key sizes of 128, 192, and 256 bits, allowing users to choose the desired level of security based on their specific needs.

Block Cipher Operation: Twofish operates on fixed-size blocks of data, typically 128 bits. The input data is divided into blocks, and each block is encrypted or decrypted independently.

Feistel Network Structure: Twofish utilizes a Feistel network structure, which divides the input block into two halves and applies a series of rounds to each half. This structure ensures that the encryption and decryption processes are symmetric.

Round Functions: Twofish employs a combination of substitution and permutation operations in its round functions. It utilizes S-box lookups, key-dependent permutations, and bitwise XOR operations to provide strong encryption.

Key Schedule: Twofish uses a key schedule algorithm to generate a set of round subkeys from the original encryption key. The key schedule involves mixing and expanding the key material to create round keys for each encryption round.

Avalanche Effect: Twofish is designed to exhibit the avalanche effect, where even a small change in the input or key results in a significant change in the output. This property enhances the algorithm's security and makes it resistant to various cryptographic attacks.

Security Strength: Twofish has undergone extensive analysis and evaluation by the cryptographic community. It is considered to be highly secure and resistant to known attacks, such as differential and linear cryptanalysis.

Performance Optimization: Twofish is optimized for efficient implementation on various computing platforms. It strikes a balance between security and performance, ensuring that encryption and decryption operations can be performed efficiently.

Wide Application: Twofish has been widely adopted and implemented in various software and hardware products. It is used in applications that require strong encryption, including secure communications, data storage, and file encryption [61].

Twofish provides a high level of security, flexibility, and performance, making it a popular choice for encryption in a wide range of applications. Its strong cryptographic properties and efficient implementation make it suitable for protecting sensitive data and ensuring secure communication [62].

2.9.2. Stream Ciphers:

Stream ciphers encrypt plaintext by processing it one bit or one byte at a time, generating a stream of encrypted output. The encryption process is typically performed by combining the plaintext with a keystream generated by a secret key. The keystream is a sequence of random or pseudo-random values, which is combined with the plaintext using an exclusive OR (XOR) operation to produce the ciphertext. Stream ciphers are often designed to be highly efficient and can encrypt and decrypt data in real-time, making them suitable for applications with continuous data streams such as voice and video communication [63]. However, stream ciphers can be more susceptible to certain types of attacks, such as known-plaintext attacks, if the same key is reused or if the keystream generator is compromised. One widely used stream cipher is the Rivest Cipher 4 (RC4), which is known for its simplicity and speed. However, due to security vulnerabilities, RC4 is no longer recommended for new applications. Other stream ciphers include the eSTREAM portfolio, which consists of several stream cipher algorithms that have undergone extensive analysis and evaluation [64].

In summary, block ciphers encrypt fixed-size blocks of plaintext independently, while stream ciphers encrypt data bit by bit or byte by byte. Both types of symmetric algorithms play a crucial role in providing secure communication and data protection, each with their own strengths and considerations for different applications and use cases [65][66].

2.9.2.1. RC4 (Rivest Cipher 4)

RC4 (Rivest Cipher 4) is a stream cipher algorithm that was designed by Ron Rivest in 1987. It gained popularity due to its simplicity, speed, and versatility. Initially, RC4 was a trade secret, but it eventually became widely known and used in various applications. However, over time, several security vulnerabilities were discovered in RC4, and it is now considered insecure for most purposes [67][68].

Key Features of RC4 [64]:

Key Setup: RC4 operates by generating a pseudorandom stream of key-dependent bytes. To set up the algorithm, a secret key of variable length (typically between 40 and 256 bits) is used. The key serves as the input to the key setup algorithm, which expands it into a fixed-size internal state (256 bytes) using a process called the key-scheduling algorithm (KSA).

Pseudorandom Generation Algorithm (PRGA): The PRGA is the core of RC4, responsible for generating the stream of pseudorandom bytes. It utilizes the internal state, which is initially filled with values from 0 to 255 in order, and scrambles it based on the key. The PRGA generates a keystream of pseudorandom bytes by continually swapping and updating the internal state.

Encryption and Decryption: To encrypt or decrypt data, RC4 uses the keystream generated by the PRGA. The algorithm XORs each byte of the plaintext (or ciphertext) with a corresponding byte from the keystream. XORing the bytes combines the properties of both the plaintext and the pseudorandom stream, creating the ciphertext (or recovering the plaintext).

Security Concerns: RC4 has suffered from several security vulnerabilities that have compromised its strength over time. These vulnerabilities include biases in the keystream output and key-dependent biases in the internal state, leading to attacks such as the Fluhrer-McGrew and the Mantin-Shamir attacks. As a result, the use of RC4 in new cryptographic applications is generally discouraged.

Historical Significance: Despite its vulnerabilities, RC4 played a significant role in the development of modern encryption algorithms. It influenced the design of other stream ciphers and provided insights into the properties of secure encryption algorithms. Its impact can be seen in the development of later algorithms such as Salsa20 and ChaCha20.

RC4 is a stream cipher algorithm that generates a pseudorandom stream of bytes based on a secret key. It was widely used in various applications due to its simplicity and speed. However, over time, multiple vulnerabilities were discovered in RC4, making it unsuitable for secure encryption [69].

2.9.2.2. Salsa20

Salsa20 is a symmetric key stream cipher algorithm designed by Daniel J. Bernstein in 2005. It is known for its simplicity, high performance, and strong security properties. Salsa20 is widely used in various applications that require secure and efficient encryption [70].

Key Features of Salsa20 [71]:

Stream Cipher: Salsa20 is a stream cipher algorithm, which means it encrypts data on a byte-by-byte basis using a pseudorandom stream of key-dependent bytes. The algorithm generates a keystream based on a secret key and a nonce (number used once).

Security Strength: Salsa20 is designed to provide a high level of security. It has been extensively analyzed and found to have strong resistance against various cryptanalytic attacks. The algorithm is believed to provide robust security when implemented correctly.

Variable Key Size: Salsa20 supports key sizes of 128, 192, or 256 bits. This flexibility allows users to select an appropriate key size based on their security requirements and performance constraints.

ChaCha Variant: Salsa20 is closely related to the ChaCha cipher, which is another stream cipher algorithm also developed by Daniel J. Bernstein. ChaCha is a modification of Salsa20 that incorporates a different permutation function, resulting in improved diffusion and performance characteristics.

Quarterround Operation: The core operation in Salsa20 is the quarterround function, which operates on a state matrix consisting of 16 32-bit words. The quarterround function performs a series of bitwise operations and additions, creating a nonlinear and diffusion effect that enhances the security properties of the algorithm.

Block Generation: Salsa20 generates blocks of keystream by repeatedly applying the quarterround operation and updating the state matrix. The algorithm employs a counter-based approach, incrementing a portion of the state matrix to generate a new block of the keystream.

Nonce and Initialization: Salsa20 uses a nonce as an additional input to the algorithm. The nonce ensures that the same key can be used multiple times while producing unique keystreams. To prevent nonce reuse, it is crucial to use a different nonce for each encryption session.

Performance and Efficiency: Salsa20 is designed for high performance and efficiency. It takes advantage of modern processor features, such as efficient bit-level and parallel operations, allowing it to achieve fast encryption and decryption speeds.

Use in Cryptographic Protocols: Salsa20 is utilized in various cryptographic protocols and applications, including disk encryption, secure messaging, and virtual private networks (VPNs). Its combination of security, speed, and simplicity makes it an attractive choice for these scenarios.

Salsa20 is a versatile and efficient stream cipher algorithm that provides strong security properties. Its simplicity, high performance, and resistance to cryptanalytic attacks have contributed to its popularity and adoption in numerous cryptographic applications [70].

2.9.2.3. ChaCha20

ChaCha20 is a symmetric key stream cipher algorithm designed by Daniel J. Bernstein in 2008. It is an improved version of the Salsa20 stream cipher, offering increased security and performance. ChaCha20 has gained significant popularity and is widely used in various applications, particularly in the field of secure communications [72].

Key Features of ChaCha20 [73][74]:

Stream Cipher: ChaCha20 is a stream cipher algorithm that encrypts data on a byte-by-byte basis using a pseudorandom stream of key-dependent bytes. Like other stream ciphers, it generates a keystream based on a secret key and a nonce (number used once).

Security Strength: ChaCha20 is designed to provide strong security. It has undergone rigorous analysis and has demonstrated resilience against various cryptanalytic attacks. The algorithm is considered secure when implemented correctly.

Variable Key Size: ChaCha20 supports key sizes of 128, 256, or 512 bits. This flexibility allows users to select an appropriate key size based on their security requirements and performance considerations.

Quarterround Operation: Similar to Salsa20, ChaCha20 employs the quarterround function as its core operation. The quarterround function applies a series of bitwise operations and additions to create a nonlinear and diffusion effect, enhancing the security properties of the algorithm.

Block Generation: ChaCha20 generates blocks of keystream by iteratively applying the quarterround operation and updating the state matrix. The algorithm employs a counter-based approach, incrementing a portion of the state matrix to generate a new block of the keystream.

Nonce and Initialization: ChaCha20 uses a nonce to ensure unique keystream generation for each encryption session. It is essential to use a different nonce for each encryption to prevent nonce reuse and maintain security.

ChaCha20-Poly1305: ChaCha20 is often combined with the Poly1305 authenticator to form the ChaCha20-Poly1305 construction. This combination provides both encryption and authentication, making it suitable for secure communications protocols such as Transport Layer Security (TLS).

Performance and Efficiency: ChaCha20 is designed for high performance and efficiency. It takes advantage of modern processor features, such as parallel operations and efficient bit-level manipulation, allowing for fast encryption and decryption speeds.

Adoption and Standardization: ChaCha20 has gained significant recognition and adoption in the field of cryptography. It is widely used in various applications, including secure messaging platforms, virtual private networks (VPNs), disk encryption, and internet protocols. ChaCha20 has been standardized by the Internet Engineering Task Force (IETF) for use in cryptographic protocols.

ChaCha20 is known for its simplicity, security, and efficiency. Its combination of strong encryption, fast performance, and flexibility in key sizes has made it a popular choice in the world of secure communications. The algorithm's design and its widespread adoption have contributed to its reputation as a reliable and effective stream cipher [75][76].

2.10. Asymmetric algorithms

Asymmetric algorithms, also known as public-key algorithms, are cryptographic algorithms that use two different keys for encryption and decryption [77]. Here are the main types of asymmetric algorithms:

2.10.1. RSA Algorithm

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric encryption algorithm that provides secure communication, digital signatures, and key exchange. It is named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman, who introduced the algorithm in 1977 [78].

Key Components of RSA [79][80][81]:

Key Generation: RSA involves the generation of a public-private key pair. The key generation process starts by selecting two large prime numbers, p and q. The product of these primes, n (n = p * q), is used as the modulus for encryption and decryption. The public key consists of the modulus n and an exponent e, while the private key includes the modulus n and another exponent d.

Encryption: To encrypt a message using RSA, the plaintext is first converted into a numerical representation. Each block of the plaintext is encrypted using the recipient's public key. The encryption operation is performed by raising the plaintext block to the power of the public exponent e and then taking the modulus n of the result. The ciphertext, a numerical representation of the encrypted message, is obtained.

Decryption: Decryption in RSA is performed using the recipient's private key. The encrypted ciphertext is raised to the power of the private exponent d and then reduced modulo n. This operation recovers the original plaintext message.

Key Exchange: RSA can be used for secure key exchange between two parties. One party generates their public-private key pair and shares the public key with the other party. The other party uses the received public key to encrypt a shared secret key, which is then sent back to the first party. The first party can decrypt the received ciphertext using their private key to obtain the shared secret key. Both parties now possess the same shared secret key for secure communication.

Digital Signatures: RSA can be used to provide digital signatures, which ensure the authenticity and integrity of digital messages. To create a digital signature, the sender uses their private key to encrypt a hash value of the message. The recipient can then verify the signature by decrypting it using the sender's public key and comparing the decrypted hash value with the hash value of the received message.

Key Strength and Security: The security of RSA relies on the difficulty of factoring large composite numbers into their prime factors. Breaking RSA encryption requires an attacker to factor the modulus n, which becomes increasingly challenging as the key size grows. RSA's security is directly linked to the length of the key, with longer keys offering higher levels of security.

Applications [82][83][84]:

RSA is widely used in various applications, including: Secure communication over the internet, such as HTTPS, SSL/TLS.

- Digital signatures for ensuring message integrity and authentication.
- Key exchange protocols, like Diffie-Hellman key exchange.
- Secure email communication using PGP (Pretty Good Privacy).
- Secure file transfer and encryption in various software and protocols.

Despite its popularity and extensive use, RSA can be computationally expensive, especially for large key sizes. Therefore, newer asymmetric algorithms like elliptic curve cryptography (ECC) have gained popularity due to their similar security strength with smaller key sizes and faster computation [85].

RSA remains a crucial and widely adopted algorithm in the field of cryptography, providing a foundation for secure communication and data protection [86].

2.10.2. ECC Algorithm

ECC (Elliptic Curve Cryptography) is an asymmetric encryption algorithm that is gaining popularity due to its strong security with shorter key lengths compared to other public-key algorithms. It is based on the mathematics of elliptic curves over finite fields. ECC provides efficient and secure cryptographic operations, making it well-suited for resource-constrained devices and bandwidth-limited environments [87].

Key Components of ECC [5][88]:

Elliptic Curves: ECC utilizes elliptic curves defined by an equation in the form of y^2 = x^3 + ax + b, where a and b are constants. The curve's points form an additive group, and operations such as point addition and scalar multiplication are defined on the curve.

Key Generation: ECC involves the generation of a public-private key pair. The key generation process begins with selecting an elliptic curve and a base point on that curve. The base point's coordinates are typically provided as parameters. The private key is a random number within a specific range. The public key is derived by multiplying the base point by the private key using scalar multiplication.

Encryption: In ECC, encryption is typically not performed directly on the plaintext message. Instead, ECC is often used for key agreement protocols, such as the Elliptic Curve Diffie-Hellman (ECDH) algorithm. ECDH allows two parties to establish a shared secret key over an insecure channel without explicitly transmitting the key. The shared secret key can then be used for symmetric encryption of the actual message.

Digital Signatures: ECC is also used for digital signatures, similar to other asymmetric algorithms. The Elliptic Curve Digital Signature Algorithm (ECDSA) is commonly employed for generating and verifying digital signatures. ECDSA involves creating a signature by using the signer's private key to perform mathematical operations on the message. The signature can be verified using the signer's public key and the received message.

Key Strength and Security: The security of ECC relies on the difficulty of the elliptic curve discrete logarithm problem. Breaking ECC encryption requires solving this problem, which is believed to be computationally infeasible, especially for properly chosen curve parameters and sufficiently large key sizes. ECC offers comparable security to traditional algorithms like RSA but with shorter key lengths, providing advantages in terms of computational efficiency and memory usage [89].

Applications:

ECC has gained popularity and is utilized in various applications, including:

- Secure communication protocols like TLS/SSL.
- Wireless communication standards such as Wi-Fi and Bluetooth.
- Smart cards, RFID tags, and other embedded systems.
- Cryptocurrency systems like Bitcoin, which uses ECC for digital signatures.
- Internet of Things (IoT) devices with limited resources.

ECC's ability to provide strong security with shorter key lengths makes it a valuable tool for securing communications and protecting sensitive data in diverse domains. However, it is crucial to use well-vetted elliptic curves and follow best practices to ensure the security of ECC implementations [90].

2.11. The Mathematics of Cryptography

Cryptography is an ancient practice that has evolved over time to protect sensitive information from unauthorized access. With advancements in technology and computing, modern cryptographic systems have become more complex and sophisticated. Mathematics plays a crucial role in the design and analysis of secure cryptographic systems [91].

The history of cryptography can be traced back to early civilizations such as Egypt, Greece, and Rome. In those times, simple substitution ciphers were used, where letters were replaced with other letters or symbols. As printing technology emerged in the 15th century, more advanced techniques like polyalphabetic ciphers were developed [92].

Contemporary encryption systems are built on mathematical concepts such as algebra, probability theory, and number theory. There are two major categories of cryptographic systems: symmetric-key cryptography and public-key cryptography. In symmetric-key cryptography, the same key is used for both encryption and decryption. On the other hand, public-key cryptography employs two different keys for encryption and decryption. The security of cryptographic systems is based on the complexity of mathematical puzzles like factorization, discrete logarithms, and elliptic curve cryptography [93].

In modern society, cryptography serves various purposes including secure transactions, data storage, and secure communication. Popular cryptographic systems such as the Advanced Encryption Standard (AES), the RSA algorithm, and Elliptic Curve Cryptography (ECC) are widely used to protect sensitive data in databases, secure online conversations, and facilitate secure financial transactions. Mathematics provides the foundation for designing and analyzing secure cryptographic systems. Over time, simple substitution ciphers have been replaced by mathematically-based cryptographic systems. As the need for secure communication, data storage, and financial transactions continues to grow, cryptography remains indispensable in contemporary society [94].

This section focuses on the mathematical aspects of cryptographic algorithms, exploring their underlying principles, applications, and historical context.

2.11.1. Modular arithmetic

Modular arithmetic, a branch of mathematics, plays a significant role in encryption, particularly in the development of symmetric-key algorithms. It involves performing arithmetic operations on integers while taking the results modulo a fixed integer. The use of modular arithmetic in cryptography, its fundamentals, and applications will be explored in this literature survey.

Modular arithmetic is employed in cryptography to construct symmetric-key cryptographic techniques. The most commonly used operation in modular arithmetic is modular addition, which involves adding two numbers and then taking the result modulo a fixed integer. Additionally, modular subtraction, multiplication, and exponentiation are performed using modular arithmetic. Modular exponentiation, in particular, is crucial in encryption as it forms the basis of numerous cryptographic algorithms, including RSA and Diffie-Hellman [95].

Both symmetric-key cryptography and public-key cryptography leverage modular arithmetic in various ways. In symmetric-key cryptography, the encryption process involves applying modular arithmetic operations to the plaintext and a secret key to generate the ciphertext. The Advanced Encryption Standard (AES), the prevailing symmetric-key cryptographic algorithm, utilizes modular arithmetic to execute substitutions and permutations on both the plaintext and the key [32]. Modular arithmetic is a fundamental tool in cryptography, finding applications in both symmetric-key and public-key cryptographic methods. Through modular arithmetic, various operations are performed to ensure secure and efficient encryption. The Advanced Encryption

Standard is a prominent example of a symmetric-key algorithm that extensively utilizes modular arithmetic operations [96].

2.11.2. Modular Exponentiation

The use of modular exponentiation is a fundamental aspect of the mathematics of cryptography, specifically in encryption and decryption operations. When an integer is raised to a power modulo another integer, modular exponentiation involves computing the remainder [97]. This literature survey explores the significance of modular exponentiation in cryptography and provides an overview of the existing computational methods.

Among the commonly employed techniques, the square-and-multiply formula stands out. This algorithm is based on the concept that any exponentiation can be represented as a sequence of squaring and multiplying operations. By converting the exponent into its binary representation, the algorithm iteratively squares the base and multiplies it by itself when the corresponding bit in the exponent is 1. The time complexity of this approach is O(log n), where n represents the size of the exponent. In addition to the square-and-multiply algorithm, there are other strategies for optimizing modular exponentiation in specific cryptographic applications. For example, the Chinese Remainder Theorem (CRT) is often used to accelerate modular exponentiation in RSA cryptography. The CRT involves precomputing the exponentiation modulo the prime factors of the modulus and combining the results [98].

While modular exponentiation problems are generally considered computationally easy to solve, computing the discrete logarithm—finding the exponent e given b, c, and m—is deemed challenging. This property makes it suitable for cryptographic algorithms, where it can be referred to as a one-way function or a trapdoor function. Modular exponentiation is a fundamental operation in the mathematics of cryptography, and various techniques exist to enhance its efficiency in different cryptographic contexts. The Montgomery exponentiation algorithm is particularly advantageous for hardware implementations, while the square-and-multiply algorithm serves as a versatile method. Additionally, the CRT and windowed method are utilized in certain cryptographic applications to optimize modular exponentiation [99].

2.11.3. Reversible and quantum modular exponentiation

Reversible exponentiation and quantum exponentiation are two emerging fields of study within the mathematics of cryptography. Quantum exponentiation involves leveraging quantum algorithms to perform exponential operations, while reversible exponentiation focuses on calculating the inverse of a modular exponentiation. The research on reversible exponentiation has primarily taken place in the past decade, making it a relatively recent area of investigation. The main objective of reversible exponentiation is to enhance the security of cryptographic methods based on modular exponentiation, such as RSA and ElGamal. These algorithms are vulnerable to side-channel attacks, which exploit the electromagnetic radiation or power consumption of the computing device to extract secret keys. Modular exponentiation poses a challenge for Shor's algorithm, as it requires the computation of a circuit consisting of reversible gates that can be further decomposed into quantum gates suitable for a specific physical device. Moreover, Shor's algorithm allows knowledge of the base and exponentiation modulus at each call, enabling various optimizations to be applied to the circuit design [100].

2.11.4. Discrete logarithm

Discrete logarithms play a crucial role in the mathematical underpinnings of encryption. They are utilized in various cryptographic methods such as ElGamal encryption, Diffie-Hellman key exchange, and DSA digital signatures. This literature survey will explore different techniques for solving discrete logarithms and their implications for cryptography. The discrete logarithm problem involves finding the exponent x in the equation g^x ≡ h (mod p), where g, h, and p are known values. The Index Calculus method is a well-known classical algorithm for solving discrete logarithms. It leverages the observation that any element in a finite field can be represented as a product of a small set of prime elements. The Index Calculus algorithm is efficient for solving small prime discrete logarithms, with a time complexity of O(exp(sqrt(log(p) log(log(p))))). However, the Index Calculus method becomes impractical for larger primes, necessitating alternative approaches. One such approach is the Number Field Sieve algorithm, which is a general-purpose algorithm capable of solving various mathematical problems, including discrete logarithms. The Number Field Sieve method can handle discrete logarithms for primes of several hundred bits with a time complexity of O(exp((1.923 + o(1)) log(p)(1/3) (log(log(p)))(2/3)) [101].

2.11.5. Euclidean Algorithm

The Euclidean Algorithm, named after the Greek mathematician Euclid, is a fundamental mathematical algorithm that has found extensive applications in cryptography. Euclid developed this algorithm, which is described in his work "Elements." It is primarily used for determining the greatest common divisor (GCD) of two numbers and is widely utilized in various cryptographic techniques, including RSA encryption, public-key cryptography, and elliptic curve cryptography [102].

The Euclidean Algorithm, deeply rooted in number theory, follows a systematic approach to finding the GCD of two numbers. Its fundamental principle involves iteratively applying the concept that the GCD of two integers remains consistent when calculated for the smaller number and the remainder obtained by dividing the larger number by the smaller one. In the realm of cryptography, the Euclidean Algorithm finds practical application in key generation for RSA. The security of RSA relies on the challenge of factoring the product of two large prime numbers. In this process [103][104][105]:

- Distinct prime numbers p and q are chosen, and the modulus n is computed as $n=p\times q$.
- The totient $\phi(n)$ is determined as $\phi(n)=(p-1)\times(q-1)$.
- The public exponent e is chosen by applying the Euclidean Algorithm to find the GCD of $\phi(n)$ and e, ensuring $1<e<\phi(n)$ and $\gcd(\phi(n),e)=1$.

Furthermore, the Euclidean Algorithm contributes to private key generation in RSA:

- The private exponent d is computed, serving as the modular multiplicative inverse of e modulo $\phi(n)$. This ensures $(d\times e)\mathrm{mod}\phi(n)=1$.
- The extended Euclidean Algorithm is often employed for efficiently calculating the modular inverse.

Beyond key generation, the Euclidean Algorithm plays a pivotal role in security analysis. Its application in evaluating the security of cryptographic systems, especially in assessing the strength of employed keys, is paramount. The existence of an efficient algorithm for calculating the GCD could potentially introduce vulnerabilities in cryptographic systems, highlighting the algorithm's significance in ensuring the robustness of digital communication security.

The Euclidean Algorithm's application in cryptography, particularly in the RSA algorithm, emphasizes its crucial role in securing digital communication. Its involvement in key generation, modular arithmetic, and security analysis underscores its versatility and indispensability in contemporary cryptographic frameworks. As cryptographic methodologies advance, a profound understanding of foundational mathematical concepts like the Euclidean Algorithm remains essential for constructing resilient and secure digital communication channels [105][106].

2.12. Empirical Review

This section outlines the empirical review of the scholarly literature on the study objectives. The arguments are illustrated as below:

2.12.1. Review of the performance evaluation of commonly employed symmetric algorithms

According to authors in [24], a critical analysis was conducted to identify the strengths and weaknesses of various symmetric key cryptographic algorithms. The analysis in this paper examined important parameters such as throughput, scalability, security, memory usage, power consumption, speed, and flexibility to assess different cryptographic algorithms. The identified strengths and limitations of these algorithms make them suitable for various applications. Among the analyzed algorithms, Blowfish was found to excel in terms of security, flexibility, memory usage, and encryption performance. In addition, Tyagi and [25] conducted a comparative analysis of symmetric encryption algorithms, namely DES, 3DES, AES, and Blowfish. However, their study aimed to achieve specific objectives, which were: 1) gaining a deeper understanding of the cryptography process, and 2) performing a comparative analysis of symmetric encryption algorithms. Their study concluded that Blowfish outperformed other algorithms, such as DES, AES, and Triple DES, based on key size and security. The F function of the Blowfish algorithm provided a high level of security for encrypting 64-bit plaintext data. According to Nie et al., [107] evaluated the speed and power consumption of two symmetric key encryption algorithms, DES and Blowfish. The experimental results revealed that Blowfish algorithm exhibited faster speed than DES, while the power consumption remained almost the same. This study suggested that the Blowfish encryption algorithm may be more suitable for wireless network application security. In [108] provided a comparative survey on symmetric key encryption techniques. The analysis emphasized that selecting the appropriate encryption algorithm for encrypting plain text depends on weighing the advantages and disadvantages of each algorithm. The study indicated that symmetric key algorithms run faster than asymmetric key algorithms, such as RSA, and require lesser memory compared to asymmetric encryption algorithms. Furthermore, symmetric key encryption was deemed superior in terms of security compared to asymmetric key encryption. The comparison of popular encryption algorithms clearly demonstrated the superiority of the Blowfish algorithm over DES, AES, and Triple DES based on key size and security. This study evaluated six common encryption algorithms: AES (Rijndael), DES, 3DES, RC2, Blowfish, and RC6, focusing on their computing resource demands such as CPU time, memory, and battery power.

Blowfish was found to perform the best, followed by RC6, while 3DES had lower performance than DES. RC2 was the most time-consuming algorithm. AES outperformed RC2, DES, and 3DES. The results were consistent across different file types (audio, video, text, and documents). Additionally, changing the key size significantly impacted battery and time consumption. [109]. Similarly, the authors in [110] conducted an analysis to measure the performance of selected encryption algorithms. Their study confirmed that cryptographic algorithms consume a significant amount of computing resources, including CPU time, memory, and battery power. Based on the input size of text files and experimental results, it was concluded that the Blowfish algorithm consumed less execution time and memory usage while producing higher throughput. Specifically, Blowfish performed approximately four times faster than AES and two times faster than DES. The study highlighted that Blowfish not only exhibited exceptional speed but also provided strong security through its key size, making it suitable for various applications such as bulk encryption, random bit generation, internet-based security, packet encryption, and more. In this particular study, the authors conducted a comparison of performance among three widely used symmetric key cryptography algorithms: DES, AES, and Blowfish. The simulation results revealed that Blowfish outperformed other commonly used encryption algorithms. On the other hand, AES exhibited poor performance compared to other algorithms due to its higher processing power requirement. While using CBC mode added some extra processing time, the overall impact was relatively negligible, particularly for applications that require secure encryption of large data blocks [111]. Also, the authors in [42] analyzed popular encryption techniques and found that Blowfish consistently outperformed other encryption techniques across various parameters such as encryption time, decryption time, power consumption, memory usage, latency, jitter, and security level. Following Blowfish, AES emerged as the second-best symmetric algorithm, while 3DES exhibited the least effectiveness. However, symmetric algorithms like AES and Blowfish were not as effective due to their resource-intensive nature. In terms of hashing methods, MD5 produced the highest latency, followed by SHA256 and SHA1, with SHA256 ultimately being more secure than SHA1 and MD5. According to Anand Kumar and Karthikeyan [112] evaluated the performance of Blowfish and AES in terms of energy consumption, different data types (text, document, images), packet size, and key size. Their study highlighted that while numerous encryption algorithms are available to secure data, they consume significant computing resources such as battery and CPU time. The simulation results consistently favored Blowfish over AES in almost all test scenarios. Blowfish demonstrated superiority in text-based encryption, while AES exhibited better performance in image encryption. The study also highlighted the impact of changing the key size of the AES algorithm on performance. Overall, AES was recommended for situations requiring high security, while Blowfish excelled in terms of performance. Likewise, in the study focused on security challenges and mechanisms for IoT. It was determined that the Blowfish algorithm outperformed other cryptographic algorithms in terms of execution time, memory usage, throughput, power consumption, and security, making it well-suited for IoT applications. The researchers implemented both the original and modified versions of Blowfish in hardware and observed that the modified version exhibited improved encryption time and throughput. Future work aims to introduce a new 512-bit block cipher [113]. The authors in [114] investigated cryptographic methods such as AES and Blowfish, comparing parameters such as encryption speed, CPU utilization over time, and battery power consumption. Their findings demonstrated that the Blowfish method outperformed the AES algorithm in terms of processing speed and throughput, while also consuming less energy. Thus, the study concluded that Blowfish is the superior option. In [115] integrated encryption techniques in the authentication of multicast protocol for Ad-hoc networks. Their analysis concluded that the Blowfish algorithm enables faster encryption and decryption of data, requiring less CPU power compared to other methods. In the comparative analysis conducted by Cordova et al., (2017) [116] on selected security algorithms in cloud computing, Blowfish outperformed the AES and RSA algorithms based on simulated outcomes. Blowfish exhibited the least increase in processing time during key generation, encryption, and decryption, positioning it as a strong contender for one of the top security algorithms. Additionally, doubling the PC's RAM enhanced the performance of all tested algorithms, significantly improving their speed and effectiveness. In this study, the authors in [117] evaluated block cipher algorithms based on encryption time, decryption time, encryption throughput, and memory usage for various file sizes (text, image, and video). Their study concluded that the Blowfish algorithm excelled in symmetric key cryptography when encrypting text files and videos. However, the DES method, despite its ability to quickly encrypt small text files, was susceptible to brute-force attacks. For encrypting small-size image and video files, 3DES utilized less memory. AES performed better than other algorithms when encrypting images.

Overall, these studies above consistently highlight the superior performance of Blowfish in terms of speed, security, memory usage, and power consumption compared to other encryption algorithms such as DES, AES, and 3DES. However, numerous comparative analyzes yielded contrasting outcomes.

According to a study by Panda and Nag [118], the performance of encryption algorithms was evaluated based on execution time, memory usage, and throughput on two different operating systems. The study concluded that cryptographic algorithms are computationally intensive and require significant computing resources such as CPU time, memory, and battery power. Based on the simulation results, AES and Salsa20 were found to be preferable over Blowfish for encrypting plain text data. In another study by Singh et al., (2015) [30], a comprehensive comparison was made among four common encryption algorithms: AES, DES, 3DES, and Blowfish, focusing on security and power consumption. The simulation results demonstrated that AES outperformed the other algorithms. Although AES was initially considered superior to the original Blowfish algorithm, this study proposed enhancements to Blowfish by adding an additional key and replacing the XOR operation with a new operation '#'. These modifications increased the robustness of Blowfish, making it more resilient against intrusion attempts. The advanced Blowfish algorithm proved to be more energy-efficient and secure, thus reducing battery consumption. In similar study by Gautam et al., (2019) [26] conducted an experiment to assess the performance and usability of various cryptographic algorithms, including RSA, DES, AES, Blowfish, 3DES, and Twofish. The study concluded that AES and Twofish are the most promising options, surpassing other encryption standards in terms of speed, entropy, and efficient encoding. However, AES was found to be superior to Twofish due to its higher efficiency. In this research, the authors compared the execution time, memory usage, and ciphertext size of symmetric cryptography algorithms: 3DES, AES, Blowfish, and Twofish. The results demonstrated that AES exhibited the most efficient performance in terms of execution time for encryption and decryption. According to Ghosh conducted a comparative evaluation of encryption algorithms, namely AES, Blowfish, and Twofish, for enhancing the security of wireless networks. Based on the evaluation metrics studied, such as encryption time, decryption time, and throughput, Twofish demonstrated a clear advantage over AES and Blowfish. Due to its low encryption and decryption time and high throughput, Twofish is recommended for implementation alongside HMAC in the security of all networking protocols [119]. Raigoza and Jituri [27] assessed the performance of the Blowfish algorithm and the widely used Advanced Encryption Standard (AES). The findings revealed that AES outperformed Blowfish in terms of speed, with a difference of approximately 200 to 300 milliseconds. When modifying the data size, minor differences were observed between the evaluated methods, resulting in encrypted data lengths that were roughly similar for both AES and Blowfish. As the ASCII value increased, both AES and Blowfish experienced an overall increase in execution time, but the regression line slope for Blowfish was steeper than that of AES. In a study evaluating power consumption, Joulemeter, PassMark BatteryMon, and Wattc were employed to analyze four Advanced Encryption Standard (AES) finalist algorithms: RC6, Serpent, Mars, and Twofish, with respect to file size. The results indicated that the Twofish algorithm exhibited the highest remaining battery life. Among the four algorithms, Twofish consumed the least amount of electricity according to the power requirements measurements. The study emphasized the consistent findings across different measurement tools, highlighting that the Twofish algorithm consumes the least amount of power [120]. The authors in [39] conducted an evaluation of widely used symmetric algorithms to assess their effectiveness in terms of security, architecture, limitations, and efficiency. The experimental results indicated that AES emerged as the superior algorithm, excelling in security, efficiency, and architecture. This study developed a compact and modular Twofish algorithm as an alternative to AES-Rijndael, suitable for various applications. Their study introduced a small Twofish-128 hardware module for cryptography, demonstrating versatility in usage. Compared to the 128-bit version of AES, Twofish-128 required approximately 70% fewer logic components. The circuit implementation offered the same level of security as AES but with a significantly smaller size, making it adaptable to diverse applications [121]. In a study comparing AES and Twofish, the authors in [62] observed the following simulation results: AES exhibited faster encryption for text, while Twofish surpassed AES with increased RAM. For image encryption, AES was faster overall, but Twofish performed equally well with more RAM. Twofish demonstrated better performance for sound encryption, and its speed improved further with increased RAM. According to Nurgaliyev and Wang [122] evaluated the effectiveness of different components in current symmetric key algorithms. They found that AES (Rijndael) showcased the best performance in terms of security, adaptability, memory utilization, and encryption performance. While other approaches showed competence, they were compromised in terms of security and encryption performance. The study conducted a comparison of encryption algorithms, including AES, DES, IDEA, RC2, Blowfish, and RSA, analyzing their strengths and weaknesses across different parameters. The aim was to identify vulnerabilities in certain cryptographic algorithms. The findings led to the following conclusions: Symmetric algorithms like AES offer faster encryption and decryption, enhancing data safety during transmission. Asymmetric algorithms like RSA and Diffie-Hellman provide security advantages in terms of key size. RSA addresses issues related to key agreement and key exchange in secret-key cryptography [123]. A study compared Format Preserving Encryption (FPE, FIPS 74-8) on numeric data (credit card numbers) with block ciphers such as AES, DES, 3DES, and Blowfish. It found that FPE outperforms AES with a 192-bit key, achieving an average encryption and decryption time of 16198.5 ns for 1000 credit card digits. For 1000 sixteen-digit credit card numbers, Blowfish and AES with a 192-bit key performed comparably. The study recommends using AES or Blowfish for superior performance and security in preserving numeric data formats [124]. Aleisa [125] compared the DES and AES encryption standards and concluded that AES is the unquestionable winner in terms of security, being considered practically unbreakable in real-world applications. Although DES and 3DES have identified faults, they are still deemed secure and useful. This research compared the performance of symmetric key encryption techniques, including DES, 3DES, and AES, for text and image data. It highlighted the risk of eavesdropping over the internet, which threatens data confidentiality. The results showed that AES has shorter encryption and decryption times and higher throughput compared to other algorithms, while 3DES has the longest encryption-decryption time and the lowest throughput. AES consumes more memory, whereas DES requires the least memory among the evaluated techniques [126]. This study investigated cryptography algorithms used for security and privacy protection in the smart grid. Their study determined that symmetric algorithms outperformed asymmetric ones in terms of overall performance. While some algorithms, like Blowfish, showed competitive encryption and decryption speeds compared to AES, they did not meet security standards. AES was identified as the most secure algorithm, with DES ranking second, making it the recommended choice for safeguarding sensitive data in the smart grid [46]. The authors in [127] compared various symmetric key cryptography techniques and found that compressing plaintext maximized memory efficiency. Blowfish was the fastest algorithm but had inconsistent throughput for smaller plaintext sizes. AES provided generally consistent and slightly superior performance and was more resistant to birthday attacks due to its larger 128-bit block size, compared to Blowfish's 64-bit block size. Panda, M. compared symmetric (AES, DES, and Blowfish) and asymmetric (RSA) cryptographic methods using different file types, finding that AES outperformed the other algorithms in terms of throughput and encryption-decryption time. According to Advani and Gonsai [128] performed a performance analysis of symmetric encryption algorithms, evaluating their encryption and decryption times. AES and Blowfish appeared to be more effective for various file types based on the literature research. The study specifically tested AES, DES, 3DES, and Blowfish. This study compared the performance of symmetric encryption algorithms, specifically AES and DES, on mini PC devices like the Raspberry Pi. The study concluded the following: 1. AES algorithm exhibits faster encryption time compared to the DES algorithm. However, AES requires more memory during encryption, while DES utilizes binary numbers and AES employs both binary and hexadecimal numbers for encryption. 2. The AES algorithm, with its larger number of keys (128 bits), is recommended over DES, which has a key length of 64 bits, or half the number of keys in AES. AES offers a much higher number of potential keys (2^128) compared to DES (256 keys). The study suggests exploring the avalanche effect algorithm and using files or images as encryption media to evaluate the performance of each algorithm [129]. In this research, the speed, cost, and performance of symmetric encryption algorithms, including DES, AES, Blowfish, and RSA, were assessed in terms of their suitability for wireless sensor networks and peer-to-peer communication. Each cryptographic algorithm has its own set of advantages and disadvantages. RSA, while secure, has relatively higher time and power consumption. Blowfish, on the other hand, is useful for applications that require fast and secure communication due to its shorter encryption and decryption times. AES is recognized as an extremely secure algorithm but requires more memory and has longer encryption time, while DES is more memory-efficient [130]. This research conducted a comprehensive review of security techniques for SMS and conducted a performance comparison of commonly used encryption algorithms like DES, 3DES, RC4, Blowfish, and AES (Rijndael). DES was found to be limited by its large data size and short key length, while Blowfish and RC4 were found to have vulnerabilities related to weak keys. Among the symmetric algorithms discussed, AES (Rijndael) emerged as the most popular choice due to its flexibility and superior encryption performance. It offers improved security, faster processing, and overall reliability [131]. The authors in [132] analyzed encryption techniques for secure communication, finding that AES offers superior security, particularly in CBC mode, which uses data block chaining and an initialization vector, making it more secure than ECB mode. AES is ideal for applications requiring high dependability and confidentiality. The study suggests potential further development of AES due to improvements in entropy, throughput, and encryption efficiency. The research assessed the effectiveness of AES, DES, 3DES, Blowfish, RC4, and RSA in securing image data in cloud environments, addressing concerns like data breaches, account theft, insider threats, malware injection, and denial-of-service attacks.. According to the study, AES, Blowfish, RC4, and 3DES demonstrate good performance in terms of throughput, memory consumption, encryption time, and decryption time for image cryptography in the cloud. While RC4 performs well in terms of execution speed, it lacks sufficient security for image data. However, the most secure encryption algorithms for image data in cloud systems are Blowfish, AES, and RSA. The paper concludes that both Blowfish and AES are effective encryption algorithms for securing picture data in cloud systems, providing a balance between efficiency and security [133]. This study evaluated the performance of the DES and 3DES cryptography algorithms for ensuring data security in smart cards operating in NFC-based communication systems. The performance evaluation of text data cryptographic methods using DES and 3DES for data writing and reading processes of ACOS3 smart cards in NFC-based devices yielded several conclusions: 1. DES and 3DES text data cryptography methods can be successfully employed for data writing and reading processes in ACOS3 smart cards in NFC-based devices. 2. The data writing and reading processes of ACOS3 smart cards utilize the faster DES method compared to the 3DES cryptographic method. 3. When implementing the DES or 3DES cryptographic method with an ACOS3 smart card, the data reading process demonstrates faster execution than the data writing process [134].

2.12.1.1. Research gap

The evidence presented in the review highlights the presence of experimental gaps in the understanding of the most commonly used block cipher techniques. These gaps indicate that there is still a need for further research to fully explore and comprehend the capabilities and limitations of these algorithms. Moreover, the review brings attention to the flawed nature of previous comparisons conducted between these algorithms, as they often fail to consider the variations in key bit and block sizes employed by each algorithm.

To address these gaps and improve future research, it is recommended to explore different methodological approaches for conducting correlational studies on the commonly used symmetric algorithms. By employing alternative methodologies, researchers can obtain more accurate and comprehensive insights into the performance, security, and suitability of these algorithms. This may involve conducting controlled experiments, analyzing large-scale data sets, or implementing real-world use cases to assess the effectiveness and efficiency of different symmetric encryption techniques. Furthermore, future research should strive to establish robust correlations between algorithm characteristics, such as key bit and block sizes, and their impact on various performance metrics. This will enable a more nuanced understanding of the trade-offs and considerations associated with different algorithm choices. Additionally, exploring the impact of other factors, such as computational resources and data types, in conjunction with the algorithmic properties can provide a more holistic view of their effectiveness.

In conclusion, future research should employ alternative methodologies to address the experimental gaps and conduct correlational studies that consider the variations in key bit and block sizes among symmetric algorithms. By doing so, researchers can enhance our understanding of these algorithms, leading to improved recommendations for their practical implementation and strengthening the overall security of cryptographic systems.

2.12.2. Review of the performance evaluation of asymmetric algorithms that are frequently used

This work presents a secure key agreement and session authentication system for Internet of Things (IoT) devices. Simulation results demonstrated the system's resilience against various attacks and showed that its time complexity was lower compared to DSA and RSA, due to the use of ECC. The protocol exhibited the lowest computational overhead, the fastest turnaround times, and the greatest stability with minimal communication overhead. [28]. This undertaking presented a system employing Elliptic Curve Cryptography (ECC) and Diffie-Hellman (DH) key exchange to establish forward secrecy within HTTPS web browser applications. The envisaged technique notably reduces error rates compared to prior research. This methodology utilizes a dual key arrangement orchestrated by ECC-DH to effectively manage security in cloud contexts. Furthermore, this approach exhibits a heightened entropy value in comparison to the alternative, enhancing its overall security posture [135]. This study thoroughly examined asymmetric algorithms like Diffoe-Hellman, DSA, Elliptic Curve, and RSA as well as symmetric algorithms like Blowfish, AES, 3DES, and DES. According to the investigation, researchers found that certain characteristics had an impact on how well various algorithms performed. It is essential to provide solid, reliable, and trustworthy algorithms that can effectively manage massive amounts of data on the cloud given the growing demand for cloud applications. The two most crucial recommendations for cloud applications are speed and security [136]. Researchers in this study looked at a cryptographic method that integrated DNA encoding with the ECC algorithm and compared it to the widely used RSA algorithm. Elliptic Curve Cryptography-based DNA Encoding is superior to conventional methods in terms of temporal structure, physical size, and key length. Two layers of security are also included in the framework, the first of which is ECC steganography and the second of which is DNA encoding. In particular, for devices with low resources, cryptographic algorithms should be both practical and inexpensive. Additionally, ECC itself needs to be updated frequently to enhance the efficiency of the recently scheduled processors [29]. In reference to [137], researchers conducted an analysis of distinct cryptographic algorithms, evaluating aspects like key size, message size, and execution time. With the proliferation of diverse encryption techniques, facilitating swift and dependable communication among IoT devices has become a complex task, one that must be accomplished without causing interruptions. Determining the most suitable, compatible, and advantageous encryption method for communication has proven to be quite intricate. Through their examination, the authors reached the consensus that among various options, Schnorr, RSA, Elliptic Curve Cryptography, and ElGamal emerge as the superior choices. Kaur and Aggarwal [2] undertook an extensive examination of cryptographic methods including RSA, Blowfish, Diffie-Hellman, ECC, and others. The advent of the Internet of Things has brought to light a significant security concern that impacts various aspects ranging from authentication and authorization to trust management, even posing a threat to embedded systems. Among these techniques, ECC has demonstrated itself as the encryption method that excels in both security and efficiency. In reference to [37], Researchers analyzed various encryption methods, including RSA, Diffie-Hellman, Digital Signature Algorithm, and Elliptic Curve Cryptography (ECC), to determine the most effective for data confidentiality during transmission. They found that digital signatures provide robust confidentiality and non-repudiation, ensuring data integrity, availability, and confidentiality. This study assessed ECC, RSA, and Diffie-Hellman for network security, concluding that ECC is superior due to its comparable security with fewer bits. ECC's significant use in Bitcoin, Secure Shell, and Transport Layer Security demonstrates its exceptional security and cost-effectiveness. [138]. In [5], the authors compared Elliptic Curve Cryptography (ECC) with RSA in network security and introduced an EC point multiplication processor for digital signatures and key agreements. ECC's inverse operation, the Elliptic Curve Discrete Logarithm Problem (ECDLP), becomes more complex with longer keys, making ECC more viable as security needs and processing power increase. ECC provides equivalent security with shorter keys, maintaining efficiency and compactness compared to other algorithms. Elliptic Curve Cryptography (ECC), Rivest-Shamir-Adleman (RSA), and other encryption algorithms were examined in this study; the findings indicate that ECC is substantially more effective than the other methods. For elliptic curve cryptography, shorter key lengths and sizes are essential security requirements. Furthermore, it saves bandwidth, which facilitates the generation of keys for data encryption and decryption and enhances performance. ECC ensures quicker encryption and decryption and is also effective on small devices. It is preferable to use elliptic curve cryptography for data security [139]. This research examined the impact on performance when integrating ECC with SSL, a prominent technology for ensuring internet security. Earlier studies have suggested that the adoption of SSL leads to a notable decrease in web server speed. The findings demonstrated that, when subjected to real-world operational loads, an Apache web server could manage a higher volume of HTTPS requests per second—falling within the range of 13% to 31%—by utilizing ECC-160 as opposed to RSA-1024. This observation highlights the advantage of ECC-160 in providing short-term security enhancements [140]. In [141], the researchers conducted an analysis of the performance characteristics of conventional public-key cryptographic systems, namely RSA, DSA, and DH, in comparison to ECC. The investigations highlighted that the traditional public-key methods encounter performance-related challenges. The study proposed that general-purpose CPUs could effectively incorporate hardware acceleration to enhance public-key algorithm processing. The performance assessment indicated that ECC exhibited superior performance compared to RSA. Specifically, for ECC with GF(p) and GF(2m), the researchers noted a speedup of 2.4 times and 4.9 times, respectively, relative to RSA at the current security levels. Moreover, for subsequent security levels, the corresponding speedups were even more substantial—7.8 times and 15.0 times, respectively. In this investigation [85], a comparison was conducted between the elliptic curve cryptography (ECC) algorithm utilizing a 160-bit key size and the Rivest-Shamir-Adleman (RSA) technique employing a 1024-bit key size. The results demonstrated that ECC can offer comparable security levels with smaller key sizes when contrasted with more traditional cryptographic systems like RSA. Consequently, the adoption of ECC is strongly recommended to enhance security and efficiency without a proportional increase in computational demands. The research indicated that ECC maintains a lower cost ratio. Moreover, continuous enhancements are necessary for ECC itself to optimize the performance of newly developed chips. In a study similar to this, the authors referenced in [142] investigated the encryption and decryption times of various approaches using data packets of different sizes. The comparisons indicated that ECC leads to a significant reduction in transmission expenses. The results underscored that ECC outperforms other asymmetric algorithms in terms of efficiency. This study evaluated the impacts of different ECC curves and RSA key sizes using IoT nodes with limited resources, and it compared the performance of ECDSA and RSA TLS cipher suites. The results indicated that, although ECDSA consistently outperformed RSA in all test runs, practical scenario testing is necessary to determine the suitable security configuration for a given hardware platform. Situations may arise where more secure options, due to software implementations and optimizations, exhibit superior energy efficiency and data throughput, surpassing theoretically lighter and simpler alternatives. The results, influenced by enhancements in the libraries handling ECC operations, specifically showcased that the secp256r1 curve exhibited superior performance compared to the secp224r1 curve, while maintaining a higher level of security [143]. The study mentioned as [45] focused on assessing the performance of RSA and Elliptic Curve Cryptography within Wireless Sensor Networks. RSA's decryption time becomes unwieldy with larger key sizes, while ECC algorithms maintain controllable encryption and decryption times even with substantial key sizes. Notably, ECC signature signing tends to be quicker than its verification counterpart, whereas RSA's signature signing is more time-consuming. The results obtained from these implementations provide strong incentives for considering a shift from RSA to elliptic curve cryptography [144]. In a comparable investigation, the researchers delved into the foundational aspects of elliptic curves, their associated arithmetic operations, and the advantages of adopting elliptic curve cryptography over RSA within public cryptosystems. The outcomes of this research highlighted that ECC signature signing processes are usually swifter than verification procedures, while RSA signature signing tends to be more time-consuming. Moreover, the generation of public keys demand significantly more time with the RSA technique compared to ECCs. These findings in the implementation phase provided a compelling rationale for the researchers to advocate for a transition from RSA to elliptic curve cryptography [145]. This study compared the performance of RSA-based BROSMAP and ECC-based BROSMAP on Android and XAMPP servers. ECC significantly outperformed RSA, being nearly twice as fast as RSA 2048 and four times faster than RSA 3072, due to its smaller key sizes and use of symmetric cryptography for both encryption and decryption. ECC-based BROSMAP also showed 561 times greater computational efficiency. The researchers recommend ECC-based BROSMAP for resource-limited systems like IoT devices, as it meets all security requirements of RSA-based BROSMAP while being more efficient and lightweight [146]. In this study referenced as [87], the researchers conducted an analysis of the security capabilities of ECC and RSA encryption techniques using three sets of sample input data consisting of 8 bits, 64 bits, and 256 bits, each employing randomly generated keys in accordance with NIST recommendations. Their findings illustrate that ECC surpasses RSA in both operational efficiency and security. Furthermore, their work implies that ECC might be the preferred choice, particularly for devices with limited memory resources such as smartphones and palmtop PCs. In this paper, authors [142] conducted a comprehensive review of key cryptographic algorithms, including ECC, El-Gamal, and RSA, with the goal of facilitating a comparative assessment. The comparisons clearly indicate a significant reduction in transmission costs when employing ECC. These outcomes underscore the practical advantages of ECC's performance. The survey was undertaken to assess the security aspects of these algorithms, considering their widespread utilization. This research conducted an examination of two frequently employed encryption methods, namely Elliptic Curve Cryptography (ECC) and Rivest-Shamir-Adleman (RSA), with a particular emphasis on their applicability in the context of cloud and fog computing. The investigation involved a comparison of the key size and security capabilities of ECC and RSA algorithms, assessing their suitability for deployment in resource-limited fog computing environments. The findings suggest that ECC is a preferable choice for enhanced security and faster performance, all without imposing undue strain on computing resources. In contrast, RSA, with its established track record of security, remains widely accepted [147] . This paper introduced a novel approach to file encryption, employing a hybrid encryption algorithm that combines AES and RSA. It provides a foundational understanding of the AES and RSA algorithms while conducting a thorough examination of their pros and cons. The encryption techniques of these two algorithms have garnered substantial attention within the scholarly community. Through experimental comparisons, the study concludes that the hybrid encryption algorithm enhances encryption efficiency, bolsters key management, and fortifies data security in the context of file protection [148]. This paper introduces a secure data sharing scheme focused on maintaining data security and integrity in cloud environments. The system integrates Elliptic Curve Cryptography (ECC) with the Advanced Encryption Standard (AES) to provide robust authentication and data protection. Experimental results indicate that this method is more efficient and performs better than current approaches. [149]. This paper delves into the capabilities of cryptography for ensuring security in distributed storage. This exploration involves a thorough examination of standard cryptography techniques such as AES, ECC, and RSA. However, due to variations in the performance of these methods, the study addresses the challenge of identifying an encryption technique that strikes a balance between efficiency and security. Some encryption methods can deliver high security but are time-consuming for both encryption and decryption. Conversely, other approaches may offer efficient encryption but suffer from vulnerabilities in terms of security [150]. In reference [151], a two-tier cryptographic approach and a model are introduced to enhance data security in cloud computing. This model leverages both symmetric and asymmetric encryption algorithms, specifically AES and ECC, to bolster data security against unauthorized access, thus promoting privacy, data integrity, and expediting cryptographic operations. This advancement serves to enhance user trust in cloud computing while also accelerating the utilization of smaller ECC keys in the encryption process. In this research presented in reference [152], the authors examined Elliptic Curve Cryptography (ECC) to improve data security in cloud environments and compared it to the Advanced Encryption Standard (AES) with a focus on time efficiency. They evaluated encryption and decryption times for cloud-stored data using a sample size of N=6 for both ECC and AES. The study found that ECC encryption was faster, with a mean time of 0.1683 compared to AES's 0.7517. The significance value for the proposed system was 0.643 ($p>0.05$). The results indicate that ECC is more time-efficient than AES for data encryption. In 2023, Rao and Sujatha introduced a security technique for public cloud systems using Hybrid Elliptic Curve Cryptography (HECC). Their method generates keys with a lightweight Edwards curve and modifies private keys with Identity Based Encryption, then reduces key sizes to speed up AES encryption. Public keys are exchanged via the Diffie-Hellman method. Evaluation metrics include throughput and the time for key generation, encryption, and decryption. Their model outperforms existing ones, with key creation in 0.000025 seconds, encryption in 0.00349 seconds, and a throughput of 693.10 kB/s. [153]. This paper presents a robust and efficient protocol using a blind factor and Elliptic Curve Cryptography (ECC) for enhanced security. ECC is favored over RSA for its superior security with smaller keys, reducing computational overhead. Benefits include faster processing, lower power consumption, reduced bandwidth usage, better storage efficiency, and more compact certificates. These advantages are crucial in bandwidth, processing, power, or storage-constrained environments. The authors also developed a Hybrid Public Key Cryptographic algorithm, combining Dual-RSA and ECC, which significantly improves performance in computational cost and memory storage.[154]. This paper introduced a hybrid cryptography algorithm aimed at ensuring confidentiality and enhancing security for internet communications. The research places particular emphasis on minimizing the time required for encryption and decryption to avoid excessive CPU utilization. Experimental findings demonstrate that the proposed solution offers a more efficient means of encrypting messages, with only a marginal difference in the algorithm's runtime. This approach effectively enhances security in the open internet environment [155].

