JOIV : Int. J. Inform. Visualization, 9(1) - January 2025 407-417

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

RC5 Performance Enhancement Based on Parallel Computing

Suaad Ali Abead [a], Nada Hussein M. Ali [b,*]

[a] Department of Computer Science, College of Science for Women, University of Baghdad, Baghdad, Iraq
[b] Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Corresponding author: *nada.husn@sc.uobaghdad.edu.iq

***Abstract*—This study aims to enhance the RC5 algorithm to improve encryption and decryption speeds in devices with limited power and memory resources. These resource-constrained applications, which range in size from wearables and smart cards to microscopic sensors, frequently function in settings where traditional cryptographic techniques because of their high computational overhead and memory requirements are impracticable. The Enhanced RC5 (ERC5) algorithm integrates the PKCS#7 padding method to effectively adapt to various data sizes. Empirical investigation reveals significant improvements in encryption speed with ERC5, ranging from 50.90% to 64.18% for audio files and 46.97% to 56.84% for image files, depending on file size. A substantial improvement of 59.90% is observed for data files sized at 1500000kb. Partitioning larger files notably reduces encryption time, while smaller files experience marginal benefits. Certain file types benefit from both strategies. Evaluation metrics include encryption execution time and throughput, consistently demonstrating ERC5's superiority over the original RC5. Moreover, ERC5 exhibits reduced power consumption and heightened throughput, highlighting its multifaceted benefits in resource-constrained environments. ERC5 is developed and tested on various file types and sizes to evaluate encryption speed, power consumption, and throughput. ERC5 significantly improves encryption speed across different file types and sizes, with notable gains for audio, image, and large data files. While partitioning smaller files only slightly improves encryption time, larger files partitioning yields faster results. Future research could explore ERC5 optimizations for different computing environments, its integration into real-time encryption scenarios, and its impact on other cryptographic operations and security protocols.**

***Keywords*— Lightweight cryptography; RC5; encryption and decryption; time consuming.**

Manuscript received 14 Apr. 2024; revised 12 Jul. 2024; accepted 15 Sep. 2024. Date of publication 31 Jan. 2025.

I. INTRODUCTION

In today's interconnected world, the proliferation of Internet of Things (IoT) devices and embedded systems is rapidly expanding, ensuring security in resource-constrained environments poses a significant challenge. Lightweight cryptography emerges as a crucial solution to address this challenge, offering robust security measures tailored specifically for devices with limited computational power, memory, and energy resources. These resource-constrained applications, ranging from tiny sensors to wearable devices and smart cards, often operate in environments where traditional cryptographic techniques are impractical due to their high computational overhead and memory requirements. Therefore, lightweight cryptography plays a pivotal role in protecting sensitive data and communications in these contexts. This study explores the significance of lightweight cryptography in providing security to resource-constrained applications, emphasizing its role in enabling secure communication, data protection, and privacy preservation amidst the constraints imposed by the devices' limited resources [1], [2], [3]. These applications produce large capacities of data that require efficient processing. In such resource-constrained environments, the traditional cryptographic methods often prove cumbersome and impractical due to their high computational demands. Lightweight cryptography offers a tailored solution, providing robust security measures while minimizing the computational burden on these devices. By optimizing algorithms and protocols for efficiency and resource conservation, lightweight cryptography ensures that security is not compromised in the face of constrained resources [4].

The RC5 algorithm is considered lightweight by nature due to its minimal memory and power requirements, rendering it suitable for deployment in devices with limited resources [5]. This characteristic is particularly advantageous in resource-constrained environments where conserving memory and power consumption is paramount. By minimizing the

computational overhead, RC5 efficiently balances security requirements with the constraints imposed by the device's capabilities. Its lightweight nature makes RC5 an attractive choice for implementation in various applications, ranging from embedded systems and IoT devices to smart cards and mobile devices. This adaptability ensures that RC5 can provide robust cryptographic protection without unduly taxing the resources available to the device, thereby facilitating secure operations in diverse scenarios [6]. RC5, standing for "Rivest Cipher," represents a pivotal advancement in the realm of cryptographic algorithms, distinguished by its rapid block encryption technique operating symmetrically [7]. What sets RC5 apart is its ingenious utilization of rotation operations based on data, a feature that contributes to its innovative approach to encryption. Unlike conventional cryptographic methods, RC5 incorporates dynamic elements such as variable block sizes, round numbers, and key lengths, endowing it with a remarkable degree of adaptability to varying performance and security requirements [8]. This adaptability is crucial in modern computing environments where the trade-off between speed and security is often a delicate balance. Moreover, RC5's exceptional speed and minimal memory requirements further underscore its prowess as an efficient cryptographic solution [9].

The algorithm's ability to swiftly encrypt and decrypt data while consuming minimal memory resources is particularly advantageous in scenarios where computational efficiency and resource conservation are paramount. This makes RC5 an appealing choice for a diverse range of applications, spanning from embedded systems and IoT devices to high-performance computing platforms. Its versatility extends to environments where memory constraints and power limitations dictate the feasibility of cryptographic implementations. By leveraging RC5, developers can ensure robust cryptographic protection without compromising on system performance or resource availability. The algorithm's ability to strike a balance between speed, security, and resource efficiency makes it a valuable asset in safeguarding sensitive data across a myriad of computing platforms.

Consequently, RC5 stands as a testament to the ingenuity and adaptability of cryptographic techniques, offering a reliable solution tailored to meet the evolving demands of modern computing landscapes [10], [11]. The RC5 algorithm utilizes several parameters to govern its operations [12], [13], [14], [15], [16], [17], [18]. One of these parameters is the size of the word ω in bits. This parameter determines the size of the word used in the algorithm's computations. The options typically include 16, 32, and 64 bits, with 32 bits often recommended as the optimal choice.

Encoding blocks into two words (A and B): Blocks of data are encoded into two words, denoted as A and B, as part of the RC5 algorithm's process.

The maximum number of rounds in algorithm $R(\mathcal{R})$: This parameter specifies the maximum number of rounds that the algorithm will iterate through during encryption or decryption. $\mathcal{R}$ can be any integer between 0 and 255 inclusive, allowing for flexibility in adjusting the algorithm's complexity and security level. The size of the secret key in bytes: The secret key's size is expressed in bytes and determines the length of the key used for encryption and decryption. Similar to the maximum number of rounds, the size of the secret key can range from 0 to 255 bytes, providing versatility in accommodating different security requirements and key lengths. These parameters collectively define the configuration and behavior of the RC5 algorithm, allowing for customization to suit various cryptographic needs and constraints.

The focal point of this research involves enhancing the RC5 algorithm to fulfill the requirements of rapid processing essential for specific applications, such as video conferencing and remote sensing imagery. Given RC5's nature as an iterative and block cipher algorithm, its performance is pivotal in scenarios prioritizing minimal processing time for effective encryption and decryption tasks. The primary aim of this study is to enhance the performance of the RC5 algorithm by harnessing the potential of multiple computation cores, rather than relying solely on individual cores. This entails distributing the computational workload across several cores to notably augment the algorithm's efficiency and speed, particularly when managing various data types. This strategy aims to optimize resource utilization and decrease encryption time, consequently addressing the pressing demand for swift cryptographic processing in real-time applications like video conferencing and remote sensing image analysis.

The structure of the paper is organized into several sections. The second