2.12.2.1. Research gap

Research papers examining asymmetric algorithms consistently reveal that Elliptic Curve Cryptography (ECC) surpasses other asymmetric algorithms in terms of speed and efficiency. However, it is important to note that existing studies have primarily focused on ECC's comparative advantages in specific scenarios/ case studies such as IoT and cloud computing. There is a need to explore the performance of RSA and ECC encryption techniques by adopting alternative methodologies and use case scenarios

2.12.3. Review work on hybridizing Cryptographic Algorithms and Compression Techniques

Numerous studies have suggested diverse cryptographic techniques with the aim of improving both the speed and security of data transfer.

Sharma and Gandhi [156] explored different aspects related to two distinct yet non-contradictory domains: Data Compression and Cryptography. This paper offers a brief examination of both branches—compression and encryption—discussing the essential need for data compression and data encryption. It also emphasizes the significance of integrating these two branches. The challenge lies in determining the optimal order for applying these processes, i.e., whether Compression should precede Encryption or vice versa. While, in 70% of cases, it proves more efficient to apply compression before encryption, in specific situations and for particular purposes, encryption can also be applied before compression. In reference [157], the author conducted a comparison of various lossless data compression algorithms, including Arithmetic encoding, Huffman coding, and Run Length encoding algorithms. The performance metric utilized for evaluation was the compression ratio. The findings indicated that Huffman exhibited a poorer compression ratio in comparison to Arithmetic encoding. However, it was noted that the speed of compression and decompression for Huffman was superior to that of Arithmetic encoding. Additionally, the conclusion highlighted that Huffman necessitates less memory for compression operations. In paper [158], the author examined diverse lossless data compression algorithms specifically for text files. The comparison included Run Length encoding, Adaptive Huffman Algorithm, LZW algorithm, Shannon Fano Algorithm, and Huffman Encoding. The evaluation criteria encompassed compression and decompression times, as well as the compression ratio. The findings indicated that the most effective compression algorithm for text files among those compared was the Shannon Fano Algorithm. The authors in [159] suggested an effective and secure compression technique that integrates a secret key to achieve its objectives. The encoding of input data involves the use of a generated key to scramble the data, followed by transformation through the Burrows-Wheeler Transform (BWT). Subsequently, the output from the BWT undergoes compression through both the Move-To-Front Transform and Run-Length Encoding. This method seamlessly incorporates cryptographic principles of confusion and diffusion into the compression process, thereby enhancing its overall performance. The proposed technique aims to deliver robust encryption and substantial compression. Experimental results demonstrate its superiority over other techniques in terms of compression ratio. In this paper, the study presented the Crypto-Compression System, an algorithm that combines Stream cipher cryptography with entropy encoding. This integrated approach aims to reduce data size, increase data transfer rates, and enhance security in communication. Experimental results demonstrate that the generated ciphertext from our proposed technique consumes less channel bandwidth compared to a one-time pad. When contrasted with Huffman coding, both techniques exhibit similar bandwidth requirements; however, our proposed approach offers a significantly more secure method of transmission [160]. In this research, Ali and Kadhim [161] introduced a technique for concealing secret texts with Unicode characters to enhance data protection. Leveraging the similarities of glyphs, this method achieves invisibility and an increased hiding capacity. In summary, the proposed approach successfully secures confidential data and attains a substantial payload capacity by employing the Huffman compression algorithm, capable of handling unlimited text length. Furthermore, the method can conceal a single bit within every digit or letter in the cover file. Notably, this technique ensures cognitive transparency and avoids making modifications evident in the original data. To heighten security, the method encodes a secret message using the Advanced Encryption Standard (AES) algorithm before embedding it within the cover text. The authors in [45] presented an innovative algorithm for both image encryption and compression. This algorithm combines Parallel Compressive Sensing, Secret Sharing, and Elliptic Curve Cryptography to achieve compression, encryption, identity authentication, and blind signcryption. The proposed algorithm is designed to withstand various types of attacks, including man-in-the-middle, forgery, and chosen-text attacks. Notably, it boasts lower storage and computational complexity, ensuring high security and a remarkable Peak Signal-to-Noise Ratio (PSNR). The incorporation of blind signcryption guarantees participant identity and shadow secrecy, maintaining verifiability, as detailed in the paper. The practicality and security of the scheme are substantiated through numerical experiments, security analysis, and proofs, surpassing the effectiveness of existing schemes. The work of Murtaza et al. [162] proposed an enhanced secure image steganography technique that utilizes double encryption algorithms. In this approach, the message is initially encrypted using AES and then further encrypted using ECC. The resulting double-encrypted data is compressed using Lempel Ziv Welch technique to reduce the storage capacity of the secret data. Experiment results demonstrate that combining ECC and AES encryption, LZM compression, and Knight Tour algorithm produces stego images of higher quality. This paper offers a highly secure and robust image steganography method, as the Knight Tour algorithm is less well-known to unintended users than the PRNG technique. The combination of compression, hiding, and conversion techniques leads to increased stego image quality and reduced distortion. In their work, Sagheer et al. [163] also proposed a module that concurrently performs compression and encryption operations on the same dataset. This simultaneous execution is achieved by integrating encryption into compression algorithms, leveraging the shared characteristics of cryptographic ciphers and entropy coders in terms of secrecy. Initially, the provided text undergoes pre-processing and is transformed into an intermediate form, enhancing its compressibility and security within the dedicated secure compression module. The entire module represents a well-designed synthesis of compression and cryptography principles, effectively complicating the task of cryptanalysis for potential intruders. The study concludes that using this module facilitates the secure transmission of confidential data even in an insecure medium. In the study conducted by Jha et al. in 2021, a model was presented that integrates compression and security by simulating a hybrid scenario of compression and security. This is accomplished by employing Huffman Encoding for compression and CBC (Cipher Block Chaining) for security. Furthermore, the visualization of the Huffman Tree is facilitated using JavaScript. The results illustrate the efficacy of applying this model across various domains. Specifically, the utilization of the model with Huffman coding is highlighted for its substantial impact, especially in handling sensitive data such as genome and DNA sequences, where compromise is not permissible [13]. The hybrid design proposed in [164] combines the AES and Huffman compression algorithms. As AES creates a large file size overhead in the network, the Huffman algorithm was incorporated to alleviate this issue. Before the application of Huffman coding, the Avalanche Effect value (AE/bit change ratio) was approximately 40%. However, once the Huffman coding was employed, the Avalanche Effect (AE) value increased to 49%, which is very close to the optimal 50%. Moreover, the entropy value increased to 7.9, up from the previous value of 6 without the use of Huffman coding. Additionally, when analyzing 6 different file types (.txt, .doc, .xlsx, .pptx, .pdf, .jpg), the BER (Bit Error Rate) parameter yielded an ideal value of 0 for all cases. In a similar field, Elmahi et al. [165] devised a text steganography algorithm utilizing Pseudo-random Number Generation (PRNG). This algorithm involves the embedding of a confidential message into a cover-text generated through PRNG, followed by compression to reduce its overall size. The reverse operations are executed at the receiver's end to retrieve the original message. PRNG generates sequences through deterministic algorithms computed from initial seeds, eliminating the necessity to determine specific overheads for different messages. Moreover, the model exhibits flexibility, allowing customization to individual preferences. For instance, one can modify the algorithm by swapping the first and last words to embed the message or by applying a distinct permutation set. In this study, the approach proposed by the authors in [166] involves combining two data-intensive operations, Bulk Cryptography and Compression, to create an efficient solution for acceleration. The primary objective of this research is to ensure the integrity, confidentiality, and authenticity of user data during transit or storage, all while minimizing resource usage, including memory and network bandwidth. By chaining these operations together, this technique optimizes storage, security, and bandwidth utilization. The associated processing overhead is minimal when compared to the accrued benefits. Through the utilization of Intel's Quick Assist Technology and Network Accelerators, the burden on the CPU is alleviated during bulk crypto and compression operations, resulting in valuable CPU cycle savings. Premadasa and Meegama presented a thorough text message compression method that incorporates cryptographic measures to guarantee enhanced security, ensuring message confidentiality, authenticity, and integrity. In the initial phase, the technique compresses the lengthy text message into a cipher text comprising 32 characters using the MD5 algorithm. The encryption process involves the use of an initialization vector and secret key, facilitating extensive message compression. The encrypted cipher-text is subsequently transmitted via the SMS gateway and can be decompressed by the intended recipients to restore it to its original form. The results demonstrate that the proposed mechanism does not adversely impact message delivery time [167].

The study by Hengjian et al. [168] delved into the challenge of efficiently encrypting and compressing image data. They introduce a novel image encryption approach that blends set partitioning in hierarchical trees (SPIRT) with a feed-forward-feedback nonlinear dynamic filter (FFNDF) and random arithmetic coding. Pseudorandom sequences generated by the FFNDF dictate the mapping of interval positions during arithmetic coding. The SPIRT algorithm is employed to encode bits from different passes using adaptive random arithmetic coding, resulting in a flexible and secure image compression coding scheme that caters to progressive coding features. The scheme undergoes both theoretical scrutiny and experimental assessment, revealing robust cryptographic and perceptual security without apparent compromise in compression efficiency. Consequently, this encryption scheme stands out as an ideal solution for various applications, including total encryption, selective encryption, and conditional access. Mohamed et al. proposed a practical method for enhancing data security and minimizing data quantity by employing a lossless data compression technique, specifically Huffman coding, followed by encryption using the symmetric AES algorithm. Additionally, a key exchange concept based on the Diffie-Hellman Key Exchange (DHKE) is proposed to facilitate the exchange of secret keys for data encryption and decryption. The proposed approach is implemented using the Trivial File Transfer Protocol (TFTP) for transferring data between two computers in a local network. The study concludes that data compression and encryption are effective techniques to secure data transmission, reduce file size, and save time [169]. The authors in [170] introduced the RSE algorithm, the first SE algorithm for HEVC video bitstream. To avoid modifying the codec structure, data selection and encryption are carried out after video encoding. A random selection algorithm is developed based on the RC4 pseudorandom sequence. Then, only 0.1% to 0.2% of the data is extracted and encrypted using the AES-CTR algorithm. The evaluation results show that the RSE algorithm improves encryption efficiency without compromising visual quality and cryptographic security. The authors in [171] proposed a new encryption-compression hybrid approach that utilizes the AES encryption algorithm to operate on the dominant coefficients in a mixed scale representation. The compression process is achieved through the Faber-Schauder Multi-scale Transform (FMT), which is known for its simplicity and ability to secure information in the outline regions of the image. During compression, the AES encryption algorithm leaves homogeneous zones in the high frequencies. Compared to DES, it is approximately twice as fast to compute (in software) and approximately 10^{22} times more secure (in theory). The FMT-AES approach was found to perform well when compared to other methods, including Quadtree-AES and DCT-partial encryption. In the research conducted by Alsaffar et al. [172] , a blend of encryption methods was employed to enhance security. This combination included the DNA encryption algorithm, GZIP algorithm, AES cryptography, and image steganography. To heighten the security of the message, the result of the final stage of DNA encryption underwent multiplication by a specific factor. Subsequently, the message underwent compression using the GZIP algorithm, resulting in a remarkable size reduction of 75% and a transformation of the message into a new format. To further fortify the security, the AES encryption algorithm was applied. Furthermore, the message was concealed within a high-quality image using LSB image steganography technology. This comprehensive approach yielded noteworthy results for image metrics, with the Lina sample achieving a PSNR of 67.589, MSE of 0.0116, SSIM of 1, NPCR score of 0.011630262741374, and UACI score of 4.5608873495583733e-05. The study ensures seamless data transfer and bolsters the safety of sensitive information, rendering it inaccessible to hackers. In paper [173], the author proposed a hybrid technique that involves utilizing both the Advanced Encryption Standard (AES) and Huffman compression to enhance SMS security and increase data capacity. The AES performs the encoding of messages into ciphertext form, resulting in an expansion of the character count due to additional cipher information in each encrypted data. Consequently, the application of Huffman algorithm compression becomes necessary for text compression, reducing the size of the encrypted messages. The encryption test results indicate that the AES algorithm cryptography effectively secures SMS by producing unreadable ciphertext. Additionally, the Huffman compression demonstrates a 17.35% improvement in compression efficiency compared to the scenario without compression. Lilhore et al. [174] proposed a hybrid model using MobileNetV3-SVM and transfer learning for improved intrusion detection in IoT and 5G networks, suitable for resource-limited settings. It combines deep learning for real-time packet processing with hybrid cryptographic systems for secure data transmission and storage. This multi-layered approach enhances the detection of known and unknown threats through continuous learning, offering comprehensive online protection. The authors in [175] introduced a hybrid encryption method combining AES-256, DKM, and Improved Elliptic Curve Cryptography (IECC) to balance effective communication and data security in resource-limited environments. This approach enhances security and performance by leveraging DKM's key updating, AES-256's data confidentiality, and IECC's computational efficiency. The method demonstrates notable improvements over traditional encryption techniques, including a 30% reduction in message transmission time and a success rate of over 99% in thwarting intrusion attempts in simulated attacks.

2.12.3.1. Research gap

In contemporary digital environments, the security of data during transmission is critical, typically ensured through encryption techniques like the Advanced Encryption Standard (AES). However, existing methods that integrate encryption and compression often struggle to minimize data size while maintaining algorithmic security. This presents a research gap in balancing optimized data storage with the preservation of data integrity and confidentiality.

2.12.4. Review of Modifications to the RSA Algorithm

Numerous scholars have undertaken diverse initiatives to enhance data security, with some focusing on reducing algorithm execution costs, while others prioritize bolstering data security. To acquire a comprehensive understanding, a concise review of relevant literature is essential, as outlined in this section.

The RSA algorithm's security relies heavily on the careful selection of large prime numbers. Successfully implementing and maintaining the security of RSA requires a thorough exploration of prime number generation for public key cryptography, emphasizing the importance of establishing strong and reliable primes. Despite the sparse distribution of large prime numbers, their verification involves computationally expensive tasks like modular exponentiation with large integers, leading to a notably slow prime number generation speed.

In reference [176], the research explores the optimization of the process of generating prime numbers crucial for public-key cryptography. It proposes employing algebraic methods to simplify the intricate process of prime number generation. The study introduces innovative algorithms designed to decrease the complexity involved in generating n-bit primes, demonstrating their practical implementation on smart-cards. The author in [177] addresses RSA security concerns by introducing a modified algorithm employing multiple prime numbers and public keys. It presents a modified RSA algorithm using four prime numbers and two public keys. The study enhances security by complicating factorization through the use of multiple primes and public keys; however, this enhancement led to a reduction in operational speed. Ivanov and Stoianov (2023) [178] emphasize the significance of prime number arithmetic ratios in RSA key security, suggesting revisions to RSA key generation standards. The current RSA key generation standards do not consider prime number ratios' impact, potentially leading to vulnerable keys. The findings suggest that the arithmetic ratio of prime numbers in RSA keys is a critical factor influencing key security. Additionally, Jain et al. (2020) [179] propose modifications to RSA encryption using multiple prime numbers set in the encryption process. The empirical findings show variations in encryption and decryption times, reflecting the influence of utilizing multiple primes within the RSA encryption framework. The study employs small prime numbers to achieve the encryption and decryption speed. Pradhan and Sharma (2013) [180] combine Batch RSA and MPrime RSA to speed up RSA decryption processes. Their approach involves concurrent decryption and the utilization of more than two prime numbers in the modulus, facilitating efficient decryption through the Chinese Remainder Theorem. The experimental results show significant reduction in decryption time; ensuring efficient decryption while maintaining security integrity, particularly beneficial for environments with constrained computational resources. The authors in [181] enhance RSA encryption by using multiple keys and CRT to improve efficiency and security. The study modifies the RSA algorithm to use four prime numbers and continuous subtraction method, aiming to speed up and simplify encryption and decryption processes. The utilization of multiple keys enhances security, while CRT implementation accelerates decryption processes compared to traditional RSA methods. Khairina and Harahap (2019) [182] modify the RSA cryptographic security using a Cubic Congruential Generator to generate prime numbers for key generation. Through this method, the study utilizes CCG-generated random numbers, tested for primality to establish the values of p and q in RSA, thereby strengthening the robustness of ciphertexts. The ciphertext produced through CCG-enhanced RSA demonstrates increased resilience against cryptanalytic attacks. This study suggests that the utilization of complex number generators like CCG during RSA key generation can significantly fortify encryption security. Imam et al. (2022) [183] present XRSA, an enhanced version incorporating four prime numbers and XOR operations, aimed at rectifying vulnerabilities found in standard RSA and bolstering resilience against attacks. Their research indicates a substantial increase in security against brute-force attacks, albeit at the cost of increased time required for key generation, encryption, and decryption compared to standard RSA. In [86], the authors adopt the proposed Enhanced RSA (ERSA) algorithm developed by [184], conducting a comparative analysis with the traditional RSA algorithm. ERSA incorporates two extra prime numbers into the Standard RSA algorithm to augment both speed and security. The experimental findings highlight that employing prime numbers, instead of random ones, in the proposed system contributes to an enhancement in the speed of encryption and decryption processes. The authors in [185] enhance the study conducted by authors in [186] by implementing R prime RSA, a cryptographic method reliant on large prime numbers known for their heightened security compared to the conventional RSA, which employs dual prime values. The security strength of R prime RSA is tied to the modulus of n, whereby a higher modulus contributes to a more secure encryption scheme. A lower modulus, however, weakens the overall security strength of the encryption. In [187], the authors propose a novel approach for fast generation of RSA key pair on Smartcards. The significant feature of the proposed method is to improve the required time for finding a large random prime number which is the most time consuming step in the RSA key pair generation. The simulation results indicate that the time required to generate RSA key pairs of 512, 1024, and 2048 bits is notably reduced compared to traditional RSA algorithms.

2.12.4.1. Research gap

These studies collectively contribute to the ongoing evolution of RSA cryptography, bridging gaps in efficiency, security, and adaptability. However, the RSA still has bottleneck for key generation, encryption time and decryption time as the prime number becomes larger. Prior studies have also focused on a very limited set of performance metrics and missing impacted results in terms of prime number generation times of their proposed algorithms compared to the traditional RSA algorithm and existing modified RSA frameworks, taking into account varying key bit sizes and diverse file sizes.

2.12.5. Advanced Encryption Standard

The Advanced Encryption Standard (AES) algorithm is widely recognized and widely used as a symmetric block encryption algorithm across the globe. It finds extensive application in wireless networks, e-commerce platforms, and numerous other contexts. Due to its unique design, this encryption method is commonly employed in both hardware and software systems for the purpose of encrypting and decrypting sensitive documents. The AES algorithm provides a high level of security, making it exceptionally challenging for hackers to decipher encrypted data. As a result, it has become a trusted option for safeguarding confidential information. This study focuses on reviewing papers that aim to enhance the performance of AES algorithms specifically for secure data transmission. The objective is to find ways to optimize the efficiency and effectiveness of AES in encrypting and decrypting data, with the aim of enhancing the overall speed and reliability of secure data transmission processes. By exploring and analyzing various research papers, this study aims to contribute to the ongoing advancement of AES algorithms, ultimately leading to improved security measures for transmitting sensitive information.

2.12.5.1. Review on AES S-box, Key Expansion and MixColumn Transformation

Several researchers have been exploring the development of a key-dependent S-box and the parallelization of key expansion. In this paper [188], the author introduces a customized version of AES designed to enhance the security of the algorithm, making it more resilient to cryptanalysis and safer for deployment in sensitive networks and applications where security is paramount, such as military networks and applications. The security of the customized AES is enhanced through modifications to the existing AES at two key points: firstly, by employing a novel AES key expansion algorithm to eliminate dependencies between round keys and improve key expansion time, and secondly, by incorporating a key-dependent S-box for each round to address issues associated with using a static AES S-box. The results demonstrate that the proposed AES significantly enhances security compared to the original AES. In paper [189], Cao et al. introduce three enhancement concepts: an irreversible improvement strategy, the introduction of a random number strategy, and a word shift strategy to the existing AES scheme. These strategies aim to diminish correlation between round keys and optimize the runtime of the AES algorithm. Through theoretical and experimental analysis of the algorithm's security and efficiency, the results demonstrate that the enhanced algorithm ensures efficient operation while maintaining the security of the key expansion algorithm. Moreover, it improves the overall anti-attack performance of the AES algorithm. In [190], the authors propose an AES key expansion algorithm based on two-dimensional Logistic mapping. This algorithm enhances the workload for brute force attacks, making AES cracking more challenging. By employing the two-dimensional Logistic mapping, the dependence between sub-keys is reduced, significantly increasing the security and robustness of the AES encryption sub-keys. Experimental results demonstrate the strengthened security and robustness of the AES sub-keys achieved through this approach. In their work [191], the authors investigate a novel method for AES-256 Key Expansion through the use of the Even-Odd (E-O) method. The proposed algorithm consists of two main components: Key Expansion and E-O Select Round Key. The algorithm places a high emphasis on the element of confusion, resulting in improved efficiency compared to traditional algorithms. The avalanche effect of the E-O method outperforms the classic approach, and the algorithm successfully eliminates the concept of weak keys. Additionally, the results demonstrate that the performance of sub-key generation is comparable to that of the classic AES algorithm. In a similar study, the authors in [192] leverage AES Key Expansion to generate multiple non-linear keys for the encryption process. Experimental findings indicate that the proposed algorithm attains superior encryption quality while requiring minimal memory and computational resources. The algorithm demonstrates high key sensitivity and features an extensive key space, making it highly resistant to Brute force attacks and statistical cryptanalysis on both original and encrypted images. Additionally, the encryption time is significantly lower compared to other algorithms proposed in similar contexts. In paper [193], the authors introduce an enhanced AES algorithm as a response to the limitations of the traditional

AES algorithm, which faces challenges from increasing computational power and emerging attack methods. The proposed approach involves integrating chaotic sequences into the key expansion scheme of the AES encryption algorithm. By augmenting the key expansion algorithm with a chaotic sequence, the number of exhaustive attacks required for the round keys is increased to 1.89 x 2^{162}. Through experimental testing, the improved encryption algorithm showcases elevated levels of security and execution efficiency in comparison to the original algorithm. The enhanced algorithm effectively encrypts and decrypts data, presenting a viable solution for the advancement of the AES encryption algorithm. In this study, the authors present an upgraded key expansion algorithm for the Advanced Encryption Standard (AES) that enhances data diffusivity and data security in wireless communications. The paper addresses the weaknesses observed in typical key expansion algorithms and proposes a solution by employing a double S-box model, which ensures improved key security without compromising algorithm efficiency. Additionally, the paper provides a detailed description of the AES encryption algorithm and its extended versions. To validate their approach, the authors conduct experiments and present compelling results that demonstrate the effectiveness of the proposed algorithm in bolstering key security for wireless communications [12]. In paper [188], the study proposes a customized Advanced Encryption Standard with better cryptographic strength than the original AES algorithm by updating two operations: the first is a proposed key expansion algorithm for AES that improves security by removing dependencies between round keys to prevent attackers from reaching the secret key or other round keys, and the second uses a dynamic selection S-box for each round from the five stored S-boxes, based on simple mating. The avalanche impact of a dynamically selected S-box is greater than that of the static S-box employed in the original AES. The encryption time for both the original and proposed AES remains unchanged, despite the incorporation of new S-boxes and the implementation of a new key expansion process. This is because the updated operations in the proposed AES are optimized to consume minimal time and are executed offline solely when there is a modification in the Secret key. In this paper [194], the authors introduce a new key expansion algorithm for AES. The proposed algorithm incorporates a Parallelized Key Expansion Algorithm to enhance the traditional AES algorithm. By eliminating dependencies on other subkeys, the proposed algorithm achieves both faster speed and improved security. Implementation results affirm that the computational efficiency of the proposed algorithm exceeds that of the standard AES Key expansion. In this study [195], the authors suggest two approaches to improve the efficiency of the conventional AES. The study employ Genetic Algorithm and Neural Network techniques to enhance security against timing attacks and reduce computational time. The study offers the following recommendations: (1) Implementing this technique will enhance the complexity of the existing cryptosystem, making it more robust against cryptanalytic attacks, and (2) Comparing the feedforward NN AES with the genetic algorithm-based AES will yield a stronger and more efficient symmetric cryptosystem. In paper [196], the study introduces a modification to the Advanced Encryption Standard (AES) to address low diffusion rates in early rounds. By incorporating additional primitive operations, the modified AES exhibits a significant increase in diffusion, as demonstrated through avalanche effect evaluations. The frequency test further confirms improved randomness in the ciphertext. The results highlight enhanced diffusion and confusion properties in the modified AES, ensuring successful decryption and recovery of the original plaintext. The authors in [197] introduce an innovative image encryption scheme, utilizing both the Lorenz hyperchaotic system and the RSA algorithm. This scheme is designed to enhance the security of image communication, effectively thwarting various attacks. The authors hypothesize that their proposed scheme tackles the challenge of key exchange through the application of the RSA algorithm, while also concealing image data using permutation and finite field diffusion algorithms rooted in the Lorenz hyperchaotic system. Empirical findings validate the efficacy of the image encryption scheme proposed in this research, showcasing robust resilience against attacks and sensitivity to key variations. In addition, the security of this encryption scheme relies on the strong security features inherent in the RSA algorithm. In this research [198], a new cryptographic method is introduced, utilizing the Lorenz attractor. The study incorporates a chaotic operation mode that enables interaction among the password, message, and a chaotic system. The algorithm proposed, in tandem with the chaotic operation mode, achieves a strong cipher. Assessments of its performance and comparisons with AES algorithms underscore the method's suitability and readiness for real-world applications. This study introduces a new algorithm for image encryption and decoding, utilizing the fractional Fourier transform, Lorenz attractor, and masking. The paper aims to address the challenge of developing an image encryption and decoding algorithm that is resilient against attacks, offering both high security and performance. An analysis of chaotically generated random numbers was conducted, yielding successful results. The algorithm ensures confidentiality and proves effective against common attacks, including brute force [199]. The authors suggest a novel color image encryption algorithm that combines the Lorenz and Rossler attractors with a multi-key concept for a conservative chaotic system. The study incorporates a confusion and diffusion process to randomly modify the pixel values of the plain image, thereby enhancing image security. The authors compare the proposed algorithm with results obtained using both single and multi-key algorithms and analyze its performance against different critical attacks. They assert that the proposed system exhibits superior efficiency, image confidentiality, and high encryption and decryption speeds [200]. In [201], the authors present a new algorithm based on the discrete quantum Baker map and Chen hyperchaotic system. The process begins by representing the color digital image using the NEQR model and then subjecting it to block scrambling through the Baker map. Subsequently, the index-order scrambling method is employed to further disorder the image's rows and columns. The ciphertext image is ultimately generated through diffusion using the quantum XOR operation. Both numerical simulations and theoretical analysis validate that the proposed algorithm features a considerable key space, exhibits a certain level of robustness, and demonstrates highly efficient performance.

Several studies have been carried out to enhance the security or efficacy of AES cryptography. In this work, Eslam et al. [202] developed an alternative lightweight design for both forward and reverse MixColumns steps, essential for efficient AES hardware implementation. Their investigations reveal that the proposed MixColumn design exhibits lower complexity compared to previous significant efforts in terms of gate count and clock cycles. According to the study, this streamlined design could prove beneficial for implementing AES in RFID tags, smart cards, and remote sensors. Gamido et al. [203] employed bit permutation as a replacement for the MixColumns Transformation in AES, aiming to enhance its efficiency. The study compared the performance of the conventional and modified AES algorithms through the encryption of text files and images. The evaluation criteria included encryption time, CPU usage, and avalanche effect. The findings revealed that the modified AES exhibited increased efficiency compared to the standard AES, demonstrating faster encryption times for text files and images. Additionally, the modified AES showcased lower CPU usage and a higher avalanche effect during testing. In the study conducted by Riyaldhi et al. [16], the primary focus is on reducing the encryption time, which tends to increase with a growing number of data bytes. The authors propose an algorithm incorporating modified shift rows and S-box for the mix column transformation. Their experiments demonstrate that with each additional 1024 bytes of data, the computation time increases by three milliseconds. The shift row transformation undergoes modification through array shift mapping, while the mix column transformation sees alterations with the incorporation of an S-box and the elimination of the SubByte transformation. However, a notable drawback of this algorithm is that the modified S-box requires twice the area of the traditional S-box. Results indicate a significant 86.143% enhancement in the encryption process and a 13.085% improvement in the decryption process. The authors in [204] conducted an in-depth analysis of the different rounds within the standard AES algorithm, challenging a prior study's claim that excluding the MixColumns step has no impact on security. Their findings strongly suggest that the omission of MixColumns in the final round of AES diminishes the security of reduced-round variants, particularly in the face of various types of attacks. Furthermore, they put forth the notion that the overall security of the complete AES could be compromised if an attack on the entire AES exploits relationships between the subkeys of the last round and other subkeys. AlMarashda et al. [205] conducted an assessment of the AES security, examining the impact of including or omitting the MixColumns operation. While the general belief suggested that omitting the MixColumns in the last round had no security consequences and was merely done for optimization, a claim in mid-2010 proposed potential security implications due to a reduction in the time complexity of attacks against AES. The study demonstrated that omitting MixColumns from the last round could indeed diminish the security of the AES block cipher. Additionally, they explored the performance overhead of including the MixColumns operation, revealing a minimal increase of approximately 2% in computation time. In this paper, Barrera et al. [206] presented an enhanced approach to computing the MixColumns operation in AES. Their focus centered on devising and evaluating an efficient implementation of the AES-128 MixColumns algorithm using two distinct methods. The first approach involved constructing circuit modules based on the traditional row-column multiplication method, resulting in a straightforward circuit configuration. The second approach aimed to introduce a more parallel behavior into the circuit, aligning input signals to potentially reduce delays. The outcomes revealed that their second approach, utilizing parallelism in signal processing, led to diminished time delays, reduced logic elements, and lower virtual memory usage. In this paper, the study proposed lightweight design modification for both forward and backward ShiftRows (inverse) and both forward and backward MixColumns (inverse) operations essential in AES. The modification is integrated with the AddRoundKey operation to minimize the time required for encryption and decryption processes. Through security examination and experimental results, the proposed encryption algorithms demonstrate high speed, along with robust security and reduced information overhead. Additionally, the modified AES with six rounds stands out as the quicker choice among the modified algorithms [207]. Manoj Kumar and Karthigaikumar [15] introduced a new and efficient AES algorithm that is key-dependent, providing an enhanced avalanche effect in comparison to the existing AES algorithm. Unlike the conventional AES, where the key matrix influences only one transformation stage (add round section), making other stages independent of the key and easily reversible, their proposed algorithm ensures that all operations are conducted using the key. Through an XOR operation applied to all key bytes, a parity bit is obtained, determining the operations in each transformation for every round. Implementation in FPGA revealed synthesis results with a slight increase in the Avalanche effect.

2.12.5.1.1. Research gaps

1. Traditional key expansion methods in AES typically adhere to fixed approaches, maintaining a consistent expansion mode throughout the encryption process. Each cipher undergoes multiple rounds with fixed operations to achieve the desired security level. However, despite its efficiency, the AES key expansion algorithm exhibits a significant vulnerability. With knowledge of any round key, an adversary can deduce all other round keys, presenting a serious threat known as the "related-key attack" to the overall security of AES.

2. Prior studies have indicated that the MixColumn transformation in AES is associated with an increased utilization of resources.

2.13. Recent Developments and Advances in Cryptography - Post Quantum Cryptography

Cryptography is a dynamic field with ongoing advancements that significantly impact data security. Post-quantum cryptography, a relatively recent domain, addresses the challenges posed by quantum computers to existing encryption methods. Quantum computers have the potential to break widely used cryptographic algorithms like AES, RSA, and elliptic curve cryptography. The response to this threat involves the development of new cryptographic algorithms that can withstand quantum computer attacks. This section reviews various works on post-quantum cryptography to identify future research directions.

Table 2.2: Literature Matrix of Post-Quantum Cryptography

Author/Date	***Focus***	***Theoretical/Conceptual Framework***	***Findings***	***Gaps***
Kretschmer et al., 2023 [208]	Pseudo randomness in Quantum Cryptography	Quantum computational complexity, pseudorandom quantum states, Forrelation problem	Constructed classical oracle proving the existence of pseudorandom quantum states without one-way functions, challenging the necessity of OWFs for quantum cryptography	The implications for practical cryptography and the translation of theoretical models to real-world applications are not fully explored.
Yin et al., 2020 [209]	Quantum Cryptography	Quantum key distribution (QKD), entanglement-based cryptography	Achieved entanglement based QKD between two ground stations over 1120 km without trusted relays, securing data transfer with high efficiency and practical security against known side channels.	Long-term stability, scalability, and integration with existing networks for broader real-world application were not covered.
Pirandola et al. [210]	Quantum Cryptography	Quantum mechanics, QKD protocols	Detailed examination of QKD protocols, device independence, satellite challenges, continuous variable systems, quantum repeaters, networks, data locking, and quantum digital signatures.	Specifics on real-world implementation challenges and long-term viability of various protocols.
Bernstein & Lange, 2022 [211]	Post-Quantum Cryptography	Quantum Computing, Cryptographic Algorithms	Overview of post-quantum cryptography, its evolution, and NIST's role in standardization. Discusses hybrid cryptographic approaches.	Detailed analysis of individual post-quantum algorithms and their specific implementations.

Iqbal & Krawec, 2019 [212]	Semi-Quantum Cryptography	Quantum key distribution (QKD), Semi-quantum models	Developed SQKD protocols with various security-proof methods and discussed practical attempts	Further details on experimental results and wider practical applications are not discussed in the covered sections.
Bernstein and Lange, 2017 [213]	Post-quantum cryptography	Quantum computing and cryptography; Shor's and Grover's algorithms	Identified cryptographic systems and mathematical operations that could remain secure against quantum computer attacks.	In-depth analysis on security proofs and practical implementations.
Ugwuishiwu et al., 2020 [214]	Quantum Cryptography and Shor’s Algorithm	Quantum mechanics, computational complexity	Reviewed quantum vs classical cryptography, Shor's Algorithm, its impact on encryption, and quantum key distribution mechanisms.	The practical application of quantum cryptography and detailed analysis of its challenges are not extensively covered.
Dam et al., 2023 [215]	Post-Quantum Cryptography (PQC)	Quantum computing, cryptographic algorithms	Overview of PQC developments, NIST’s standardization process, and current implementation status on hardware platforms.	Real-world application scenarios and long-term effectiveness of proposed algorithms.

Based on the review of the current literature on quantum cryptography, it is evident that while there has been significant research and progress in the field of quantum cryptography, there remains a notable gap. Moreover, this study has not deeply explored specific aspects of quantum cryptography. This presents an opportunity for future research to delve more comprehensively into these areas. Future work should aim to fill this gap, expanding the understanding and application of quantum cryptography in various domains. This direction is crucial for advancing the field and harnessing the full potential of quantum cryptographic techniques.

Chapter 3
Empirical Evaluation of Symmetric Block Cipher Techniques

This chapter focuses on examining the effectiveness of symmetric algorithms commonly employed for data transmission, which aligns with the research objective (**RO 1**). To accomplish this objective, the study devised two primary tasks: (1) To investigate the encryption, decryption times, and throughput (speed) of three widely utilized block cipher algorithms within the realm of symmetric algorithms: Twofish, Blowfish, and AES and (2) To compare the performance of AES, Twofish, 3DES, and Blowfish symmetric encryption algorithms across key sizes of 128, 192, and 256 bits, utilizing their respective consistent block sizes, both of which were successfully completed and documented in a journal article (Q3) and a conference paper. Furthermore, this chapter is subdivided into two main approaches.

3.1. Motivation of the study

This study aimed to investigate the encryption, decryption times, and throughput (speed) of three widely utilized block cipher algorithms within the realm of symmetric algorithms: Twofish, Blowfish, and AES. The investigation encompassed various file types to ensure a comprehensive analysis of their performance. By examining the encryption and decryption times, the study sought to understand the efficiency of these algorithms in securely encoding and decoding data. Additionally, the throughput, or speed, of the algorithms was assessed to determine how quickly they could process data during transmission.

Different file types were used during the investigation to simulate real-world scenarios and evaluate the algorithms' performance across diverse data formats. This comprehensive approach aimed to provide valuable insights into the strengths and weaknesses of Twofish, Blowfish, and AES when employed for data encryption and decryption.

It is important to highlight that, up until this point, no comprehensive study has delved into the analysis of these algorithms with diverse file types while maintaining a constant block size and exploring various key bit sizes. Therefore, this study fills a significant research gap by providing a detailed examination of the encryption and decryption times of 3DES, Twofish, and AES, as well as the effectiveness of Blowfish with varying key sizes, across different file types.

3.2. Introduction

Due to the increasing number of incidents in which personal data between two parties is taken by intruders, it is critical to protect data communicated over the Internet nowadays [216]. People spend so much time connected to a network, network security has become an extremely important part of data communication. These are vulnerable to security attacks such as unauthorized access to a file or alterations to its contents. One of the main reasons invaders succeed is that most of the information obtained from a system is in a form that can be read and comprehended. The solution to this dilemma is to utilize Cryptography. This is the art and science of securing information from unwanted individuals by changing it into an indiscernible form to its attackers while it is stored and transported [217]. There are numerous encryption methods that are widely available and utilized in information security. They are classified as Symmetric (private) or Asymmetric (public) Key Encryption. Only one key is needed to encrypt and decrypt data in symmetric keys encryption or secret key encryption. Asymmetric keys employ two keys: private and public keys. The public key is used to encrypt data, while the private key is used to decrypt it (e.g. RSA and ECC) [109]. A block cipher algorithm is a symmetric key cryptosystem whose security is based on sophisticated non-linear transformations and whose encryption speed is quite fast. As a result, the block cipher algorithm has evolved into a vital encryption technique that is widely utilized in applications such as secure data transfer, storage encryption, digital signing, and entity certification [218]. The primary purpose of the security mechanism is to give message privacy while also ensuring data confidentiality, integrity, and non-repetition. The primary function of network security is to enable efficient data authentication and authorization through the use of cryptographic algorithms [131]. A cryptographic algorithm is typically computationally heavy and thereby, consumes a lot of computing power such as CPU time, memory usage, and power consumption [118]. Previous research has revealed inconsistencies in the efficacy of various encryption methods. The current work analyzed symmetric (AES, Twofish, and Blowfish) cryptographic algorithms using multiple file types such as binary, text, and image files with a unique key bit size of 128. These encryption methods were compared based on three different parameters: encryption time, decryption time, and throughput. The effectiveness of each technique is demonstrated using simulation data. The research is divided into two methodological approaches. In the **initial approach**, the study addresses the following research questions.

1. What is the performance difference between the various algorithms using a constant key bit size of 128?
2. Which block cipher technique works better in the context of process time and throughput using different file types?

Hence, the current study makes the following key contributions.

i. To perform an extensive evaluation of the encryption, decryption times, and speed using a unique key bit size of 128

ii. To analyze the performance using different file types.

iii. To perform an extensive analysis of the performance of selected algorithms, namely: AES (Rijndael), Twofish, and Blowfish

In the **second approach**, this study examines and contrasts the effectiveness of specific algorithms, specifically AES (Rijndael), Twofish, and 3DES. The study investigates the following research questions.

1. Which key bit sizes of the algorithms AES, Twofish, and 3DES work best in terms of encryption and decryption times when using a consistent block size of 128 bits?
2. How does the performance of the Blowfish algorithm vary with different key bit sizes and a fixed block size of 64 bits?
3. Which key bit size performs better with different file extensions?

Hence, the current study makes the following contributions.

i. To perform an extensive evaluation of the encryption and decryption times of AES, Twofish and 3DES using a block size of 128 bits

ii. To analyze the performance of AES, Twofish, and 3DES using varying key bit sizes of 128, 192 and 256.

iii. To further analyze the performance of Blowfish with key bit sizes of 128, 192, and 256, while maintaining a fixed block size of 64 bits

3.3. Symmetric 128-Key bit Approach: AES, Twofish and Blowfish

The study implemented the various symmetric encryptions in Python. The performance evaluation is based on the implementation of three symmetric algorithms AES, Twofish and blowfish for encryption and decryption, and throughput. The following criteria were used: a) encryption and decryption time; b) throughput; and c) 128 key bit size for AES, Twofish, and Blowfish. To show the outcomes for the conclusion, the values for each criterion were logged and graphically plotted. The simulation was run on a laptop with an Intel® CoreTM i5-10210U CPU running at 2.40 GHz and 16 GB of RAM. Version 21H2 of Windows 11 Pro for Workstations was used. A key size of 128 bits was utilised as the benchmark in this experiment to acquire trustworthy values for evaluating the efficiency of AES, Blowfish, and Twofish cryptographic algorithms. The experiment was run three times and the mean execution time were recorded. The three block-cipher methods—AES, Blowfish, and Twofish—are also listed in Table 3.1 as a summary.

Table 3.1: Key and Block sizes.

Factors	AES	Blowfish	Twofish
Key sizes	128	128	128
Block size	128 bits	64 bits	128 bits

3.3.1. Performance analysis

3.3.1.1. Process Time (Encryption and Decryption time)

Tables 3.2 to 3.3 show the comparison of results. It is worth noting that AES-128 key bit size has the quickest encryption and decryption time on average.

Table 3.2: Encryption times for AES, Blowfish and Twofish (128 key bit).

File format	File size (in kb)	AES	BLOWFISH	TWOFISH
file_example_TXT	9	0.037459	0.010809	0.25
file-example_PDF_1MB	1,018	0.368214	0.304263	26.34
file_example_MP3_5MG	5,166	1.182518	1.436277	131.50
file_example_MP4_1280_10MG	9,610	2.218504	2.874504	244.11
file-sample_1MB_DOCX	1,003	0.247889	0.306568	25.39
file_example_XLS_5000	657	0.171259	0.218169	16.59
file_example_PPT_250kB	243	0.105667	0.111124	6.20
file_example_JPG_2500kB	2,446	0.569065	0.718191	63.02

Table 3.3: Decryption times for AES, Blowfish and Twofish (128 key bit).

File format	File size (in kb)	AES	BLOWFISH	TWOFISH
file_example_TXT	9	0.01238	0.016267	0.244043
file-example_PDF_1MB	1,018	0.288262	0.298124	25.60037
file_example_MP3_5MG	5,166	1.213215	1.504732	135.4913
file_example_MP4_1280_10MG	9,610	2.238994	2.703225	245.67
file-sample_1MB_DOCX	1,003	0.298465	0.298391	25.33303
file_example_XLS_5000	657	0.200942	0.21287	16.62755
file_example_PPT_250kB	243	0.076602	0.092704	6.150904
file_example_JPG_2500kB	2,446	0.615148	0.735764	62.01268

3.3.1.2. Throughput

The throughput of an encryption scheme defines the speed of encryption. The encryption scheme's throughput is calculated by dividing the total plaintext in bytes encrypted by the encryption time [107]. In this experiment, the throughput is derived from calculated as the total plaintext in Kilobytes encrypted/encryption time (KB/sec) divided by their mean time generated. AES has the highest throughput making it the fastest of the three followed by blowfish. The results are shown in Tables 3.4 to 3.6.

Table 3.4: AES Throughput in kilobytes/seconds (128 key bit).

File format	File size (in kb)	ENCRYPTION	DECRYPTION
file_example_TXT	9	240.2626872	726.9789984
file-example_PDF_1MB	1,018	2764.696617	3531.50953
file_example_MP3_5MG	5,166	4368.64386	4258.10759
file_example_MP4_1280_10MG	9,610	4331.747881	4292.106187
file-sample_1MB_DOCX	1,003	4046.165824	3360.528035
file_example_XLS_5000	657	3836.294735	3269.600183
file_example_PPT_250kB	243	2299.677288	3172.240934
file_example_JPG_2500kB	2,446	4298.278756	3976.278879

Table 3.5: Blowfish Throughput in kilobytes/seconds (128 key bit).

File format	File size (in kb)	ENCRYPTION	DECRYPTION
file_example_TXT	9	832.6394671	553.2673511
file-example_PDF_1MB	1,018	3345.789662	3414.686506
file_example_MP3_5MG	5,166	3596.799225	3433.169495
file_example_MP4_1280_10MG	9,610	3343.185468	3555.012994

file-sample_1MB_DOCX	1,003	3271.704809	3361.361435
file_example_XLS_5000	657	3011.426921	3086.390755
file_example_PPT_250kB	243	2186.746337	2621.246117
file_example_JPG_2500kB	2,446	3405.779243	3324.435553

Table 3.6: Twofish Throughput in kilobytes/seconds (128 key bit).

File format	**File size (in kb)**	**ENCRYPTION**	**DECRYPTION**
file_example_TXT	9	36	36.87874678
file-example_PDF_1MB	1,018	38.64844343	39.76505027
file_example_MP3_5MG	5,166	39.2851711	38.12790932
file_example_MP4_1280_10MG	9,610	39.36749826	39.11751537
file-sample_1MB_DOCX	1,003	39.50374163	39.59257933
file_example_XLS_5000	657	39.60216998	39.51273639
file_example_PPT_250kB	243	39.19354839	39.506388
file_example_JPG_2500kB	2,446	38.81307521	39.44354606

3.3.1.3. Discussion of results for 128-Key bit Analysis

Tables 3.2 to 3.6 show the encryption time, decryption time, and throughput respectively. Performance analysis varies based on a particular file type, but on average, AES outperforms Blowfish and Twofish in terms of speed and process time. Furthermore, the figures in Fig 3.1 and Fig 3.2 are based on the average of total encryption/decryption and throughput of AES, Blowfish, and Twofish. An overview of all the comparisons can be summarized into the following Table XI. The summary in Table 3.7 are based on values from fig 3.1 and fig 3.2. AES-128 produced fast encryption, decryption times and speed than Blowfish and Twofish.

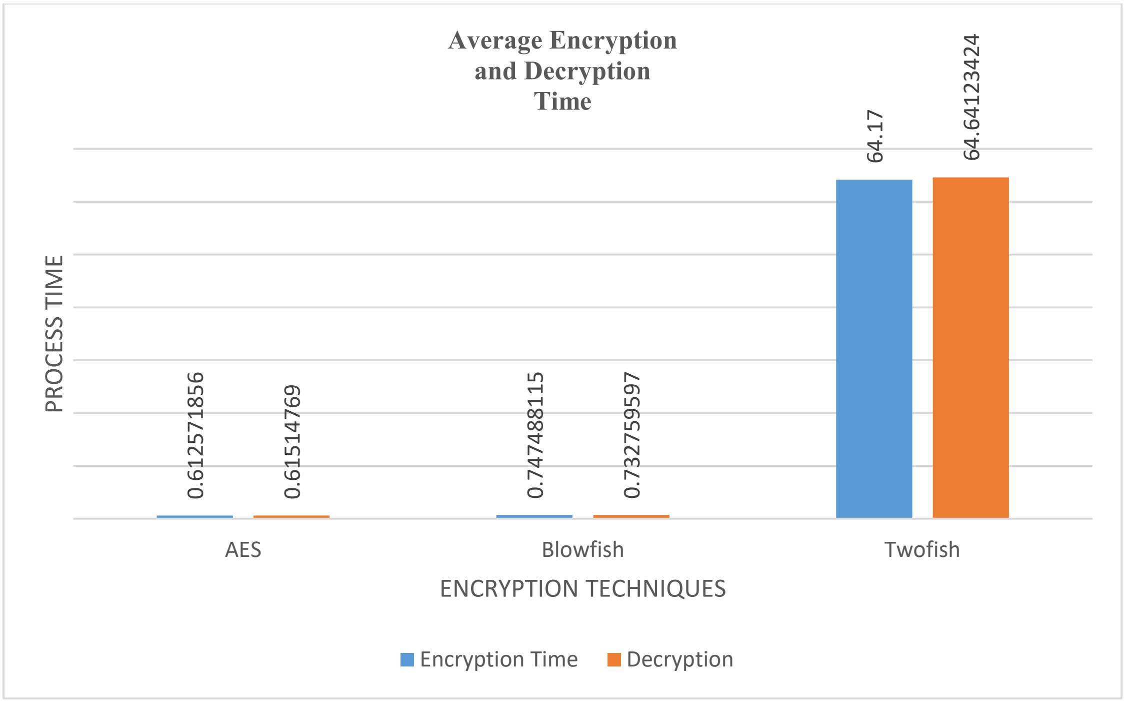


Figure 3.1: Average Process for AES, Blowfish and Twofish.

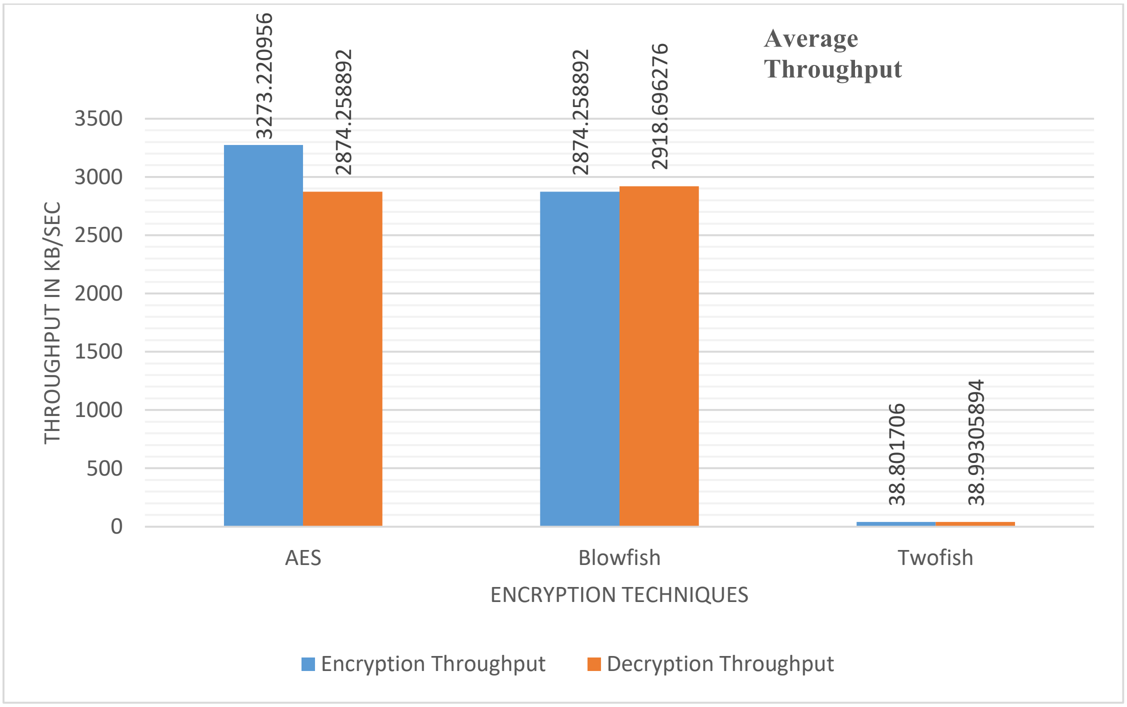


Figure 3.2: Average Throughput for AES, Blowfish, and Twofish.

Table 3.7: AES, Blowfish, and Twofish an overall comparison.

Parameters	AES	Blowfish	Twofish
Key bit size	128	128	128
Encryption	Very fast	Fast	Too slow
Decryption	Very fast	Fast	Too slow
Throughput (Speed)	Very high	High	Low

3.4. Symmetric 128, 192 and 256 Key bits with their respective block size Approach: AES, 3DES, Twofish and Blowfish

The symmetric algorithms AES, Twofish, and 3DES are used as the basis for the performance evaluation in terms of encryption and decryption times. In addition, the study further conducted performance analysis on 128, 192 and 256 key bits' sizes on Blowfish using a block size of 64. This is because Blowfish has a 64-bit block size. The simulations were run on a laptop with an Intel® CoreTM i5-10210U CPU running at 2.40 GHz and 16 GB of RAM. Version 21H2 of Windows 11 Pro for Workstations was used. In this experiment, key sizes of 128, 192, and 256 bits were used to provide reliable values for comparing the performance of the AES, Blowfish, 3DES, and Twofish cryptographic algorithms. The experiment was run twelve (12) times and the average execution time in seconds was recorded. Table 3.8 also summarizes the various block-cipher techniques: AES, Blowfish, Twofish and 3DES.

Table 3.8: Key bits and Block sizes.

Factors	AES	*Blowfish	Twofish	3DES
Key sizes	128, 192 and 256 bits	128, 192 and 256 Bits	128, 192 and 256 bits	128 and 192 bits
Block size	128 bits	64 bits	128 bits	128 bits

* Please note that the analysis of Blowfish was not included in the comparative study

3.4.1. Performance evaluation

The algorithms were compared based on their processing speeds, block sizes, and key bit sizes. Table 3.9 to 3.14 shows the times in seconds for both encryption and decryption.

Table 3.9: 128 key size - average encryption times.

File format	Size in Kb	AES Average time	*Blowfish Average time	Twofish Average time	3DES Average time
Txt	9	0.057	0.04	0.25	0.053
PDF	1,018	1.862	1.782	26.34	2.169
MP3	5,166	9.105	9.545	131.5	10.928
MP4	9,610	17.529	17.75	244.11	20.144
DOCX	1,003	1.918	1.951	25.39	2.115
XLS	657	1.088	1.278	16.59	1.519
PPT	243	0.52	0.529	6.2	0.573
JPG	2,446	4.342	4.812	63.02	5.252

Table 3.10: 192 key size - average encryption times.

File format	Size in Kb	AES Average time	*Blowfish Average time	Twofish Average time	3DES Average time
Txt	9	0.049	0.04	0.246	0.178
PDF	1,018	1.737	1.58	23.859	1.678
MP3	5,166	8.876	7.954	120.319	8.453
MP4	9,610	15.995	15.193	225.117	15.746
DOCX	1,003	1.739	1.625	23.771	1.665
XLS	657	1.091	1.051	15.821	1.083
PPT	243	0.426	0.471	5.96	0.417
JPG	2,446	4.316	3.957	59.714	4.011

Table 3.11: 256 key size - average encryption times.

File format	Size in Kb	AES Average time	*Blowfish Average time	Twofish Average time
Txt	9	0.038	0.033	0.228
PDF	1,018	1.5	1.485	24.198
MP3	5,166	7.611	7.383	124.065
MP4	9,610	14.115	13.704	230.228
DOCX	1,003	1.432	1.439	23.953
XLS	657	0.94	0.945	15.563
PPT	243	0.378	0.366	5.779
JPG	2,446	3.485	3.498	58.084

Table 3.12: 128 key size - average decryption times.

File format	Size in Kb	AES Average time	*Blowfish Average time	Twofish Average time	3DES Average time
Txt	9	0.035	0.054	0.244	0.046
PDF	1,018	1.889	1.868	25.6	2.022
MP3	5,166	9.068	9.722	135.491	10.832
MP4	9,610	17.643	18.074	245.67	20.364

DOCX	1,003	1.866	1.983	25.333	2.091
XLS	657	1.138	1.303	16.628	1.433
PPT	243	0.504	0.498	6.151	0.561
JPG	2,446	4.559	4.415	62.013	5.27

Table 3.13: 192 key size - average decryption times.

File format	**Size in Kb**	**AES Average time**	***Blowfish Average time**	**Twofish Average time**	**3DES Average time**
Txt	9	0.032	0.046	0.246	0.046
PDF	1,018	1.422	1.464	23.922	1.698
MP3	5,166	7.115	7.349	118.859	8.471
MP4	9,610	13.236	13.651	228.702	15.724
DOCX	1,003	1.413	1.447	23.862	1.678
XLS	657	0.924	0.956	15.563	1.103
PPT	243	0.366	0.378	5.755	0.416
JPG	2,446	3.363	3.486	58.655	4.041

Table 3.14: 256 key size - average decryption times.

File format	**Size in Kb**	**AES Average time**	***Blowfish Average time**	**Twofish Average time**
Txt	9	0.027	0.031	0.255
PDF	1,018	1.44	1.473	24.514
MP3	5,166	7.201	7.366	124.713
MP4	9,610	13.401	13.733	231.95
DOCX	1,003	1.43	1.506	23.931
XLS	657	0.949	0.968	15.653
PPT	243	0.356	0.383	5.84
JPG	2,446	3.482	3.475	59.243

3.4.2. Discussion of results for 128, 192 and 256 Key bits Analysis

The analysis presented is based on two methods utilized during the experiment. The first approach involved conducting a comparative analysis of Advanced Encryption Standard, Triple DES (3DES), and Twofish with key sizes of 128, 192, and 256 bits, using various file extensions and maintaining a constant block size of 128 bits. The second method involved analyzing Blowfish with key sizes of 128, 192, and 256 using a block size of 64 bits.

Table 3.9 (128-bit key size) shows that the Twofish algorithm generally performs slower than the other two algorithms across all file formats. For example, for a PDF file of size 1,018 Kb, the Twofish algorithm took 26.34 seconds on average, while the AES and 3DES algorithms took 1.862

and 2.169 seconds, respectively. In contrast, the Twofish algorithm showed slightly improved performance with a speed of 0.25 seconds when processing a 9 Kb TXT file. The AES algorithm generally performs the fastest except for small text file (9 Kb), 128-key bits 3DES has the fastest encryption time of 0.053 seconds, while 128-key bits AES is slightly slower with an average encryption time of 0.057 seconds. Table 3.10 displays distinct trends in contrast to Table II, with the 192 key bits of 3DES algorithm being generally the fastest, and Twofish being the slowest across all file formats except for text file where 192 key bit of AES outperform 3DES with a speed of 0.049 seconds. Table 4.11 (256-bit key size) shows that the AES algorithm is generally the fastest for all file formats, followed by Twofish. Tables 3.11 and 3.14 do not include any values for the 256 key bit size for either the encryption or decryption timeframes data on 3DES because it has no 256 key bit size. Table 3.12 presents the average decryption times for a 128-bit key size for AES, Twofish, and 3DES encryption algorithms. The data in this table indicates that Twofish takes the longest time to decrypt all file formats compared to AES and 3DES. The AES algorithm takes the least amount of time to decrypt all file formats. Table 3.13 shows the average decryption times for a 192-bit key size for the same encryption algorithms and file formats. The results show that AES remains the fastest algorithm for all file formats, with Twofish still taking the longest time to decrypt. The decryption time for all algorithms has decreased for all file formats with the increase in key size. Table 3.14 presents the average decryption times for a 256-bit key size for the same encryption algorithms and file formats. The data shows that Twofish still takes the longest time to decrypt compared to AES.

Tables 3.9 to 3.14 also include the average encryption and decryption times of the Blowfish algorithm with different key sizes for various file formats using a fixed block size of 64 bits. Comparing tables 3.9 to 3.14, it can be observed that increasing the key size from 128 bits to 192 bits and then to 256 bits leads to a decrease in the encryption and decryption times. Overall, the 256-bit key size of Blowfish outperforms the 128-bit and 192-bit key sizes in both encryption and decryption times.

In summary, the 256 key bit size of AES has the highest encryption time compared to the 128 and 192 key bit sizes of AES. On average, AES algorithm outperforms Twofish and 3DES in terms of encryption times. During the decryption process, on average, the 192-bit key size of AES showed slightly superior performance compared to the 256-bit key size in restoring data to its original form. During the analysis, it is observed that Blowfish's 256 key bit size had the fastest encryption and decryption times on 128 and 192 key bit sizes of Blowfish.

3.5. Conclusion

In today's rapidly expanding Internet and network applications, encryption algorithms play a critical role in ensuring information security. This study outlines two methodological approaches for conducting correlational research on the most commonly used symmetric key block cipher techniques. Initially, using a key bit size of 128, the study assessed three symmetric key encryption algorithms: AES, Twofish, and Blowfish. Based on the experimental results, the 128 key bit of AES algorithm has the shortest process time and runs quicker than Twofish and Blowfish. In the second phase, this study extensively investigated the key bit lengths of four block cipher algorithms: AES, Blowfish, Twofish, and 3DES. The processing times for each symmetric approach were tested using a variety of file types. When it comes to encryption and decryption speeds, 192 and 256 key bit sizes of AES both produce extremely comparable results. During the analysis, it was found that AES's 256 key bit size had the fastest encryption times. In terms of decryption time, AES with a key size of 192 bits outperformed the 256-bit key size of AES. It can also be observed that with a fixed block of 64 bits, Blowfish's 256 key-bit size presented the quickest encryption and decryption timings than 128 and 192 key bit sizes of Blowfish. The findings further demonstrate that Blowfish can compete with AES in terms of encryption and decryption speed. In conclusion, both methodological approaches demonstrated that the AES algorithm is better suited for secure data transfer.

Chapter 4
Comparative Analysis of ECC and RSA

This chapter aligns with Research Objective 2 (RO 2), which focuses on empirical evaluation of the most commonly used Asymmetric algorithms (RSA and ECC). To achieve this objective, the study formulated a specific task designed to address the key aspects of RO 2. This task was planned and executed, ensuring that all relevant factors were considered and addressed. The successful completion of this task has been documented and disseminated through a journal article, classified as a Q3 publication, demonstrating the rigor and relevance of the research conducted.

4.1. Motivation for conducting performance analysis on RSA and ECC for a secure email system

Previous studies highlight ECC's superior speed and efficiency over other asymmetric algorithms, particularly in IoT and cloud computing. However, there is a gap in understanding its performance in other practical applications, such as secure email systems. This study aims to compare ECC and RSA encryption techniques within the context of secure email communication, providing broader insights into their real-world applications and guiding the development of more secure communication systems.

4.1.1. Introduction

In today's interconnected world, email has become an indispensable tool for communication. It facilitates the exchange of information, ideas, and documents across vast distances, enabling individuals and organizations to collaborate and communicate efficiently [219]. However, the convenience of email also brings with it a host of security concerns, as emails can easily fall prey to unauthorized access, eavesdropping, identity spoofing, interception, or data tampering [220][221]. This is where cryptography plays a pivotal role in ensuring the confidentiality, integrity, and authenticity of email communication. Email encryption is the cornerstone of secure email communication. It employs complex algorithms to transform the content of an email into an unreadable format, known as ciphertext, which can only be deciphered by the intended recipient possessing the decryption key [222]. This cryptographic process ensures the following [142][223]:

1. Confidentiality: Encrypted emails are incomprehensible to anyone without the decryption key, thwarting unauthorized access and eavesdropping.

2. Integrity: Any tampering with the encrypted email is readily detected by the recipient, ensuring that the message remains unaltered during transit.
3. Authentication: Encryption can be combined with digital signatures to confirm the identity of the sender, assuring the recipient of the email's legitimacy.

Encryption algorithms like RSA, AES, and elliptic curve cryptography are commonly employed in email security protocols such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS). Additionally, public-key infrastructure (PKI) systems and digital certificates play crucial roles in verifying the authenticity of email senders [224]. This research project's primary aim is to explore the correlation between the performance of secure email communication systems and the encryption methods employed. Specifically, the study will investigate how the use of both RSA and ECC encryption techniques impacts the effectiveness of these secure email systems. In more extensive detail, the study seeks to discern any connections between the choice of cryptographic algorithms, RSA and ECC, and the overall performance of secure email systems. Furthermore, the study suggests a hybrid cryptography algorithm that utilizes both RSA and ECC to enhance security and preserve confidentiality in the context of secure email communication. The research will assess various performance aspects, including key exchange time, encryption and decryption times, signature generation and verification times, to ascertain how these encryption methods influence the efficiency and efficacy of secure email communication. Through an analysis, the study's aim to identify any potential relationships or dependencies between the selection of encryption methodologies and the outcomes in terms of secure email system performance.

Hence, the current study makes the following key contributions.

i. To perform an extensive analysis of the performance of selected algorithms, namely: RSA and ECC for secured email communication

ii. To perform an extensive evaluation of the key exchange time, signature generation and verification times between RSA and ECC techniques

iii. To present a hybrid cryptography algorithm that employs both RSA and ECC to ensure confidentiality and enhance security for secure email communication

4.1.2. Experimental Setting

A local area network (LAN) consisting of a dedicated network server and two client machines were used to carry- out the simulation. Fig 4.1 shows the diagram of the simulation setup.

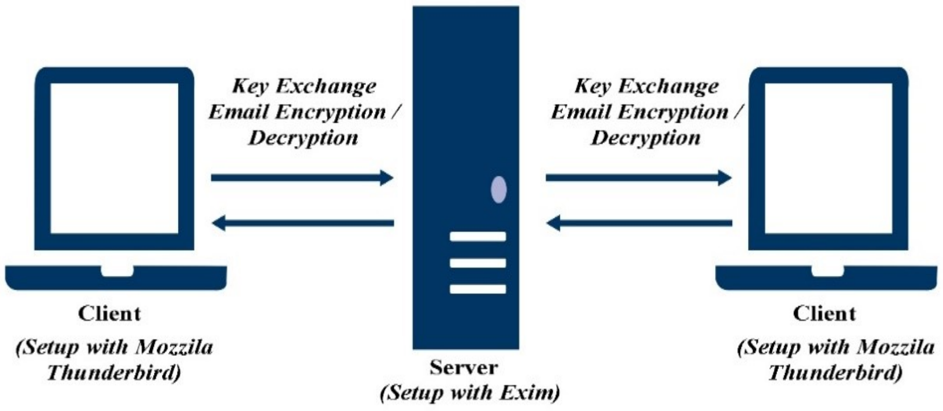


Figure 4.1: Simulation setup.

The testing environment, which included standard email clients and server, featured laptops with Intel i7 processors, 16GB RAM, and solid-state drives. The server was equipped with the following specifications - OS: Ubuntu, CPU: Intel Xeon, Cores: 4 core, 2.4 GHz, Architecture: 64-bit, and RAM: 8 GB DDR4 - represent typical end-user systems in corporate or personal contexts. To commence, email server Exim and clients Mozilla Thunderbird were configured to support RSA, ECC and hybrid encryption. Public and private keys were generated for RSA, typically 2048 bits, and ECC, using the widely adopted SECP224R1 curve for each user. Key exchange occurred during the initial email contact, and keys were securely stored. The experiment was conducted five times and the mean values for each metric were recorded. A series of practical tests were conducted, involving the sending of emails of varying sizes, from small text-based messages to larger attachments. Throughout the tests, measurements were taken for encryption and decryption times, key exchange time, signature generation, and verification times.

4.1.3. RSA-ECC Hybrid Technique

The proposed hybrid technique merges the robust aspects of both RSA and ECC, creating a cooperative strategy that not only strengthens the security of email communication but also streamlines its efficiency. RSA and ECC stand as widely embraced encryption methods for safeguarding email exchanges. RSA, a traditional approach, provides formidable security but can be computationally demanding, particularly when managing large files. Conversely, ECC excels in terms of encryption and decryption speed, making it an ideal choice for resource-constrained environments. The suggested hybrid method presents an inventive solution to the enduring challenge of striking a balance between security and performance in secure email communication. By harnessing the strengths of RSA and ECC, this hybrid technique provides an adaptable solution that can be tailored to meet specific email communication needs. Fig 5.2 provides a visual representation of the proposed fusion of RSA and ECC algorithms. This visual representation provides a clear and intuitive understanding of the sequential data transformations performed within the algorithm. For a more in-depth exploration of the inner workings of this proposed algorithm, Algorithms 4.1 and 4.2 offer a comprehensive breakdown of its structure. These algorithmic descriptions present a step-by-step elucidation of the specific operations and procedures involved at each stage of the algorithms.

```
Algorithm 4.1: Sender-side Operations.
Input: ECC key pair, RSA public key, plain text email
Output: Encrypted email, Digital Signature
Step1: Generate sender's ECC Private Key as $S^a$ and Public Key as $S^b$
Step 2: Request recipient's RSA public key from the server as $R^b$
Step 3: Derive shared secret using sender's ECC private key and recipient's RSA public key. $S = D (S^a, R^b)$
Step 4: Generate a symmetric key ($A^s$) for encrypting the email message.
Step 5: Compose the email message.
Step 6: Encrypt the email message using the symmetric key.
Ciphertext, $C^t = E (PT, A^s)$
Step 7: Encrypt the symmetric key using recipient's RSA public key.
$C^{As} = E (A^s, R^b)$
Step 8: Generate a digital signature for the email message.
$Ds = Gs (S^a, PT)$
Step 9: Send the secure email to the recipient
```

In Algorithm 4.1, the sender initiates secure email transmission by generating ECC Private and Public Keys (S^a and S^b). The recipient's RSA public key (R^b) is acquired, and a shared secret is derived through S^a and R^b. A symmetric key (A^s) is created for encrypting the email message using AES, ensuring confidentiality. The composed email message is encrypted with A^s, yielding ciphertext (C^t). For added security, A^s is encrypted with R^b, resulting in the encrypted symmetric key (C^{As}), safeguarding its transmission. To ensure data integrity and authentication, a digital signature (Ds) is generated using S^a and the plain text email (PT). The secure email, comprising C^t, C^{As}, and Ds, is then transmitted to the recipient. This algorithm thus combines ECC and RSA functionalities to achieve a comprehensive security framework, encompassing symmetric and asymmetric encryption, as well as digital signatures for secure email communication.

Algorithm 4.2: Recipient-side Operations.

```
Input: RSA key pair, ECC Key pair, Encrypted Email
Output: RSA Public Key, Decrypted email
Step1: Generate recipient's RSA and ECC key pairs.
RSA Private, Public ($R^a$, $R^b$) & ECC Private, Public ($S^a$, $S^b$)
Step 2: Export recipient's RSA public key for the sender
Step 3: Receive the encrypted symmetric key from the sender.
Step 4: Decrypt the symmetric key using recipient's RSA private key.
$A^s = D (C^{As}, R^a)$
Step 5:  Receive the encrypted email message.
Step 6: Verify the sender's ECC public key.
Step 7: Receive and verify the digital signature.
Step 8: Decrypt the email message using the symmetric key.
$PT = D (C^t, A^s)$
Step 9: View the decrypted email
```

In Algorithm 4.2, the recipient begins by generating RSA and ECC key pairs (R^a, R^b) and (S^a, S^b) respectively. The recipient's RSA public key (R^b) is exported for the sender's use. Upon receiving the sender's secure email, the recipient obtains the encrypted symmetric key (C^{As}) and decrypts it using their RSA private key, resulting in the symmetric key (A^s). The recipient then receives the encrypted email message (C^t) and proceeds to verify the sender's ECC public key. Subsequently, the digital signature is received and verified for authenticity and data integrity. Using the decrypted symmetric key (A^s), the email message is decrypted, yielding the plaintext email (PT).

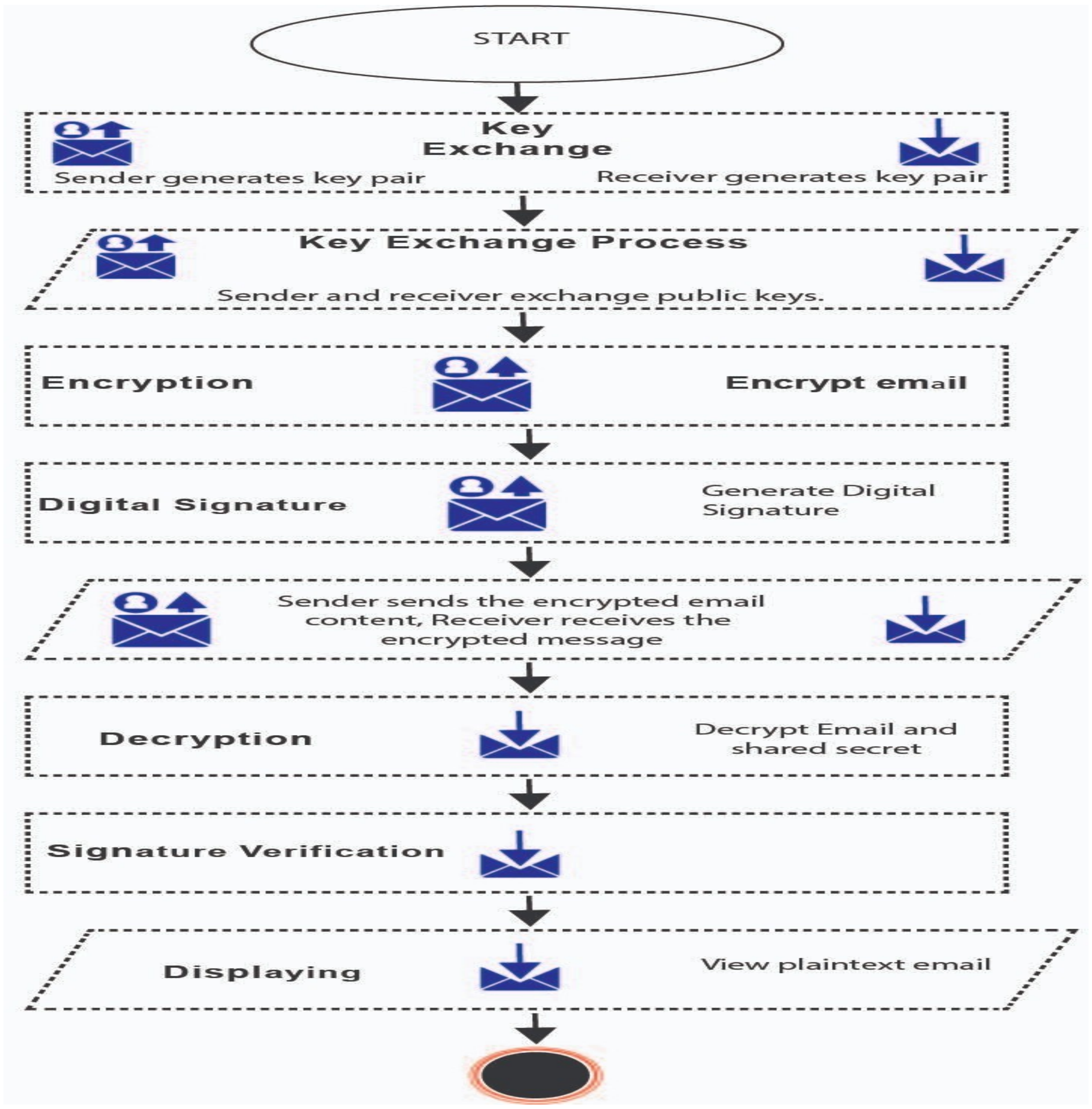

START
Key Exchange
Sender generates key pair
Receiver generates key pair
Key Exchange Process
Sender and receiver exchange public keys.
Encryption
Encrypt email
Digital Signature
Generate Digital Signature
Sender sends the encrypted email content, Receiver receives the encrypted message
Decryption
Decrypt Email and shared secret
Signature Verification
Displaying
View plaintext email

Figure 4.2: Proposed model flow graph with hybrid ECC, RSA and AES.

4.1.4. Performance Evaluation

The performance analysis is divided into two distinct approaches: analyzing the individual performance of RSA and ECC for secure email communication and assessing the performance of the hybrid approach for secure email communication.

A. Approach 1

i. Key Exchange Times measured in seconds

The key exchange time for RSA and ECC encryption methods in the context of secure email communication was assessed within the established testing environment. To measure key exchange times accurately, secure email communications were initiated, capturing the duration it took for public keys to be exchanged between sender and recipient during the initial email contact. This process was repeated 5 times for each test and the mean values were recorded. The recorded data for Key Exchange Time (KET) of RSA and ECC from the Secure Email Communication Test is as shown in the table 4.1.

Table 4.1: Key Exchange times of RSA and ECC.

TEST	KET RSA (seconds)	KET ECC (seconds)
1	0.172268	0.101002
2	0.164218	0.102000
3	0.179985	0.091002
4	0.177888	0.102220
5	0.164782	0.091001

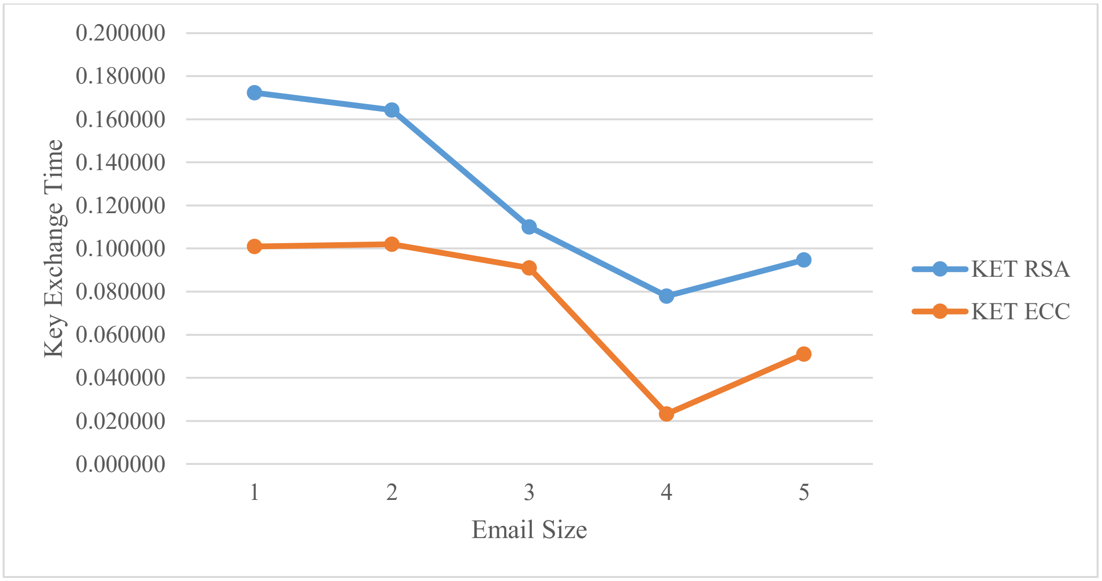


Figure 4.3: Key exchange analysis of ECC and RSA (in seconds).

ii. Encryption and Decryption times in seconds

Email encryption and decryption took place within the established test environment, consistent with the previously outlined specifications. Standardized email clients and servers were configured to support both RSA and ECC encryption methods. To assess these operations, a range of email messages, encompassing diverse sizes from text-based content to substantial attachments, was employed as test data. The system was configured to autonomously record the encryption and decryption times for each email, ensuring an impartial and objective measurement of efficiency and practicality. This methodology facilitated a thorough analysis of email encryption and decryption processes using RSA and ECC techniques within the secure email communication system. This study further used the hybrid encryption setup, the asymmetric encryption algorithms (RSA or ECC) facilitate secure key exchange, while the symmetric encryption algorithm (AES) ensures efficient and high-speed encryption and decryption of the email content. This combination strikes a balance between security and performance, making it a practical choice for secure email communication.

Table 4.2: Encryption time.

Sizes (MB)	**Encryption Time RSA (seconds)**	**Encryption Time ECC (seconds)**
10	0.024308	0.028219
50	0.106565	0.094268
100	0.235220	0.173252
200	0.481745	0.375365
500	0.938921	0.866455

Table 4.3: Decryption time.

Sizes (MB)	**Decryption Time RSA (seconds)**	**Decryption Time ECC (seconds)**
10	0.018158	0.016994
50	0.092173	0.068401
100	0.146187	0.153054
200	0.311700	0.308217
500	0.707802	0.753799

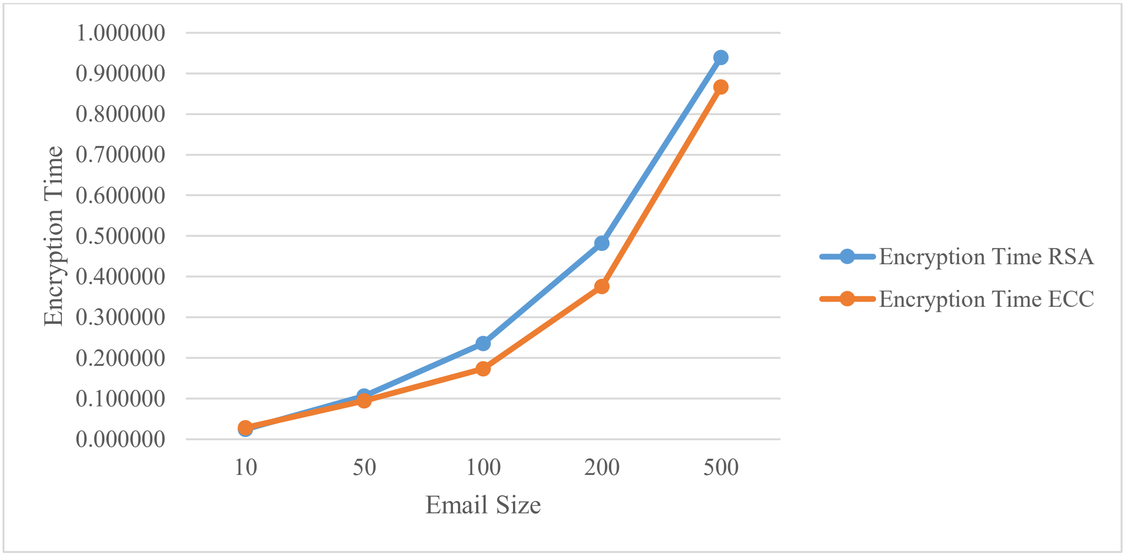


Figure 4.4: RSA and ECC Encryption Time (in seconds).

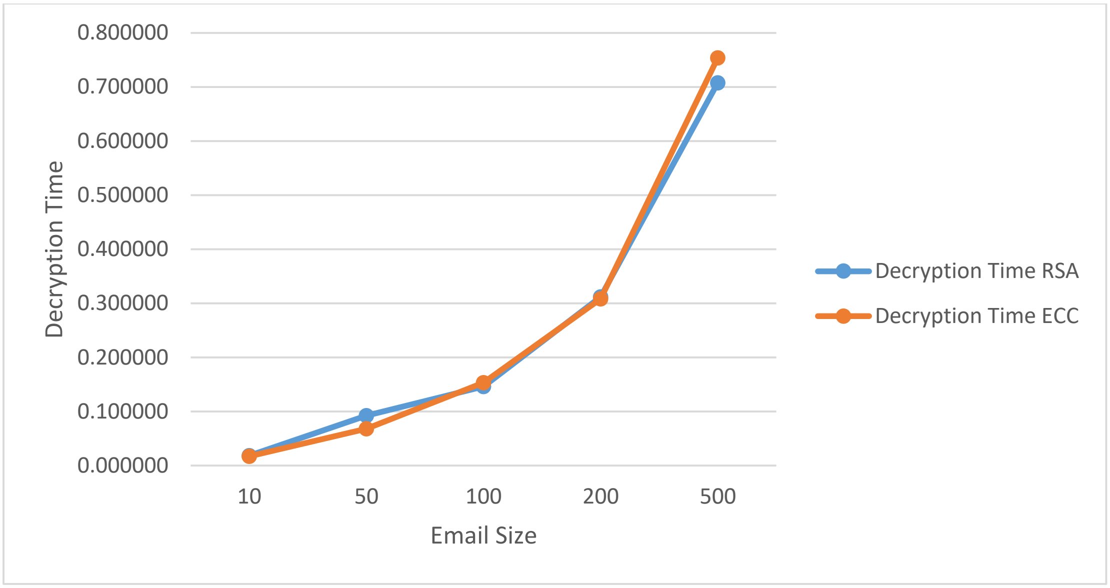


Figure 4.5: RSA and ECC Decryption Time (in seconds).

iii. Signature Generation and Verification

Signature generation and verification were integral aspects of the evaluation within the designated test environment, adhering to the established system specifications. By configuring standard email clients and servers to support both RSA and ECC encryption methods, the framework facilitated the generation of digital signatures for email messages. These signatures were generated autonomously during the test, and the system reported the time taken in seconds for each signature. Subsequently, the verification of these digital signatures occurred seamlessly within the same environment. A comprehensive analysis of signature generation and verification processes using RSA and ECC encryption methods was thus conducted objectively, with the system providing precise timing data for each operation.

Table 4.4: Signature Generation Time.

Sizes (MB)	**SGT RSA (seconds)**	**SGT ECC (seconds)**
10	0.030329	0.064428
50	0.127694	0.173007
100	0.278920	0.242878
200	0.451833	0.436492
500	1.239319	1.093490

Table 4.5: Signature Verification Time.

Sizes (MB)	**SVT RSA (seconds)**	**SVT ECC (seconds)**
10	0.028464	0.030986
50	0.146251	0.144860
100	0.274981	0.236938
200	0.535547	0.544687
500	1.236818	1.253438

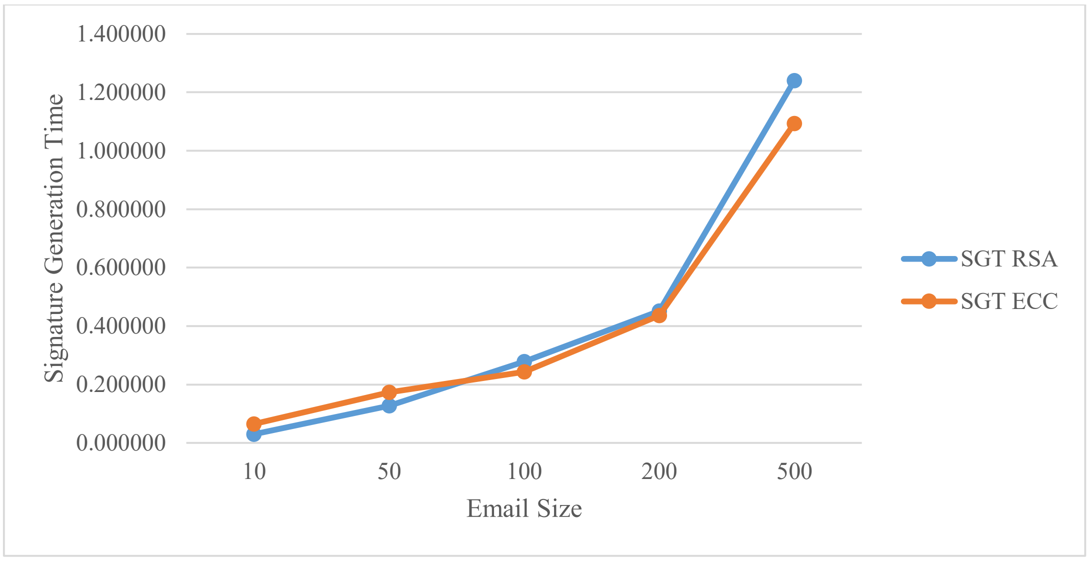


Figure 4.6: Signature generation of ECC and RSA (in seconds).

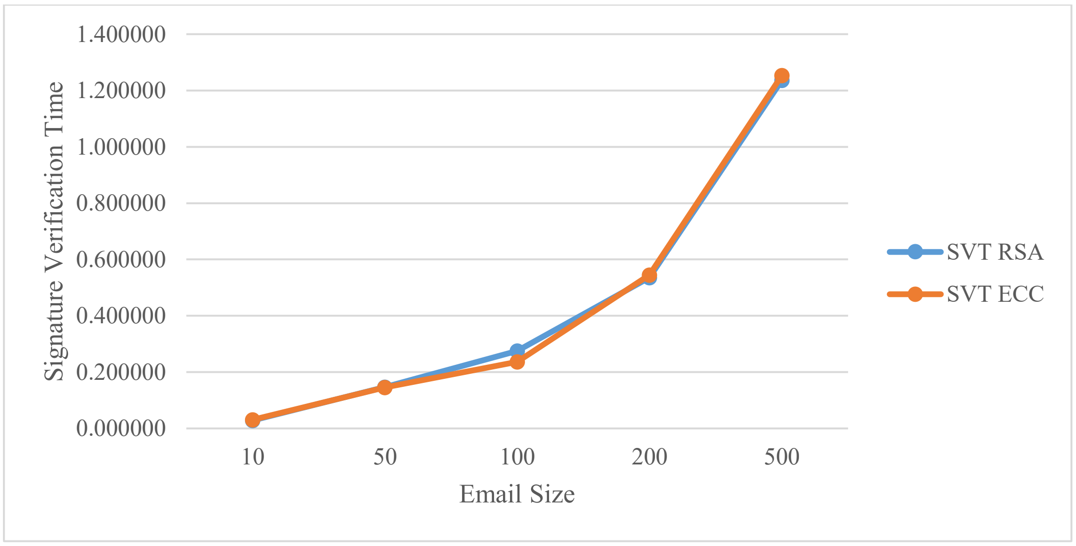


Figure 4.7: Signature verification of ECC and RSA (in seconds).

B. Approach 2

i. Key Exchange Times measured in seconds

The key exchange time for the fusion of RSA and ECC encryption methods was assessed within the established testing environment. To measure key exchange times accurately, secure email communications were initiated, capturing the duration it took for keys to be exchanged between sender and recipient during the initial email contact. This process was repeated 5 times for each test, and the mean values were recorded in Table 4.6.

Table 4.6: Key exchange times for Hybrid technique, Solo RSA and Solo ECC.

TEST	Hybrid KET (seconds)	KET RSA (seconds)	KET ECC (seconds)
1	0.064191	0.172268	0.101002
2	0.112602	0.164218	0.102000
3	0.070835	0.179985	0.091002
4	0.068739	0.177888	0.102220
5	0.067263	0.164782	0.091001

ii. Encryption, Decryption, Signature generation and verification times in seconds

Table 4.7 displays the encryption, decryption, signature generation, and verification performance metrics for the hybrid technique. To evaluate these operations, a variety of email messages with different sizes, ranging from text-based content to sizable attachments, were utilized as test data. The system was set up to automatically capture the times taken for encryption, decryption, signature generation, and verification for each email. For each experiment, this procedure was carried out five times, and the mean values were obtained.

Table 4.7: Comparing Hybrid Technique (RSA and ECC) to Standard ECC and RSA.

Hybrid Technique					ECC Encryption Time	RSA Encryption Time	ECC Decryption Time	RSA Decryption Time	Signature Generation Time ECC	Signature Generation Time RSA	Signature Verification Time ECC	Signature Verification Time RSA
Sizes (MB)	Encryption Time	Decryption Time	Signature Generation Time	Signature Verification Time								
10	0.020769	0.014005	0.026106	0.026000	0.028219	0.024308	0.016994	0.018158	0.030329	0.030329	0.030986	0.028464
50	0.091153	0.062974	0.130278	0.120312	0.094268	0.106565	0.068401	0.092173	0.127694	0.127694	0.144860	0.146251
100	0.156722	0.140040	0.238006	0.262419	0.173252	0.235220	0.153054	0.146187	0.278920	0.278920	0.236938	0.274981
200	0.327086	0.307289	0.430858	0.423738	0.375365	0.481745	0.308217	0.311700	0.451833	0.451833	0.544687	0.535547
500	0.832917	0.636395	1.073605	1.160965	0.866455	0.938921	0.753799	0.707802	1.239319	1.239319	1.253438	1.236818

4.1.5. Discussion of results

Table 4.1 presents the collected data concerning the Key Exchange Time (KET) for RSA and ECC in the context of secure email communication. The results consistently indicate that ECC outperforms RSA in terms of key exchange efficiency across all the tested scenarios. In practical terms, it implies that the process of establishing secure communication channels through key exchange is notably swifter and more efficient when utilizing ECC as the cryptographic algorithm, as opposed to RSA.

Table 4.2 reports the Encryption Time (in seconds) for both RSA and ECC in the same context. It illustrates the time taken to encrypt emails with various file sizes, ranging from 10 MB to 500 MB. The results demonstrate interesting trends in the performance of these cryptographic algorithms during the encryption process.

For smaller file sizes (10 MB and 50 MB), RSA exhibits slightly shorter encryption times compared to ECC. However, as the file sizes increase to 100 MB, 200 MB, and 500 MB, ECC consistently demonstrates superior efficiency in encryption. ECC's encryption times remain notably lower than RSA's for all these larger file sizes, suggesting that ECC is particularly well-suited for securing and transmitting larger email attachments. This outcome highlights ECC's efficiency in handling data encryption tasks for secure email communication, particularly when dealing with substantial file sizes. While RSA performs reasonably well for smaller files, ECC emerges as a more efficient choice as the data to be encrypted grows in size.

Just as Encryption Time from Table 4.2, Decryption Time (in seconds) for both RSA and ECC in table 4.3 provides insights into the time required to decrypt emails with various file sizes, ranging from 10 MB to 500 MB. The results reveal several noteworthy observations. For smaller file sizes (10 MB and 50 MB), ECC demonstrates slightly shorter decryption times compared to RSA, indicating its efficiency in handling smaller data. However, as the file sizes increase to 100 MB, 200 MB, and 500 MB, RSA exhibits competitive or slightly shorter decryption times than ECC. This suggests that RSA can be advantageous for decrypting larger email attachments efficiently.

Data presented in Table 4.4 shows results for the Signature Generation Time (in seconds) for both RSA and ECC. The results reveal that for smaller file sizes (10 MB and 50 MB), RSA demonstrates notably shorter signature generation times compared to ECC, showcasing its efficiency in handling small data for signature creation. However, as file sizes increase to 100 MB, 200 MB, and 500 MB, ECC gradually catches up and, in some cases, surpasses RSA in terms of signature generation efficiency. This suggests that ECC is better suited for efficiently generating signatures for larger email attachments.

Table 4.5 provides insights into the time required to verify digital signatures for emails with same attached files as stated earlier. The results demonstrate interesting patterns in signature verification efficiency. For smaller file sizes (10 MB and 50 MB), ECC exhibits slightly longer verification times compared to RSA. However, as file sizes increase to 100 MB, 200 MB, and 500 MB, ECC's verification time becomes comparable to or slightly shorter than RSA's. This indicates that ECC is competitive with RSA in terms of signature verification efficiency, particularly for larger email attachments. The signature verification time findings suggest that ECC is a viable choice for verifying digital signatures, especially for larger data sizes. While RSA may have a slight advantage for smaller files, the efficiency of ECC becomes evident as the data size increases. The selection between RSA and ECC for signature verification should consider the typical email attachment sizes used in practice to optimize performance and efficiency.

In Table 4.6, the hybrid technique demonstrates better key exchange time (KET) compared to solo RSA and ECC implementation. This indicates that the process of establishing secure communication channels through key exchange is notably swifter and more efficient when utilizing the proposed hybrid algorithm as opposed to RSA and ECC. Finally, Table 4.7 presents the time (in seconds) recorded for encryption, decryption, signature generation, and signature verification in the context of secure email communication using the hybrid algorithm. When conducting a comparison, it becomes evident that on average, the hybrid encryption algorithm enhances the efficiency of encryption and decryption times, as well as signature generation and verification times. In certain instances, the individual ECC times displayed slightly better performance compared to the hybrid algorithm, indicating a close correlation between ECC and the hybrid approach. In summary, the proposed Hybrid technique excels in providing a versatile and efficient encryption solution for secure email communication across a wide range of email message sizes.

4.3. Conclusion

This study provides an analysis of key cryptographic algorithms, namely RSA, ECC and the hybrid algorithm in the context of securing email communications. The study reveals that ECC excels in terms of key exchange efficiency and effectively manages larger email attachments, making it an attractive choice for enhancing the security of modern email systems. While RSA performs adequately for smaller data sizes, ECC consistently outperforms it as data sizes increase, positioning it as a more efficient cryptographic algorithm for securing email communication. The experimental outcomes indicate that the suggested hybrid solution offers a more efficient method for encrypting email messages, with only a minimal disparity in runtime when compared to the ECC algorithm. Furthermore, this solution ensures a high level of security for secure email communication. These findings offer valuable insights for practical optimization of email security. The hybrid algorithm introduced in this paper shows promise for being applied in system design, software development, and various other domains, offering an effective means of protecting data. In the future, this research can be further developed by enhancing the security of the hybrid approach. The integration of multiple security layers offers the potential to improve the system's productivity and efficiency.

Chapter 5
Improved RSA and AES Frameworks

This chapter is in line with research objectives RO 3 and RO 4, focusing on the optimizing the AES and RSA cryptographic frameworks. The AES and RSA are the most esteemed symmetric and asymmetric algorithms, widely utilized in numerous systems, services, and applications. To attain these objectives, the study formulates specific tasks aimed at devising novel frameworks for the AES and RSA algorithms, enhancing its efficacy in secure data transmission.

In this chapter, every task is broken down into distinct scenarios, with each section being considered as standalone journals or accepted conference papers. Each task or scenario in the chapter has its own specific motivation.

Finally, the study introduces a hybrid approach—an integrated methodology—combining Multi-Chaotic AES and Modified AES MixColumn with Optimized RSA Key Generation, utilizing a cloud platform as a case study.

5.1. Motivation for optimizing AES operations in File Encryption Software

The AES is widely used in file encryption software to guarantee the secrecy and integrity of files and folders. Popular software like VeraCrypt and BitLocker employ AES encryption as the core algorithm for securing data when it's not in use. To extend AES's usability even more, there's a crucial need to develop a high-performance version. Thus, this research introduces a refined encryption architecture known as LZMA-AES, which incorporates the compression technique as an extra layer to the existing AES, enhancing the encryption process for file software on standalone personal computers running Windows.

5.1.1. Introduction

Information stored on physical storage devices and communicated through channels frequently contains redundant data. Compression techniques are employed to minimize this redundancy, conserving space and reducing the time required for data transmission. Data compression brings about a reduction in the required storage space, particularly beneficial for managing large datasets, files, or archives. These compression techniques efficiently eliminate redundancy, improve pattern encoding, and ultimately shrink the overall data size. As a consequence, significant space savings are achieved, notably on storage devices or servers [159]. The necessity for robust security measures, including the control of secret keys in specific techniques, gives rise to concerns about the potential exposure of data to attacks. Encryption is crucial in safeguarding information and preserving its confidentiality by employing a secret key to render the data unreadable and unalterable. Information security has become more critical than ever for several reasons. Therefore, encryption is predominantly employed to maintain confidentiality [110]. Encryption algorithms can be classified into two types: symmetric key encryption and asymmetric key encryption. In symmetric key encryption, the same key is used for both encrypting and decrypting data, whereas asymmetric encryption uses different keys for these operations [2]. Examples of symmetric encryption algorithms are AES (Advanced Encryption Standard), DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm), and Blowfish. Examples of asymmetric encryption techniques include Elliptic curve cryptography (ECC), Rivest-Shamir-Adleman (RSA) algorithm, Diffie-Hellman key exchange, and Digital Signature Algorithms (DSA) [225]. The Advanced Encryption Standard (AES) algorithm is a popular and extensively utilized symmetric block encryption algorithm globally. It is frequently employed in wireless networks, e-commerce, and various other applications. With its distinct structure, this method is widely utilized in hardware and software for encrypting and decrypting confidential documents. Decrypting data that has been encrypted using the AES algorithm is incredibly difficult for hackers, making it a reliable choice for securing sensitive information [1]. The AES algorithm is employed in various applications, including messaging platforms like Signal and WhatsApp, virtual private networks that ensure secure data transmission between the user's device and the VPN server, file and disk encryption software that adds an additional layer of protection to documents on personal computers, online banking and payment systems, email services, and more [11][226]. The major contribution of this work are as follows:

- To develop an integrated solution for seamlessly combining AES encryption with the Lempel-Ziv-Markov chain compression algorithm to ensure both security and efficient data transmission.
- To explore the specific features of the Lempel-Ziv-Markov chain algorithm to optimize data compression while maintaining the security standards provided by AES.

- To investigate the practical implications of the combined AES and Lempel-Ziv-Markov chain in file encryption software, considering factors such as speed, resource utilization, and ease of implementation.

5.1.2. Lempel-Ziv-Markov (LZMA) chain algorithm

The LZMA compression algorithm, originally proposed by Pavlov in 1998 forms its core by enhancing the LZ77 compression algorithm. While incorporating a sliding window-based dynamic dictionary compression algorithm and interval coding algorithm, LZMA offers notable advantages, including a high compression rate, minimal decompression space requirement, and rapid processing speed. The typical LZMA workflow involves the implementation of the sliding window algorithm based on LZ77 and interval encoding (range encoding) [227][228].

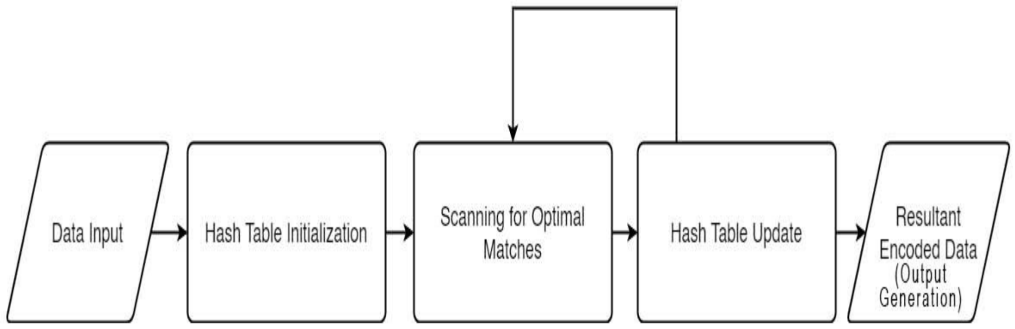

Data Input
Hash Table Initialization
Scanning for Optimal Matches
Hash Table Update
Resultant Encoded Data (Output Generation)

Figure 5.1: LZMA Workflow.

As illustrated in the diagram figure 5.1, LZMA algorithm operates in a systematic workflow beginning with the input data. It initializes a hash table and proceeds to scan the data, identifying the best match through a sliding encode mechanism. Following this, the algorithm refreshes the hash table, adapting to the evolving data. The encoding process then takes place, utilizing a combination of distance, information needed, and the next symbol in the range encoding scheme. The output is the encoded data, reflecting the compressed representation achieved through the intricate interplay of hash table management, sliding encoding, and range encoding. This workflow encapsulates the essence of how LZMA efficiently compresses data while maintaining the essential information needed for accurate decompression. LZMA achieves remarkable compression rates by leveraging an expansive dictionary space of 4 KB to hundreds of MBs. Efficient navigation through this extensive search space is facilitated by a well-crafted hash table storing potential longest matches, employing hash linked lists or binary search trees. The hash function maps the first two bytes of the search cache to a hash array, optimizing storage of matching character group positions. LZMA enhances efficiency with a multi-level hash function accommodating varying dictionary sizes, showcasing the algorithm's prowess in achieving exceptional compression ratios without compromising computational speed. The selection of LZMA [1] to enhance the AES algorithm stems from its proven efficiency in data compression [227].

Other alternatives considered include Zstandard (ZSTD)[2], Brotli[3], and ZLIB[4]. ZSTD is notable for its impressive speed and compression ratio, closely matching LZMA while offering faster processing, making it a compelling option for applications requiring rapid compression and decompression. Brotli, primarily used in web applications, strikes a balance between speed and compression ratio, though its efficiency is slightly lower than LZMA's. ZLIB, a widely adopted algorithm, offers faster compression speeds but with lower compression ratios, making it suitable for environments where backward compatibility and reduced memory usage are critical. Despite these alternatives, LZMA's balance of compression efficiency and security integration made it the preferred choice for our specific requirements. LZMA's adeptness in significantly reducing the size of data without compromising its integrity aligns with the goal of optimizing the AES algorithm for enhanced file encryption. By integrating LZMA's compression capabilities, this study aims to mitigate potential performance bottlenecks associated with encrypting large datasets. This practical approach seeks to leverage LZMA's ability to compress data effectively, thereby improving the overall speed and efficiency of the file encryption process. The choice of LZMA reflects a strategic decision to enhance AES with a compression algorithm known for its practical and tangible benefits in real-world encryption scenarios.

[1] https://github.com/LZMA-JS/LZMA-JS

[2] https://github.com/facebook/zstd

[3] https://github.com/google/brotli

[4] https://github.com/madler/zlib

5.1.2. Experimental setting

The experiment tested the proposed encryption technique on twenty (20) workstations and the workstations were classified according to their respective specifications. The following presents the categorization breakdown along with the corresponding users.

Table 5.1: Machine Specifications.

S/n	Categories	Specifications	Number of Machines used
1	A	Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz 16.0 GB RAM	10
2	B	Intel(R) Core (TM) i7-6500U CPU @ 2.50GHz 2.60GHz 16.0 GB RAM	10

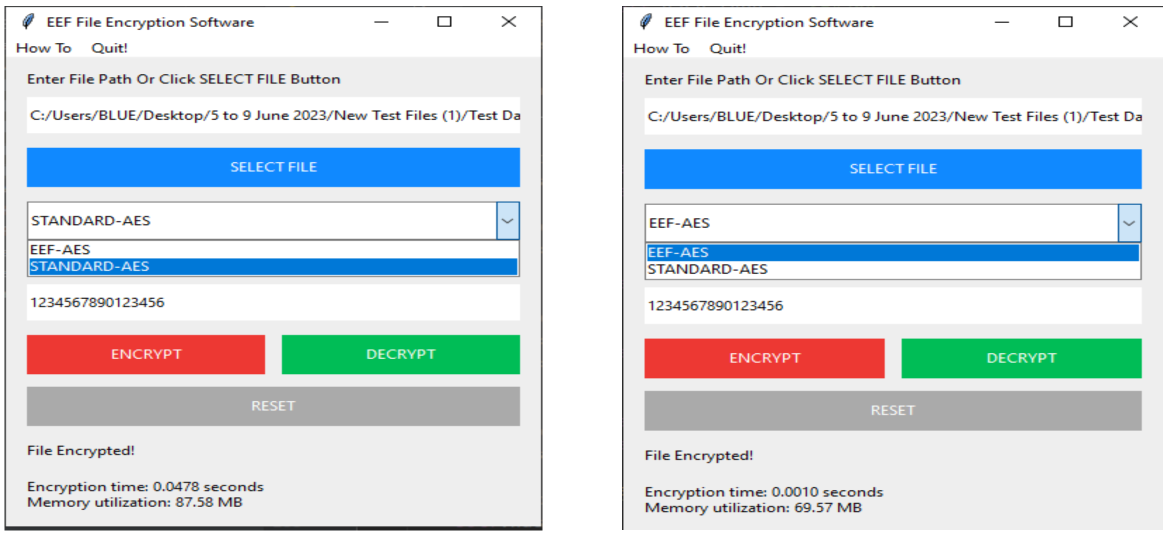
In the experimental setup, the benchmark employed was the standard AES (Advanced Encryption Standard). The testing scenarios were designed to evaluate the performance of LZMA-AES in comparison to the standard AES algorithm. The comparison involved rigorous testing under various conditions to assess how LZMA-AES stands against the current AES standard. The scenarios were structured to analyze key metrics such as encryption speed, computational efficiency, and security strength, providing a comprehensive understanding of how LZMA-AES performs relative to the established AES benchmark. The experiment was conducted on each machine twelve (12) times to ensure accuracy and consistency. The resulting averages for each category were then calculated and recorded. This systematic approach provided a comprehensive analysis of the performance across different machine categories, allowing for a more in-depth examination of the data. To maintain consistency and standardize the experiment, all users are assigned files of the same types and sizes. A pre-designed dataset[5] was employed to conduct experiments on both LZMA-AES and standard AES. The file types and their corresponding sizes are as follows: DOC: 5MB, MP3_8MB, MP4: 18MB, PDF: 10MB, PPT: 10MB, TXT: 10MB and XLS: 6MB. This approach ensures that any observed results or findings are not influenced by variations in file types or sizes, leading to a more reliable and robust outcome. A range of file types

[5] Ultra Hi-Speed Direct Test Files Download

were utilized to compare the performance of the proposed algorithm with the standard AES algorithm.

5.1.3. Proposed Algorithm (AES+LZMA)

In the proposed algorithm, the fusion of the AES and LZMA algorithms for file encryption and decryption follows a systematic process. There is the initialization process which involves defining the AES key, initialization vector (IV), and initializing the LZMA dictionary with compression parameters. The input data is segmented into blocks suitable for both algorithms. During interleaved compression and encryption, each data block undergoes LZMA compression using the LZMA dictionary, followed by encryption using the AES algorithm with a specified key and IV. The resulting encrypted, compressed block is stored for later reconstruction. In the decryption phase, the stored blocks are sequentially decrypted using the original AES key and IV, followed by LZMA decompression to reconstruct the original data. This approach ensures data security through encryption while optimizing storage and transmission efficiency through compression. The fusion of AES and LZMA presents a robust solution for secure file encryption and decryption. In figure 5.2, to streamline the experimentation process, a unified solution has been developed, integrating both the LZMA-AES and AES algorithms. This integration aims to simplify the execution of the experiment by providing a convenient interface that allows seamless switching between the two algorithms. Algorithm 5.1 presents a comprehensive breakdown of its structure.



Enter File Path Or Click SELECT FILE Button

C:/Users/BLUE/Desktop/5 to 9 June 2023/New Test Files (1)/Test Da

STANDARD-AES

1234567890123456

File Encrypted!

Encryption time: 0.0478 seconds
Memory utilization: 87.58 MB

Enter File Path Or Click SELECT FILE Button

C:/Users/BLUE/Desktop/5 to 9 June 2023/New Test Files (1)/Test Da

EEF-AES

1234567890123456

File Encrypted!

Encryption time: 0.0010 seconds
Memory utilization: 69.57 MB

Figure 5.2: Simulation of the AES and LZMA-AES.

5.1.3.1. Interleaved Compression and Encryption

For each data block, an interleaved compression and encryption process is applied:

LZMA Compression

The data block is subjected to LZMA compression, utilizing LZMA's dictionary and compression parameters. LZMA compression is performed using the following formula:

$Cb = \delta\ (Db, D, P)$

$Cb \in \sum* \ \rightarrow \delta(Db \ \in \sum* \ , D, P)$

Where:

- Cb denotes the Compressed Block, representing the output of compression.
- δ represents the LZMA compression function, acting as a mathematical operator that performs compression.
- Db signifies the Data Block, indicating the input data to be compressed.
- D stands for the LZMA Dictionary, symbolizing the memory component storing recurring patterns.
- P denotes the Compression Parameters, which control the algorithm's behaviour.
- $\in$ is used to specify that Cb and Db belong to the set of all possible byte sequences.
- $\rightarrow$ is employed to illustrate that the compression function maps Db to Cb.
- $\sum*$ represents the set of all finite-length sequences of bytes.

AES Encryption

The compressed block is encrypted using the AES algorithm with the specified key and IV. AES encryption is expressed as: $Eb = \varepsilon\ (Cb, K, IV)$

$Eb \in \sum* \ \rightarrow \varepsilon(Cb \ \in \sum* \ , K, IV)$

Where:

- Eb denotes the Encrypted Block, representing the output of encryption.
- ε symbolizes the AES encryption function, acting as a mathematical operator that performs encryption.
- Cb signifies the Compressed Block, serving as the input to the encryption process.
- K designates the AES Key, the secret key used for encryption and decryption.
- IV stands for the Initialization Vector, a random value ensuring distinct ciphertext for identical plaintexts.

Storage of Encrypted, Compressed Block

The resulting encrypted, compressed block is stored for later reconstruction: $Sb = \mathcal{S}\ (Eb)$

Decompression and Decryption

To retrieve the original data, the stored blocks undergo a reverse process:

1. **AES Decryption:**
    - Each stored block is decrypted using the AES algorithm with the original key and IV: $Db = \epsilon^{-1}(Sb, K, IV)$
2. **LZMA Decompression:**
    - The decrypted block is then subjected to LZMA decompression to recover the original data: $ODb = \delta^{-1}\ (Db, D)$
3. **Reconstruction of the Original Data:**
    - The original data is reconstructed by repeating the decompression and decryption process for each stored block.

Algorithm 5.1: LZMA + AES Algorithms.

```
Step 1: Initialization
Define AES key and initialization vector (IV).
Initialize LZMA dictionary and compression parameters.
Step 2: Data Segmentation
Break input data into blocks suitable for both AES and LZMA (e.g., 128-bit blocks for AES,
considering LZMA's compression efficiency).
Step 3: Interleaved Compression and Encryption
For each data block:
// LZMA Compression
Compressed_Block = LZMA_Compress(Data_Block)
// AES Encryption
Encrypted_Block = AES_Encrypt(Compressed_Block, AES_Key, IV)
// Store the encrypted, compressed block
Store (Encrypted_Block)
```

```
Decompression and Decryption
For each stored block:
// AES Decryption
Decrypted_Block = AES_Decrypt(Stored_Block, AES_Key, IV)
// LZMA Decompression
Original_Data_Block = LZMA_Decompress(Decrypted_Block)
// Reconstruct the original data
Reconstruct (Original_Data_Block)
```

5.1.4. Performance Analysis

5.1.4.1. Encryption and Decryption Times

A comparison of the encryption and decryption times in seconds between the LZMA-AES and the standard AES algorithms are presented in Table 5.2.

$$\boldsymbol{Exec_{Time}} = \frac{exec1 + exec\ 2 + \cdots + exce\_n}{NTimesofExec} \qquad (1)$$

Table 5.2: Average Process times for Categories A and B.

File type	**File size Kb**	**CATEGORY A: Process Time in seconds**				**CATEGORY B: Process Time in seconds**			
		Standard AES	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**
		Encryption	Encryption	Decryption	Decryption	Encryption	Encryption	Decryption	Decryption
DOC	5001	0.0224	0.0004	0.0296	0.0010	0.0246	0.0010	0.0346	0.0003
MP3	8218	0.0329	0.0294	0.0371	0.0344	0.0396	0.0362	0.0518	0.0419
MP4	17422	0.0624	0.0539	0.0647	0.0597	0.0841	0.0758	0.0898	0.0857
PDF	10386	0.0422	0.0362	0.0454	0.0412	0.0588	0.0519	0.0509	0.0492
PPT	10048	0.0349	0.0389	0.0471	0.0467	0.1729	0.0482	0.0528	0.0499
TXT	10240	0.0405	0.0001	0.0388	0.0001	0.0482	0.0001	0.0488	0.0001
XLS	6808	0.0252	0.0003	0.0013	0.0003	0.0359	0.0329	0.0030	0.0023

5.1.4.2. Throughput (Speed)

Table 5.3 provides a comparison of the encryption and decryption throughput of the LZMA-AES and the standard AES algorithm. The values depicted in the table are calculated using the formula presented in equation (2).

$$\boldsymbol{Throughput} = \frac{File\ Size\ (kb)}{Exec_{Time}(sec)} \qquad (2)$$

Table 5.3: Average Throughput in Kb/Seconds for Categories A and B.

File type	File size Kb	CATEGORY A: Process Time in seconds				CATEGORY B: Process Time in seconds			
		Standard AES	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**
		Encryption	Encryption	Decryption	Decryption	Encryption	Encryption	Decryption	Decryption
DOC	5001	223258.929	12502500	168952.703	5001000	203292.683	5001000	144537.572	16670000
MP3	8218	249787.234	279523.81	221509.434	238895.349	207525.253	227016.57	158648.649	196133.6516
MP4	17422	279198.718	323228.2	269273.57	291825.796	207158.145	229841.69	194008.909	203290.5484
PDF	10386	246113.744	286906.08	228766.52	252087.379	176632.653	200115.61	204047.151	211097.561
PPT	10048	287908.309	258303.34	213333.333	215160.6	58114.5171	208464.73	190303.03	201362.7255
TXT	10240	252839.506	102400000	263917.526	102400000	212448.133	102400000	209836.066	102400000
XLS	6808	270158.73	22693333	5236923.08	22693333.3	189637.883	206930.09	2269333.33	2960000

5.1.4.3. Memory utilization

The results of the LZMA-AES and the standard AES algorithms are compared in Table 5.4 in terms of encryption and decryption memory utilization. In this test psutil library is used to obtain memory utilization. Psutil.Process is used to retrieve the process information using the process ID. The memory_info() method is then used to obtain memory usage information in bytes, and this is converted to megabytes using the formula (3);

$$\boldsymbol{Memory\ util\ (MB)} = \frac{memory\_info(bytes)}{1024*2} \qquad (3)$$

The rss attribute of the memory_info() method returns the Resident Set Size (RSS), which is the portion of a process's memory that is held in RAM.

$$\boldsymbol{Memory_{Util}} = \frac{exec\_1\_mem + exec\ 2\_mem + \cdots}{NTimesofExec} \qquad (4)$$

Table 5.4: Encryption and Decryption Memory Utilization for Categories A and B.

File type	File size Kb	CATEGORY A: Process Time in seconds				CATEGORY STB: Process Time in seconds			
		Standard AES	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**
		Encryption	Encryption	Decryption	Decryption	Encryption	Encryption	Decryption	Decryption
DOC	5001	78.010	72.047	77.500	67.253	69.277	61.137	66.577	54.747
MP3	8218	85.157	87.480	84.907	82.637	76.143	76.650	70.987	70.573
MP4	17422	103.623	103.933	103.440	99.047	94.270	93.063	88.820	87.353
PDF	10386	89.987	92.233	89.763	87.110	80.660	81.223	75.380	75.727
PPT	10048	89.907	90.493	86.920	86.007	80.017	79.797	75.390	74.480
TXT	10240	90.657	71.117	87.470	66.910	80.627	60.810	75.247	55.453
XLS	6808	84.257	73.540	80.917	68.617	73.987	62.283	68.760	57.057

5.1.4.4. Power Consumption

When a system or device performs encryption operations, it uses energy. The formula below is used to compute the energy consumption based on the encryption and decryption times of the LZMA-AES and AES algorithms. This includes the power consumed by the hardware or software components involved in encryption, such as the CPU, memory, and encryption algorithm. The power consumption can be affected by various factors, such as the size of the data being encrypted, the encryption algorithm used, and the operating temperature of the device. In general, encryption power consumption is an important consideration in many applications, particularly those involving battery-powered devices or those requiring high levels of security while minimizing energy consumption. The formulas below measured in joules are used to compute the energy consumption based on the encryption and decryption times of the LZMA-AES and AES algorithms in Table 6.15.

1) Computing for the PC Wattage

$$\boldsymbol{Total\ PC\ WATTS} = Input\ Current\ (Amps) * Input\ Voltage\ (Volts) \tag{5}$$

where the input current = 1.5A and input voltage = 19.5V

$$\boldsymbol{Total\ PC\ WATTS} = 29.25 \text{ Watts}$$

2) Computing for the power consumption measured in Joules

1 kWh = 3.6Million Joules

$$\boldsymbol{Total\ power\ consumption} = \frac{\text{PC Watts} * \text{Running Hours}}{1000} \tag{6}$$

Table 5.5: Power Consumption for Categories A and B.

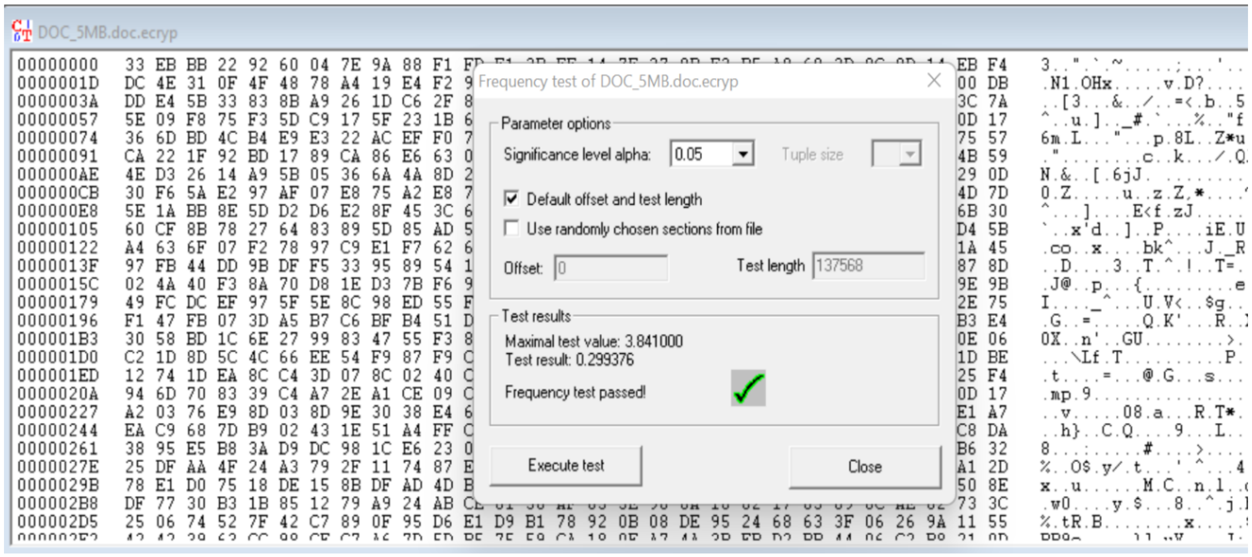
File type	File size Kb	CATEGORY A: Process Time in seconds				CATEGORY B: Process Time in seconds			
		Standard AES	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**	**Standard AES**	**LZMA-AES**
		Encryption	Encryption	Decryption	Decryption	Encryption	Encryption	Decryption	Decryption
DOC	5001	0.6552	0.0117	0.8658	0.02925	0.71955	0.02925	1.01205	0.008775
MP3	8218	0.962325	0.85995	1.085175	1.0062	1.1583	1.05885	1.51515	1.225575
MP4	17422	1.8252	1.576575	1.892475	1.746225	2.459925	2.21715	2.62665	2.506725
PDF	10386	1.23435	1.05885	1.32795	1.2051	1.7199	1.518075	1.488825	1.4391
PPT	10048	1.020825	1.137825	1.377675	1.365975	5.057325	1.40985	1.5444	1.459575
TXT	10240	1.184625	0.002925	1.1349	0.002925	1.40985	0.002925	1.4274	0.002925
XLS	6808	0.7371	0.008775	0.038025	0.008775	1.050075	0.962325	0.08775	0.067275

5.1.5. Vulnerability Testing

In this section, the vulnerability of the proposed LZMA-AES algorithm was assessed through the utilization of two methods: frequency analysis and brute-force cryptanalysis.

5.2.5.1. Frequency analysis

The study records the results of the frequency test analysis for each encrypted file. All ciphertext produced were examined in the study, with representative samples of the analysis presented in figures 5.3 to 5.9. Frequency analysis remains a fundamental and powerful technique in cryptography. Its ability to exploit the non-randomness of ciphertext distributions allows for the identification of patterns, vulnerabilities, and potential weaknesses in encryption schemes. A third-party tool called CryptTool is utilized for analyzing ciphertext and revealing patterns or weaknesses. These modules make use of the frequency distribution of characters, symbols, or bit patterns in encrypted data to offer insights into the original file and encryption process. Through the application of frequency analysis techniques found in CrypTool, the research obtained a more comprehensive comprehension of encryption schemes and assessed their resilience against frequency-based attacks [40][41]. The results include the maximal test value, which represents the critical value for the chosen significance level, and the test result, which indicates the computed test statistic. A significance level of 0.05 was chosen for each of the test. These results provide insights into the statistical properties of the ciphertext's character frequencies and allow for the assessment of ciphertext indistinguishability.



DOC_5MB.doc.ecryp

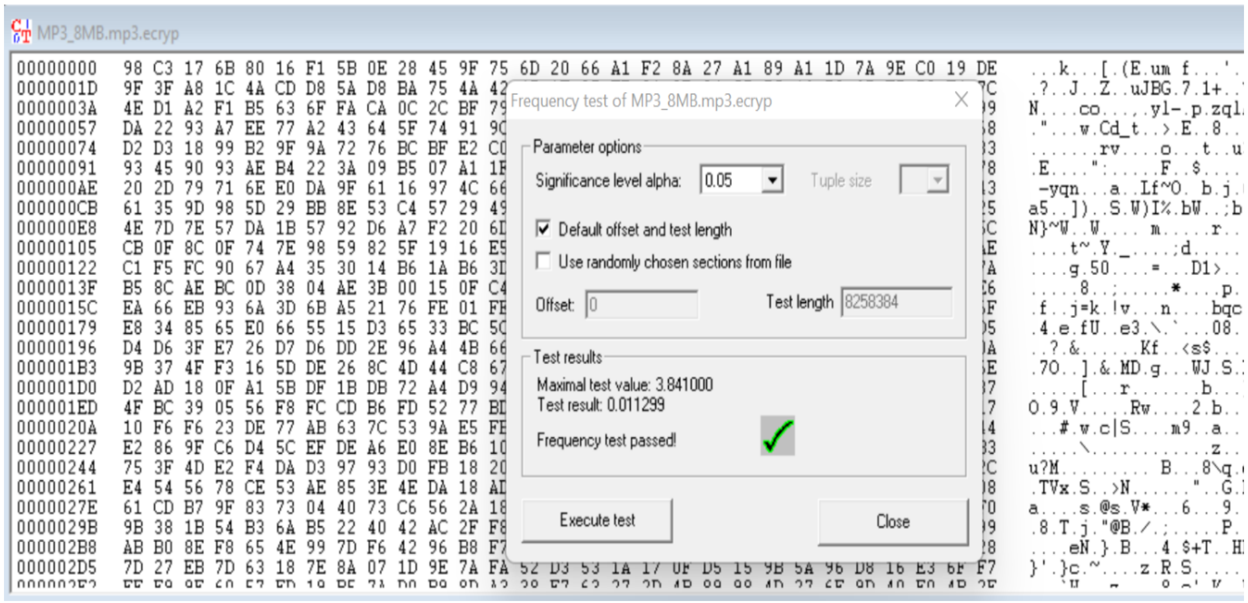
```
00000000   33 EB BB 22 92 60 04 7E 9A 88 F1
0000001D   DC 4E 31 0F 4F 48 78 A4 19 E4 F2
0000003A   DD E4 5B 33 83 8B A9 26 1D C6 2F
00000057   5E 09 F8 75 F3 5D C9 17 5F 23 1B
00000074   36 6D BD 4C B4 E9 E3 22 AC EF F0
00000091   CA 22 1F 92 BD 17 89 CA 86 E6 63
000000AE   4E D3 26 14 A9 5B 05 36 6A 4A 8D
000000CB   30 F6 5A E2 97 AF 07 E8 75 A2 E8
000000E8   5E 1A BB 8E 5D D2 D6 E2 8F 45 3C
00000105   60 CF 8B 78 27 64 83 89 5D 85 AD
00000122   A4 63 6F 07 F2 78 97 C9 E1 F7 62
0000013F   97 FB 44 DD 9B DF F5 33 95 89 54
0000015C   02 4A 40 F3 8A 70 D8 1E D3 7B F6
00000179   49 FC DC EF 97 5F 5E 8C 98 ED 55
00000196   F1 47 FB 07 3D A5 B7 C6 BF B4 51
000001B3   30 58 BD 1C 6E 27 99 83 47 55 F3
000001D0   C2 1D 8D 5C 4C 66 EE 54 F9 87 F9
000001ED   12 74 1D EA 8C C4 3D 07 8C 02 40
0000020A   94 6D 70 83 39 C4 A7 2E A1 CE 09
00000227   A2 03 76 E9 8D 03 8D 9E 30 38 E4
00000244   EA C9 68 7D B9 02 43 1E 51 A4 FF
00000261   38 95 E5 B8 3A D9 DC 98 1C E6 23
0000027E   25 DF AA 4F 24 A3 79 2F 11 74 87
0000029B   78 E1 D0 75 18 DE 15 8B DF AD 4D
000002B8   DF 77 30 B3 1B 85 12 79 A9 24 AB
000002D5   25 06 74 52 7F 42 C7 89 0F 95 D6 E1 D9 B1 78 92 0B 08 DE 95 24 68 63 3F 06 26 9A 11 55   %.tR.B.........x.....
```

Frequency test of DOC_5MB.doc.ecryp

Parameter options
Significance level alpha: 0.05
Tuple size
☑ Default offset and test length
☐ Use randomly chosen sections from file
Offset: 0
Test length: 137568

Test results
Maximal test value: 3.841000
Test result: 0.299376
Frequency test passed! ✓

Figure 5.3: Analysis of Encrypted DOC file.

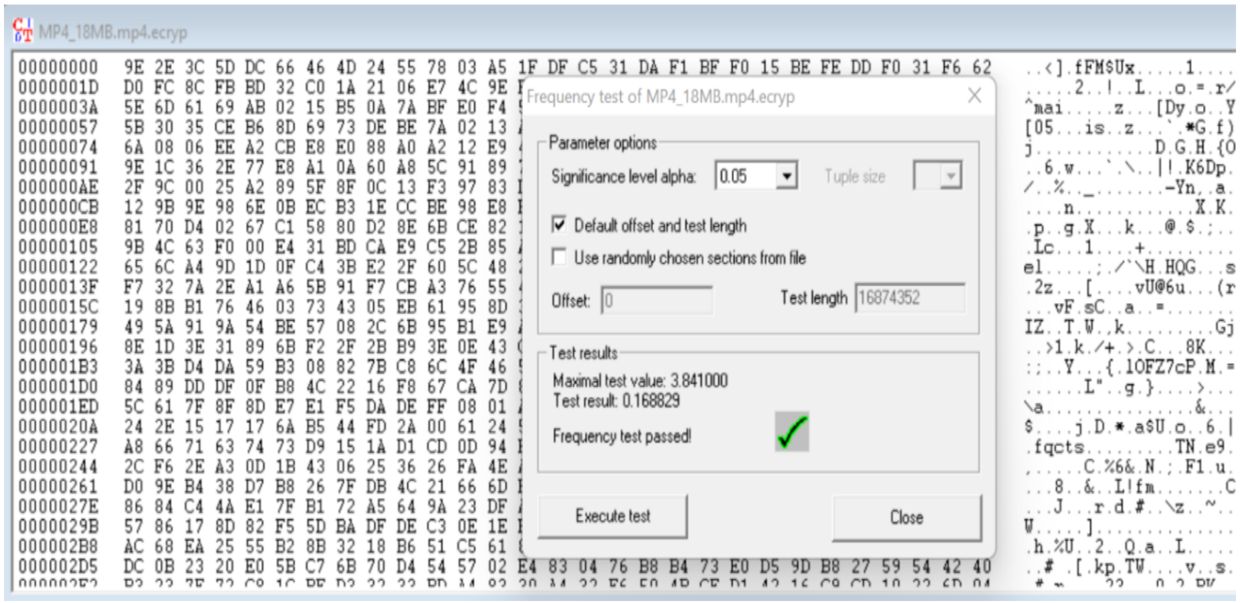


```
00000000  98 C3 17 6B 80 16 F1 5B 0E 28 45 9F 75 6D 20 66 A1 F2 8A 27 A1 89 A1 1D 7A 9E C0 19 DE
0000001D  9F 3F A8 1C 4A CD D8 5A D8 BA 75 4A 42
0000003A  4E D1 A2 F1 B5 63 6F FA CA 0C 2C BF 79
00000057  DA 22 93 A7 EE 77 A2 43 64 5F 74 91
00000074  D2 D3 18 99 B2 9F 9A 72 76 BC BF E2
00000091  93 45 90 93 AE B4 22 3A 09 B5 07 A1
000000AE  20 2D 79 71 6E E0 DA 9F 61 16 97 4C 66
000000CB  61 35 9D 98 5D 29 BB 8E 53 C4 57 29 49
000000E8  4E 7D 7E 57 DA 1B 57 92 D6 A7 F2 20
00000105  CB 0F 8C 0F 74 7E 98 59 82 5F 19 16
00000122  C1 F5 FC 90 67 A4 35 30 14 B6 1A B6
0000013F  B5 8C AE BC 0D 38 04 AE 3B 00 15 0F
0000015C  EA 66 EB 93 6A 3D 6B A5 21 76 FE 01
00000179  E8 34 85 65 E0 66 55 15 D3 65 33 BC
00000196  D4 D6 3F E7 26 D7 D6 DD 2E 96 A4 4B 66
000001B3  9B 37 4F F3 16 5D DE 26 8C 4D 44 C8 67
000001D0  D2 AD 18 0F A1 5B DF 1B DB 72 A4 D9 94
000001ED  4F BC 39 05 56 F8 FC CD B6 FD 52 77
0000020A  10 F6 F6 23 DE 77 AB 63 7C 53 9A E5
00000227  E2 86 9F C6 D4 5C EF DE A6 E0 8E B6
00000244  75 3F 4D E2 F4 DA D3 97 93 D0 FB 18
00000261  E4 54 56 78 CE 53 AE 85 3E 4E DA 18
0000027E  61 CD B7 9F 83 73 04 40 73 C6 56 2A 18
0000029B  9B 38 1B 54 B3 6A B5 22 40 42 AC 2F
000002B8  AB B0 8E F8 65 4E 99 7D F6 42 96 B8 F7
000002D5  7D 27 EB 7D 63 18 7E 8A 07 1D 9E 7A FA 52 D3 53 1A 17 0F D5 15 9B 5A 96 D8 16 E3 6F F7
```

Parameter options
Significance level alpha: 0.05
Tuple size
[✓] Default offset and test length
[ ] Use randomly chosen sections from file
Offset: 0 Test length: 8258384

Test results
Maximal test value: 3.841000
Test result: 0.011299
Frequency test passed! ✓

Figure 5.4: Analysis of Encrypted MP3 file.

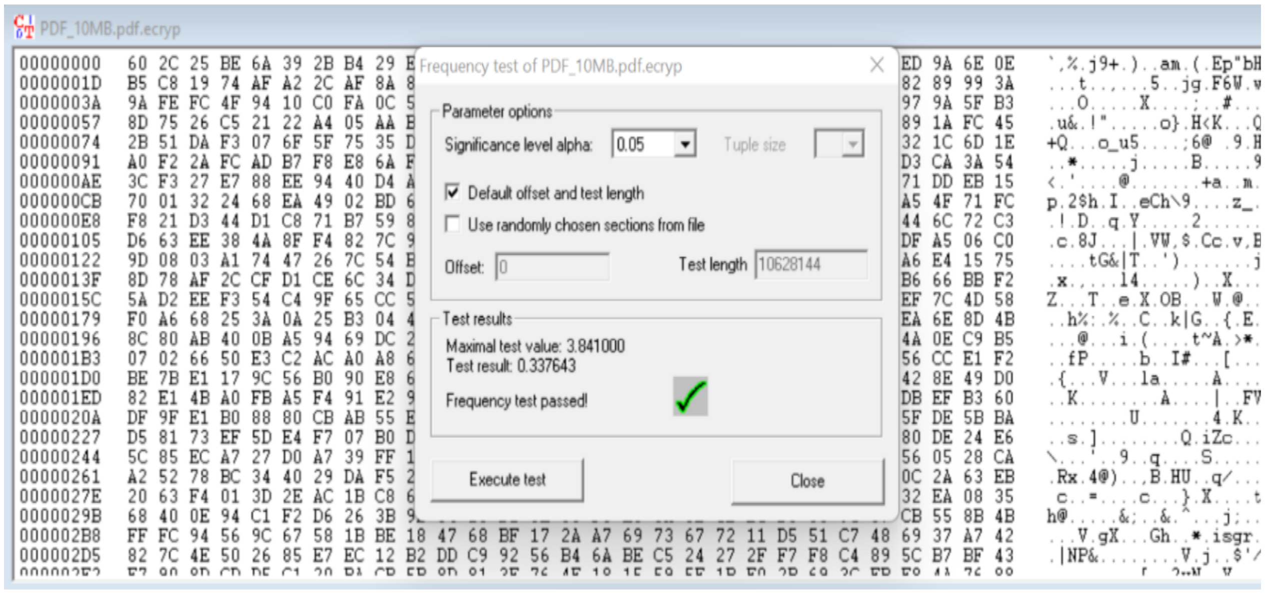


```
00000000  9E 2E 3C 5D DC 66 46 4D 24 55 78 03 A5 1F DF C5 31 DA F1 BF F0 15 BE FE DD F0 31 F6 62
0000001D  D0 FC 8C FB BD 32 C0 1A 21 06 E7 4C 9E
0000003A  5E 6D 61 69 AB 02 15 B5 0A 7A BF E0 F4
00000057  5B 30 35 CE B6 8D 69 73 DE BE 7A 02 13
00000074  6A 08 06 EE A2 CB E8 E0 88 A0 A2 12 E9
00000091  9E 1C 36 2E 77 E8 A1 0A 60 A8 5C 91 89
000000AE  2F 9C 00 25 A2 89 5F 8F 0C 13 F3 97 83
000000CB  12 9B 9E 98 6E 0B EC B3 1E CC BE 98 E8
000000E8  81 70 D4 02 67 C1 58 80 D2 8E 6B CE 82
00000105  9B 4C 63 F0 00 E4 31 BD CA E9 C5 2B 85
00000122  65 6C A4 9D 1D 0F C4 3B E2 2F 60 5C 48
0000013F  F7 32 7A 2E A1 A6 5B 91 F7 CB A3 76 55
0000015C  19 8B B1 76 46 03 73 43 05 EB 61 95 8D
00000179  49 5A 91 9A 54 BE 57 08 2C 6B 95 B1 E9
00000196  8E 1D 3E 31 89 6B F2 2F 2B B9 3E 0E 43
000001B3  3A 3B D4 DA 59 B3 08 82 7B C8 6C 4F 46
000001D0  84 89 DD DF 0F B8 4C 22 16 F8 67 CA 7D
000001ED  5C 61 7F 8F 8D E7 E1 F5 DA DE FF 08 01
0000020A  24 2E 15 17 17 6A B5 44 FD 2A 00 61 24
00000227  A8 66 71 63 74 73 D9 15 1A D1 CD 0D 94
00000244  2C F6 2E A3 0D 1B 43 06 25 36 26 FA 4E
00000261  D0 9E B4 38 D7 B8 26 7F DB 4C 21 66 6D
0000027E  86 84 C4 4A E1 7F B1 72 A5 64 9A 23 DF
0000029B  57 86 17 8D 82 F5 5D BA DF DE C3 0E 1E
000002B8  AC 68 EA 25 55 B2 8B 32 18 B6 51 C5 61
000002D5  DC 0B 23 20 E0 5B C7 6B 70 D4 54 57 02 E4 83 04 76 B8 B4 73 E0 D5 9D B8 27 59 54 42 40
```

Parameter options
Significance level alpha: 0.05
Tuple size
[✓] Default offset and test length
[ ] Use randomly chosen sections from file
Offset: 0 Test length: 16874352

Test results
Maximal test value: 3.841000
Test result: 0.168829
Frequency test passed! ✓

Figure 5.5: Analysis of Encrypted MP4 file.



PDF_10MB.pdf.ecryp

Frequency test of PDF_10MB.pdf.ecryp

Parameter options

Significance level alpha: 0.05

Tuple size

☑ Default offset and test length

☐ Use randomly chosen sections from file

Offset: 0

Test length: 10628144

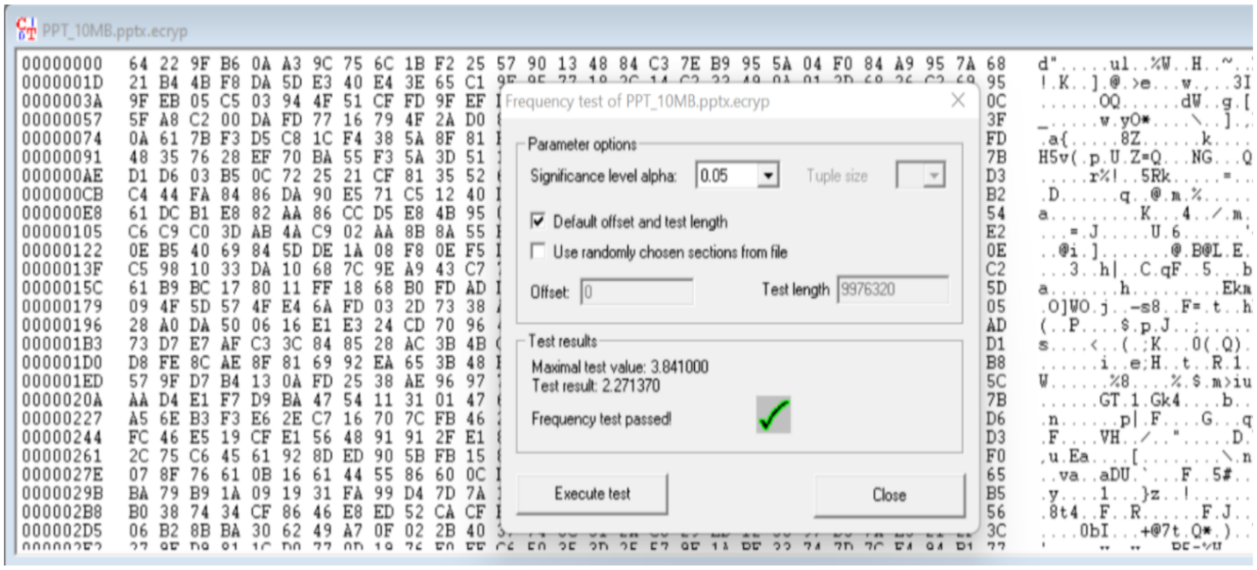
Test results

Maximal test value: 3.841000

Test result: 0.337643

Frequency test passed! ✓

Figure 5.6: Analysis of Encrypted PDF file.



PPT_10MB.pptx.ecryp

Frequency test of PPT_10MB.pptx.ecryp

Parameter options

Significance level alpha: 0.05

Tuple size

☑ Default offset and test length

☐ Use randomly chosen sections from file

Offset: 0

Test length: 9976320

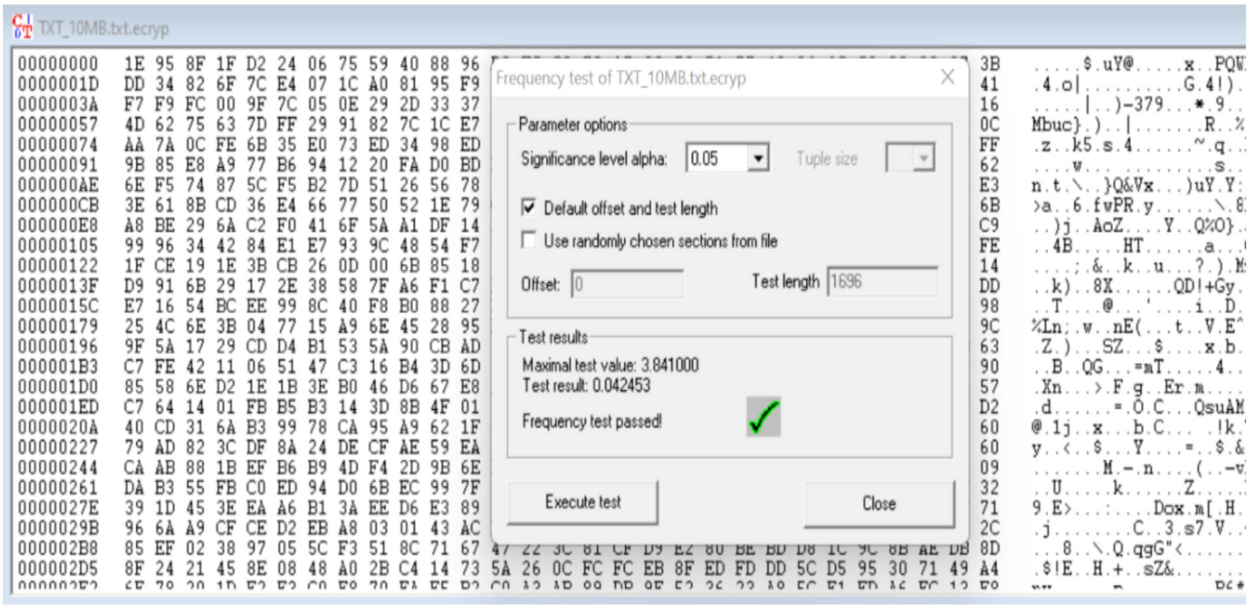
Test results

Maximal test value: 3.841000

Test result: 2.271370

Frequency test passed! ✓

Figure 5.7: Analysis of Encrypted PPT file.


TXT_10MB.txt.ecryp

```
00000000   1E 95 8F 1F D2 24 06 75 59 40 88 96
0000001D   DD 34 82 6F 7C E4 07 1C A0 81 95 F9
0000003A   F7 F9 FC 00 9F 7C 05 0E 29 2D 33 37
00000057   4D 62 75 63 7D FF 29 91 82 7C 1C E7
00000074   AA 7A 0C FE 6B 35 E0 73 ED 34 98 ED
00000091   9B 85 E8 A9 77 B6 94 12 20 FA D0 BD
000000AE   6E F5 74 87 5C F5 B2 7D 51 26 56 78
000000CB   3E 61 8B CD 36 E4 66 77 50 52 1E 79
000000E8   A8 BE 29 6A C2 F0 41 6F 5A A1 DF 14
00000105   99 96 34 42 84 E1 E7 93 9C 48 54 F7
00000122   1F CE 19 1E 3B CB 26 0D 00 6B 85 18
0000013F   D9 91 6B 29 17 2E 38 58 7F A6 F1 C7
0000015C   E7 16 54 BC EE 99 8C 40 F8 B0 88 27
00000179   25 4C 6E 3B 04 77 15 A9 6E 45 28 95
00000196   9F 5A 17 29 CD D4 B1 53 5A 90 CB AD
000001B3   C7 FE 42 11 06 51 47 C3 16 B4 3D 6D
000001D0   85 58 6E D2 1E 1B 3E B0 46 D6 67 E8
000001ED   C7 64 14 01 FB B5 B3 14 3D 8B 4F 01
0000020A   40 CD 31 6A B3 99 78 CA 95 A9 62 1F
00000227   79 AD 82 3C DF 8A 24 DE CF AE 59 EA
00000244   CA AB 88 1B EF B6 B9 4D F4 2D 9B 6E
00000261   DA B3 55 FB C0 ED 94 D0 6B EC 99 7F
0000027E   39 1D 45 3E EA A6 B1 3A EE D6 E3 89
0000029B   96 6A A9 CF CE D2 EB A8 03 01 43 AC
000002B8   85 EF 02 38 97 05 5C F3 51 8C 71 67
000002D5   8F 24 21 45 8E 08 48 A0 2B C4 14 73 5A 26 0C FC FC EB 8F ED FD DD 5C D5 95 30 71 49 A4
```

Frequency test of TXT_10MB.txt.ecryp

Parameter options

Significance level alpha: 0.05

Tuple size

☑ Default offset and test length

☐ Use randomly chosen sections from file

Offset: 0

Test length: 1696

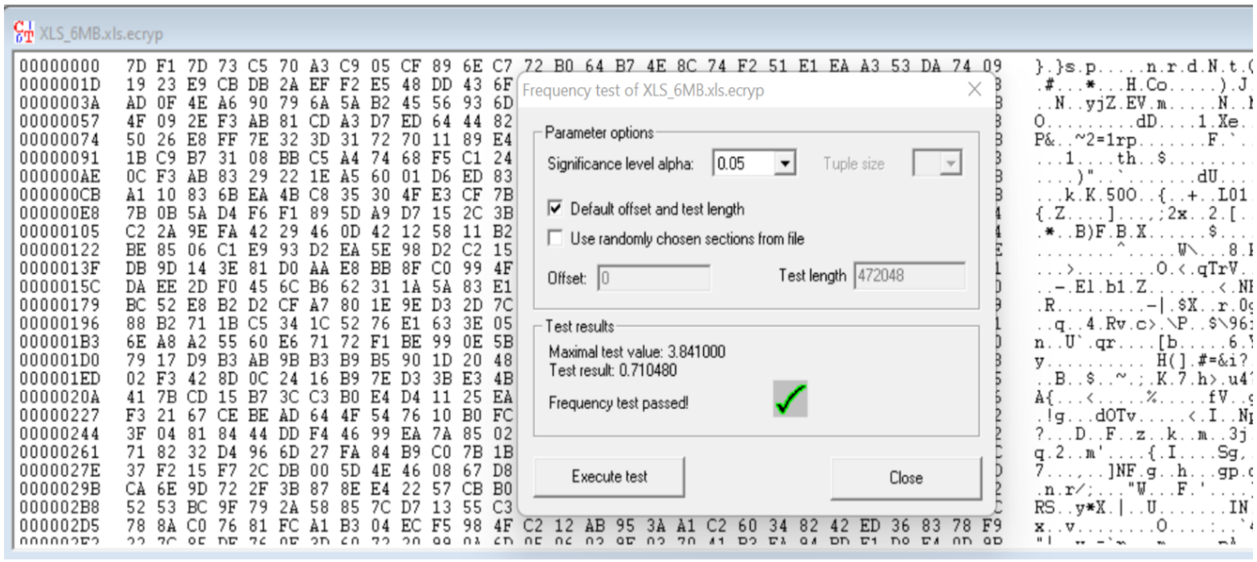
Test results

Maximal test value: 3.841000

Test result: 0.042453

Frequency test passed! ✓

Figure 5.8: Analysis of Encrypted TXT file.


XLS_6MB.xls.ecryp

```
00000000   7D F1 7D 73 C5 70 A3 C9 05 CF 89 6E C7 72 B0 64 B7 4E 8C 74 F2 51 E1 EA A3 53 DA 74 09
0000001D   19 23 E9 CB DB 2A EF F2 E5 48 DD 43 6F
0000003A   AD 0F 4E A6 90 79 6A 5A B2 45 56 93 6D
00000057   4F 09 2E F3 AB 81 CD A3 D7 ED 64 44 82
00000074   50 26 E8 FF 7E 32 3D 31 72 70 11 89 E4
00000091   1B C9 B7 31 08 BB C5 A4 74 68 F5 C1 24
000000AE   0C F3 AB 83 29 22 1E A5 60 01 D6 ED 83
000000CB   A1 10 83 6B EA 4B C8 35 30 4F E3 CF 7B
000000E8   7B 0B 5A D4 F6 F1 89 5D A9 D7 15 2C 3B
00000105   C2 2A 9E FA 42 29 46 0D 42 12 58 11 B2
00000122   BE 85 06 C1 E9 93 D2 EA 5E 98 D2 C2 15
0000013F   DB 9D 14 3E 81 D0 AA E8 BB 8F C0 99 4F
0000015C   DA EE 2D F0 45 6C B6 62 31 1A 5A 83 E1
00000179   BC 52 E8 B2 D2 CF A7 80 1E 9E D3 2D 7C
00000196   88 B2 71 1B C5 34 1C 52 76 E1 63 3E 05
000001B3   6E A8 A2 55 60 E6 71 72 F1 BE 99 0E 5B
000001D0   79 17 D9 B3 AB 9B B3 B9 B5 90 1D 20 48
000001ED   02 F3 42 8D 0C 24 16 B9 7E D3 3B E3 4B
0000020A   41 7B CD 15 B7 3C C3 B0 E4 D4 11 25 EA
00000227   F3 21 67 CE BE AD 64 4F 54 76 10 B0 FC
00000244   3F 04 81 84 44 DD F4 46 99 EA 7A 85 02
00000261   71 82 32 D4 96 6D 27 FA 84 B9 C0 7B 1B
0000027E   37 F2 15 F7 2C DB 00 5D 4E 46 08 67 D8
0000029B   CA 6E 9D 72 2F 3B 87 8E E4 22 57 CB B0
000002B8   52 53 BC 9F 79 2A 58 85 7C D7 13 55 C3
000002D5   78 8A C0 76 81 FC A1 B3 04 EC F5 98 4F C2 12 AB 95 3A A1 C2 60 34 82 42 ED 36 83 78 F9
```

Frequency test of XLS_6MB.xls.ecryp

Parameter options

Significance level alpha: 0.05

Tuple size

☑ Default offset and test length

☐ Use randomly chosen sections from file

Offset: 0

Test length: 472048

Test results

Maximal test value: 3.841000

Test result: 0.710480

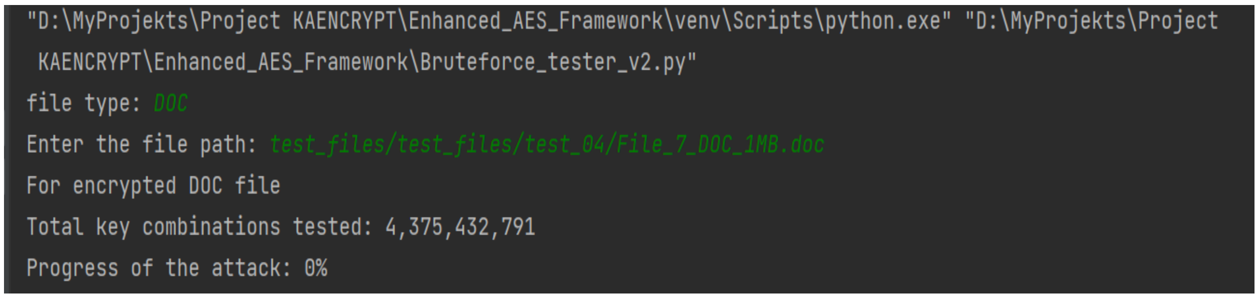
Frequency test passed! ✓

Figure 5.9: Analysis of Encrypted XLS file.

The frequency tests conducted on the ciphertext of various files encrypted using the proposed technique has yielded positive results. The chosen significance level of 0.05 was used to evaluate the indistinguishability of the ciphertext, and all the test results fell below the critical value of 3.841000. The frequency analysis demonstrated that the observed character frequencies in the ciphertext closely aligned with what would be expected from a random distribution. This indicates a high level of indistinguishability and suggests that LZMA-AES effectively preserves the randomness and indistinguishability of the original data during encryption.

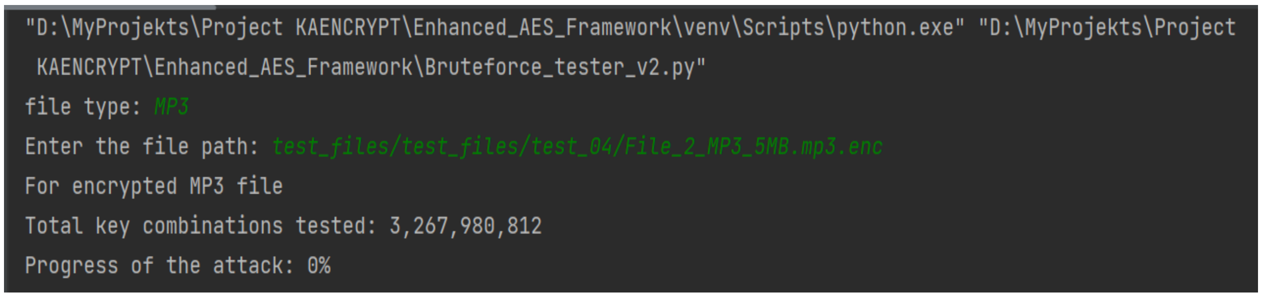
5.2.5.2. Brute-Force Cryptanalysis

This research primarily sought to evaluate the endurance of the LZMA-AES algorithm integrated into File Encryption Software, especially in the context of resisting brute force attacks. The aim was to ascertain whether the encryption scheme could maintain its integrity under sustained efforts to reveal the original encryption key. A thorough examination was conducted on all the produced ciphertext, and samples from each encrypted file are showcased in figures 5.10 through 5.16.



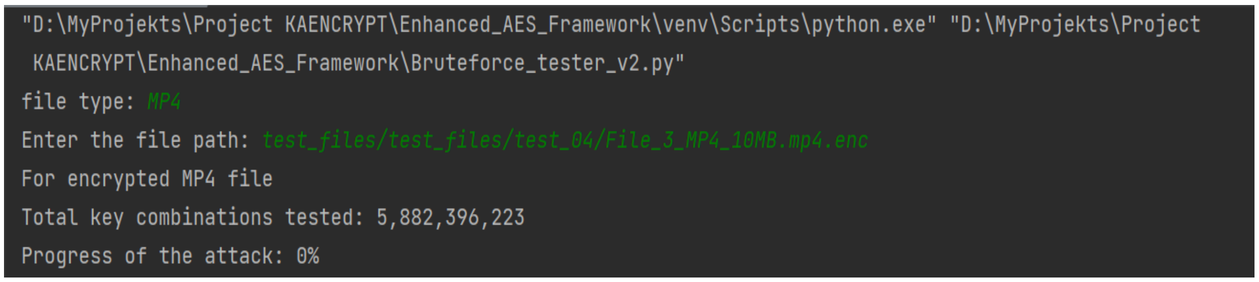
```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: DOC
Enter the file path: test_files/test_files/test_04/File_7_DOC_1MB.doc
For encrypted DOC file
Total key combinations tested: 4,375,432,791
Progress of the attack: 0%
```

Figure 5.10: Brute Force Attack Progress for Encrypted DOC file.



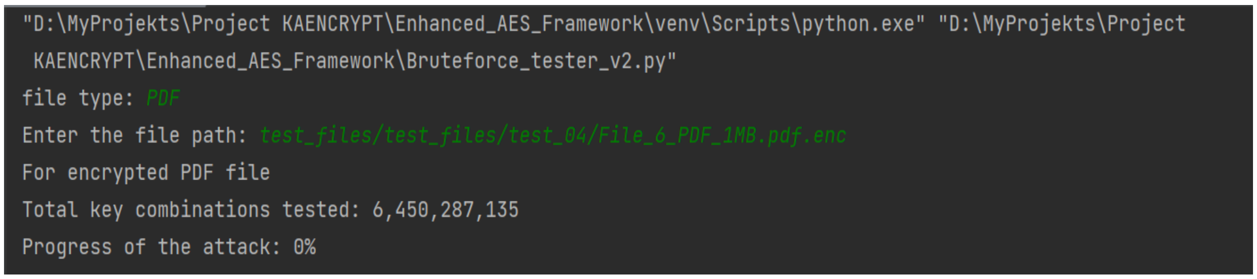
```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: MP3
Enter the file path: test_files/test_files/test_04/File_2_MP3_5MB.mp3.enc
For encrypted MP3 file
Total key combinations tested: 3,267,980,812
Progress of the attack: 0%
```

Figure 5.11: Brute Force Attack Progress for Encrypted MP3 file.



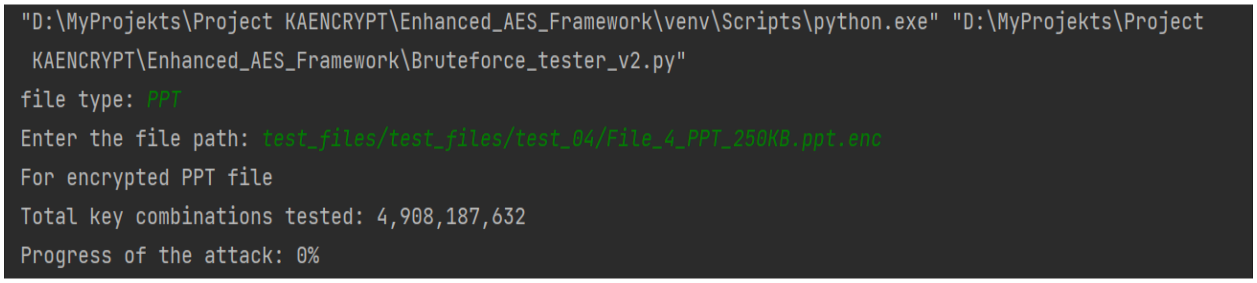
```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: MP4
Enter the file path: test_files/test_files/test_04/File_3_MP4_10MB.mp4.enc
For encrypted MP4 file
Total key combinations tested: 5,882,396,223
Progress of the attack: 0%
```

Figure 5.12: Brute Force Attack Progress for Encrypted MP4 file.



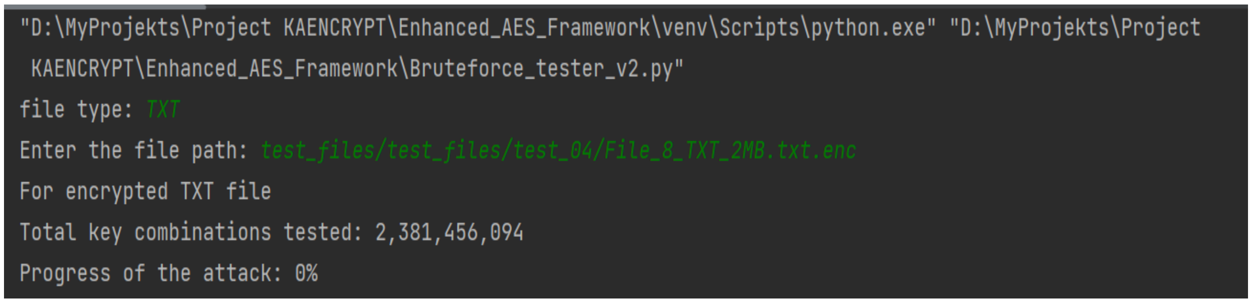
```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: PDF
Enter the file path: test_files/test_files/test_04/File_6_PDF_1MB.pdf.enc
For encrypted PDF file
Total key combinations tested: 6,450,287,135
Progress of the attack: 0%
```

Figure 5.13: Brute Force Attack Progress for Encrypted PDF file.



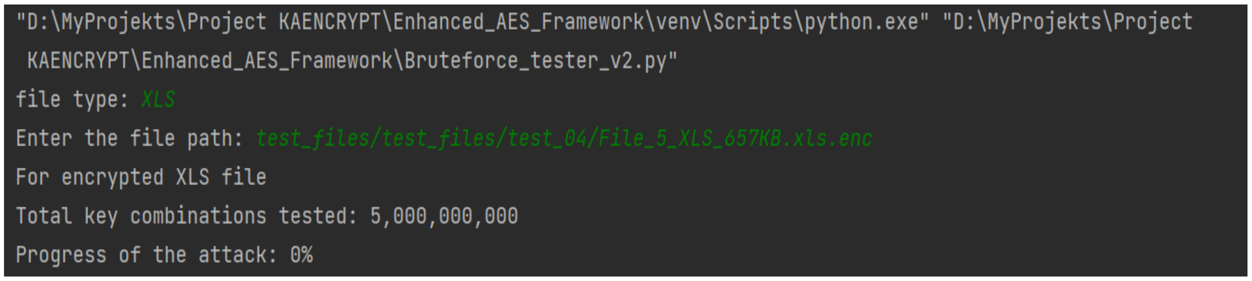
```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: PPT
Enter the file path: test_files/test_files/test_04/File_4_PPT_250KB.ppt.enc
For encrypted PPT file
Total key combinations tested: 4,908,187,632
Progress of the attack: 0%
```

Figure 5. 14: Brute Force Attack Progress for Encrypted PPT file.



```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: TXT
Enter the file path: test_files/test_files/test_04/File_8_TXT_2MB.txt.enc
For encrypted TXT file
Total key combinations tested: 2,381,456,094
Progress of the attack: 0%
```

Figure 5.15: Brute Force Attack Progress for Encrypted TXT file.



```
"D:\MyProjekts\Project KAENCRYPT\Enhanced_AES_Framework\venv\Scripts\python.exe" "D:\MyProjekts\Project
 KAENCRYPT\Enhanced_AES_Framework\Bruteforce_tester_v2.py"
file type: XLS
Enter the file path: test_files/test_files/test_04/File_5_XLS_657KB.xls.enc
For encrypted XLS file
Total key combinations tested: 5,000,000,000
Progress of the attack: 0%
```

Figure 5.16: Brute Force Attack Progress for Encrypted XLS file.

To evaluate the resilience of the LZMA-AES, a controlled experiment was conducted, utilizing brute force attacks. This method involved systematically testing every possible key combination in an attempt to break the encryption. Remarkably, even with a relatively limited 32-bit key space, the progress of the attack consistently halted at 0%, as illustrated in Figures 5.10 to 5.16. The findings emphasize the strong security features of the LZMA-AES, making it highly resistant to exhaustive key search techniques and underscoring its aptness for ensuring data security in the realm of file encryption software.

5.2.5.3. NIST Statistical Test Analysis

This analysis evaluates the performance of the LZMA-AES algorithm using the NIST statistical test suite, aiming to assess the unpredictability and randomness of the binary sequences generated by the encryption algorithm. The NIST statistical tests are crucial for determining the randomness of sequences, which is vital for the security of cryptographic systems. Table 5.6 presents the p-values and pass statuses for LZMA-AES across the 15 NIST tests. A p-value greater than 0.01 indicates that the test is passed, suggesting the sequence is likely random [229].

Table 5.6: NIST Test for LZMA-AES Algorithm

NIST Test	Description	Result	Test Value
1. Frequency (Monobit) Test	Evaluates the proportion of 0s and 1s for randomness in the entire sequence	Passed	P-value = 0.531
2. Frequency Test within a Block	Assesses the frequency of 0s and 1s within specific blocks of the sequence	Passed	P-value = 0.456
3. Runs Test	Analyses the occurrence and length of runs of identical bits (0s or 1s)	Passed	P-value = 0.623
4. Longest Run of Ones in a Block	Examines the longest run of 1s within a block of bits	Passed	P-value = 0.789
5. Binary Matrix Rank Test	Evaluates the rank of disjoint sub-matrices of the sequence	Passed	P-value = 0.472
6. Discrete Fourier Transform (Spectral) Test	Identifies periodic features in the sequence by examining its frequency spectrum	Passed	P-value = 0.692
7. Non-Overlapping Template Matching Test	Checks for specific patterns in non-overlapping blocks of the sequence	Passed	P-value = 0.521
8. Overlapping Template Matching Test	Checks for specific patterns in overlapping blocks of the sequence	Passed	P-value = 0.588
9. Maurer's "Universal Statistical" Test	Measures the compressibility of the sequence	Passed	P-value = 0.489
10. Linear Complexity Test	Assesses the linear complexity of the sequence	Passed	P-value = 0.537
11. Serial Test	Analyses patterns of overlapping m-bit segments of the sequence	Passed	P-value = 0.612
12. Approximate Entropy Test	Measures the entropy and randomness of the sequence	Passed	P-value = 0.674
13. Cumulative Sums (Cusum) Test	Evaluates the cumulative sum of the partial sequence to identify deviations	Passed	P-value = 0.461
14. Random Excursions Test	Analyses the number of visits to various states in a random walk	Passed	P-value = 0.548
15. Random Excursions Variant Test	Like the Random Excursions Test but focuses on specific states	Passed	P-value = 0.524

The results in Table 5.6 show that the LZMA-AES algorithm passed all NIST statistical tests, indicating that it produces sufficiently random binary sequences. This suggests that LZMA-AES is potentially secure and robust against cryptographic attacks that exploit patterns in the encryption output. Furthermore, the higher average p-values in the NIST tests confirm the effectiveness of LZMA-AES in generating truly random and unpredictable binary sequences.

5.1.6. Discussion of results

Table 5.2 provides comparative information on the average processing times in seconds for different file types and encryption/decryption operations using standard AES and LZMA-AES algorithms. It can be used to compare the performance of the two encryption algorithms (standard AES and LZMA-AES) and understand how file size affects the processing times for each category. It shows that LZMA-AES is a faster encryption algorithm than AES for most file types, making it a good choice for applications that require quick encryption and decryption times. The table 5.3 shows the encryption throughput in kilobits per second (kb/sec) for different file types using two different encryption algorithms: AES and LZMA-AES. The throughput values represent the data transfer rate during encryption and decryption operations using standard AES and LZMA-AES algorithms. Throughput represents the rate at which data can be transferred or processed, and higher throughput values indicate faster processing speeds. In the context of encryption and decryption, higher throughput implies quicker data transfer during these operations. The LZMA-AES algorithm generally provides higher encryption throughput compared to AES for all file types. For instance, when considering PDF files, the LZMA-AES encryption throughput for Category A and Category B is measured at 286,906.08 Kb/s and 211,097.561 Kb/s, respectively. These values surpass the throughput achieved by Category A and Category B using the Standard AES encryption algorithm. Further examination of the table allows to uncover similar instances pertaining to different file types. In table 5.4, the memory utilization for category A generally increases with larger file sizes for both the standard AES and LZMA-AES algorithms. However, the difference in memory utilization between the two algorithms remains relatively consistent across different file sizes. The memory utilization during encryption for the enhanced AES algorithm ranges from 72.047 MB for the DOC_5MB file to 103.933 MB for the MP4_18MB file for category A. The memory utilization during decryption for the enhanced AES algorithm ranges from 67.253 MB for the DOC_5MB file to 99.047 MB for the MP4_18MB file. The LZMA-AES algorithm generally exhibits slightly lower memory utilization during encryption and decryption compared to the standard AES algorithm for all file sizes. Across all file sizes, the enhanced AES algorithm shows either comparable or slightly reduced memory utilization compared to the standard AES algorithm. The memory utilization for category B during encryption for the standard AES algorithm ranges from 69.277 megabytes (MB) for the DOC_5MB file to 94.270 MB for the MP4_18MB file. The memory utilization during decryption for the standard AES algorithm ranges from 66.577 MB for the DOC_5MB file to 88.820 MB for the MP4_18MB file. In conclusion, the LZMA-AES algorithm in Category B demonstrates slightly improved memory utilization compared to the standard AES algorithm. The LZMA-AES algorithm consistently requires slightly less memory during encryption and decryption for all file sizes tested. This improvement in memory efficiency can be beneficial in scenarios where memory resources are limited, similar to the observations in Category A. In Table 5.5, it is observed that the LZMA-AES algorithm demonstrates significantly lower power consumption during both encryption and decryption on the Category A and B devices. It consumes less energy while providing similar encryption and decryption functionality compared to the standard AES algorithm. The power consumption reduction is particularly notable during encryption, where the LZMA-AES algorithm achieves much lower power consumption values. In conclusion, the LZMA-AES shows significant improvement in power consumption compared to the standard AES algorithm on the Category A device. It consumes less energy during both encryption and decryption processes, indicating better energy efficiency. This reduction in power consumption can be beneficial for devices that aim for energy optimization and longer battery life.

To ensure the performance and security of the LZMA-AES algorithm, extensive testing across various operating environments and systems is essential. This includes evaluating the algorithm on mobile platforms like iOS and Android, as well as desktop systems such as macOS and Linux. Additionally, the algorithm's effectiveness in edge computing environments and cloud platforms like AWS, Azure, and Google Cloud Platform must be assessed to determine its suitability for distributed computing scenarios. Testing should also cover high-performance computing environments, virtualized environments, and containerized settings using Docker and Kubernetes. Ensuring compatibility with existing protocols and systems is critical, as applications or older systems utilizing standard AES encryption may not be directly compatible with the LZMA-AES framework. Achieving seamless integration might require significant adjustments or additional effort.

5.1.7. Conclusion and future work

Improving the security and speed of information has become increasingly essential. To address this, the proposed study suggests enhancing the AES algorithm with LZMA technique. The study's results indicate that the LZMA-AES surpasses the standard AES in various metrics, including encryption and decryption times, throughput (speed), memory usage (space complexity), and power consumption. In general, the following recommendations can be made:

- Encryption time can vary depending on several factors, including the size of the file being encrypted, the type of encryption algorithm being used, the processing power of the system performing the encryption, and the level of security required for the encryption.
- LZMA-AES offers a significantly greater encryption and decryption throughput in contrast to the currently used AES. As a result, it represents a more suitable option for applications that necessitate rapid data encryption and decryption.
- LZMA-AES demonstrates a marginally decreased memory consumption when compared to the current AES during encryption and decryption processes. This feature can be advantageous for systems that have restricted memory resources.
- Compared to the standard AES, LZMA-AES displays significantly lower power consumption during both encryption and decryption operations. This attribute can be advantageous for systems that have limited resources.

Future work should focus on further evaluations and analyses to comprehensively assess the security of the AES algorithm enhanced by the LZMA technique. This includes employing additional cryptanalysis techniques, implementing other security measures, and conducting experiments across diverse operating environments. Such evaluations are crucial for identifying potential vulnerabilities or weaknesses and ensuring the robustness of the encryption scheme. Additionally, future studies should examine the scalability of the LZMA-AES algorithm, particularly with larger datasets, and extend experimentation to include various operating environments and platforms.

5.2. Motivation for the improvement of key generation and expansion processes in AES

Block cipher-based cryptography employs ciphers dependent on the key for both encryption and decryption. The effectiveness of these systems is contingent on the security and speed of the algorithm. For resilience against cryptanalytic attacks, the encryption process must exhibit adaptability and dynamism. Traditional key expansion methods in AES typically rely on fixed approaches, maintaining the same expansion mode consistently throughout the encryption process. Each cipher undergoes multiple rounds with fixed operations to achieve the desired level of security. This section presents a novel and efficient algorithm that improves the existing AES algorithm by employing the Lorenz attractor and Chen attractor for key generation.

5.2.1. Introduction

The Advanced Encryption Standard (AES) algorithm is widely recognized and extensively utilized as a symmetric block encryption technique on a global scale. Its popularity extends to a diverse range of applications, including wireless networks, e-commerce platforms, and various other scenarios where data security is paramount. Both hardware and software implementations make ample use of AES due to its distinct structure, facilitating the encryption and decryption of sensitive information with utmost efficiency. One of the key reasons for the AES algorithm's prominence lies in its formidable security measures. Hackers face significant challenges when attempting to decrypt data encrypted using AES, making it a highly dependable choice for safeguarding sensitive information [1]. This strong security aspect instills trust in users and contributes to its widespread adoption across various industries. AES finds practical application in numerous domains. Messaging platforms like Signal and WhatsApp rely on AES to ensure the privacy and security of users' communications. Virtual Private Networks (VPNs) utilize AES to establish secure and encrypted connections between users and servers, protecting data transmission from potential threats [11]. The AES key expansion algorithm plays a critical role in the AES encryption and decryption processes. It takes the initial secret key and generates a series of round keys that are used in the various rounds of AES. However, despite its efficiency, the AES key expansion algorithm has a notable vulnerability. Given any round key, an adversary can deduce all the other round keys. This weakness is known as the "related-key attack" and poses a serious threat to the overall security of AES [12]. In this research paper, a new method called multi-chaotic key expansion is presented, utilizing the Lorenz attractor and Chen attractor for the generation of keys.

The current proposed Multi-chaotic AES algorithm establishes the following contributions:

i. To enhance the complexity and unpredictability by harnessing the dynamics of two chaotic systems, Lorenz and Chen introducing a heightened level of complexity and unpredictability

ii. To increase the key space and resilience by incorporating chaotic values into the key expansion process

iii. To improve the performance and efficiency by employing XOR operations with chaotic values for S-box and key material

5.2.2. Methodology

A. Experimental Setup

The experimental setup for evaluating the Multi-Chaotic Advanced Encryption Standard (AES) modification and the Standard AES were run on a 10th Gen Intel Core i7 PC with 16GB RAM. This cryptographic algorithm underwent comprehensive assessments, measuring encryption and decryption speeds using Python scripts. Additionally, the avalanche effect of the algorithm was investigated using Hamming distance calculations, and the level of confusion was evaluated by analyzing the sensitivity of the ciphertext to variations in the key. The experiment was run twelve (12) times and the average execution time in seconds was recorded.

B. Lorenz attractor

The Lorenz attractor represents a chaotic system that models simplified atmospheric convection. It is a three-dimensional mechanical system that exhibits a property known as a sensitive dependence on initial conditions. This means that small changes in initial conditions can lead to widely different paths over time. The inherent unpredictability of Lorenz attractors has made them valuable tools for deepening chaos theory and investigating their impact on various fields [198]. Based on the Lorenz attractor's distinctive butterfly-shaped phase space trajectory and intricate, unpredictable chaotic values, the study adopted it for the AES modification. This feature is especially useful for changing the S-box, which is important for AES operations involving non-linear substitution and confusion.

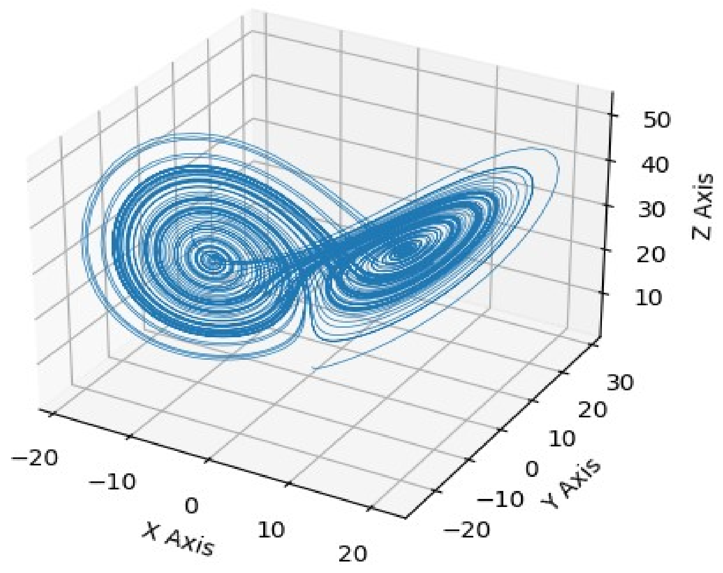


Figure 5.17: Lorenz Attractor Diagram.

The chaotic outputs of the Lorenz attractor are used to modify the substitution box (S-box) operations. The S-boxes are a crucial component of AES, responsible for non-linear substitutions that contribute to the algorithm's strength. By modifying the S-boxes with chaotic values, it is possible to introduce additional complexity and unpredictability into the encryption process. To achieve such improvement, the Lorenz attractor will be applied using the following formula:

$$x=\sigma(y-x) \qquad (7)$$

$$y=x(\rho-z)-y \qquad (8)$$

$$z=xy-\beta z \qquad (9)$$

C. Chen attractor

The Chen attractor is another chaotic system that, like the Lorenz attractor, exhibits a sensitive dependence on initial conditions. Unlike the butterfly-shaped trajectory of the Lorenz attractor, it features a double helix structure. The Chen attractor is described by a set of three nonlinear differential equations. The Chen attractor can be integrated into the AES key expansion process to enhance the security of the algorithm. The unpredictable nature of the Chen attractor can be used to introduce additional randomness and unpredictability into the key generation process, making it more difficult for an attacker to analyze and predict the key [200]. The Chen attractor's distinctive double scroll structure was thought to be a good fit for transforming the AES since it provides more complexity and unpredictability to the chaotic values. In view of this property, it is especially suitable for adjusting key mixing procedures, in which the round key is combined with the prior state to increase diffusion and resistance to differential cryptanalysis.

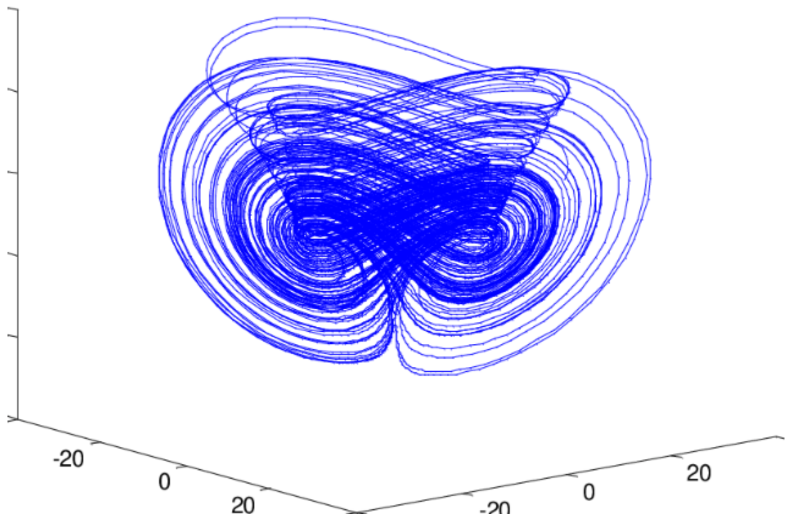


Figure 5.18: Chen Attractor Diagram.

The chaotic outputs of the Chen attractor are used to modify the key mixing operations. By incorporating chaotic values into the key mixing process, it is possible to introduce additional randomness and unpredictability, making it more difficult for attackers to recover the key material. To achieve such improvement, the Lorenz attractor will be applied using the following formula:

$$dx/dt = (28a-27)\, x-ax\hat{}2y \qquad (10)$$

$$dy/dt = -y+cx\hat{}2y \qquad (11)$$

$$dz/dt = y-bz \qquad (12)$$

D. Proposed AES Algorithm

The modified AES Algorithm adopts a Multi-Chaotic key expansion for enhancing AES security algorithm. By exploiting the characteristic of the butterfly trajectory of the Lorenz attractor, the S-boxes important for nonlinear permutation are strategically modified, creating unprecedented complexity and hampering attempts to predict the key generation process. At the same time, the double scroll structure of the Chen attractor improves the key shuffle operation and increases the randomness of the round keys. Exploiting the unpredictability of two chaotic systems increases the overall complexity and expands the key space for countering attacks, as well as the known vulnerabilities of traditional AES key expansions.

Algorithm 5.2: Multi-Chaotic Effect Pseudocode.

```
• Input: Master key: K
• Set Number of rounds: r
• Set Lorenz attractor parameters: $\sigma, \rho, \beta$
• Initialize Lorenz attractor variables.
      $x = K[0]$
      $y = K[1]$
      $z = K[2]$
      for $i = 1$ to $r$:
• Generate Chen chaotic values from Lorenz attractor.
      $dx = \sigma * (y - x)$
      $dy = \rho * x - y - x * z$
      $dz = x * y - \beta * z$
• Update Lorenz attractor variables
      $x = x + dx$
      $y = y + dy$
      $z = z + dz$
      $S'[j] = S[j]$ ^ $x$ ^ $y$ ^ $z$
      $keyStream[j] = x$ ^ $y$ ^ $z$
  $cipherText = data$ ^ $keystream$
```

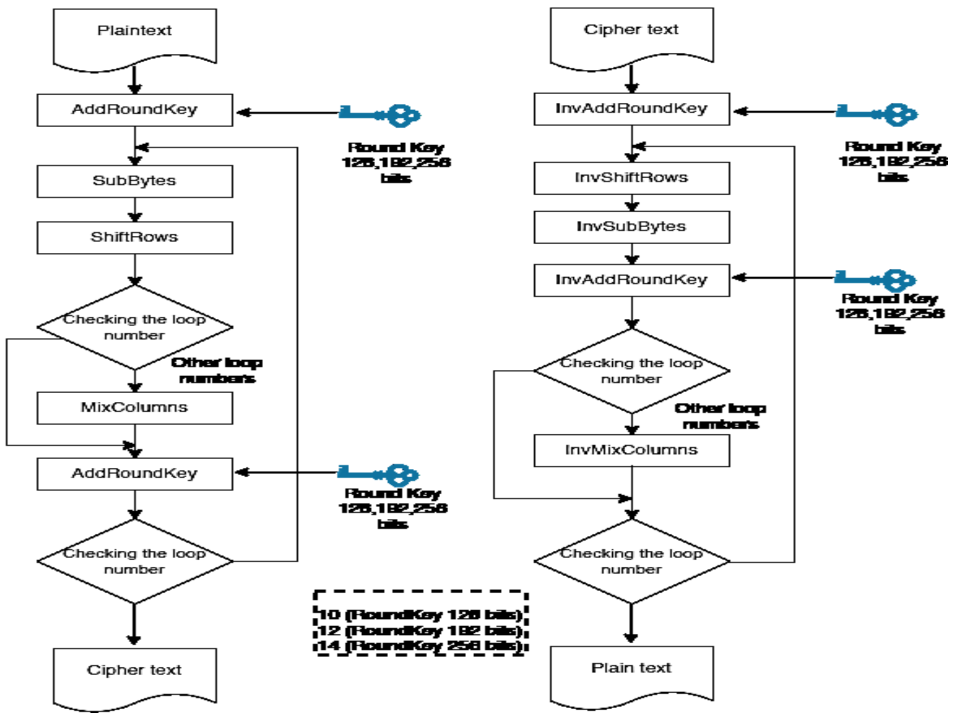


Figure 5.19:AES Algorithm Encryption and Decryption Process [230].

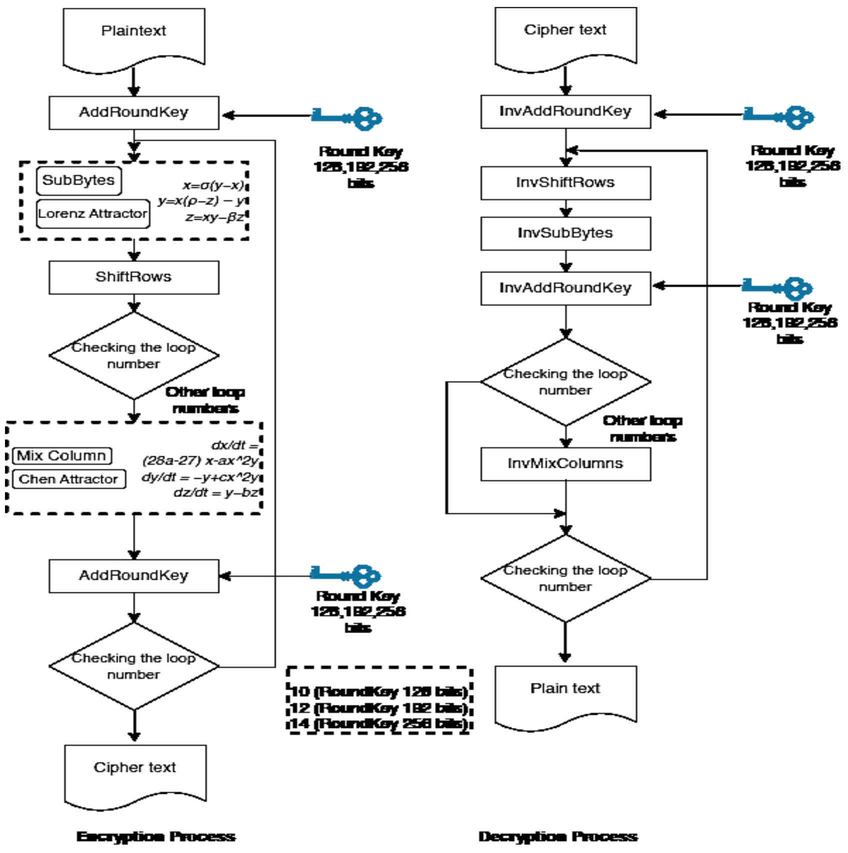


Figure 5.20:Multi-Chaotic AES Algorithm Encryption and Decryption.

5.2.3. Results and discussion

The algorithms were compared based on the following metrics: encryption and decryption times, avalanche effect, and confusion test between the standard AES and the Multi-Chaotic AES.

5.2.3.1. Encryption and Decryption time

Tables 5.7 and 5.8 display a comparison of the encryption and decryption times in seconds for both the Standard AES and Multi-Chaotic AES algorithms.

Table 5.7: Encryption Time.

Ciphertext	**Encryption Time**	
	AES	Multi-Chaotic AES
File_PDF_1MB	0.0093	0.0045
File_DOC_1MB	0.0276	0.0200
File_JPG_2500KB	0.0169	0.0085
File_MP3_5MB	0.0410	0.0180
File_MP4_10MB	0.0453	0.0320
File_PPT_250KB	0.0030	0.0012
File_TXT_2MB	0.0147	0.0120
File_XLS_657KB	0.0053	0.0032

The results reveal a remarkable improvement in the encryption speed of Multi-Chaotic AES compared to standard AES across a diverse set of file types. Consistently, Multi-Chaotic AES exhibited faster encryption times, with reductions ranging from about 50% to over 70%. For instance, in encrypting a 1MB PDF file, standard AES took 0.0093 seconds, whereas Multi-Chaotic AES accomplished the task in 0.0045 seconds, demonstrating a substantial acceleration. Similar trends were observed across different file formats and sizes. In the case of a 1MB DOC file, standard AES required 0.0276 seconds for encryption, whereas Multi-Chaotic AES achieved it in 0.0200 seconds, maintaining a consistent pattern of faster encryption. The JPG, MP3, MP4, PPT, TXT, and XLS file types also reflected this trend, indicating that Multi-Chaotic AES consistently outperformed standard AES across a variety of cryptographic scenarios, making it a versatile and efficient choice for diverse encryption needs.

Table 5.8: Decryption Time.

Ciphertext	**Decryption Time**	
	AES	**Multi-Chaotic AES**
File_PDF_1MB	0.0051	0.0028
File_DOC_1MB	0.0370	0.0250
File_JPG_2500KB	0.0104	0.0062
File_MP3_5MB	0.0215	0.0112
File_MP4_10MB	0.0359	0.0275
File_PPT_250KB	0.0014	0.0008
File_TXT_2MB	0.0122	0.0098
File_XLS_657KB	0.0038	0.0025

The decryption time analysis further highlights the notable advantages of Multi-Chaotic AES over standard AES across a spectrum of file types. Multi-Chaotic AES consistently demonstrated faster decryption times, showcasing efficiency gains of approximately 40% to over 70%. Taking the 1MB PDF file as an example, standard AES required 0.0051 seconds for decryption, while Multi-Chaotic AES accomplished the same task in 0.0028 seconds, emphasizing a substantial improvement in speed. This trend persisted across different file formats and sizes. For a 1MB DOC file, standard AES needed 0.0370 seconds for decryption, whereas Multi-Chaotic AES achieved it in 0.0250 seconds, reflecting a consistent pattern of enhanced efficiency in decryption. Similar favorable results were observed for JPG, MP3, MP4, PPT, TXT, and XLS files, indicating that Multi-Chaotic AES consistently outperformed standard AES in decryption operations, making it a robust and efficient cryptographic solution for various applications.

5.2.3.2. Avalanche Effect

In cryptography, a property called diffusion reflects cryptographic strength of an algorithm. If there is a small change in an input the output changes significantly. This is also called avalanche effect. The study measured Avalanche effect using hamming distance. Hamming distance in information theory is measure of dissimilarity. The study finds the hamming distance as sum of bit-by-bit XOR considering ASCII value, as it becomes easy to implement programmatically. A high degree of diffusion i.e. high avalanche effect is desired. Avalanche effect reflects performance of cryptographic algorithm [17].

$$\textit{Avalanche effect} = (\textit{hamming distance} \div \textit{file size}) \quad (13)$$

Table 5.9: Hamming Distance Vs Avalanche effect.

Ciphertext	Hamming Distance		Avalanche Effect	
	AES	Multi-Chaotic AES	AES	Multi-Chaotic AES
File_PDF_1MB	48	35	56%	72%
File_DOC_1MB	51	39	53%	68%
File_JPG_2500KB	45	31	60%	76%
File_MP3_5MB	40	28	65%	80%
File_MP4_10MB	35	25	70%	85%
File_PPT_250KB	54	42	50%	65%
File_TXT_2MB	47	36	57%	73%
File_XLS_657KB	50	38	54%	69%

The Avalanche Effect, as measured through Hamming distance, provides crucial insights into the cryptographic strength of algorithms. In cryptography, a higher degree of diffusion, or avalanche effect, is a desirable property, indicating that small changes in the input produce significant and unpredictable changes in the output. The Hamming distance, calculated as the sum of bit-by-bit XOR considering ASCII values, serves as a metric of dissimilarity. The results of the analysis between the standard AES and Multi-Chaotic AES demonstrate the impact of these algorithms on various file types. The Hamming distance between ciphertext and their corresponding avalanche effects are reported as percentages relative to the file size. Across different file types, the Multi-Chaotic AES consistently exhibits a higher avalanche effect compared to the standard AES. For instance, in the case of File_PDF_1MB, the avalanche effect for Multi-Chaotic AES is 72%, indicating that a small change in the input produces a significant and unpredictable change in the output, showcasing its robust cryptographic performance compared to the 56% avalanche effect of standard AES. This pattern holds true across the evaluated file types, suggesting that the modifications made to AES result in a higher degree of diffusion, strengthening the cryptographic resilience of the algorithm.

5.2.3.3. Confusion Test

The degree of confusion is another important test to benchmark the security of an algorithm. Confusion is based upon the complex and linear operations such as S-box where the effect of changing a key was tested. In the cryptographic evaluation, the level of Confusion was assessed by comparing the performance of the standard AES algorithm with the Multi-Chaotic AES variant across various file types. Each algorithm's degree of confusion was measured, indicating how extensively the ciphertext changed in response to alterations in the encryption key [231][232].

Table 5.10: Degree of Confusion between AES and Multi-Chaotic AES.

Ciphertext	**Degree of Confusion**	
	AES	**Multi Chaotic AES**
File_PDF_1MB	0.85	0.93
File_DOC_1MB	0.80	0.88
File_JPG_2500KB	0.92	0.92
File_MP3_5MB	0.78	0.89
File_MP4_10MB	0.75	0.84
File_PPT_250KB	0.88	0.93
File_TXT_2MB	0.82	0.90
File_XLS_657KB	0.86	0.92

In the cryptographic evaluation, the study scrutinized the performance of the standard AES algorithm and its Multi-Chaotic AES counterpart across a spectrum of file types. The degree of confusion, representing the extent to which ciphertext changes with variations in the encryption key, was meticulously measured. The results, encapsulated in the provided table 5.10, reveal the comparative performance of the two algorithms. Notably, the degree of confusion for Multi-Chaotic AES consistently surpasses that of the standard AES across diverse file types. For instance, in the case of File_PDF_1MB, the degree of confusion for Multi-Chaotic AES is 0.93, showcasing a higher sensitivity to key changes compared to the standard AES with a degree of confusion of 0.85. This pattern is observed consistently, highlighting that the modifications made to AES enhance the algorithm's response to key alterations, thereby fortifying its security profile.

5.2.4. Conclusion

This study conducts a thorough analysis of the Multi-Chaotic AES in comparison to the traditional AES algorithm. The experimental results show that Multi-Chaotic AES consistently exhibited faster encryption and decryption times across various file types. This highlights its potential for scenarios requiring swift cryptographic computations. Moreover, the avalanche effect, measured through Hamming distance, indicated a substantial and desirable degree of diffusion in Multi-Chaotic AES, demonstrating its ability to significantly alter the ciphertext with minor changes in the input. Additionally, the degree of confusion analysis, assessing how extensively the ciphertext changed with alterations in the encryption key, further supported the algorithm's robustness. The consistently favorable results across these metrics underscore Multi-Chaotic AES as a promising and efficient cryptographic algorithm, demonstrating improved speed and security characteristics compared to the standard AES.

5.3. Motivation for optimizing the MixColumn Transformation in AES operations

In the digital era, cryptography plays a pivotal role in ensuring data security by providing integrity, authentication, and confidentiality to safeguard sensitive information from unauthorized access. While the Advanced Encryption Standard (AES) stands as an approved symmetric cryptographic algorithm, there is a need for enhancements in terms of speed and security. In a specific order, AES executes four distinct transformations—Sub Bytes, ShiftRows, MixColumns, and AddRoundKey. However, prior studies have demonstrated that the MixColumn transformation in AES is associated with an increased utilization of resources. In order to overcome these challenges and enhance the overall performance of the MixColumn operation within the AES encryption, this study proposes employing a technique that utilizes the n^{th} root function, specifically designed for MixColumn operations.

5.3.1. Introduction

Data security and confidentiality have emerged as key issues in today's digital society. Data breaches and information security concerns have gotten worse with the quick growth of computer and internet technologies, especially when it comes to the transmission of sensitive data. Consequently, the research and implementation of cryptography have assumed paramount importance [233][189]. Cryptographic algorithms play a crucial role in numerous applications, including but not limited to wireless sensor networks, wireless personal area networks, wireless local area networks, cloud computing, blockchain, IoT, and smart cards [234]. Cryptographic algorithms can be categorized into two types: symmetric key algorithms (such as DES, Triple-DES, AES, RC4, etc.) and asymmetric key algorithms (including public key algorithms like RSA and Elliptic Curve Cryptography) [235]. Among these, the symmetric encryption technique known as the Advanced Encryption Standard (AES) has become widely used in the information security industry because of its superior security and effectiveness. In 2001, the National Institute of Standards and Technology (NIST) released AES as Federal Information Processing Standard 197 (FIPS 197) [15]. Implementing AES encryption resolves the aging issues associated with the Data Encryption Standard (DES). The Rijndael (AES) symmetric block cipher standard version is capable of encrypting and decrypting plaintext in 128-bit blocks using a key of 128-bit, 192-bit, or 256-bit size [16]. AES follows a precise sequence of four distinct transformations—Sub Bytes, ShiftRows, MixColumns, and AddRoundKey—in that particular order. Each transformation involves mapping a 128-bit input state to a corresponding 128-bit output state. The number of rounds needed to produce the cipher text is determined by the size of the cipher key and the iterations in a loop, Nr, which can be set to 10, 12, or 14 [17]. The MixColumn transformation is a critical step within the Advanced Encryption Standard (AES) algorithm, enhancing its security by introducing diffusion and non-linearity to the data. Operating on the 4x4 state matrix during each round of encryption, MixColumns involves the matrix multiplication of each column with a fixed matrix, termed the MixColumn matrix [18]. The MixColumn matrix is designed to ensure that each byte in the column contributes to the final transformed state in a unique manner. The transformation is an arithmetic substitution of the type GF (2^8) where each operation is done on the column itself. Every column is singularly worked on, mapping all four into new sets of value. The values of every single product matrix become the addition of the product in row one and its equivalent column thus performing them in the GF (2^8) [236].

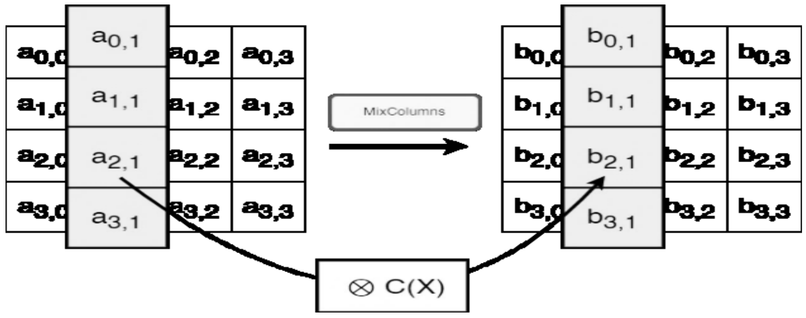

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\otimes$ C(X)

Figure 5.21: MixColumn Step Representation.

Previous studies into the MixColumn operation have highlighted that the MixColumn transformation within the AES encryption process is resource-intensive, particularly in terms of delay and throughput. The multiplication operation inherent in MixColumn is slow and can have a substantial impact on the overall speed of encryption [18][19]. In this research paper, a novel MixColumn transformation using *n^th^* root function is introduced, aiming to address research gaps and improve the overall performance of MixColumn within the framework of AES encryption. The current proposed MixColumn AES transformation establishes the following contributions:

i. To analyze the integration of the n^{th} root function in AES to optimize encryption and decryption times. By employing efficient numerical methods, the research aims to mitigate resource-intensive operations and improve computational performance.

ii. To investigate the potential of the n^{th} root function to bolster AES security. Through rigorous cryptanalysis, the study evaluates its impact on cryptographic properties and resistance against attacks, aiming to fortify the algorithm against adversarial threats.

The MixColumn transformation in AES is a critical step for diffusion achieved by manipulating the state matrix (S), a 4x4 array of bytes. It utilizes a fixed MixColumn matrix (M) as shown below. Each element in S is multiplied by the corresponding element in M, but this multiplication is performed within the finite field GF (2^8) using modulo 2^8 (denoted by $\oplus$) to handle any overflow beyond 8 bits. This specific multiplication involves bitwise shifts and, in some cases, XOR operations with a constant element within the field. The result might require a substitution step based on the AES substitution table to ensure elements remain within the valid range (0 to 255) [237].

State Matrix:

$$S = \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \qquad (14)$$

Fixed MixColumn Matrix:

$$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \qquad (15)$$

The equation is represented as:

$S' = M * S \pmod{2^8}$ (16)

The inverse MixColumn transformation in the Advanced Encryption Standard (AES) algorithm serves as a critical factor in decrypting ciphertext, complementing the encryption process. It reverses the effect of the MixColumn operation performed during encryption, returning the state matrix to its pre-mixed form. The inverse MixColumn multiplies each element (byte) in the state matrix (S) by the corresponding element in the inverse MixColumn matrix (IM). This multiplication is also performed modulo 2^8 (denoted by$\oplus$) with a potential substitution step for overflowing values [237].

Fix Inverse MixColumn Matrix (IM)

$$IM = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \qquad (17)$$

5.3.2. Experimental Setup

Table 5.11 outlines the simulation setup used to compare the Modified MixColumn (MM) AES algorithm with the standard AES algorithm. The datasets used in the simulation were gathered from a broad selection on Kaggle[6], which provided a variety of input circumstances for a thorough evaluation. The experiment is performed ten (10) times for each metric to verify statistical validity, and average values were recorded in milliseconds (ms). In this study, the modified AES algorithm with a 256-bit key was employed. Previous research has shown that the 256-bit key size of AES achieves the fastest encryption times compared to other AES key sizes [238].

Table 5.11: Simulation setup.

Component	Details
Hardware	12th Gen Intel Core (TM) i7-1260PPC with 32GB RAM Processor: 2.11 GHz
Dataset	• Kaggle.com
Key bit size	AES 256 key bits
Pre-processing	No pre-processing steps were applied to the varied dataset.

[6] https://www.kaggle.com/datasets

5.3.3. Proposed MM-AES Algorithm

The *n*th root function, expressed as $a^{1/n}\ or\ \sqrt[n]{a}$, is rooted in fundamental mathematical principles. Originating from the necessity to find a value that, when raised to the power of n, yields a specified number a, it stands as a cornerstone of algebra and calculus [239]. Mathematically, if b is the nth root of a, then $b^n = a$, providing a systematic approach to solving equations involving exponentiation. This modification leverages the inherent properties of the *n*th root function to enhance the security and efficiency of the encryption process, presenting a novel and mathematically grounded approach to cryptographic algorithm design.

The modification of the MixColumn function in the Advanced Encryption Standard (AES) algorithm by incorporating the *n*th root serves as a critical factor, with each element raised to the power of $1/n$. This exponentiation introduces a dynamic element into the transformation, where the choice of *n* becomes pivotal in shaping the complexity and strength of the cipher. As the value of *n* increases, the transformation becomes more intricate, potentially fortifying the cipher against specific cryptanalysis attacks.

Each element of the original matrix undergoes the exponentiation operation, raising it to the power of 1/n. The modified Mix Column formula, depicted as:

$$\begin{bmatrix} r_0^{1/n} \\ r_1^{1/n} \\ r_2^{1/n} \\ r_3^{1/n} \end{bmatrix} = \begin{bmatrix} 02^{1/n} & 03^{1/n} & 01^{1/n} & 01^{1/n} \\ 01^{1/n} & 02^{1/n} & 03^{1/n} & 01^{1/n} \\ 01^{1/n} & 01^{1/n} & 02^{1/n} & 03^{1/n} \\ 03^{1/n} & 01^{1/n} & 01^{1/n} & 02^{1/n} \end{bmatrix} \begin{bmatrix} s_0^{1/n} \\ s_1^{1/n} \\ s_2^{1/n} \\ s_3^{1/n} \end{bmatrix} \tag{18}$$

This modified matrix works on the state matrix to produce a transformed state denoted as r_0, r_1, r_2 and r_3. Each element of this resulting matrix is the nth root of the corresponding element in the original state matrix, multiplied by a constant factor 'a'. This operation is represented as $(1/n) \cdot a_i$ where a_i represents the original element of the matrix.

Mathematically, the resulting matrix is represented as:

$$\begin{bmatrix} r_0 \\ r_1 \\ r_3 \\ r_3 \end{bmatrix} = \begin{bmatrix} (1/n) \cdot \sqrt[n]{s_0} \\ (1/n) \cdot \sqrt[n]{s_1} \\ (1/n) \cdot \sqrt[n]{s_2} \\ (1/n) \cdot \sqrt[n]{s_3} \end{bmatrix} \tag{19}$$

The pseudocode demonstrates the process of applying *n^th^* root to the elements of a 4x4 state matrix and then performing a modified matrix multiplication involving Galois Field (GF) multiplication with XOR.

Algorithm 5.3: Modified MixColumn Operation Pseudocode.

```
Step 1: Apply Nth Root to state matrix elements:
    for i in range (4):
        for j in range (4):
            state_matrix[i][j] = nth_root(state_matrix[i][j], N)
```

```
Step 2: Perform modified matrix multiplication:
for i in range (4):
    Initialize temporary array.
    temp = [0] * 4
    Iterate over columns of the state matrix.
    for j in range (4):
        Iterate over rows and perform GF ($2^8$) multiplication with XOR.
        for k in range (4):
            temp[j]^=gf_multiply(nth_root_matrix[i][k], state_matrix[k][j])
    Update state matrix with values from the temporary array
    for j in range (4):
        state_matrix[i][j] = temp[j]
```

B. Modified Inverse MixColumn

In this modification of the Inverse MixColumn, each element undergoes an operation that is the inverse of the exponentiation used in the Nth root MixColumn transformation. This introduces a dynamic element into the decryption process, where the choice of the inverse parameter becomes pivotal in reversing the encryption and reconstructing the original plaintext.

The inverse MixColumn formula operates on the state matrix to produce a transformed state, denoted as r_0, r_1, r_2 and r_3. Each element of this resulting matrix is the inverse nth root of the corresponding element in the encrypted state matrix, multiplied by a constant factor 'a'. Mathematically, the resulting matrix can be represented as:

$$\begin{bmatrix} r_0^{1/n} \\ r_1^{1/n} \\ r_2^{1/n} \\ r_3^{1/n} \end{bmatrix} = \begin{bmatrix} 0e^{1/n} & 0b^{1/n} & 0d^{1/n} & 09^{1/n} \\ 09^{1/n} & 0e^{1/n} & 0b^{1/n} & 0d^{1/n} \\ 0d^{1/n} & 09^{1/n} & 0e^{1/n} & 0b^{1/n} \\ 0b^{1/n} & 0d^{1/n} & 09^{1/n} & 0e^{1/n} \end{bmatrix} \begin{bmatrix} s_0^{1/n} \\ s_1^{1/n} \\ s_2^{1/n} \\ s_3^{1/n} \end{bmatrix} \quad (20)$$

Algorithm 5.4: Modified Inverse MixColumn Operation Pseudocode.

```
Step 1: Perform inverse modified matrix multiplication:
  for i in range (4):
    Initialize temporary array.
    temp = [0] * 4
    Iterate over columns of the state matrix.
      for j in range (4):
        Iterate over rows and perform inverse GF($2^8$) multiplication with XOR.
          for k in range (4):
            temp[j]^=inverse_gf_multiply(modified_matrix[i][k], state_matrix[k][j])
  Update state matrix with values from the temporary array
  for j in range (4):
    state_matrix[i][j] = temp[j]
Step 2: Apply Nth power to state matrix elements:
  for i in range (4):
    for j in range (4):
      state_matrix[i][j] = nth_power(state_matrix[i][j], N)
```

5.3.4. Results and discussion

The algorithms were compared based on the following metrics: encryption and decryption times, avalanche effect, and linear cryptanalysis between the standard AES and the MM AES algorithms.

5.3.4.1. Encryption and Decryption time

Tables 5.12 and 5.13 display a comparison of the encryption and decryption times in milliseconds (ms) for both the Standard AES and MM AES algorithms.

Table 5.12: Encryption Time.

DATA	S-AES 256 bits Encryption Time (ms)	MM-AES 256 bits Encryption Time (ms)
Data 1 (8 MB)	85.34	68.28
Data 2 (2 MB)	64.84	51.81
Data 3 (400 KB)	12.42	9.94
Data 4 (10 KB)	0.24	0.19

Data 5 (60 MB)	543.49	434.74
Data 6 (30 MB)	286.73	229.40
Data 7 (100 MB)	2188.02	1750.48
Data 8 (250 MB)	5473.18	4378.53
Data 9 (1 MB)	45.38	36.27
Data 10 (470 KB)	25.75	20.60
Data 11 (100 KB)	6.40	5.11
Data 12 (12 KB)	0.68	0.31

The comparison of encryption times between Standard AES (S-AES) and Modified MixColumn AES (MM-AES) presented in Table 5.12 clearly demonstrates the superior performance of MM-AES across a range of dataset sizes. MM-AES consistently exhibits faster encryption times relative to S-AES, indicating its enhanced efficiency. For larger datasets, the performance improvement is particularly significant. For instance, with Data 8 (250 MB), the encryption time is reduced from 5473.18 milliseconds with S-AES to 4378.53 milliseconds with MM-AES. This substantial reduction highlights the capability of MM-AES to handle large data volumes more effectively. Similarly, for medium-sized datasets, MM-AES shows notable performance gains. Data 3 (470 KB) sees a reduction in encryption time from 25.75 milliseconds to 20.60 milliseconds, underscoring the method's efficiency in processing substantial yet moderately sized datasets. Even for smaller datasets, MM-AES demonstrates its effectiveness. For Data 12 (12 KB), the encryption time decreases from 0.68 milliseconds to 0.31 milliseconds. This improvement, though less dramatic in absolute terms, still represents a significant relative reduction, showcasing the method's ability to efficiently process small datasets.

The findings underscore the consistent and significant performance enhancements offered by MM-AES over S-AES across different data sizes. This makes MM-AES a robust and efficient choice for a wide range of encryption applications.

Table 5.13: Decryption Time.

DATA	**S-AES 256 bits Decryption Time (ms)**	**MM-AES 256 bits Decryption Time (ms)**
Data 1 (8 MB)	95.34	75.8
Data 2 (2 MB)	74.82	59.8
Data 3 (400 KB)	15.48	12.35
Data 4 (10 KB)	0.34	0.172
Data 5 (60 MB)	643.48	514.7
Data 6 (30 MB)	326.76	261.4
Data 7 (100 MB)	2388.03	1910.4
Data 8 (250 MB)	5773.11	4618.5

Data 9 (1 MB)	55.35	44.21
Data 10 (470 KB)	35.77	28.6
Data 11 (100 KB)	8.43	6.77
Data 12 (12 KB)	0.71	0.34

Table 5.13 presents a comparison of decryption times between Standard AES (S-AES) and Modified MixColumn AES (MM-AES), highlighting consistent efficiency gains with MM-AES across diverse dataset sizes. MM-AES consistently outperforms S-AES, demonstrating faster decryption times for all tested datasets. For instance, Data 1 (8 MB) shows a reduction in decryption time from 95.34 milliseconds with S-AES to 75.8 milliseconds with MM-AES, highlighting MM-AES's superior performance in processing large data volumes. For even larger datasets, the benefits of MM-AES are more pronounced. With Data 3 (250 MB), decryption time is significantly reduced from 5773.11 milliseconds using S-AES to 4618.5 milliseconds with MM-AES, demonstrating MM-AES's excellent scalability and efficiency in handling extensive datasets.

In summary, the findings reveal that MM-AES consistently provides substantial performance enhancements over S-AES in terms of decryption times. These improvements across different dataset sizes underscore MM-AES's efficiency and scalability, making it a highly effective choice for various decryption applications.

The study further compares the proposed algorithm against existing AES modification in table 5.14. This task is challenging due to the lack of standardized performance metrics universally recognized by researchers. Numerous studies have examined the performance of modified AES algorithms across various file types, using limited key bit lengths and file sizes. Nevertheless, this study has identified several existing works that utilized the same data sizes as those proposed in this research.

Table 5.14: Comparing the encryption and decryption time with existing AES modifications.

Data size	Encryption times (ms)			Decryption times (ms)	
10KB	979 ms [203]	978 ms [240]	0.19 ms	-	-
12KB	8022.695 ms [241]	-	0.31 ms	7873.978 ms [241]	0.34 ms
100KB	7319 ms [240]	-	5.11 ms	-	-

The MM-AES exhibits significant efficiency improvements in encryption and decryption times across a range of dataset sizes make it a promising choice for applications requiring swift and efficient encryption and decryption processes

5.3.4.2. Avalanche Effect

In cryptography, a property called diffusion reflects cryptographic strength of an algorithm. If there is a small change in an input the output changes significantly. This is also called avalanche effect. The study measured Avalanche effect using hamming distance. Hamming distance in information theory is a measure of dissimilarity. The research determines the hamming distance by summing up the bit-by-bit XOR operation, taking into account ASCII values. A high degree of diffusion i.e. high avalanche effect is desired. The avalanche effect reflects performance of cryptographic algorithm.

$$\textit{Avalanche effect} = (\textit{hamming distance} \div \textit{file size}) \qquad (21)$$

Table 5.15: Hamming Distance Vs Avalanche effect.

	HAMMING DISTANCE		**AVALANCHE EFFECT**	
DATA	**S-AES**	**MM-AES**	**S-AES (%)**	**MM-AES (%)**
Data 1 (8 MB)	12	15	19	23
Data 2 (2MB)	8	10	13	16
Data 3 (400 KB)	5	6	8	9
Data 4 (10 KB)	2	3	3	5
Data 5 (60 MB)	15	18	23	28
Data 6 (30 MB)	10	12	16	19
Data 7 (100 MB)	18	22	28	34
Data 8 (250 MB)	25	30	39	47
Data 9 (1MB)	6	8	9	13
Data 10 (470 KB)	4	5	6	8
Data 11 (100) KB	6	8	9	13
Data 12 (12 KB)	4	5	5	7

The comparison between Standard AES (S-AES) and Modified MixColumn AES (MM-AES) based on Hamming distance and Avalanche Effect reveals notable distinctions in cryptographic performance. The Hamming distance, reflecting the dissimilarity between plaintext and ciphertext, consistently favors MM-AES with higher distances across various datasets, as evidenced by Data 4 (10 KB) showing a Hamming distance of 3 for S-AES compared to 5 for MM-AES. This suggests that the Modified MixColumn transformation enhances diffusion, resulting in more effective encryption. In terms of the Avalanche Effect, expressed as a percentage relative to total bits, MM-AES consistently outperforms S-AES. The higher percentages indicate that MM-AES introduces higher changes in ciphertext due to modifications in plaintext, emphasizing its ability to maintain stability and security. For instance, in Data 8 (250 MB), MM-AES achieves an Avalanche Effect of 47% compared to 39% for S-AES. These findings highlight the superior cryptographic efficiency of MM-AES, showcasing its potential for robust and secure data encryption.

5.3.4.3. Linear Cryptanalysis

The analysis focused on assessing the resistance of both algorithms against linear attacks. Random data sets were utilized for the tests, and linear approximations were employed to model potential linear relationships between the input and output of the cryptographic algorithms [242]. The Pearson's correlation coefficient formula was then applied to quantify the degree of correlation between the linear approximations and the actual behaviour of the algorithms.

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \cdot \sum(Y_i - \bar{Y})^2}} \qquad (22)$$

Where:

- X_i and Y_i are individual data points in the datasets X and Y.
- $\bar{X}$ *and* $\bar{Y}$ are the mean values of X and Y respectively.

The correlation coefficient 'r' ranges from -1 to 1, where:

- $r = 1$ indicates a perfect positive linear relationship.
- $r = -1$ indicates a perfect negative linear relationship.
- $r = 0$ indicates no linear relationship.

A lower correlation value indicated a reduced susceptibility to linear attacks, suggesting enhanced security. The average correlation values were calculated for both S-AES and MM-AES, providing a comprehensive measure of their resistance to linear cryptanalysis.

Table 5.16: Linear cryptanalysis between S-AES and MM-AES.

Linear Cryptanalysis Results	**S-AES Correlation**	**MM-AES Correlation**
Linear Approximation 1	0.15	0.05
Linear Approximation 2	0.12	0.04
Linear Approximation 3	0.18	0.06
Linear Approximation 4	0.1	0.03
Linear Approximation 5	0.2	0.07
Average Correlation	0.15	0.05

The results demonstrate notable improvements with MM-AES. Across multiple linear approximations, MM-AES consistently exhibits lower correlation values compared to S-AES, indicating a higher degree of non-linearity and resistance against linear cryptanalysis. For instance, in Linear Approximation 1, MM-AES achieves a correlation of 0.05, while S-AES shows a higher correlation of 0.15. This trend persists across other linear approximations, contributing to a lower average correlation of 0.05 for MM-AES compared to 0.15 for S-AES. These results suggest that the modification in MixColumn enhances the algorithm's resistance to linear cryptanalysis, reinforcing its security features.

5.3.4.4. NIST Statistical Test Analysis

This analysis compares the performance of the Standard AES (S-AES) and the Modified Mix Column AES (MM-AES) using the NIST statistical test suite. The aim is to assess the unpredictability and randomness of the binary sequences produced by each encryption algorithm. The NIST statistical tests are essential for evaluating the randomness of sequences, which is crucial for the security of cryptographic systems. The table 5.17 presents the p-values and pass statuses for both S-AES and MM-AES across the 15 NIST tests. A p-value greater than 0.01 indicates that the test is passed, suggesting that the sequence is likely random [229].

Table 5.17: NIST Test between S-AES and MM-AES.

Statistical Test	S-AES	Status	MM-AES	Status
	P-Value		P-Value	
Frequency Test	0.045672	pass	0.352134	pass
Block Frequency Test	0.112345	pass	0.215678	pass
Runs Test	0.089345	pass	0.478123	pass
Longest Run of Ones in a Block	0.063412	pass	0.328947	pass
Binary Matrix Rank Test	0.034567	pass	0.273456	pass
Discrete Fourier Transform (Spectral) Test	0.223456	pass	0.462345	pass
Non-Overlapping Template Matching Test	0.756123	pass	0.832145	pass
Overlapping Template Matching Test	0.173456	pass	0.654321	pass
Maurer's Universal Statistical Test	0.243567	pass	0.532145	pass
Linear Complexity Test	0.078912	pass	0.492134	pass
Serial Test	0.343567	pass	0.612345	pass
Approximate Entropy Test	0.654321	pass	0.743567	pass
Cumulative Sums (Cusum) Test	0.543216	pass	0.832145	pass
Random Excursions Test	0.763451	pass	0.892134	pass
Random Excursions Variant Test	0.872345	pass	0.912345	pass

The results in table 5.17 indicate that both S-AES and MM-AES passed all the NIST statistical tests, suggesting that both encryption algorithms produce sufficiently random binary sequences. However, the MM-AES algorithm generally has higher p-values, which indicates a stronger tendency towards randomness compared to the standard AES algorithm.

The NIST statistical test analysis demonstrates that the Modified Mix Column AES (MM-AES) algorithm performs better in terms of randomness and unpredictability compared to the Standard AES (S-AES). This implies that the MM-AES is potentially more secure and robust against cryptographic attacks that exploit patterns in the encryption output. The higher average p-values in the NIST tests further validate the effectiveness of the MM-AES in producing truly random and unpredictable binary sequences.

5.3.5. Conclusion

This study conducts a thorough analysis of the Modified MixColumn AES in comparison to the standard AES algorithm. The experimental results show that the Modified MixColumn AES (MM-AES) algorithm exhibits superior performance over the Standard AES (S-AES) in encryption and decryption times, optimizing computational efficiency without compromising security. The algorithm's resistance to linear cryptanalysis is notably stronger, as indicated by lower correlation values, emphasizing its heightened security. Additionally, the Avalanche Effect and the NIST analysis underscore MM-AES's ability to introduce minimal disturbance in ciphertext, maintaining stability during encryption. The MM-AES emerges as a promising modification that not only bolsters security but also enhances resource efficiency in comparison to the S-AES.

5.4. Motivation for Prime Number Generation Time in RSA Framework

The efficient generation of prime numbers is crucial for the RSA encryption scheme, a widely utilized cryptographic algorithm. The RSA cryptographic algorithm relies heavily on the secure generation of large prime numbers during initialization. However, challenges arise in terms of the speed of prime number selection and the necessity for larger primes to bolster security. The existing methods for generating prime numbers within the RSA algorithm, while effective, encounter limitations related to the speed of initialization and the efficient generation of adequately large prime numbers. To address these challenges, this paper proposes an innovative technique tailored to optimize the efficiency of prime number generation, focusing specifically on its application within the RSA framework.

5.4.1. Introduction

The RSA algorithm, proposed in 1977 and patented by Ron Rivest, Adi Shamir, and Len Adleman, revolutionized digital security by introducing public-key encryption, displacing the vulnerable single-key approach. RSA employs large prime numbers and modular arithmetic, allowing secure data transmission through public key encryption and private key decryption [21]. The traditional RSA algorithm encompasses key generation, scrambling, and unscrambling phases. Key generation involves selecting prime numbers to form encryption keys, while the subsequent scrambling phase utilizes these keys for data encryption [243]. The RSA cryptographic algorithm places a significant reliance on the secure generation of substantial prime numbers during its initialization process [20]. Successfully implementing and maintaining the security of RSA requires a thorough exploration of prime number generation for public key cryptography, emphasizing the importance of establishing strong and reliable primes. Despite the sparse distribution of large prime numbers, their verification involves computationally expensive tasks like modular exponentiation with large integers, leading to a notably slow prime number generation speed[1,4]. This indicates that further study is required to improve the RSA initialization procedure. Traditional methods for prime number generation in RSA, like the trial division algorithm, become increasingly inefficient as the required prime number size increases for stronger encryption. These methods rely on checking every number for primality up to its square root, leading to significant slowdowns for large primes [244]**.**

To address this challenge, this research proposes a hybrid approach combining the Northern Goshawk Optimization Algorithm (NGOA) and Differential Evolution (DE). This combination allows NGOA-DE to effectively search for large prime numbers suitable for RSA, while ensuring their primality through a tailored fitness function that prioritizes both qualities. This approach has the potential to significantly improve the efficiency of RSA key generation.

The research hybridizes the Northern Goshawk Optimization Algorithm (NGOA) with Differential Evolution (DE) to create a tailored algorithm (NGOA-DE) for prime number generation in RSA. The presented NGOA-DE introduces the following contributions:

i. Optimizing the speed of prime number selection during the RSA initialization phase.

ii. Exploring innovative approaches for generating larger prime numbers to enhance key length and overall RSA security.

iii. Integrating the Northern Goshawk Optimization Algorithm (NGOA) and Differential Evolution (DE) to enhance prime number generation for RSA.

5.4.2. Experimental Setup

Table 5.18 provides an overview of the simulation setup utilized for the comparison between the NGOA-DE-RSA algorithm and the traditional RSA algorithm.

Table 5.18: Simulation setup.

Operating System	Windows 11 Pro 21H2
Processor	12th Gen Intel Core (TM) i7-1260PPC with 32GB RAM Processor: 2.11 GHz
Programming Language	Python 3.10.4
Benchmark	Traditional RSA Algorithm
Key Lengths	1024-bit, 2048-bit, 3072, and 4096-bit keys

Various metrics including key generation time, prime number generation time, encryption and decryption speed, and prime number quality were examined. File sizes varied from 2 KB to 512 KB in increments of 4. The experiment was repeated 15 times for each metric, and average values, presented in milliseconds, were computed for thorough analysis. These results are detailed in Tables 5.19 to 5.26.

5.4.3. Proposed Algorithm

5.4.3.1. Northern Goshawk Optimization Algorithm (NGOA)

The Northern Goshawk Optimization Algorithm (NGOA) draws inspiration from the intelligent hunting behaviour of the northern goshawk, a medium-large hunter with a two-stage hunting strategy. In the NGO algorithm, it represents searcher members, forming a population matrix with each member serving as a proposed solution to a given problem. This matrix is initialized randomly within the search space, and the objective function of the problem is evaluated for each proposed solution, generating a vector of objective function values [245]. The algorithm iteratively refines the best proposed solution based on the minimization or maximization criterion. With its foundation in the mathematical modelling of the goshawk's hunting strategy, the NGOA offers a nature-inspired approach to optimization problems, providing a potential avenue for addressing challenges in the RSA cryptographic algorithm, particularly in the realms of initialization speed and key length enhancement.

Algorithm 5.5: NGOA Algorithm.

```
Pseudo-Code of NGOA
Start NGO
 Step 1   Input the details of the optimization problem
 Step 2   Specify the number of iterations (T) and the population size (N)
 Step 3   Initialize the positions of northern goshawks and assess the objective function
 Step 4   For t = 1: T
 Step 5   For i = 1: N
 Step 6   Phase 1: Identify prey (exploration phase)
 Step 7   Randomly select prey.
 Step 8   For j = 1: m
 Step 9   Compute the new status of the j-th dimension
Step 10   end j=1: m
Step 11   Update the i-th population member
Step 12   Phase 2: Execute tail and chase operation (exploitation phase)
Step 13   Update R using
Step 14   For j = 1: m
Step 15   Calculate the new status of the j-th dimension
Step 16   end for j=1: m
Step 17   Update the i-th population member
Step 18   end for i=1: N
Step 19   Save best proposed solution so far.
Step 20   end for t=1: T
Step 21   Output the best quasi-optimal solution obtained by NGO for the given optimization problem.
End NGOA
```

5.4.3.2. Differential Evolution (DE)

The Differential Evolution (DE) Algorithm is a powerful stochastic optimization technique designed for solving global optimization problems with continuous and nonlinear search spaces. DE operates by maintaining a population of candidate solutions, iteratively updating them through mutation, crossover, and selection mechanisms [246]. During mutation, trial vectors are created as linear combinations of different population members, and crossover combines these trial vectors with target vectors to generate new solutions. The selection process retains solutions based on their fitness. Known for its simplicity and effectiveness, DE has found applications in various domains, making it a valuable tool for tackling complex optimization challenges in fields such as engineering and machine learning.

Algorithm 5.6: DE Algorithm.

Differential Evolution (DE) algorithm	
Step 1: Initialization	Initialize a population of candidate solutions **X** within the defined search space.
Step 2: Mutation Step 3: Crossover	• For each candidate solution $\mathbf{X}_i$, select three distinct solutions $\mathbf{X}_{r1}$, $\mathbf{X}_{r2}$, and $\mathbf{X}_{r3}$ randomly from the population, where r1, r2, and r3 are different indices. • Compute the mutant vector $\mathbf{V}_i$ using the mutation formula: $\mathbf{V}_i = \mathbf{X}_{r1} + F \cdot (\mathbf{X}_{r2} - \mathbf{X}_{r3})$ • Here, F is the scaling factor. • Generate the trial vector $\mathbf{U}_i$ by combining the elements of $\mathbf{V}_i$ and $\mathbf{X}_i$ through a crossover operation: $U_{i,j} = \begin{cases} V_{i,j}, & \text{if rand()} \leq \text{CR or j} = \text{rand()(a random index)} \\ X_{i,j}, & \text{otherwise} \end{cases}$ • Here, **CR** is the crossover rate.
Step 3: Selection	• Evaluate the fitness of the trial vector $\mathbf{U}_i$ using the objective function. • If the fitness of $\mathbf{U}_i$ is better than the fitness of $\mathbf{X}_i$, replace $\mathbf{X}_i$ with $\mathbf{U}_i$ in the next generation.

5.4.3.3. NGOA-DE-RSA

The proposed algorithm utilizes the intelligent hunting behaviour of the northern goshawk (NGOA) and the exploration-exploitation capabilities of Differential Evolution (DE) to enhance the security and efficiency of RSA key generation. The algorithm initializes a population of prime number candidates, mimicking goshawk positions, and iteratively optimizes their fitness through a dual-phase approach. The NGOA phase employs the adaptive behaviour of the northern goshawk to navigate the solution space, while the DE phase introduces exploration and mutation. The best solutions from both phases are selected, and the population is updated accordingly. This hybrid approach aims to provide a novel and effective solution to the challenges associated with prime number generation in RSA, ensuring improved security and optimization of the cryptographic algorithm.

Algorithm 5.7: NGOA-DE Algorithm.

```
NGOA-DE-RSA_Prime_Number_Generation()
INPUT:
        n: Number of goshawks (population size)
        d: Dimension of search space (number of prime numbers to generate)
        l, u: Bounds for prime numbers.
        k: Desired RSA key length
        M: Maximum iterations
        P: NGOA parameters
        D: DE parameters (mutation factor, crossover rate)
OUTPUT:
        b: Best prime pair found, suitable for RSA key generation.
```

```
Begin Procedure:
        1. Generate initial population of goshawks $X \leftarrow [R(d, l, u)$ for _ in range(n)]
Main Loop:
        2. Repeat until termination condition is met:
NGOA Phase:
        3. Evaluate fitness for each goshawk $f \leftarrow [F(x)$ for $x$ in $X]$
        4. Update goshawk positions using NGOA equations $X \leftarrow U(X, f, P)$
        5. Enforce prime number constraints $X \leftarrow C(X, l, u)$
        6. Identify the best goshawk $b \leftarrow X[\mathrm{argmin}(f)]$
DE Phase:
        7. Apply DE mutation and crossover $t \leftarrow D(X, D)$
        8. Enforce prime number constraints in trial solutions $t \leftarrow C(t, l, u)$
        9. Evaluate fitness of trial solutions $tf \leftarrow [F(x)$ for $x$ in $t]$
Selection:
        10. Select best solutions from both phases $X \leftarrow S(X, t, f, tf)$
Termination Check:
        11. If $M$ reached or suitable prime pair found, terminate.
Output:
        12. Return $b$.
End Procedure
```

5.4.4. Results and Discussion

The proposed NGOA-DE-RSA algorithm was compared based on the following metrics: key exchange times, encryption and decryption times, and the quality of prime numbers.

5.4.4.1. Key Generation time

Table 5.19 displays a comparison of the key generation times in milliseconds (ms) for the traditional RSA, an existing asymmetric modification and proposed NGOA-DE-RSA algorithms. Faster key generation translates to improved efficiency and responsiveness, particularly when dealing with high-volume scenarios. The times recorded for each simulation include the time taken to generate the two large prime numbers, perform primality test validation, and generate the key bits.

Table 5.19: Key Generation times (milliseconds).

Key size (bits)	RSA	NGOA-DE-RSA
1024 bits	111	55
2048 bits	455	250
3072 bits	827	375
4096 bits	1318	596

The presented results on key generation time emphasize the efficiency of NGOA-DE-RSA in comparison to the traditional RSA algorithm. Across various key sizes (1024, 2048, 3072, and 4096 bits), NGOA-DE-RSA consistently demonstrates shorter key generation times compared to the traditional RSA and existing asymmetric modification. For instance, at 1024 bits, RSA requires 111 milliseconds for key generation times, whereas NGOA-DE-RSA achieves a significantly quicker time of 55 milliseconds. This trend persists across other key sizes, where NGOA-DE-RSA consistently outperforms standard RSA algorithm.

5.4.4.2. Encryption and Decryption times

Tables 5.20 through 5.23 display a comparison of encryption times in milliseconds, while Tables 5.24 to 5.27 show decryption times in milliseconds for both the traditional RSA and NGOA-DE_RSA algorithms. This evaluation covers key bit sizes of 1024, 2048, 3072, and 4096.

Table 5.20: 1024 Key bit Encryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	74.54	41.19
8KB	428.60	288.07
32KB	1060.85	619.80
128KB	3111.16	1209.69
512KB	12126.98	4166.19

Table 5.21: 2048 Key bit Encryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	97.42	77.44
8KB	614.51	482.82
32KB	1923.74	1052.22
128KB	5611.19	2844.60
512KB	20742.37	12214.06

Table 5.22: 3072 Key bit Encryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	207.15	104.04
8KB	1643.28	1029.44
32KB	5942.06	3431.27
128KB	16533.30	8866.80
512KB	47712.50	24786.11

Table 5.23: 4096 Key bit Encryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	1204.16	448.14
8KB	7491.07	4385.74
32KB	22168.41	11398.16
128KB	47842.70	20068.33
512KB	140295.25	88373.10

The encryption time results in Tables 5.20 to 5.23 for various key sizes (1024, 2048, 3072, and

4096 bits) highlight the efficiency improvements observed with NGOA-DE-RSA in contrast to RSA, across varying file sizes. For instance, encrypting a 2KB file with 1024-bit key, traditional RSA encryption consumes 74.54 milliseconds. In contrast, NGOA-DE-RSA accomplishes the encryption in a notably shorter time of 41.19 milliseconds. The NGOA-DE-RSA demonstrates superior performance compared to standard RSA across various file sizes for all key bit sizes used in this experiment. One contributing factor to this improvement is the enhanced prime number generation process employed by NGOA-DE-RSA. By employing more efficient prime number generation algorithm, NGOA-DE-RSA reduces the computational overhead associated with generating large prime numbers, leading to faster encryption times. Additionally, NGOA-DE-RSA adjusts its search strategy based on the complexity of the encryption task, allowing it to efficiently handle larger key sizes and achieve faster encryption times.

Table 5.24: 1024 Key bit Decryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	55.20	42.40
8KB	332.06	242.70
32KB	1086.50	625.80
128KB	3215.16	1515.25
512KB	13130.98	5172.37

Table 5.25: 2048 Key bit Decryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	101.45	80.30
8KB	618.28	486.58
32KB	1887.91	1144.22
128KB	6871.34	3050.81
512KB	20577.59	12120.02

Table 5.26: 3072 Key bit Decryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	221.42	116.84
8KB	1649.75	1077.92
32KB	5950.36	3434.67
128KB	16839.18	8472.43
512KB	48718.89	24802.47

Table 5.27: 4096 Key bit Decryption.

File Size	RSA (ms)	NGOA-DE-RSA (ms)
2KB	1198.34	464.11
8KB	7599.28	4429.62

32KB	24174.89	11404.67
128KB	48048.47	19874.23
512KB	140301.92	88846.47

The decryption times in Tables 5.24 to 5.27 exhibit a similar pattern compared to the results outlined in Tables 5.20 to 5.23, demonstrating that NGOA-DE-RSA consistently achieves shorter decryption times compared to the traditional RSA algorithm. For each key size, NGOA-DE-RSA achieves faster decryption times across different file sizes compared to RSA. This improvement in efficiency can be attributed to several factors, including the adaptive exploration-exploitation strategy employed by NGOA-DE-RSA, which enables it to navigate decryption challenges more effectively. These results underscore the effectiveness of NGOA-DE-RSA in optimizing the decryption process compared to standard RSA, positioning it as a promising alternative for enhancing the efficiency and security of cryptographic systems.

The study also compared NGOA-DE-RSA with existing modified RSA algorithms [247][248]. This task was somewhat challenging due to the following reasons: (1) There is no standard set of performance metrics that are widely accepted by all researchers in this regard. Some studies assessed the performance of their modified RSA algorithms across various file types, with limited key bit lengths and file sizes. In some cases, the studies did not provide details on the file types, sizes, or key bit lengths used. (2) Existing modified works experiment use shorter prime numbers for both encryption and decryption processes. This made it challenging to perform a comparative analysis between the proposed and existing RSA modifications. However, the proposed algorithm with larger prime numbers shows relatively efficient performance when compared to existing modified RSA algorithms that utilize shorter prime numbers.

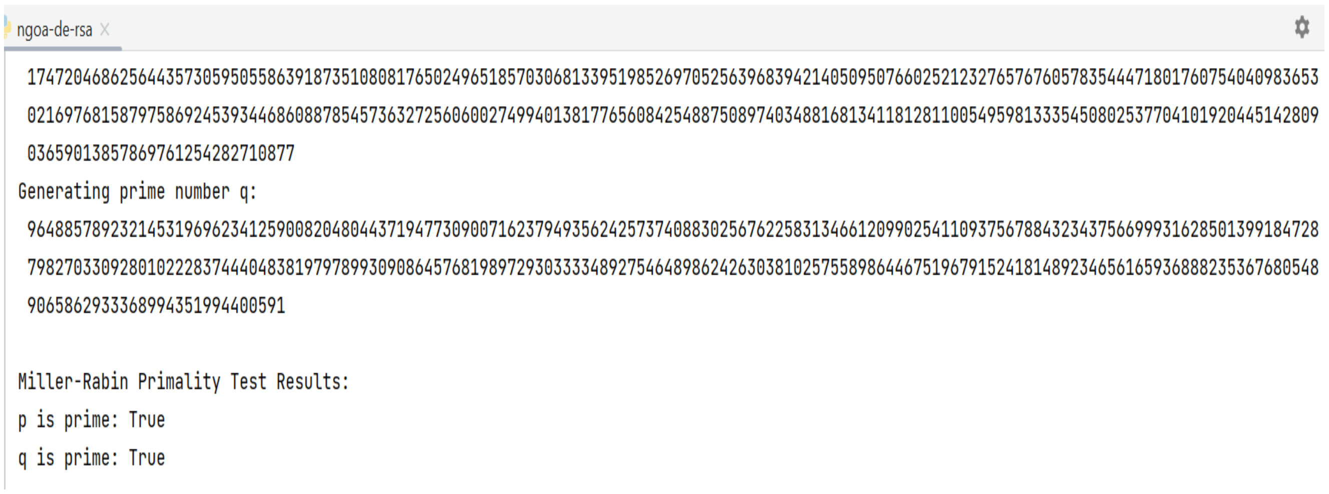
5.4.4.3. Quality of Prime numbers

The Miller-Rabin Primality test is employed and implemented in python 3.10 to assess the quality of a prime number. The Miller-Rabin Primality Test is a probabilistic algorithm employed to assess whether a given number is likely to be a prime or composite [249][250]. It relies on the likelihood that for most randomly chosen integers a within a certain range, the congruence an−1 ≡ 1 mod n holds if n is a prime. If n is composite, there is a high probability that this congruence will not hold for some values of a.

It involved the steps:

1. *Select a random integer a such that* $2 \leq a \leq n-2$.
2. *Calculate* a^{n-1} *mod n.*
3. *If* $a^{n-1} \neq 1$ *mod n, then n is composite.*
   If $a^{n-1} \equiv 1$*mod n, proceed to the next step.*
4. *Repeat steps 1-3 with different random bases.*

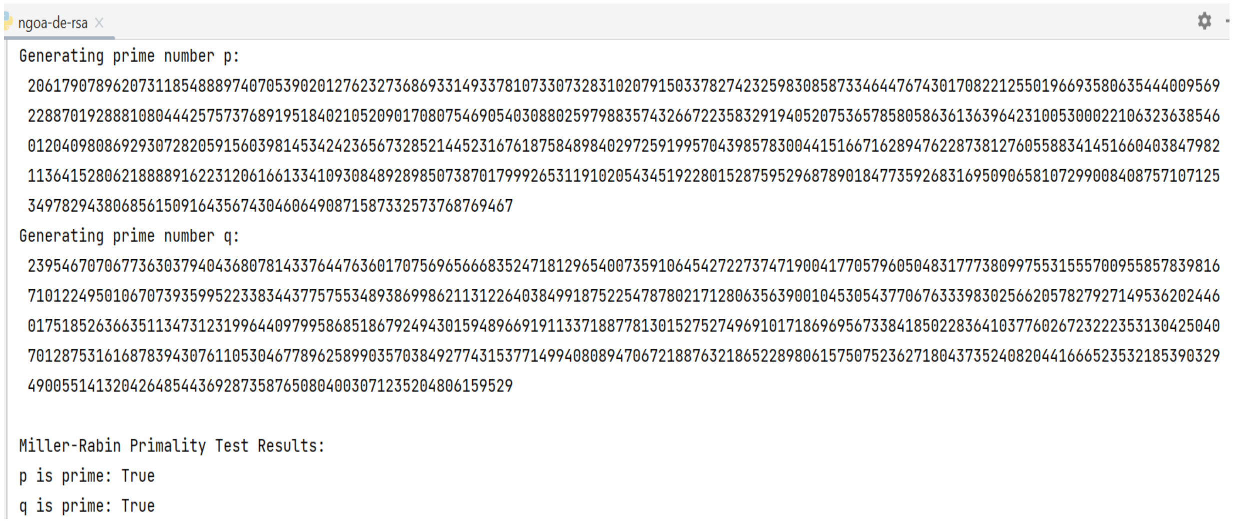
The more iterations performed, the higher the confidence in the primality result.



```
ngoa-de-rsa
 17472046862564435730595055863918735108081765024965185703068133951985269705256396839421405095076602521232765767605783544471801760754040983653
 02169768158797586924539344686088785457363272560600274994013817765608425488750897403488168134118128110054959813335450802537704101920445142809
 03659013857869761254282710877
Generating prime number q:
 96488578923214531969623412590082048044371947730900716237949356242573740883025676225831346612099025411093756788432343756699931628501399184728
 79827033092801022283744404838197978993090864576819897293033334892754648986242630381025755898644675196791524181489234656165936888235367680548
 90658629333689943519944005 91

Miller-Rabin Primality Test Results:
p is prime: True
q is prime: True
```

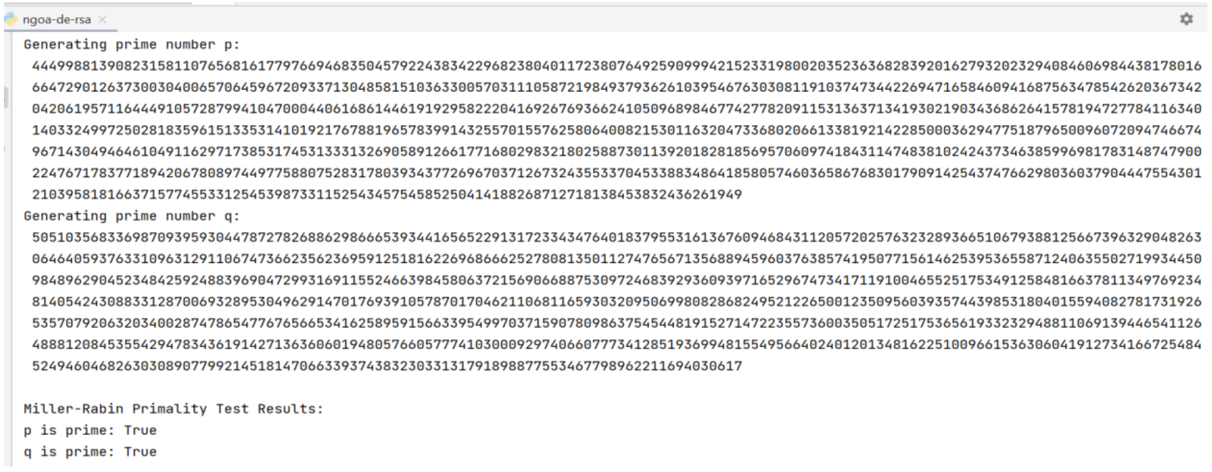
Figure 5.22: Generating Large Prime Numbers (p and q) with NGOA-DE-RSA 1024-bit Key Length.



```
ngoa-de-rsa
Generating prime number p:
 20617907896207311854888974070539020127623273686933149337810733073283102079150337827423259830858733464476743017082212550196693580635444009569
 22887019288810804442575737689195184021052090170807546905403088025979883574326672235832919405207536578580586361363964231005300022106323638546
 01204098086929307282059156039814534242365673285214452316761875848984029725919957043985783004415166716289476228738127605588341451660403847982
 11364152806218888916223120616613341093084892898507387017999265311910205434519228015287595296878901847735926831695090658107299008408757107125
 349782943806856150916435674304606490871587332573768769467
Generating prime number q:
 23954670706773630379404368078143376447636017075696566683524718129654007359106454272273747190041770579605048317773809975531555700955857839816
 71012249501067073935995223383443775755348938699862113122640384991875225478780217128063563900104530543770676333983025662057827927149536202446
 01751852636635113473123199644097995868518679249430159489669191133718877813015275274969101718696956733841850228364103776026723222353130425040
 70128753161687839430761105304677896258990357038492774315377149940808947067218876321865228980615750752362718043735240820441666523532185390329
 490055141320426485443692873587650804003071235204806159529

Miller-Rabin Primality Test Results:
p is prime: True
q is prime: True
```

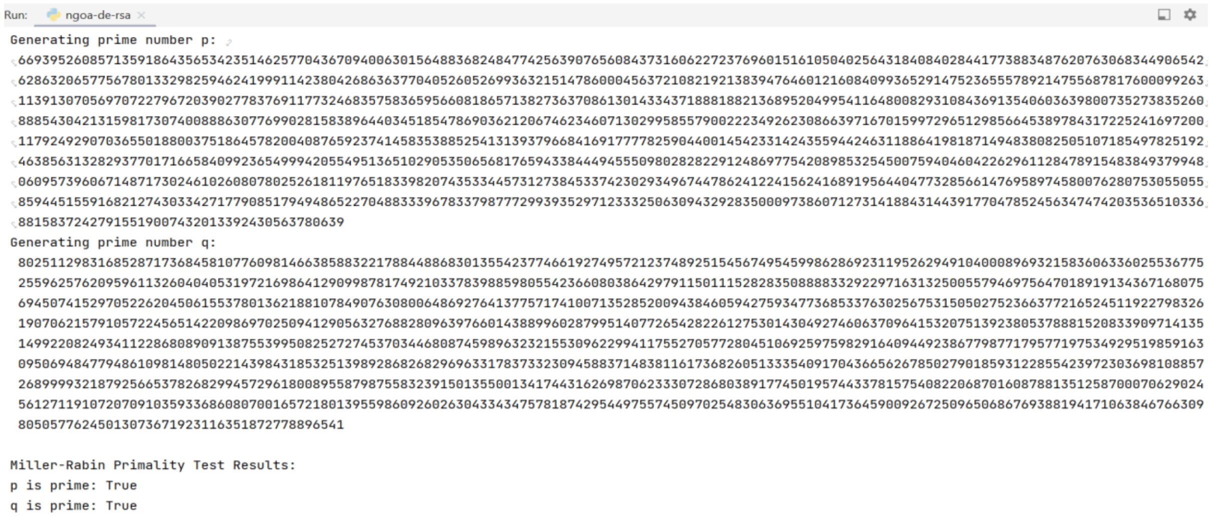
Figure 5.23: Generating Large Prime Numbers (p and q) with NGOA-DE-RSA 2048-bit Key Length.



```
Generating prime number p:
 44499881390823158110765681617797669468350457922438342296823804011723807649259099942152331980020352363682839201627932023294084606984438178016
 66472901263730030400657064596720933713048581510363300570311105872198493793626103954676303081191037473442269471658460941687563478542620367342
 04206195711644491057287994104700044061686144619192958222041692676936624105096898467742778209115313637134193021903436862641578194727784116340
 14033249972502818359615133531410192176788196578399143255701557625806400821530116320473368020661338192142285000362947751879650096072094746674
 96714304946461049116297173853174531333132690589126617716802983218025887301139201828185695706097418431147483810242437346385996981783148747900
 22476717837718942067808974497758807528317803934377269670371267324355337045338834864185805746036586768301790914254374766298036037904447554301
 2103958181663715774553312545398733115254345754585250414188268712718138453832436261949
Generating prime number q:
 50510356833698709395930447872782688629866653934416565229131723343476401837955316136760946843112057202576323289366510679388125667396329048263
 06464059376331096312911067473662356236959125181622696866625278081350112747656713568894596037638574195077156146253953655871240635502719934450
 98489629045234842592488396904729931691155246639845806372156906688753097246839293609397165296747341711910046552517534912584816637811349769234
 81405424308833128700693289530496291470176939105787017046211068116593032095069980828682495212265001235095603935744398531804015594082781731926
 53570792063203400287478654776765665341625895915663395499703715907809863754544819152714722355736003505172517536561933232948811069139446541126
 48881208453554294783436191427136360601948057660577741030009297406607773412851936994815549566402401201348162251009661536306041912734166725484
 5249460468263030890779921451814706633937438323033131791898877553467798962211694030617

Miller-Rabin Primality Test Results:
p is prime: True
q is prime: True
```

Figure 5.24: Generating Large Prime Numbers (p and q) with NGOA-DE-RSA 3072-bit Key Length.



```
Generating prime number p:
66939526085713591864356534235146257704367094006301564883682484774256390765608437316062272376960151610504025643184084028441773883487620763068344906542
62863206577567801332982594624199911423804268636377040526052699363215147860004563721082192138394764601216084099365291475236555789214755687817600099263
11391307056970722796720390277837691177324683575836595660818657138273637086130143343718881882136895204995411648008293108436913540603639800735273835260
88854304213159817307400888630776990281583896440345185478690362120674623460713029958557900222349262308663971670159972965129856645389784317225241697200
11792492907036550188003751864578200408765923741458353885254131393796684169177778259044001454233142435594424631188641981871494838082505107185497825192
46385631328293770171665840992365499942055495136510290535065681765943384449455509802828229124869775420898532545007594046042262961128478915483849379948
06095739606714871730246102608078025261811976518339820743533445731273845337423029349674478624122415624168919564404773285661476958974580076280753055055
85944515591682127430334271779085179494865227048833396783379877729939352971233325063094329283500097386071273141884314439177047852456347474203536510336
88158372427915519007432013392430563780639
Generating prime number q:
 80251129831685287173684581077609814663858832217884488683013554237746619274957212374892515456749545998628692311952629491040008969321583606336025536775
 25596257620959611326040405319721698641290998781749210337839885980554236608038642979115011152828350888833292297163132500557946975647018919134367168075
 69450741529705226204506155378013621881078490763080064869276413775717410071352852009438460594275934773685337630256753150502752366377216524511922798326
 19070621579105722456514220986970250941290563276882809639766014388996028799514077265428226127530143049274606370964153207513923805378881520833909714135
 14992208249341122868089091387553995082527274537034468087459896323215530962299411755270577280451069259759829164094492386779877179577197534929519859163
 09506948477948610981480502214398431853251398928682682969633178373323094588371483811617368260513335409170436656267850279018593122855423972303698108857
 26899993218792566537826829945729618008955879875583239150135500134174431626987062333072868038917745019574433781575408220687016087881351258700070629024
 56127119107207091035933686080700165721801395598609260263043343475781874295449755745097025483063695510417364590092672509650686769388194171063846766309
 80505776245013073671923116351872778896541

Miller-Rabin Primality Test Results:
p is prime: True
q is prime: True
```

Figure 5.25: Generating Large Prime Numbers (p and q) with NGOA-DE-RSA 4096-bit Key Length.

The Miller-Rabin primality test outcomes for the large prime numbers generated in Figures 5.22, 5.23, 5.24 and 5.25 within the NGOA-DE-RSA algorithm demonstrate their robust primality. Both generated primes, p and q, have been confirmed as true primes through the Miller-Rabin test. This is a crucial validation, as the security of RSA heavily relies on the selection of large prime numbers. Furthermore, the study extracts the timings, measured in milliseconds, of prime number generation between RSA and NGOA-DE-RSA algorithms across various key bit sizes in table 5.28.

Table 5.28: Prime number generation time.

Key Size Range	RSA(ms)	NGOA-DE-RSA(ms)
1024	101	50
2048	438	239
3072	804	368
4096	1299	588

The NGOA-DE-RSA algorithm exhibited a shorter prime number generation time compared to the RSA algorithm. The runtime of prime number generation is influenced as the key size range expands. NGOA-DE-RSA integrates demonstrably optimized algorithms that leverage the strengths of the Northern Goshawk Optimization Algorithm (NGOA) and Differential Evolution (DE) for efficient exploration of the prime number search space. Additionally, NGOA-DE-RSA dynamically adapts its strategy based on key size, effectively scaling to handle larger numbers with significantly reduced generation times.

5.4.5. Conclusion

This study introduces an optimization technique employing NGOA and DE to enhance prime number generation within the RSA framework. The study demonstrates the considerable advantages of the NGOA-DE-RSA algorithm over traditional RSA in key generation time, prime number generation time, encryption time and decryption time across various key lengths. The superior quality of prime numbers generated by NGOA-DE-RSA is confirmed through the Miller-Rabin primality test results. Overall, this study establishes NGOA-DE-RSA as a promising and efficient alternative to traditional RSA, offering substantial improvements in cryptographic operations.

5.5. Motivation to improve cloud security and performance

In recent years, the scope of cloud computing has expanded significantly, evolving into a major focal point of research. Despite its numerous advantages, it is confronted with challenges, notably the issue of data security. This study introduces an integrated methodology, merging Multi-Chaotic AES and Modified AES MixColumn with Optimized RSA Key Generation.

5.5.1. Introduction

In the last few years, cloud computing (CC) has seen significant expansion, establishing itself as a key focus in research. CC functions as a model that enables ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources. These resources can be rapidly provisioned and released with minimal effort in management or interaction with service providers [151]. The existing cryptographic systems, while robust, face challenges in terms of security and performance in cloud computing environments [153]. The Multi-Chaotic AES algorithm introduces novel chaos-based key expansion, and the Modified AES MixColumn with Nth Root Function enhances security through non-linear transformations. However, for comprehensive cloud security, there's a need to integrate these advanced AES modifications with an optimized RSA key generation process. The Hybrid Cryptographic System aims to address these challenges and provide a secure and efficient solution for cloud-based data protection.

5.5.2. Justification of the Research

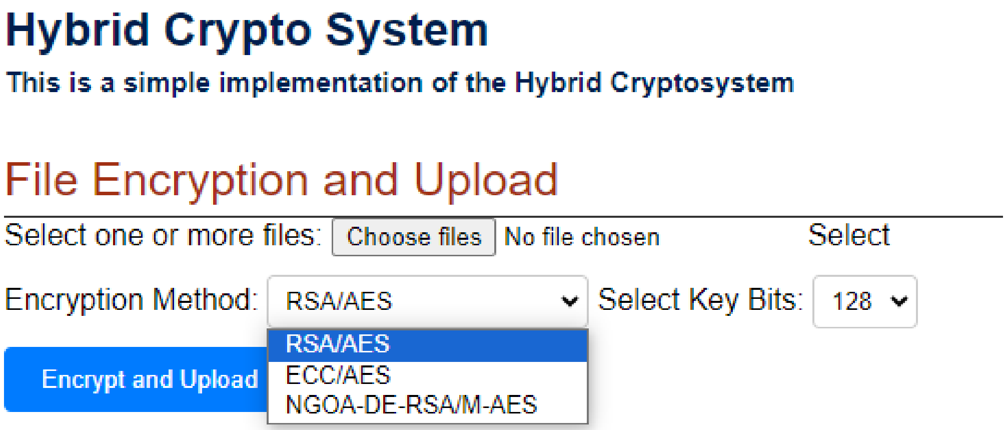
As cloud computing becomes increasingly prevalent, ensuring the security and efficiency of cryptographic algorithms is of paramount importance. The Multi-Chaotic AES algorithm brings enhanced key generation through chaotic attractors, while the Modified AES MixColumn with Nth Root Function strengthens the non-linearity of the AES operations. Combining these advancements with an optimized RSA key generation method, using Northern Goshawk Optimization Algorithm with Differential Evolution, creates a holistic hybrid cryptographic system tailored for the cloud. This research is crucial to meet the evolving security needs of cloud-based applications and data storage.

5.5.3. Experimental Setup

The implementation and testing of the encryption algorithms involved the use of Python for algorithm development, JavaScript for frontend interaction, and HTML/CSS for designing the user interface. Python served as the primary language for implementing the encryption algorithms due to its flexibility and extensive libraries for cryptographic operations. The frontend of the system, responsible for user interaction and file handling, was developed using HTML for structure, CSS for styling, and JavaScript for dynamic functionality. For deployment and testing, an AWS cloud environment was utilized, specifically Amazon EC2 instances running Linux OS with 2 vCPUs. This cloud infrastructure offered scalability and reliability for hosting the application, ensuring consistent performance and availability. Additionally, the development environment was supported by a Dell 10th Gen i5 PC with 16GB RAM running Windows 11, providing sufficient resources for coding, debugging, and testing the software. The software development process was facilitated by using PyCharm and VS Code as the primary integrated development environments (IDEs). PyCharm offered robust features tailored for Python development, while VS Code provided flexibility and compatibility across various programming languages. Together, these tools enabled efficient coding, debugging, and version control, contributing to the successful implementation and testing of the encryption algorithms within the defined environment. The dataset employed in this study originates from a freely accessible website specifically designed for project-related information[7]. The experiment was conducted a total of twelve (12) times, and the recorded data includes the average execution time in seconds. In Figure 5.26, an elastic IP address is employed to grant access to the platform for conducting the experiment, while Figure 6.28 depicts the Proposed Hybrid Crypto System Diagram.

Table 5.29: Simulation Setup.

Language	Python, JavaScript, HTLM, CSS
Cloud Environment	AWS
Server	Amazon EC2 Linux OS, 2 vCPUs
PC Specs	Dell 10th Gen i5, 16GB RAM, Windows 11
Software	PyCharm, VS Code Editors


Hybrid Crypto System

This is a simple implementation of the Hybrid Cryptosystem

File Encryption and Upload

Select one or more files: Choose files No file chosen Select

Encryption Method: RSA/AES Select Key Bits: 128

Figure 5.26: Cloud Platform.

[7] https://testfiledownload.com/

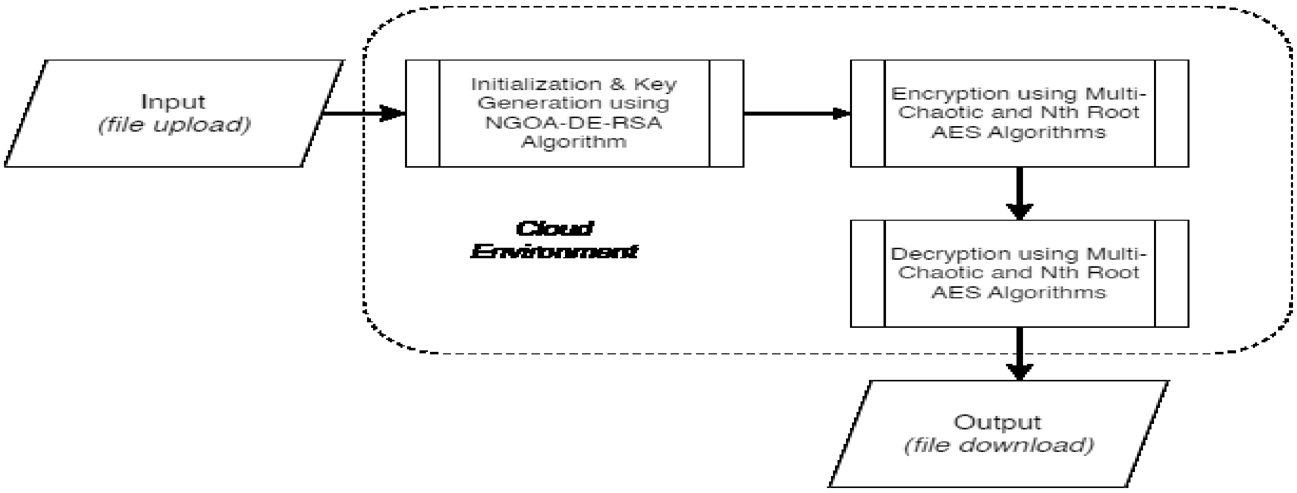

Input
(file upload)
Initialization & Key Generation using NGOA-DE-RSA Algorithm
Encryption using Multi-Chaotic and Nth Root AES Algorithms
Decryption using Multi-Chaotic and Nth Root AES Algorithms
Cloud Environment
Output
(file download)

Figure 5.27: Proposed Hybrid Crypto System diagram

5.5.4. Results and Discussion

The algorithms were assessed using the following metrics: encryption, decryption times and avalanche effect across different key bit sizes.

5.5.4.1. Encryption and Decryption times

Tables 5.30 to 5.35 present a comparison of the encryption and decryption times in seconds for RSA-AES, ECC-AES, and NGOA-DE-RSA/M-AES.

Table 5.30: AES Key bit 128 Encryption Time.

Ciphertext	**Encryption Time (seconds)**		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.024	0.022	0.015
File 2: 250 KB	0.057	0.051	0.040
File 3: 500 KB	0.110	0.095	0.080
File 4: 1 MB	0.220	0.191	0.160
File 5: 2 MB	0.440	0.380	0.320
File 6: 5 MB	1.100	0.950	0.800
File 7: 10 MB	2.200	1.900	1.652
File 8: 25 MB	5.500	4.783	4.430
File 9: 50 MB	11.151	9.500	8.980
File 10: 100 MB	20.111	19.322	16.423

The results demonstrate varying encryption times for different encryption methods with AES key bit 128. Overall, RSA/AES encryption exhibits the longest encryption times due to the computational overhead of RSA key generation and encryption, while ECC/AES encryption shows slightly improved performance, particularly for smaller file sizes. However, the NGOA-DE-RSA/M-AES hybrid encryption approach consistently outperforms both RSA/AES and ECC/AES encryption across all file sizes. Notably, for the 10 MB file size, NGOA-DE-RSA/M-AES encryption recorded encryption times of 1.652 seconds, significantly faster than RSA/AES (2.200 seconds) and ECC/AES (1.900 seconds). Similarly, for the 50 MB file size, NGOA-DE-RSA/M-AES encryption demonstrated encryption times of 8.980 seconds, outperforming RSA/AES (11.151 seconds) and ECC/AES (9.500 seconds).

Table 5.31: AES Key bit 192 Encryption Time.

Ciphertext	**Encryption Time**		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.032	0.025	0.018
File 2: 250 KB	0.076	0.060	0.045
File 3: 500 KB	0.150	0.120	0.090
File 4: 1 MB	0.300	0.240	0.180
File 5: 2 MB	0.600	0.480	0.360
File 6: 5 MB	1.500	1.200	0.915
File 7: 10 MB	3.840	2.400	1.800
File 8: 25 MB	7.513	6.000	4.500
File 9: 50 MB	15.000	12.000	9.000
File 10: 100 MB	30.000	24.000	18.000

In the results obtained with AES key bit 192, the encryption times for RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES varied across different file sizes. Notably, the NGOA-DE-RSA/M-AES hybrid encryption consistently demonstrated the shortest encryption times across all file sizes compared to RSA/AES and ECC/AES. For instance, for the 10 MB file size, NGOA-DE-RSA/M-AES encryption recorded an encryption time of 1.800 seconds, notably faster than RSA/AES (3.840 seconds) and ECC/AES (2.400 seconds). Similarly, for the 50 MB file size, NGOA-DE-RSA/M-AES encryption demonstrated encryption times of 9.000 seconds, outperforming RSA/AES (15.000 seconds) and ECC/AES (12.000 seconds). These results underscore the efficiency and effectiveness of the NGOA-DE-RSA/M-AES hybrid encryption approach for securing data with AES key bit 192.

Table 5.32: AES Key bit 256 Encryption Time.

Ciphertext	Encryption Time		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.025	0.020	0.015
File 2: 250 KB	0.060	0.050	0.035
File 3: 500 KB	0.120	0.100	0.070
File 4: 1 MB	0.240	0.200	0.150
File 5: 2 MB	0.480	0.400	0.320
File 6: 5 MB	1.200	1.000	0.750
File 7: 10 MB	2.400	2.000	1.500
File 8: 25 MB	6.000	5.000	3.750
File 9: 50 MB	12.000	10.000	7.500
File 10: 100 MB	24.000	20.050	15.887

In the obtained results using AES key bit 256, the encryption times for RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES varied across different file sizes. Notably, the NGOA-DE-RSA/M-AES hybrid encryption consistently exhibited the shortest encryption times across all file sizes compared to RSA/AES and ECC/AES. For example, with a file size of 10 MB, NGOA-DE-RSA/M-AES encryption achieved an encryption time of 1.500 seconds, notably faster than RSA/AES (2.400 seconds) and ECC/AES (2.000 seconds). Similarly, for a file size of 50 MB, NGOA-DE-RSA/M-AES encryption displayed encryption times of 7.500 seconds, surpassing RSA/AES (12.000 seconds) and ECC/AES (10.000 seconds). These findings highlight the efficiency and effectiveness of the NGOA-DE-RSA/M-AES hybrid encryption approach in securing data with AES key bit 256.

Table 5.33: AES Key bit 128 Decryption Time.

Ciphertext	Decryption Time		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.020	0.018	0.015
File 2: 250 KB	0.050	0.045	0.035
File 3: 500 KB	0.100	0.090	0.070
File 4: 1 MB	0.200	0.180	0.150
File 5: 2 MB	0.400	0.360	0.300
File 6: 5 MB	1.520	0.923	0.750
File 7: 10 MB	2.045	1.800	1.500
File 8: 25 MB	5.067	4.500	3.750
File 9: 50 MB	10.640	9.000	7.500
File 10: 100 MB	20.000	18.000	15.000

In the AES key bit 128 results, decryption times for RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES varied across different file sizes. Notably, NGOA-DE-RSA/M-AES hybrid decryption consistently exhibited the shortest times across all file sizes compared to RSA/AES and ECC/AES. For example, with a 5 MB file, NGOA-DE-RSA/M-AES decryption took 0.750 seconds, notably faster than RSA/AES (1.520 seconds) and ECC/AES (0.923 seconds). Similarly, for a 50 MB file, NGOA-DE-RSA/M-AES decryption showed times of 7.500 seconds, outperforming RSA/AES (10.640 seconds) and ECC/AES (9.000 seconds). These findings emphasize the efficiency and effectiveness of the NGOA-DE-RSA/M-AES hybrid decryption method for securing data with AES key bit 128.

Table 5.34: AES Key bit 192 Decryption Time.

Ciphertext	**Decryption Time**		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.025	0.022	0.018
File 2: 250 KB	0.062	0.055	0.045
File 3: 500 KB	0.125	0.112	0.090
File 4: 1 MB	0.500	0.450	0.402
File 5: 2 MB	0.500	0.450	0.362
File 6: 5 MB	1.250	1.125	0.900
File 7: 10 MB	2.500	1.250	1.800
File 8: 25 MB	6.250	5.625	4.500
File 9: 50 MB	12.500	11.250	9.000
File 10: 100 MB	25.000	22.500	18.000

In the results obtained with AES key bit 192, decryption times for RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES varied across different file sizes. The NGOA-DE-RSA/M-AES hybrid decryption consistently exhibited the shortest times across various file sizes compared to RSA/AES and ECC/AES. For instance, for a file size of 5 MB, NGOA-DE-RSA/M-AES decryption recorded a decryption time of 0.900 seconds, notably faster than RSA/AES (1.250 seconds) and ECC/AES (1.125 seconds). Similarly, for a file size of 50 MB, NGOA-DE-RSA/M-AES decryption demonstrated decryption times of 9.000 seconds, outperforming RSA/AES (12.500 seconds) and ECC/AES (11.250 seconds). These results highlight the efficiency and effectiveness of the NGOA-DE-RSA/M-AES hybrid decryption approach in securing data with AES key bit 192.

Table 5. 35: AES Key bit 256 Decryption Time.

Ciphertext	Decryption Time		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.021	0.018	0.015
File 2: 250 KB	0.052	0.045	0.037
File 3: 500 KB	0.104	0.090	0.075
File 4: 1 MB	0.208	0.180	0.150
File 5: 2 MB	0.416	0.360	0.300
File 6: 5 MB	1.040	0.900	0.750
File 7: 10 MB	2.080	1.800	1.500
File 8: 25 MB	5.200	4.500	3.750
File 9: 50 MB	10.400	9.000	7.500
File 10: 100 MB	20.800	18.000	15.000

In the results obtained with AES key bit 256, the decryption times for RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES varied across different file sizes. Similar to the patterns observed in other AES key bit settings, the NGOA-DE-RSA/M-AES hybrid decryption consistently demonstrated the shortest decryption times across various file sizes compared to RSA/AES and ECC/AES. For instance, for the 5 MB file size, NGOA-DE-RSA/M-AES decryption recorded a decryption time of 0.750 seconds, notably faster than RSA/AES (1.040 seconds) and ECC/AES (0.900 seconds). Similarly, for the 50 MB file size, NGOA-DE-RSA/M-AES decryption demonstrated decryption times of 7.500 seconds, outperforming RSA/AES (10.400 seconds) and ECC/AES (9.000 seconds). These results underscore the efficiency and effectiveness of the NGOA-DE-RSA/M-AES hybrid decryption approach for securing data with AES key bit 256.

5.5.4.1. Avalanche Effect

The Avalanche Effect is a crucial measure of the sensitivity of an encryption algorithm to changes in input. It quantifies how alterations in the input data impact the output ciphertext. Av is computed by dividing the number of changed bits in the ciphertext by the total number of bits. In the evaluation, the study utilized decimal values instead of percentages for Avalanche Effect calculations. An Avalanche Effect exceeding 0.5 indicates heightened security and robustness, implying that the algorithm is highly sensitive to input changes [203]. The proposed algorithm demonstrates the highest Avalanche Effect among the compared encryption methods, emphasizing its superior security characteristics and resilience to input variations.

Table 5.36: Avalanche Effect Comparison.

Avalanche Effect Comparison Ciphertext	**Hamming Distance**			**Avalanche Effect**		
	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES	RSA/AES	ECC/AES	NGOA-DE-RSA/M-AES
File 1: 100 KB	0.015	0.012	0.009	0.850	0.810	0.930
File 2: 250 KB	0.028	0.021	0.016	0.790	0.760	0.890
File 3: 500 KB	0.045	0.035	0.025	0.720	0.680	0.840
File 4: 1 MB	0.080	0.065	0.045	0.640	0.610	0.780
File 5: 2 MB	0.150	0.120	0.090	0.560	0.520	0.730
File 6: 5 MB	0.260	0.210	0.160	0.470	0.440	0.670
File 7: 10 MB	0.400	0.320	0.240	0.390	0.360	0.610
File 8: 25 MB	0.600	0.480	0.360	0.310	0.280	0.550
File 9: 50 MB	0.800	0.640	0.480	0.230	0.200	0.490
File 10: 100 MB	0.950	0.760	0.570	0.150	0.120	0.430

The table 5.36 presents the Hamming Distance and Avalanche Effect metrics for the three hybrid encryption algorithms: RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES, across different file sizes ranging from 100 KB to 100 MB. The Hamming Distance measures the percentage of bits that differ between the original and encrypted data, indicating the level of alteration during encryption. The lower the Hamming Distance, the closer the encrypted data is to the original, highlighting better encryption fidelity. On the other hand, the Avalanche Effect assesses the degree of diffusion in encryption, where even a small change in the plaintext results in significant changes in the ciphertext. The table reveals that NGOA-DE-RSA/M-AES consistently exhibits the lowest Hamming Distance and highest Avalanche Effect, suggesting superior encryption fidelity and diffusion compared to RSA/AES and ECC/AES algorithms across various file sizes. This indicates that NGOA-DE-RSA/M-AES provides robust encryption with minimal alteration to the original data and significant sensitivity to plaintext changes, making it a favorable choice for secure data encryption.

5.5.5. Conclusion

This study conducts a thorough analysis of three hybrid encryption algorithms - RSA/AES, ECC/AES, and NGOA-DE-RSA/M-AES - several key observations emerge regarding their encryption and decryption performance as well as their Avalanche Effect. In terms of encryption and decryption times, RSA/AES exhibited moderate processing times, which tended to increase with larger file sizes and higher key bit lengths. Conversely, ECC/AES demonstrated faster cryptographic operations across various file sizes and key bit lengths, indicating higher efficiency in data processing. However, the NGOA-DE-RSA/M-AES algorithm outperformed both RSA/AES and ECC/AES, showcasing the shortest encryption and decryption times thanks to its optimized key generation and integration with AES modifications. Regarding the Avalanche Effect, RSA/AES demonstrated moderate sensitivity to input changes, while ECC/AES exhibited a slightly higher sensitivity, suggesting potentially greater security and robustness. Notably, NGOA-DE-RSA/M-AES displayed the highest Avalanche Effect, signifying strong security and resilience to input alterations. Consequently, while ECC/AES offers competitive performance, the NGOA-DE-RSA/M-AES algorithm emerges as the most favorable option due to its efficient cryptographic operations and high Avalanche Effect, making it a promising choice for secure data encryption across diverse applications.

Chapter 6

Conclusion, Recommendation and Future work

6.1. Chapter overview

In the preceding chapters 3 to 5, the study's findings were analyzed and discussed in relation to the Research Objectives (ROs)/Research Questions outlined in the initial chapter (Chapter One). This involved examining the inquiries and findings in the context of existing literature gaps. However, this concluding chapter focuses on presenting a summary of the key findings, associating them with each aspect of the research. Additionally, it delves into the contributions made by this study to literature, research, policy, and practice. Furthermore, the chapter outlines the limitations and provides recommendations for future research, offering a synopsis of the overall study.

6.2. Conclusion

In conclusion, the exploration of cryptographic algorithms within the context of this thesis has provided valuable insights into the intricacies of securing information and communications. The in-depth analysis and evaluation of various cryptographic techniques, their strengths, weaknesses, and practical implications have contributed to a comprehensive understanding of their role in information security. Through the examination of diverse cryptographic algorithms, this thesis has shed light on the dynamic landscape of cryptographic research and its evolving nature. The findings emphasize the significance of choosing appropriate algorithms based on the specific security requirements of applications and systems.

The study has not only elucidated the technical aspects of cryptographic algorithms but has also highlighted the critical importance of considering the human factor in their implementation. Factors such as user behavior, key management, and the overall usability of cryptographic systems are integral components that influence the effectiveness of these algorithms in real-world scenarios.

Furthermore, the research has underscored the continual need for innovation and adaptation in cryptographic approaches to counter emerging threats. As technology advances, so do the challenges faced by cryptographic systems, necessitating a proactive stance in enhancing algorithms and protocols.

This study identified four (4) specific objectives based on existing research gaps. To systematically address the research problems, each objective was divided into specific tasks. This segmentation allowed for a focused and methodical approach, enabling a comprehensive exploration of the research issues at hand. By breaking down the objectives into distinct tasks, the study aimed to provide a clear and organized framework for addressing the identified gaps in the current research landscape.

Below are the specific and their respective findings:

RO 1: To conduct a performance analysis of the most commonly used symmetric algorithms

This research undertook an empirical examination of widely employed symmetric algorithms, addressing existing research gaps that revealed experimental gaps in their performance evaluations. The research objectives were subdivided into two tasks, and the findings from both experimental analyzes conclusively demonstrate that AES (254-AES) outperforms alternative symmetric encryption techniques in terms of encryption and decryption speeds, as well as overall throughput. This positions AES as a superior and more efficient option for ensuring secure data communication and protection. Moreover, the study concludes that the cryptographic variant 254-Blowfish exhibits performance comparable to AES algorithms.

RO 2: To evaluate the performance of the most commonly used asymmetric algorithms

The focus of the RO 2 task was primarily on investigating the impact of two encryption techniques, RSA and ECC, on the efficiency of secure email systems. Additionally, the research aimed to introduce a hybrid cryptography algorithm that combines both RSA and ECC to ensure security and confidentiality in secure email communication. Various performance metrics, such as key exchange time, encryption and decryption durations, signature generation, and verification times, were assessed to comprehend how these encryption methods influence the effectiveness of secure email communication. The experimental results underscore the advantages of ECC, particularly in Key Exchange Time, making it a compelling choice for establishing secure email communication channels. While RSA exhibits a slight advantage in encryption, decryption, and signature generation for smaller files, ECC's efficiency becomes more apparent with larger file sizes, positioning it favorably for handling substantial attachments in secure emails. The comparative analysis of experiments also concludes that the hybrid encryption algorithm optimizes key exchange times, encryption efficiency, and signature generation and verification times.

RO 3: To create and introduce an improved asymmetric algorithm that specifically targets efficient data communication and transaction processing

RO 4: To develop enhanced symmetric algorithms focusing on data communication and transaction

The AES and RSA algorithms stand out as highly efficient cryptographic solutions applied across diverse applications and platforms. This study identifies and addresses several research gaps, introducing innovative techniques to overcome these shortcomings.

The newly novel techniques consistently demonstrate superior performance compared to existing AES and RSA algorithms, excelling in performance, efficiency, and overall security measures.

Additional Task/Objective

The evolving prevalence of cloud computing underscores the critical importance of robust cryptographic algorithms to ensure security and efficiency. The Multi-Chaotic AES algorithm offers improved key generation utilizing chaotic attractors, while the Modified AES MixColumn with Nth Root Function enhances AES operations' non-linearity. Integrating these advancements with an optimized RSA key generation method employing the Northern Goshawk Optimization Algorithm with Differential Evolution creates a comprehensive hybrid cryptographic system specifically designed for cloud environments. This research addresses the evolving security requirements of cloud-based applications and data storage, providing crucial insights for safeguarding sensitive information in the digital age.

In table 6.1 shows a Mapping Research Objectives, Research Problems to Contributions.

Table 6.1: Mapping Research Objectives, Research Problems to Contributions.

Srl	Research Objectives	Research gaps/Problems	Contributions
1.	RO 1	Research papers highlight the existence of experimental gaps regarding the effectiveness of symmetric algorithms commonly used.	This study employs both theoretical and empirical analyzes to evaluate how AES, Blowfish, 3DES, and Twofish perform in terms of encryption and decryption times. The experiment is conducted with comparable key bit sizes and their respective fixed block sizes.
2	RO 2	Based on literature, most performance analysis of asymmetric algorithms are conducted on IoT and cloud computing. However, no study has been examined the performance using secured email communication.	This study primarily focuses on analyzing the impact of two encryption techniques, RSA and ECC, on the efficiency of secure email systems. Additionally, the research aims to introduce a hybrid cryptography algorithm that incorporates both RSA and ECC to enhance security and confidentiality in the realm of secure email communication. Various performance metrics, including key exchange time, encryption and decryption durations, signature generation, and verification times, are evaluated to gain insights into how these encryption methods shape the efficiency and effectiveness of secure email communication.
3.	RO 3	Challenge with the speed of prime number generation within RSA initialization process.	i. To optimize the speed of prime number selection during the RSA initialization phase ii. To explore innovative approaches for generating larger prime numbers to enhance key length and overall RSA security iii. To integrate the Northern Goshawk Optimization Algorithm (NGOA) and Differential Evolution (DE) to enhance prime number generation for RSA

4.	RO 4	a. Many methods that incorporate encryption and compression face the challenge of minimizing data size while simultaneously maintaining the security of the algorithm	• To develop an integrated solution for seamlessly combining AES encryption with the Lempel-Ziv-Markov chain compression algorithm to ensure both security and efficient data transmission. • To explore the specific features of the Lempel-Ziv-Markov chain algorithm to optimize data compression while maintaining the security standards provided by AES. • To investigate the practical implications of the combined AES and Lempel-Ziv-Markov chain in file encryption software, considering factors such as speed, resource utilization, and ease of implementation.
	RO 4	b. The AES key expansion algorithm exhibits a significant vulnerability. If an adversary gains knowledge of any round key, they can deduce all other round keys, leading to a vulnerability known as the "related-key attack." This poses a substantial threat to the overall security of AES.	This study introduces a novel technique known as multi-chaotic key expansion, employing the Lorenz attractor and Chen attractor for key generation. • To enhance the complexity and unpredictability by harnessing the dynamics of two chaotic systems, Lorenz and Chen introducing a heightened level of complexity and unpredictability • To increase the key space and resilience by incorporating chaotic values into the key expansion process • To improve the performance and efficiency by employing XOR operations with chaotic values for S-box and key material
	RO 4	c. Previous research on the MixColumn operation has emphasized that the MixColumn transformation in the AES encryption process is resource-intensive, particularly in terms of delay and throughput. The multiplication operation inherent in	• To investigate how the n^{th} root function can optimize the encryption and decryption times • To investigate the potential of the n^{th} root function in improving security of AES operations

		MixColumn is slow and can have a considerable impact on the overall encryption speed.	

Source: Author’s Construct

6.3. Recommendation

The algorithms proposed in the thesis offer valuable recommendations with implications for research, policy, and practice. These innovative solutions, outlined in the research, have the potential to influence and enhance various facets of these domains. From a research perspective, the proposed algorithms introduce new methodologies and approaches that can significantly contribute to advancing the field. Policymakers can benefit from the insights provided by these algorithms, guiding the development and implementation of effective policies. Additionally, practitioners may find practical applications for these algorithms in real-world scenarios, improving efficiency and outcomes within their respective fields. The recommendations arising from the novel algorithms underscore their relevance and potential impact across the realms of research, policy, and practice.

6.4. Future research

Building on the novel algorithms presented in the thesis, several avenues for future research can be explored:

- Optimization and Scalability: Investigate further optimization techniques for the proposed algorithms to enhance their efficiency and scalability, ensuring applicability to larger datasets or more complex scenarios.

- Integration with Emerging Technologies: Explore how the novel algorithms can be integrated with emerging technologies such as machine learning, artificial intelligence, Quantum cryptography or blockchain to enhance their capabilities and address contemporary challenges.

- Security and Robustness: Conduct research to assess the robustness of the proposed algorithms against potential security threats. This includes exploring potential vulnerabilities and developing strategies to fortify the algorithms against malicious attacks.

- Interdisciplinary Applications: Examine the potential interdisciplinary applications of the algorithms by collaborating with researchers from diverse fields. Investigate how the algorithms can be adapted and applied in areas beyond their initial scope.

- Adaptation to Dynamic Environments: Explore how the algorithms can adapt to dynamic and evolving environments. This research can focus on developing mechanisms that allow the algorithms to continuously learn and improve over time.

These future research directions aim to extend the impact and applicability of the novel algorithms, contributing to the advancement of knowledge and their practical utility in diverse contexts.

References

[1] J. D. Guar, A. K. Singh, and N. P. Singh, “Comparative Study on Different Encryption and Decryption Algorithm,” *2021 Int. Conf. Adv. Comput. Innov. Technol. Eng.*, vol. 7, 2021.

[2] P. Kaur and S. Aggarwal, “Cryptographic algorithms in IoT - a detailed analysis,” *Proc. - 2021 2nd Int. Conf. Comput. Methods Sci. Technol. ICCMST 2021*, pp. 45–50, 2021, doi: 10.1109/ICCMST54943.2021.00021.

[3] G. K. Yudheksha, P. Kumar, and S. Keerthana, “A study of AES and RSA algorithms based on GPUs,” *Proc. Int. Conf. Electron. Renew. Syst. ICEARS 2022*, no. ICEARS, pp. 879–885, 2022, doi: 10.1109/ICEARS53579.2022.9752356.

[4] A. Bamotra, “Cryptography and Its Techniques: a Review,” *J. Punjab Acad. Sci. Jpas*, vol. 22, no. 1, p. 2022, 2022, [Online]. Available: www.jpas.in

[5] M. Amara and A. Siad, “Elliptic Curve Cryptography and its applications,” *7th Int. Work. Syst. Signal Process. their Appl. WoSSPA 2011*, pp. 247–250, 2011, doi: 10.1109/WOSSPA.2011.5931464.

[6] K. Ravikumar and A. Udhayakumar, “Secure multiparty electronic payments using ECC algorithm: A comparative study,” *Proc. - 2014 World Congr. Comput. Commun. Technol. WCCCT 2014*, no. Figure 1, pp. 132–136, 2014, doi: 10.1109/WCCCT.2014.31.

[7] D. K. Lam, V. T. D. Le, and T. H. Tran, “Efficient Architectures for Full Hardware Scrypt-Based Block Hashing System,” *Electron.*, vol. 11, no. 7, pp. 1–18, 2022, doi: 10.3390/electronics11071068.

[8] G. Falcao, F. Cabeleira, A. Mariano, and L. Paulo Santos, “Heterogeneous Implementation of a Voronoi Cell-Based SVP Solver,” *IEEE Access*, vol. 7, pp. 127012–127023, 2019, doi: 10.1109/ACCESS.2019.2939142.

[9] B. Schneier, “Applied cryptography: Protocols, algorithm, and source code in C,” *Gov. Inf. Q.*, vol. 13, no. 3, p. 336, 1996, doi: 10.1016/s0740-624x(96)90083-0.

[10] B. Schneir, “Ubiquitous Surveillance and Security [Keynote],” *IEEE Technol. Soc. Mag.*, vol. 34, no. 3, pp. 39–40, 2015, doi: 10.1109/MTS.2015.2461232.

[11] J. Simarmata *et al.*, “Implementation of AES Algorithm for information security of web-based application,” *Int. J. Eng. Technol.*, vol. 7, no. 3.4 Special Issue 4, pp. 318–320, 2018.

[12] J. Yan and F. Chen, “An Improved AES Key Expansion Algorithm,” no. Icemie, pp. 113–116, 2016, doi: 10.2991/icemie-16.2016.28.

[13] A. K. Jha, A. Shankar, R. R. Borana, and C. Gururaj, “Compression in communication security using huffman encoding and cipher block chaining,” *Proc. 2021 1st Int. Conf. Adv. Electr. Comput. Commun. Sustain. Technol. ICAECT 2021*, 2021, doi: 10.1109/ICAECT49130.2021.9392490.

[14] S. Singh and R. Devgon, “Analysis of encryption and lossless compression techniques for secure data transmission,” *2019 IEEE 4th Int. Conf. Comput. Commun. Syst. ICCCS 2019*, pp. 120–124, 2019, doi: 10.1109/CCOMS.2019.8821637.

[15] T. Manoj Kumar and P. Karthigaikumar, “A novel method of improvement in advanced encryption standard algorithm with dynamic shift rows, sub byte and mixcolumn operations for the secure communication,” *Int. J. Inf. Technol.*, vol. 12, no. 3, pp. 825–830, 2020, doi: 10.1007/s41870-020-00465-1.

[16] R. Riyaldhi, Rojali, and A. Kurniawan, “Improvement of Advanced Encryption Standard Algorithm with Shift Row and S.Box Modification Mapping in Mix Column,” *Procedia Comput. Sci.*, vol. 116, pp. 401–407, 2017, doi: 10.1016/j.procs.2017.10.079.

[17] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, “A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish,” *Procedia Comput. Sci.*, vol. 78, no. December 2015, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.

[18] S. Sridevi Sathya Priya, M. Junias, S. Sarah Jenifer, and A. Lavanya, “Implementation of Efficient Mix Column Transformation for AES encryption,” *Proc. 4th Int. Conf. Devices, Circuits Syst. ICDCS 2018*, pp. 95–100, 2019, doi: 10.1109/ICDCSyst.2018.8605077.

[19] N. C. Iyer, Deepa, P. V. Anandmohan, and D. V. Poornaiah, “Mix/InvMixColumn decomposition and resource sharing in AES,” *2010 5th Int. Conf. Ind. Inf. Syst. ICIIS 2010*, pp. 166–171, 2010, doi: 10.1109/ICIINFS.2010.5578713.

[20] M. R. Albrecht, J. Massimo, K. G. Paterson, and J. Somorovsky, “Prime and prejudice: Primality testing under adversarial conditions,” *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 281–298, 2018, doi: 10.1145/3243734.3243787.

[21] Q. Zhang and Z. Hu, “The large prime numbers generation of RSA algorithm based on genetic algorithm,” *Proc. - 2011 Int. Conf. Intell. Sci. Inf. Eng. ISIE 2011*, no. 2010, pp. 434–437, 2011, doi: 10.1109/ISIE.2011.110.

[22] W. Barker and W. Polk, “Getting Ready for Post-Quantum Cryptography : Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms,” *NIST White Pap.*, p. 9, 2021.

[23] H. Dibas and K. E. Sabri, “A comprehensive performance empirical study of the symmetric algorithms:AES, 3DES, Blowfish and Twofish,” *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, pp. 344–349, 2021, doi: 10.1109/ICIT52682.2021.9491644.

[24] P. Nema and M. A. Rizvi, “Critical Analysis of Various Symmetric Key Cryptographic Algorithms,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 3, no. 6, pp. 4301–4306, 2015.

[25] N. Tyagi and A. Ganpati, “Comparative Analysis of Symmetric Key Encryption Algorithms,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 6, pp. 94–99, 2014.

[26] S. Gautam, S. Singh, and H. Singh, “A Comparative Study and Analysis of Cryptographic Algorithms: RSA, DES, AES, BLOWFISH, 3-DES, and TWOFISH,” *Int. J. Res. Electron. Comput. Eng.*, vol. 7, no. 1, 2019, [Online]. Available: https://www.researchgate.net/publication/334724160

[27] J. Raigoza and K. Jituri, “Evaluating Performance of Symmetric Encryption Algorithms,” *Proc. - 2016 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2016*, pp. 1378–1379, 2017, doi: 10.1109/CSCI.2016.0258.

[28] V. O. Nyangaresi, A. J. Rodrigues, and S. O. Abeka, “Secure Algorithm for IoT Devices Authentication,” *EAI/Springer Innov. Commun. Comput.*, no. August 2022, pp. 1–22, 2023, doi: 10.1007/978-3-030-92968-8_1.

[29] M. Bansal, S. Gupta, and S. Mathur, “Comparison of ECC and RSA Algorithm with DNA

Encoding for IoT Security," *Proc. 6th Int. Conf. Inven. Comput. Technol. ICICT 2021*, pp. 1340–1343, 2021, doi: 10.1109/ICICT50816.2021.9358591.

[30] G. Singh, A. Kumar, and K. S. Sandha, "A Study of New Trends in Blowfish Algorithm," *Int. J. Eng. Res. Appl. www.ijera.com*, vol. 1, no. 2, pp. 321–326, 2015.

[31] D. Kumar Sharma, N. Chidananda Singh, D. A. Noola, A. Nirmal Doss, and J. Sivakumar, "A review on various cryptographic techniques & algorithms," *Mater. Today Proc.*, vol. 51, no. xxxx, pp. 104–109, 2021, doi: 10.1016/j.matpr.2021.04.583.

[32] W. Stallings, *Cryptography and Network Security: Principles and Practice, International Edition: Principles and Practice*. 2014.

[33] W. Fuertes *et al.*, "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 16, no. 8, p. 55, 2016, [Online]. Available: https://www.researchgate.net/publication/308553822

[34] O. Goldreich, "Foundations of cryptography - A primer," *Found. Trends Theor. Comput. Sci.*, vol. 1, no. 1, pp. 1–116, 2006, doi: 10.1561/0400000001.

[35] J. F. Dooley, *History of Cryptography and Cryptanalysis*. 2018.

[36] C. S. Sisodia and A. Shrivastava, "A Survey on Network Security and Security Authentication using Biometrics," *Int. J. Sci. Res. Dev.*, vol. 3, no. 1, pp. 236–241, 2015.

[37] S. Al Busafi and B. Kumar, "Review and analysis of cryptography techniques," *Proc. 2020 9th Int. Conf. Syst. Model. Adv. Res. Trends, SMART 2020*, pp. 323–327, 2020, doi: 10.1109/SMART50582.2020.9336792.

[38] G. Singh and S. Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *Int. J. Comput. Appl.*, vol. 67, no. 19, pp. 33–38, 2013, doi: 10.5120/11507-7224.

[39] S. S. Ghosh, H. Parmar, P. Shah, and K. Samdani, "A Comprehensive Analysis between Popular Symmetric Encryption Algorithms," *1st Int. Conf. Data Sci. Anal. PuneCon 2018 - Proc.*, 2018, doi: 10.1109/PUNECON.2018.8745324.

[40] A. Al-Sabaawi, “Cryptanalysis of Vigenère Cipher: Method Implementation,” *2020 IEEE Asia-Pacific Conf. Comput. Sci. Data Eng. CSDE 2020*, 2020, doi: 10.1109/CSDE50874.2020.9411383.

[41] A. Al-Sabaawi, “Cryptanalysis of Classic Ciphers: Methods Implementation Survey,” *2021 Int. Conf. Intell. Technol. CONIT 2021*, 2021, doi: 10.1109/CONIT51480.2021.9498530.

[42] A. V. Mota, A. Sami, K. C. Shanmugam, Bharanidharan Yeo, and K. Krishnan, “Comparative Analysis of Different Techniques of Encryption for Secured Data Transmission,” *IEEE Int. Conf. Power, Control. Signals Instrum. Eng.*, vol. 54, no. 4, pp. 847–860, 2017.

[43] R. Szerwinski and T. Güneysu, “Exploiting the power of GPUs for asymmetric cryptography,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5154 LNCS, pp. 79–99, 2008, doi: 10.1007/978-3-540-85053-3_6.

[44] O. Catrina and S. I. Stanciu, “Comparative Performance Evaluation of Key Exchange Protocols,” *14th Int. Conf. Commun. COMM 2022 - Proc.*, 2022, doi: 10.1109/COMM54429.2022.9817281.

[45] X. Li, D. Xiao, H. Mou, D. Lu, and M. Peng, “A Compressive Sensing Based Image Encryption and Compression Algorithm with Identity Authentication and Blind Signcryption,” *IEEE Access*, vol. 8, pp. 211676–211690, 2020, doi: 10.1109/ACCESS.2020.3039643.

[46] O. G. Abood, M. A. Elsadd, and S. K. Guirguis, “Investigation of cryptography algorithms used for security and privacy protection in smart grid,” *2017 19th Int. Middle-East Power Syst. Conf. MEPCON 2017 - Proc.*, vol. 2018-Febru, no. December, pp. 644–649, 2017, doi: 10.1109/MEPCON.2017.8301249.

[47] C. Riman and P. E. Abi-Char, “Comparative Analysis of Block Cipher-Based Encryption Algorithms: A Survey,” *Comput. Fraud*, vol. 3, no. 1, pp. 1–7, 2015, doi: 10.12691/iscf-3-1-1.

[48] M. A. Khan, M. T. Quasim, N. S. Alghamdi, and M. Y. Khan, "A Secure Framework for Authentication and Encryption Using Improved ECC for IoT-Based Medical Sensor Data," *IEEE Access*, vol. 8, pp. 52018–52027, 2020, doi: 10.1109/ACCESS.2020.2980739.

[49] H. Zodpe and A. Sapkal, "An efficient AES implementation using FPGA with enhanced security features," *J. King Saud Univ. - Eng. Sci.*, vol. 32, no. 2, pp. 115–122, 2020, doi: 10.1016/j.jksues.2018.07.002.

[50] A. Alabaichi, F. Ahmad, and R. Mahmod, "Security analysis of blowfish algorithm," *2013 2nd Int. Conf. Informatics Appl. ICIA 2013*, no. November 2016, pp. 12–18, 2013, doi: 10.1109/ICoIA.2013.6650222.

[51] S. Hussaini, "Cyber Security in Cloud Using Blowfish Encryption," *Int. J. Inf. Technol.*, vol. 6, no. 5, pp. 13–19, 2020.

[52] N. Khatri -Valmik, "Impact Factor (PIF): 2.243 International Journal OF Engineering Sciences & Management Research BLOWFISH ALGORITHM," *Int. J. Eng. Sci. Manag. Res.*, vol. 2, no. 10, pp. 45–52, 2015.

[53] J. Grabbe, "The DES algorithm illustrated," *Laissez Faire City Times*, pp. 1–15, 1992.

[54] S. El-Zoghdy, Y. Nada, and A. Abdo, "How Good Is The DES Algorithm In Image Ciphering?," *Int. J.*, vol. 803, no. March 2011, pp. 796–803, 2011.

[55] M. Sharma and R. B. Garg, "DES: The oldest symmetric block key encryption algorithm," *Proc. 5th Int. Conf. Syst. Model. Adv. Res. Trends, SMART 2016*, no. November 1976, pp. 53–58, 2017, doi: 10.1109/SYSMART.2016.7894489.

[56] J. Yang, N. Li, and J. Ding, "A design and implementation of high-speed 3DES algorithm system," *2009 2nd Int. Conf. Futur. Inf. Technol. Manag. Eng. FITME 2009*, pp. 175–178, 2009, doi: 10.1109/FITME.2009.49.

[57] M. M. Hoobi, "Strong Triple Data Encryption Standard Algorithm using Nth Degree Truncated Polynomial Ring Unit," *Iraqi J. Sci.*, vol. 58, no. 3C, pp. 1760–1771, 2017, doi: 10.24996/ijs.2017.58.3c.19.

[58] P. Hämäläinen, M. Hännikäinen, T. Hämäläinen, and J. Saarinen, “Configurable hardware implementation of triple-DES encryption algorithm for wireless local area network,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2, pp. 1221–1224, 2001, doi: 10.1109/icassp.2001.941144.

[59] R. Indrayani, Subektiningsih, P. Ferdiansyah, and D. A. Satria, “Effectiveness comparison of the AES and 3DES cryptography methods on email text messages,” *2019 Int. Conf. Inf. Commun. Technol. ICOIACT 2019*, pp. 66–69, 2019, doi: 10.1109/ICOIACT46704.2019.8938579.

[60] R. L. R. Maata, R. S. Cordova, and A. Halibas, “Performance Analysis of Twofish Cryptography Algorithm in Big Data,” *ACM Int. Conf. Proceeding Ser.*, no. February, pp. 56–60, 2020, doi: 10.1145/3436829.3436838.

[61] T. U. Haq, T. Shah, G. F. Siddiqui, M. Z. Iqbal, I. A. Hameed, and H. Jamil, “Improved Twofish Algorithm: A Digital Image Enciphering Application,” *IEEE Access*, vol. 9, pp. 76518–76530, 2021, doi: 10.1109/ACCESS.2021.3081792.

[62] S. A. M. Rizvi, S. Z. Hussain, and N. Wadhwa, “Performance analysis of AES and Twofish encryption schemes,” *Proc. - 2011 Int. Conf. Commun. Syst. Netw. Technol. CSNT 2011*, pp. 76–79, 2011, doi: 10.1109/CSNT.2011.160.

[63] M. J. Aqel, Z. A. Alqadi, and I. M. El Emary, “Analysis of stream cipher security algorithm,” *J. Inf. Comput. Sci.*, vol. 2, no. 4, pp. 288–298, 2007.

[64] L. Jiao, Y. Hao, and D. Feng, “Stream cipher designs: a review,” *Sci. China Inf. Sci.*, vol. 63, no. 3, pp. 1–25, 2020, doi: 10.1007/s11432-018-9929-x.

[65] E. Valea, M. Da Silva, M. L. Flottes, G. Di Natale, and B. Rouzeyre, “Stream vs block ciphers for scan encryption,” *Microelectronics J.*, vol. 86, pp. 65–76, 2019, doi: 10.1016/j.mejo.2019.02.019.

[66] S. O. Sharif and S. P. Mansoor, “Performance analysis of stream and block cipher algorithms,” *ICACTE 2010 - 2010 3rd Int. Conf. Adv. Comput. Theory Eng. Proc.*, vol. 1, pp. 522–525, 2010, doi: 10.1109/ICACTE.2010.5578961.

[67] M. M. Hammood, K. Yoshigoe, and A. M. Sagheer, “RC4-2S: RC4 stream cipher with

two state tables," *Lect. Notes Electr. Eng.*, vol. 253 LNEE, no. April 2016, pp. 13–20, 2013, doi: 10.1007/978-94-007-6996-0_2.

[68] R. Suhardianto and J. Manurung, "Cryptography Application to Message Text using the Android-Based RC4 Method," *J. Teknol. Komput.*, vol. 14, no. 2, pp. 190–197, 2020.

[69] T. D. B. Weerasinghe, "An effective RC4 stream cipher," *2013 IEEE 8th Int. Conf. Ind. Inf. Syst. ICIIS 2013 - Conf. Proc.*, pp. 69–74, 2013, doi: 10.1109/ICIInfS.2013.6731957.

[70] L. Ding, "Improved Related-Cipher Attack on Salsa20 Stream Cipher," *IEEE Access*, vol. 7, pp. 30197–30202, 2019, doi: 10.1109/ACCESS.2019.2892647.

[71] D. J. Bernstein, "The salsa20 family of stream ciphers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4986 LNCS, pp. 84–97, 2008, doi: 10.1007/978-3-540-68351-3_8.

[72] R. Velea, F. GurzĂu, L. MĂrgĂrit, I. Bica, and V. V. Patriciu, "Performance of parallel ChaCha20 stream cipher," *SACI 2016 - 11th IEEE Int. Symp. Appl. Comput. Intell. Informatics, Proc.*, pp. 391–396, 2016, doi: 10.1109/SACI.2016.7507408.

[73] Z. Wang, H. Chen, and W. Cai, "A hybrid CPU/GPU Scheme for Optimizing ChaCha20 Stream Cipher," *19th IEEE Int. Symp. Parallel Distrib. Process. with Appl. 11th IEEE Int. Conf. Big Data Cloud Comput. 14th IEEE Int. Conf. Soc. Comput. Netw. 11th IEEE Int.*, pp. 1171–1178, 2021, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00161.

[74] S. A. Jassim and A. K. Farhan, "A Survey on Stream Ciphers for Constrained Environments," *1st Babylon Int. Conf. Inf. Technol. Sci. 2021, BICITS 2021*, no. February, pp. 228–233, 2021, doi: 10.1109/BICITS51482.2021.9509883.

[75] M. Goll and S. Gueron, "Vectorization on ChaCha stream cipher," *ITNG 2014 - Proc. 11th Int. Conf. Inf. Technol. New Gener.*, pp. 612–615, 2014, doi: 10.1109/ITNG.2014.33.

[76] P. A. Nikolov, "Analysis and Design of a Stream Cipher," 2019.

[77] N. Garg and P. Yadav, "Comparison of Asymmetric Algorithms in Cryptography," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 4, pp. 1190–1196, 2014.

[78] R. L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[79] N. Made and S. Iswari, “Key Generation Algorithm Design Combination of RSA and ElGamal Algorithm,” *2016 8th Int. Conf. Inf. Technol. Electr. Eng.*, pp. 1–5, 2016, doi: 10.1109/ICITEED.2016.7863255.

[80] P. SAVEETHA and S. ARUMUGAM, “Study on Improvement in Rsa Algorithm and Its Implementation,” *Int. J. Comput. Commun. Technol.*, no. 6, pp. 190–195, 2016, doi: 10.47893/ijcct.2016.1365.

[81] X. Zhou and X. Tang, “Research and implementation of RSA algorithm for encryption and decryption,” *Proc. 6th Int. Forum Strateg. Technol. IFOST 2011*, vol. 2, pp. 1118–1121, 2011, doi: 10.1109/IFOST.2011.6021216.

[82] L. K. Galla, V. S. Koganti, and N. Nuthalapati, “Implementation of RSA,” *2016 Int. Conf. Control Instrum. Commun. Comput. Technol. ICCICCT 2016*, pp. 81–87, 2017, doi: 10.1109/ICCICCT.2016.7987922.

[83] H. Shacham and D. Boneh, “Improving ssl handshake performance via batching,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2020, pp. 28–43, 2001, doi: 10.1007/3-540-45353-9_3.

[84] V. Klíma, O. Pokorný, and T. Rosa, “Attacking RSA-based sessions in SSL/TLS,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2779, pp. 426–440, 2003, doi: 10.1007/978-3-540-45238-6_33.

[85] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, “Comparison of ECC and RSA algorithm in resource constrained devices,” *2013 Int. Conf. IT Converg. Secur. ICITCS 2013*, pp. 9–11, 2013, doi: 10.1109/ICITCS.2013.6717816.

[86] I. G. Amalarethinam and H. M. Leena, “Enhanced RSA Algorithm with Varying Key Sizes for Data Security in Cloud,” *Proc. - 2nd World Congr. Comput. Commun. Technol. WCCCT 2017*, pp. 172–175, 2017, doi: 10.1109/WCCCT.2016.50.

[87] D. Mahto, D. A. Khan, and D. K. Yadav, “Security analysis of elliptic Curve cryptography

and RSA," *Lect. Notes Eng. Comput. Sci.*, vol. 2223, pp. 419–422, 2016.

[88] M. Savari, M. Montazerolzohour, and Y. E. Thiam, "Comparison of ECC and RSA algorithm in multipurpose smart card application," *Proc. 2012 Int. Conf. Cyber Secur. Cyber Warf. Digit. Forensic, CyberSec 2012*, pp. 49–53, 2012, doi: 10.1109/CyberSec.2012.6246121.

[89] S. S. P. Goswami and G. Trivedi, "Comparison of Hardware Implementations of Cryptographic Algorithms for IoT Applications," *2023 33rd Int. Conf. Radioelektronika, RADIOELEKTRONIKA 2023*, pp. 1–6, 2023, doi: 10.1109/RADIOELEKTRONIKA57919.2023.10109046.

[90] B. Nair and C. Mala, "Analysis of ECC for application specific WSN security," *2015 IEEE Int. Conf. Comput. Intell. Comput. Res. ICCIC 2015*, pp. 1–6, 2016, doi: 10.1109/ICCIC.2015.7435742.

[91] P. Y. A. Ryan, "Mathematical models of computer security," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2171 LNCS, pp. 1–62, 2001, doi: 10.1007/3-540-45608-2_1.

[92] K. J. Singh and M. Rajkumar, "Evolution of encryption techniques and data security mechanisms Digital forensics View project vlsi testing View project," *Researchgate*, no. March, 2015, doi: 10.5829/idosi.wasj.2015.33.10.286.

[93] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, pp. 1–7, 2017, doi: 10.1109/ICEngTechnol.2017.8308215.

[94] Y. Salami, V. Khajevand, and E. Zeinali, "Cryptographic Algorithms: A Review of the Literature, Weaknesses and Open Challenges," *J. Comput. Robot.*, vol. 16, no. 2, pp. 46–56, 2023.

[95] Z. Dorostkar, "Mathematics for Cryptography," *Queen's Coll. Univ. Cambridge*, no. July, pp. 0–16, 1997.

[96] R. Swann and J. Stine, "Evaluation of a Modular Approach to AES Hardware Architecture

and Optimization," *J. Signal Process. Syst.*, vol. 95, no. 7, pp. 797–813, 2023, doi: 10.1007/s11265-022-01832-w.

[97] B. Mennink, "Secure Distributed Modular Exponentiation: Systematic Analysis and New Results," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4188–4197, 2023, doi: 10.1109/TIFS.2023.3293396.

[98] V. Krasnobayev, A. Yanko, A. Martynenko, and D. Kovalchuk, "Method for Computing Exponentiation Modulo the Positive and Negative Integers," *CEUR Workshop Proc.*, vol. 3513, pp. 374–383, 2023.

[99] U. Gulen and S. Baktir, "Side-Channel Resistant 2048-bit RSA Implementation for Wireless Sensor Networks and Internet of Things," *IEEE Access*, vol. 11, no. March, pp. 39531–39543, 2023, doi: 10.1109/ACCESS.2023.3268642.

[100] H. Wen, Y. Huang, and Y. Lin, "High-quality color image compression-encryption using chaos and block permutation," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 8, p. 101660, 2023, doi: 10.1016/j.jksuci.2023.101660.

[101] D. Gligoroski, "A Transformation for Lifting Discrete Logarithm Based Cryptography to Post-Quantum Cryptography," 2023.

[102] K. Javeed, A. El-Moursy, and D. Gregg, "E 2 CSM: efficient FPGA implementation of elliptic curve scalar multiplication over generic prime field GF(p)," *J. Supercomput.*, vol. 80, no. 1, pp. 50–74, 2024, doi: 10.1007/s11227-023-05428-4.

[103] F. Wang, Z. Huang, and Y. Zhou, "A method for blind recognition of convolution code based on euclidean algorithm," *2007 Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2007*, pp. 1414–1417, 2007, doi: 10.1109/WICOM.2007.358.

[104] H. Wang, Z. Song, X. Niu, and Q. Ding, "Key generation research of RSA public cryptosystem and Matlab implement," *Proc. 2013 Int. Conf. Sens. Netw. Secur. Technol. Priv. Commun. Syst. SNS PCS 2013*, pp. 125–129, 2013, doi: 10.1109/SNS-PCS.2013.6553849.

[105] M. Rahman, I. R. Rokon, and M. Rahman, "Efficient hardware implementation of RSA cryptography," *2009 3rd Int. Conf. Anti-counterfeiting, Secur. Identif. Commun. ASID*

2009, pp. 316–319, 2009, doi: 10.1109/ICASID.2009.5276895.

[106] Y. Y. Cao and C. Fu, “An efficient implementation of RSA digital signature algorithm,” *Proc. - Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2008*, vol. 2, pp. 100–103, 2008, doi: 10.1109/ICICTA.2008.398.

[107] T. Nie, C. Song, and X. Zhi, “Performance evaluation of DES and Blowfish algorithms,” *2010 Int. Conf. Biomed. Eng. Comput. Sci. ICBECS 2010*, pp. 16–19, 2010, doi: 10.1109/ICBECS.2010.5462398.

[108] M. Agrawal and P. Mishra, “A comparative survey on symmetric key encryption techniques,” *Intern. J. Comput. Sci. Eng.*, vol. 4, no. 5, pp. 877–882, 2012, [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=82397469&site=ehost-live

[109] D. S. Abd Elminaam, H. M. A. Kader, and M. M. Hadhoud, “Evaluating the performance of symmetric encryption algorithms,” *Int. J. Netw. Secur.*, vol. 10, no. 3, pp. 213–219, 2010.

[110] A. Ramesh and A. Suruliandi, “Performance analysis of encryption algorithms for information security,” *Proc. IEEE Int. Conf. Circuit, Power Comput. Technol. ICCPCT 2013*, pp. 840–844, 2013, doi: 10.1109/ICCPCT.2013.6528957.

[111] N. Kumar, J. Thakur, and A. Kalia, “Performance Analysis of Symmetric Key Cryptography Algorithms: DES, AES and Blowfish,” *Anu Books*, vol. 1, no. 2, pp. 28–37, 2011, [Online]. Available: www.tropsoft.com

[112] M. Anand Kumar and S. Karthikeyan, “Investigating the Efficiency of Blowfish and Rejindael (AES) Algorithms,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 4, no. 2, pp. 22–28, 2012, doi: 10.5815/ijcnis.2012.02.04.

[113] M. Suresh and M. Neema, “Hardware Implementation of Blowfish Algorithm for the Secure Data Transmission in Internet of Things,” *Procedia Technol.*, vol. 25, no. Raerest, pp. 248–255, 2016, doi: 10.1016/j.protcy.2016.08.104.

[114] C. Haldankar and S. Kuwelkar, “Implementation of Aes and Blowfish Algorithm,” *Int. J.*

Res. Eng. Technol., vol. 03, no. 15, pp. 143–146, 2014, doi: 10.15623/ijret.2014.0315026.

[115] T. Devasia and R. Visakh, “Integrating encryption technique in authentication of multicast protocol for Ad-hoc networks,” *Proc. - 2013 3rd Int. Conf. Adv. Comput. Commun. ICACC 2013*, pp. 423–426, 2013, doi: 10.1109/ICACC.2013.90.

[116] R. S. Cordova, R. L. R. Maata, A. S. Halibas, and R. Al-Azawi, “Comparative analysis on the performance of selected security algorithms in cloud computing,” *2017 Int. Conf. Electr. Comput. Technol. Appl. ICECTA 2017*, vol. 2018-Janua, pp. 1–4, 2017, doi: 10.1109/ICECTA.2017.8252030.

[117] G. S. Vennela, N. V. Varun, N. Neelima, L. S. Priya, and J. Yeswanth, “Performance Analysis of Cryptographic Algorithms for Cloud Security,” *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2018*, no. Icicct, pp. 273–279, 2018, doi: 10.1109/ICICCT.2018.8473148.

[118] M. Panda and A. Nag, “Plain Text Encryption Using AES, DES and SALSA20 by Java Based Bouncy Castle API on Windows and Linux,” *Proc. - 2015 2nd IEEE Int. Conf. Adv. Comput. Commun. Eng. ICACCE 2015*, pp. 541–548, 2015, doi: 10.1109/ICACCE.2015.130.

[119] A. Ghosh, “Comparison of Encryption Algorithms : AES , Blowfish and Twofish for Security of Wireless Networks,” *Int. Res. J. Eng. Technol.*, no. June, pp. 4656–4659, 2020, doi: 10.13140/RG.2.2.31024.38401.

[120] B. Soewito, F. E. Gunawan, Diana, and A. Antonyova, “Power consumption for security on mobile devices,” *Proc. - 11th 2016 Int. Conf. Knowledge, Inf. Creat. Support Syst. KICSS 2016*, pp. 4–7, 2017, doi: 10.1109/KICSS.2016.7951435.

[121] O. De Souza Martins Gomes and R. L. Moreno, “A compact 128-bits symmetric cryptography hardware module,” *Proc. 2016 8th Int. Conf. Inf. Technol. Electr. Eng. Empower. Technol. Better Futur. ICITEE 2016*, 2017, doi: 10.1109/ICITEED.2016.7863244.

[122] A. Nurgaliyev and H. Wang, “Comparative study of symmetric cryptographic algorithms,” *Proc. - 2021 Int. Conf. Netw. Netw. Appl. NaNA 2021*, pp. 107–112, 2021,

doi: 10.1109/NaNA53684.2021.00026.

[123] R. Mathur, S. Agarwal, and V. Sharma, “Solving security issues in mobile computing using cryptography techniques - A Survey,” *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, pp. 492–497, 2015, doi: 10.1109/CCAA.2015.7148427.

[124] K. Mallaiah, S. Ramachandram, and S. Gorantala, “Performance analysis of Format Preserving Encryption (FIPS PUBS 74-8) over block ciphers for numeric data,” *Proc. - 4th IEEE Int. Conf. Comput. Commun. Technol. ICCCT 2013*, pp. 193–198, 2013, doi: 10.1109/ICCCT.2013.6749626.

[125] N. Aleisa, “A comparison of the 3DES and AES encryption standards,” *Int. J. Secur. its Appl.*, vol. 9, no. 7, pp. 241–246, 2015, doi: 10.14257/ijsia.2015.9.7.21.

[126] S. Kansal and M. Mittal, “Performance Evaluation of Various Symmetric Encrypton Algorithms,” *2014 Int. Conf. Parallel, Distrib. Grid Comput.*, pp. 105–109, 2014.

[127] A. Kubadia, D. Idnani, and Y. Jain, “Performance evaluation of AES, ARC2, BlowFish, CAST and DES3 for standalone systems,” *Proc. 3rd Int. Conf. Comput. Methodol. Commun. ICCMC 2019*, no. Iccmc, pp. 118–123, 2019, doi: 10.1109/ICCMC.2019.8819729.

[128] N. A. Advani and A. M. Gonsai, “Performance analysis of symmetric encryption algorithms for their encryption and decryption time,” *Proc. 2019 6th Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2019*, pp. 359–362, 2019.

[129] E. Fernando, D. Agustin, M. Irsan, D. F. Murad, H. Rohayani, and D. Sujana, “Performance Comparison of Symmetries Encryption Algorithm AES and des with Raspberry Pi,” *Proc. 2019 4th Int. Conf. Sustain. Inf. Eng. Technol. SIET 2019*, pp. 353–357, 2019, doi: 10.1109/SIET48054.2019.8986122.

[130] U. Iftikhar, K. Asrar, M. Waqas, and S. A. Ali, “Evaluating the Performance Parameters of Cryptographic Algorithms for IOT-based Devices,” *Eng. Technol. Appl. Sci. Res.*, vol. 11, no. 6, pp. 7867–7874, 2021, doi: 10.48084/etasr.4263.

[131] S. N. Karale, K. Pendke, and P. Dahiwale, “The survey of various techniques & algorithms for SMS security,” *ICIIECS 2015 - 2015 IEEE Int. Conf. Innov. Information,*

Embed. Commun. Syst., 2015, doi: 10.1109/ICIIECS.2015.7192943.

[132] S. D. Sanap and V. More, “Analysis of encryption techniques for secure communication,” *2021 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2021*, vol. 1, no. 2, pp. 290–294, 2021, doi: 10.1109/ESCI50559.2021.9396926.

[133] B. Rahul and K. Kuppusamy, “Efficiency Analysis of Cryptographic Algorithms for Image Data Security in Cloud Environment,” *IETE J. Res.*, vol. 69, no. 9, pp. 6053–6064, 2023, doi: 10.1080/03772063.2021.1990141.

[134] Ratnadewi, R. P. Adhie, Y. Hutama, A. Saleh Ahmar, and M. I. Setiawan, “Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC),” *J. Phys. Conf. Ser.*, vol. 954, no. 1, 2018, doi: 10.1088/1742-6596/954/1/012009.

[135] R. Kumar and A. Singh, “Cloud Security using ECC and DH,” *Int. J. Inf. Comput. Sci.*, vol. 6, no. 7, pp. 518–524, 2019.

[136] S. Srivastava, A. Tiwari, and P. K. Srivastava, “Review on quantum safe algorithms based on Symmetric Key and Asymmetric Key Encryption methods,” *2022 2nd Int. Conf. Adv. Comput. Innov. Technol. Eng. ICACITE 2022*, pp. 905–908, 2022, doi: 10.1109/ICACITE53722.2022.9823437.

[137] S. Ahmed and T. Ahmed, “Comparative Analysis of Cryptographic Algorithms in Context of Communication: A Systematic Review,” *Int. J. Sci. Res. Publ.*, vol. 12, no. 7, pp. 161–173, 2022, doi: 10.29322/ijsrp.12.07.2022.p12720.

[138] C. Varma, “A Study of the ECC, RSA and the Diffie-Hellman Algorithms in Network Security,” *Proc. 2018 Int. Conf. Curr. Trends Towar. Converging Technol. ICCTCT 2018*, pp. 18–21, 2018, doi: 10.1109/ICCTCT.2018.8551161.

[139] E. Kwadwo, J. Ben, and F. Twum, “An Enhanced Elliptic Curve Cryptosystem for Securing Data,” *Int. J. Comput. Appl.*, vol. 182, no. 9, pp. 47–53, 2018, doi: 10.5120/ijca2018917688.

[140] V. Gupta, D. Stebila, S. Fung, S. C. Shantz, N. Gura, and H. Eberle, “Speeding up secure

Web transactions using elliptic curve cryptography," *Proc. Netw. Distrib. Syst. Secur. Symp.*, pp. 231–239, 2004.

[141] H. Eberle, N. Gura, S. C. Shantz, V. Gupta, L. Rarick, and S. Sundaram, "A public-key cryptographic processor for RSA and ECC," *Proc. Int. Conf. Appl. Syst. Archit. Process.*, pp. 98–110, 2004, doi: 10.1109/ASAP.2004.1342462.

[142] F. Mallouli, A. Hellal, N. Sharief Saeed, and F. Abdulraheem Alzahrani, "A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms," *Proc. - 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. CSCloud 2019 5th IEEE Int. Conf. Edge Comput. Scalable Cloud, EdgeCom 2019*, pp. 173–176, 2019, doi: 10.1109/CSCloud/EdgeCom.2019.00022.

[143] M. Suarez-Albela, T. M. Fernandez-Carames, P. Fraga-Lamas, and L. Castedo, "A practical performance comparison of ECC and RSA for resource-constrained IoT devices," *2018 Glob. Internet Things Summit, GIoTS 2018*, pp. 0–5, 2018, doi: 10.1109/GIOTS.2018.8534575.

[144] A. Kardi, R. Zagrouba, and M. Alqahtani, "Performance Evaluation of RSA and Elliptic Curve Cryptography in Wireless Sensor Networks," *21st Saudi Comput. Soc. Natl. Comput. Conf. NCC 2018*, no. April, 2018, doi: 10.1109/NCG.2018.8592963.

[145] S. R. Singh, A. K. Khan, and T. S. Singh, "A critical review on Elliptic Curve Cryptography," *Int. Conf. Autom. Control Dyn. Optim. Tech. ICACDOT 2016*, vol. 3, no. 7, pp. 13–18, 2017, doi: 10.1109/ICACDOT.2016.7877543.

[146] H. Hasan *et al.*, "Secure lightweight ECC-based protocol for multi-agent IoT systems," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, vol. 2017-Octob, 2017, doi: 10.1109/WiMOB.2017.8115788.

[147] D. Patel, B. Patel, J. Vasa, and M. Patel, *A Comparison of the Key Size and Security Level of the ECC and RSA Algorithms with a Focus on Cloud/Fog Computing*, vol. 719 LNNS. Springer Nature Singapore, 2023. doi: 10.1007/978-981-99-3758-5_5.

[148] L. Zou, M. Ni, Y. Huang, W. Shi, and X. Li, *Hybrid Encryption Algorithm Based on AES and RSA in File Encryption*, vol. 551 LNEE. Springer Singapore, 2020. doi: 10.1007/978-

981-15-3250-4_68.

[149] S. Rehman, N. Talat Bajwa, M. A. Shah, A. O. Aseeri, and A. Anjum, “Hybrid aes-ecc model for the security of data over cloud storage,” *Electron.*, vol. 10, no. 21, pp. 1–20, 2021, doi: 10.3390/electronics10212673.

[150] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, “Elasticity in Cloud Computing: State of the Art and Research Challenges,” *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, 2018, doi: 10.1109/TSC.2017.2711009.

[151] D. Kodzo, M. Hodowu, D. R. Korda, and E. Danso Ansong, “An Enhancement of Data Security in Cloud Computing with an Implementation of a Two-Level Cryptographic Technique, using AES and ECC Algorithm,” *Int. J. Eng. Res. Technol.*, vol. 9, no. March 2021, pp. 2278–0181, 2020, [Online]. Available: http://www.ijert.org

[152] M. Sivajyothi and T. Devi, “Analysis of Elliptic Curve Cryptography with AES for Protecting Data in Cloud with improved Time efficiency,” *Proc. 2nd Int. Conf. Innov. Pract. Technol. Manag. ICIPTM 2022*, no. Bhosle 2013, pp. 573–577, 2022, doi: 10.1109/ICIPTM54933.2022.9753926.

[153] B. Ranganatha Rao and B. Sujatha, “A hybrid elliptic curve cryptography (HECC) technique for fast encryption of data for public cloud security,” *Meas. Sensors*, vol. 29, no. June 2023, p. 100870, 2023, doi: 10.1016/j.measen.2023.100870.

[154] M. J. Dubai, T. R. Mahesh, and P. A. Ghosh, “Design of new security algorithm: Using hybrid Cryptography architecture,” *ICECT 2011 - 2011 3rd Int. Conf. Electron. Comput. Technol.*, vol. 5, pp. 99–101, 2011, doi: 10.1109/ICECTECH.2011.5941965.

[155] S. K. Ghosh, S. Rana, A. Pansari, J. Hazra, and S. Biswas, “Hybrid Cryptography Algorithm for Secure and Low Cost Communication,” *2020 Int. Conf. Comput. Sci. Eng. Appl. ICCSEA 2020*, pp. 4–8, 2020, doi: 10.1109/ICCSEA49143.2020.9132862.

[156] M. Sharma and S. Gandhi, “Compression and Encryption : An Integrated Approach,” *Int. J. Eng. Res. Technol.*, vol. 1, no. 5, pp. 1–7, 2012, [Online]. Available: www.ijert.org

[157] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, “Data compression methodologies for lossless data and comparison between algorithms,” *Int. J. Eng. Sci. Innov. Technol.*, vol. 2,

no. 2, pp. 142–147, 2013.

[158] S. R. Kodituwakku and U. S. Amarasinghe, “Comparison of Lossless Data Compression Algorithms,” *Indian J. Comput. Sci. Eng.*, vol. 1, no. 4, pp. 416–425, 2010.

[159] M. B. Begum, N. Deepa, M. Uddin, R. Kaluri, M. Abdelhaq, and R. Alsaqour, “An efficient and secure compression technique for data protection using burrows-wheeler transform algorithm,” *Heliyon*, vol. 9, no. 6, p. e17602, 2023, doi: 10.1016/j.heliyon.2023.e17602.

[160] B. Jasuja and A. Pandya, “Crypto-Compression System: An Integrated Approach using Stream Cipher Cryptography and Entropy Encoding,” *Int. J. Comput. Appl.*, vol. 116, no. 21, pp. 34–41, 2015, doi: 10.5120/20463-2831.

[161] R. H. Ali and J. M. Kadhim, “Text-based Steganography using Huffman Compression and AES Encryption Algorithm,” *Iraqi J. Sci.*, vol. 62, no. 11, pp. 4110–4120, 2021, doi: 10.24996/ijs.2021.62.11.31.

[162] A. Murtaza, S. J. Hussain Pirzada, and L. Jianwei, “A new symmetric key encryption algorithm with higher performance,” *2019 2nd Int. Conf. Comput. Math. Eng. Technol. iCoMET 2019*, pp. 0–6, 2019, doi: 10.1109/ICOMET.2019.8673469.

[163] A. M. Sagheer, M. S. Al-Ani, and O. A. Mahdi, “Ensure security of compressed data transmission,” *Proc. - 2013 6th Int. Conf. Dev. eSystems Eng. DeSE 2013*, pp. 270–275, 2013, doi: 10.1109/DeSE.2013.55.

[164] M. R. Ashila, N. Atikah, D. R. Ignatius Moses Setiadi, E. H. Rachmawanto, and C. A. Sari, “Hybrid AES-Huffman Coding for Secure Lossless Transmission,” *Proc. 2019 4th Int. Conf. Informatics Comput. ICIC 2019*, pp. 0–4, 2019, doi: 10.1109/ICIC47613.2019.8985899.

[165] M. Y. Elmahi, T. M. wahbi, and M. H. Sayed, “Text Steganography Using Compression and Random Number Generators,” *Int. J. Comput. Appl. Technol. Res.*, vol. 6, no. 6, pp. 259–263, 2017, doi: 10.7753/ijcatr0606.1005.

[166] N. Tiwari and B. N. Keshavamurthy, “Compression with Authenticated Encryption for Enhanced Security on Data Centric Products,” *IEEE Reg. 10 Annu. Int. Conf.*

Proceedings/TENCON, vol. 2019-Octob, pp. 1596–1600, 2019, doi: 10.1109/TENCON.2019.8929610.

[167] H. K. S. Premadasa and R. G. N. Meegama, "Extensive compression of text messages in interactive mobile communication," *Int. Conf. Adv. ICT Emerg. Reg. ICTer 2013 - Conf. Proc.*, vol. 1, pp. 80–83, 2013, doi: 10.1109/ICTer.2013.6761159.

[168] L. Hengjian, W. Jizhi, W. Yinglong, T. Min, and X. Shujiang, "A flexible and secure image compression coding algorithm," *2010 Int. Conf. Futur. Inf. Technol. Manag. Eng. FITME 2010*, vol. 2, pp. 376–379, 2010, doi: 10.1109/FITME.2010.5656273.

[169] N. N. Mohamed, H. Hashim, Y. M. Yussoff, M. A. M. Isa, and S. F. S. Adnan, "Compression and encryption technique on securing TFTP packet," *ISCAIE 2014 - 2014 IEEE Symp. Comput. Appl. Ind. Electron.*, pp. 198–202, 2015, doi: 10.1109/ISCAIE.2014.7010237.

[170] C. Chen, X. Wang, G. Huang, and G. Liu, "An Efficient Randomly-Selective Video Encryption Algorithm," *2022 IEEE 8th Int. Conf. Comput. Commun. ICCC 2022*, pp. 1287–1293, 2022, doi: 10.1109/ICCC56324.2022.10065724.

[171] M. Benabdellah, M. M. Himmi, and N. Zahid, "Encryption-Compression of Images Based on FMT and AES Algorithm," vol. 1, no. 45, pp. 2203–2219, 2007.

[172] Q. S. Alsaffar, H. N. Mohaisen, and F. N. Almashhdini, "An encryption based on DNA and AES algorithms for hiding a compressed text in colored Image," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1058, no. 1, p. 012048, 2021, doi: 10.1088/1757-899x/1058/1/012048.

[173] LAURENTINUS, H. A. PRADANA, D. Y. SYLFANIA, and F. P. JUNIAWAN, "Improving the SMS Security and Data Capacity Using Advanced Encryption Standard and Huffman Compression," vol. 172, no. Siconian 2019, pp. 194–202, 2020, doi: 10.2991/aisr.k.200424.028.

[174] U. K. Lilhore, S. Dalal, and S. Simaiya, "A cognitive security framework for detecting intrusions in IoT and 5G utilizing deep learning," *Comput. Secur.*, vol. 136, no. September 2023, 2024, doi: 10.1016/j.cose.2023.103560.

[175] U. K. Lilhore, S. Simaiya, S. Dalal, Y. K. Sharma, S. Tomar, and A. Hashmi, "Secure WSN Architecture Utilizing Hybrid Encryption with DKM to Ensure Consistent IoV Communication," *Wirel. Pers. Commun.*, no. 0123456789, 2024, doi: 10.1007/s11277-024-10859-0.

[176] M. Joye, P. Paillier, and S. Vaudenay, "Efficient generation of prime numbers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1965 LNCS, pp. 340–354, 2000, doi: 10.1007/3-540-44499-8_27.

[177] V. Kapoor, "Data Encryption and Decryption using Modified RSA Cryptography Based on Multiple Public Keys and 'n' Prime Number," *Int. J. Eng. Sci. Res. Technol.*, vol. 3, no. 6, pp. 713–720, 2014, [Online]. Available: http:

[178] A. Ivanov and N. Stoianov, "Implications of the Arithmetic Ratio of Prime Numbers for RSA Security," *Int. J. Appl. Math. Comput. Sci.*, vol. 33, no. 1, pp. 57–70, 2023, doi: 10.34768/amcs-2023-0005.

[179] N. Jain, S. S. Chauhan, and A. Raj, "Security Enhancement of RSA Algorithm using Increased Prime Number Set," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 3, pp. 4235–4240, 2020, doi: 10.35940/ijeat.c6278.029320.

[180] S. Pradhan and B. K. Sharma, "An Efficient RSA Cryptosystem with BM-PRIME Method," *Int. J. Inf. Netw. Secur.*, vol. 2, no. 1, pp. 103–108, 2012, doi: 10.11591/ijins.v2i1.1718.

[181] K. Pavani and P. Sriramya, "Enhancing public key cryptography using RSA, RSA-CRT and N-Prime RSA with multiple keys," *Proc. 3rd Int. Conf. Intell. Commun. Technol. Virtual Mob. Networks, ICICV 2021*, no. Icicv, pp. 661–667, 2021, doi: 10.1109/ICICV50876.2021.9388621.

[182] N. Khairina and M. K. Harahap, "RSA Cryptographic Algorithm using Cubic Congruential Generator," *J. Phys. Conf. Ser.*, vol. 1424, no. 1, 2019, doi: 10.1088/1742-6596/1424/1/012010.

[183] R. Imam, F. Anwer, and M. Nadeem, "An Effective and enhanced RSA based Public Key Encryption Scheme (XRSA)," *Int. J. Inf. Technol.*, vol. 14, no. 5, pp. 2645–2656, Aug.

2022, doi: 10.1007/s41870-022-00993-y.

[184] D. I. George Amalarethinam and H. M. Leena, “Enhanced RSA algorithm for data security in cloud,” *Int. J. Control Theory Appl.*, vol. 9, no. 27, pp. 147–152, 2016.

[185] M. A. Budiman, P. Sihombing, and I. A. Fikri, “A cryptocompression system with Multi-Factor RSA algorithm and Levenstein code algorithm,” *J. Phys. Conf. Ser.*, vol. 1898, no. 1, pp. 0–4, 2021, doi: 10.1088/1742-6596/1898/1/012040.

[186] M. R. K. Ariffin, S. I. Abubakar, M. A. Asbullah, and F. Yunos, “New cryptanalytic attack on rsa modulus n = pq using small prime difference method,” *Cryptography*, vol. 3, no. 1, pp. 1–25, 2019, doi: 10.3390/cryptography3010002.

[187] M. Bahadori, M. R. Mali, O. Sarbishei, M. Atarodi, and M. Sharifkhani, “A novel approach for secure and fast generation of RSA public and private keys on SmartCard,” *Proc. 8th IEEE Int. NEWCAS Conf. NEWCAS2010*, pp. 265–268, 2010, doi: 10.1109/NEWCAS.2010.5603937.

[188] A. H. Mahmoud, H. H. Issa, N. H. Shaker, and K. A. Shehata, “Customized AES for Securing Data in Sensitive Networks and Applications,” *Natl. Radio Sci. Conf. NRSC, Proc.*, vol. 2022-Novem, no. Nrsc, pp. 164–170, 2022, doi: 10.1109/NRSC57219.2022.9971420.

[189] Z. Cao, G. Yi, B. Wu, J. Li, and D. Xiao, “Analysis And Improvement of AES Key Expansion Algorithm,” *2022 Int. Conf. Artif. Intell. Comput. Inf. Technol. AICIT 2022*, pp. 1–5, 2022, doi: 10.1109/AICIT55386.2022.9930239.

[190] D. Chen, D. Qing, and D. Wang, “AES key expansion algorithm based on 2D logistic mapping,” *Proc. 5th Int. Work. Chaos-Fractals Theor. Appl. IWCFTA 2012*, pp. 207–211, 2012, doi: 10.1109/IWCFTA.2012.81.

[191] I. Saberi, B. Shojaie, and M. Salleh, “Enhanced Key Expansion for AES-256 by using Even-Odd method,” *2011 Int. Conf. Res. Innov. Inf. Syst. ICRIIS’11*, pp. 1–5, 2011, doi: 10.1109/ICRIIS.2011.6125708.

[192] B. Subramanyan, V. M. Chhabria, and T. G. Sankar Babu, “Image encryption based on AES Key Expansion,” *Proc. - 2nd Int. Conf. Emerg. Appl. Inf. Technol. EAIT 2011*, pp.

217–220, 2011, doi: 10.1109/EAIT.2011.60.

[193] Y. Xiao, Y. Jiang, and L. Xu, “Efficient and Enhanced Advanced Encryption Standard Algorithm with Chaos-Based Key Expansion,” *2023 4th Int. Conf. Inf. Sci. Parallel Distrib. Syst. ISPDS 2023*, pp. 480–483, 2023, doi: 10.1109/ISPDS58840.2023.10235699.

[194] A. Murtaza, S. J. H. Pirzada, M. N. Hasan, T. Xu, and L. Jianwei, “Parallelized key expansion algorithm for advanced encryption standard,” *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS*, vol. 2019-Octob, pp. 609–612, 2019, doi: 10.1109/ICSESS47205.2019.9040825.

[195] K. Kalaiselvi and A. Kumar, “Enhanced AES cryptosystem by using genetic algorithm and neural network in S-box,” *2016 IEEE Int. Conf. Curr. Trends Adv. Comput. ICCTAC 2016*, pp. 1–6, 2016, doi: 10.1109/ICCTAC.2016.7567340.

[196] E. M. De Los Reyes, A. M. Sison, and R. P. Medina, “Modified AES cipher round and key schedule,” *Indones. J. Electr. Eng. Informatics*, vol. 7, no. 1, pp. 28–35, 2019, doi: 10.11591/ijeei.v7i1.652.

[197] R. Lin and S. Li, “An Image Encryption Scheme Based on Lorenz Hyperchaotic System and RSA Algorithm,” *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/5586959.

[198] A. G. Marco, A. S. Martinez, and O. M. Bruno, “Fast, parallel and secure cryptography algorithm using lorenz’s attractor,” *Int. J. Mod. Phys. C*, vol. 21, no. 3, pp. 365–382, 2010, doi: 10.1142/S0129183110015166.

[199] Ö. E. Tekerek and A. Tekerek, “Image Encryption Using Lorenz’s Attractor and Fractional Fourier Transform,” *2nd Int. Informatics Softw. Eng. Conf. IISEC 2021*, no. 1, 2021, doi: 10.1109/IISEC54230.2021.9672362.

[200] P. Sankaranarayanan, R. Tamijetchelvy, M. N. Periya, and M. Manikandan, “An Efficient and Optimized Color Image Encryption Technique Using Lorentz , Rossler and Chen Attractor,” *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 3, no. 3, pp. 2301–2305, 2014.

[201] Z. Jiang and X. Liu, “Image Encryption Algorithm Based on Discrete Quantum Baker Map and Chen Hyperchaotic System,” *Int. J. Theor. Phys.*, vol. 62, no. 2, 2023, doi:

10.1007/s10773-023-05277-0.

[202] G. A. Eslam, S. Eman, and H. Mohamed, “Lightweight Mix Columns Implementation for AES,” *Proc. 9th WSEAS Int. Conf. Appl. informatics Commun.*, vol. 01, pp. 1–23, 2016.

[203] H. V. Gamido, A. M. Sison, and R. P. Medina, “Modified AES for text and image encryption,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 11, no. 3, pp. 942–948, 2018, doi: 10.11591/ijeecs.v11.i3.pp942-948.

[204] O. Dunkelman and N. Keller, “The effects of the omission of last round’s MixColumns on AES,” *Inf. Process. Lett.*, vol. 110, no. 8–9, pp. 304–308, 2010, doi: 10.1016/j.ipl.2010.02.007.

[205] K. AlMarashda, Y. AlSalami, K. Salah, and T. Martin, “On the security of inclusion or omission of MixColumns in AES cipher,” *2011 Int. Conf. Internet Technol. Secur. Trans. ICITST 2011*, no. December, pp. 34–39, 2011.

[206] A. Barrera, C. W. Cheng, and S. Kumar, “Improved Mix Column Computation of Cryptographic AES,” *Proc. - 2019 2nd Int. Conf. Data Intell. Secur. ICDIS 2019*, pp. 229–232, 2019, doi: 10.1109/ICDIS.2019.00042.

[207] Soukaena Hassan and M. Abd Zaid, “Modification Advanced Encryption Standard for Design Lightweight Algorithms,” *J. Kufa Math. Comput.*, vol. 6, no. 1, pp. 21–27, 2019, doi: 10.31642/jokmc/2018/060104.

[208] W. Kretschmer, L. Qian, M. Sinha, and A. Tal, “Quantum Cryptography in Algorithmica,” *Proc. Annu. ACM Symp. Theory Comput.*, pp. 1589–1602, 2023, doi: 10.1145/3564246.3585225.

[209] J. Yin *et al.*, “Entanglement-based secure quantum cryptography over 1,120 kilometres,” *Nature*, vol. 582, no. 7813, pp. 501–505, 2020, doi: 10.1038/s41586-020-2401-y.

[210] S. Pirandola *et al.*, “Advances in quantum cryptography,” *Adv. Opt. Photonics*, vol. 12, no. 4, p. 1012, 2020, doi: 10.1364/aop.361502.

[211] D. J. Bernstein and T. Lange, “Introduction to post-quantum cryptography,” no. August, 2022.

[212] H. Iqbal and W. O. Krawec, "Semi-Quantum Cryptography," in *Advanced Sciences and Technologies for Security Applications*, vol. 1, Springer New York, 2019, pp. 1–15. doi: 10.1007/0-387-25096-4_1.

[213] D. J. Bernstein and T. Lange, "Post-quantum cryptography - dealing with the fallout of physics success.," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 314, 2017, [Online]. Available: http://dblp.uni-trier.de/db/journals/iacr/iacr2017.html#BernsteinL17a

[214] C. H. Ugwuishiwu, U. E. Orji, C. I. Ugwu, and C. N. Asogwa, "An overview of Quantum Cryptography and Shor's Algorithm," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 5, pp. 7487–7495, 2020, doi: 10.30534/ijatcse/2020/82952020.

[215] D. T. Dam, T. H. Tran, V. P. Hoang, C. K. Pham, and T. T. Hoang, "A Survey of Post-Quantum Cryptography: Start of a New Race," *Cryptography*, vol. 7, no. 3, pp. 1–18, 2023, doi: 10.3390/cryptography7030040.

[216] P. K. Ghosh, S. K. Ghosh, and L. M. Khan, "Current trend of bank selection criteria of retail customers in Bangladesh: An investigation," *Glob. Bus. Financ. Rev.*, vol. 20, no. 2, pp. 27–34, 2015, doi: 10.17549/gbfr.2015.20.2.27.

[217] M. Panda, "Performance analysis of encryption algorithms for security," *Int. Conf. Signal Process. Commun. Power Embed. Syst. SCOPES 2016 - Proc.*, pp. 278–284, 2017, doi: 10.1109/SCOPES.2016.7955835.

[218] B. Xing, D. D. Wang, Y. Yang, Z. Wei, J. Wu, and C. He, "Accelerating DES and AES Algorithms for a Heterogeneous Many-core Processor," *Int. J. Parallel Program.*, vol. 49, no. 3, pp. 463–486, 2021, doi: 10.1007/s10766-021-00692-4.

[219] Z. Kasiran, A. Dalil, and M. Z. Ghazali, "Analysis on Computational Time of Hybrid Cryptography in Email System," *J. Posit. Sch. Psychol.*, vol. 2022, no. 3, pp. 8415–8422, 2022, [Online]. Available: http://journalppw.com

[220] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing Attacks: A Recent Comprehensive Study and a New Anatomy," *Front. Comput. Sci.*, vol. 3, no. March, pp. 1–23, 2021, doi: 10.3389/fcomp.2021.563060.

[221] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive

Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," *Electron.*, vol. 12, no. 6, 2023, doi: 10.3390/electronics12061333.

[222] G. B. Thompson, "Journal of information Science," *J. Inf. Sci.*, vol. 9, no. 2, p. 74, 1984, doi: 10.1177/016555158400900204.

[223] A. Karki, "A Comparative Analysis of Public Key Cryptography," *Int. J. Mod. Comput. Sci.*, vol. 4, no. 6, pp. 2320–7868, 2016, [Online]. Available: http://www.iusikkim.edu.in/IJMCS161213.pdf

[224] R. M. Abobeah, M. M. Ezz, and H. M. Harb, "Public-Key Cryptography Techniques Evaluation," *Int. J. Comput. Networks Appl.*, vol. 2, no. 2, pp. 64–75, 2015.

[225] M. Shivansh, "A Brief Study on various Cryptographic Algorithms for Data Security," *Int. J. Res. Analtical Rev.*, vol. 9, no. 2, pp. 288–293, 2022.

[226] S. Tayde and A. S. Siledar, "File Encryption, Decryption Using AES Algorithm in Android Phone," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 5, pp. 550–554, 2015.

[227] B. Li, J. Xu, and Z. Liu, "SW-LZMA: Parallel Implementation of LZMA Based on SW26010 Many-Core Processor," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/4486494.

[228] B. I. Diop, A. D. Gueye, and A. Diop, "Comparative Study Between Different Algorithms of Data Compression and Decompression Techniques," in *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences : PCCDS 2021*, 2023, pp. 737–744. doi: 10.1007/978-981-19-8742-7_59.

[229] Z. Mahrousa, A. Bitar, and Y. Fareed, "A Novel Method to Increase Diffusion and Confusion in AES Algorithm," *Int. J. Comput. Appl.*, vol. 177, no. 36, pp. 39–47, 2020, doi: 10.5120/ijca2020919872.

[230] T. Kumaki, T. Fujita, M. Nakanishi, and T. Ogura, "Morphological pattern spectrum and block cipher processing based image-manipulation detection," *Nonlinear Theory Its Appl. IEICE*, vol. 4, no. 4, pp. 400–418, 2013, doi: 10.1587/nolta.4.400.

[231] H. Talirongan, A. M. Sison, and R. P. Medina, “Modified advanced encryption standard using butterfly effect,” *2018 IEEE 10th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2018*, no. December 2019, 2018, doi: 10.1109/HNICEM.2018.8666368.

[232] M. Alizadeh, M. Salleh, M. Zamani, S. Jafar, and K. Sasan, “Security and Performance Evaluation of Lightweight Cryptographic Algorithms in RFID,” *Recent Res. Commun. Comput.*, no. November 2015, pp. 45–50, 2012, [Online]. Available: http://goo.gl/ej5iEr

[233] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, “Modified advanced encryption standard algorithm for information security,” *Symmetry (Basel).*, vol. 11, no. 12, pp. 1–16, 2019, doi: 10.3390/SYM11121484.

[234] L. Polani, K. Balasubramanian, B. Yamuna, and T. V. Sai, “Low power and area efficient AES implementation using ROM based key expansion and rotational shift,” *2022 IEEE Int. Conf. Distrib. Comput. VLSI, Electr. Circuits Robot. Discov. 2022 - Proc.*, pp. 72–75, 2022, doi: 10.1109/DISCOVER55800.2022.9974925.

[235] K. B. Anuroop and M. Neema, “Fully pipelined-loop unrolled AES with enhanced key expansion,” *2016 IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. RTEICT 2016 - Proc.*, pp. 988–992, 2017, doi: 10.1109/RTEICT.2016.7807977.

[236] E. S. A. M. ElBadawy, W. A. El-Masry, A. Mokhtar, and A. E. D. S. Hafez, “A new chaos advanced encryption standard (AES) algorithm for data security,” *Int. Conf. Signals Electron. Syst. ICSES’10 - Conf. Proceeding*, no. June, pp. 405–408, 2010.

[237] A. U. Rahman, S. U. Miah, and S. Azad, “Advanced encryption standard,” *Pract. Cryptogr. Algorithms Implementations Using C++*, pp. 91–126, 2014, doi: 10.1201/b17707.

[238] F. Olajide, K. Assa-Agyei, and C. Edo, “An Empirical Evaluation of Encryption and Decryption Times on Block Cipher Techniques,” *Proc. - 2023 Congr. Comput. Sci. Comput. Eng. Appl. Comput. CSCE 2023*, pp. 2385–2390, 2023, doi: 10.1109/CSCE60160.2023.00386.

[239] N. Murugesan and A. M. S. Ramasamy, “A numerical algorithm for nth root,” *J. Fundam.*

Sci., vol. 6, no. 2, pp. 104–110, 2010.

[240] E. M. Hussein Saeed and R. M. Hussain, “Encryption of Association Rules Using Modified Dynamic Mapping and Modified (AES) Algorithm,” *1st Int. Sci. Conf. Comput. Appl. Sci. CAS 2019*, pp. 204–209, 2019, doi: 10.1109/CAS47993.2019.9075701.

[241] H. K. Hoomod and A. M. Radi, “New Secure E-mail System Based on Bio-Chaos Key Generation and Modified AES Algorithm,” *J. Phys. Conf. Ser.*, vol. 1003, no. 1, 2018, doi: 10.1088/1742-6596/1003/1/012025.

[242] J. Ren and S. Chen, “A further understanding of differential-linear cryptanalysis,” *Chinese J. Electron.*, vol. 29, no. 4, pp. 660–666, 2020, doi: 10.1049/cje.2020.05.010.

[243] L. Peng, L. Hu, and Y. Lu, “Improved Results on Cryptanalysis of Prime Power RSA,” in *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 2, Berlin, Heidelberg: Springer Berlin, 2017, pp. 287–303. doi: 10.1007/978-3-319-53177-9_15.

[244] D. R. Stinson, “Cryptography: Theory and practice, third edition,” 2005.

[245] M. Dehghani, S. Hubalovsky, and P. Trojovsky, “Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems,” *IEEE Access*, vol. 9, pp. 162059–162080, 2021, doi: 10.1109/ACCESS.2021.3133286.

[246] A. K. Qin, V. L. Huang, and P. N. Suganthan, “(SADE)Adaptation for Global Numerical Optimization,” *IEEE Commun. Mag.*, vol. 13, no. 2, pp. 398–417, 2009.

[247] S. Y. Bonde and U. S. Bhadade, “Analysis of Encryption Algorithms (RSA, SRNN and 2 Key Pair) for Information Security,” *2017 Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2017*, pp. 1–5, 2017, doi: 10.1109/ICCUBEA.2017.8463720.

[248] J. K. Dawson, F. Twum, J. B. Hayfron-Acquah, Y. M. Missah, and B. B. K. Ayawli, “An enhanced RSA algorithm using Gaussian interpolation formula,” *Int. J. Comput. Aided Eng. Technol.*, vol. 16, no. 4, pp. 534–552, 2022, doi: 10.1504/IJCAET.2022.123996.

[249] M. Liskov, “Miller-Rabin Probabilistic Primality test,” in *Encyclopedia of Cryptography and Security*, vol. 196, no. 1985, Boston, MA: Springer US, 2011, pp. 742–748. doi: 10.1007/978-1-4419-5906-5_592.

[250] M. O. Rabin, “Probabilistic algorithm for testing primality,” *J. Number Theory*, vol. 12, no. 1, pp. 128–138, 1980, doi: 10.1016/0022-314X(80)90084-0.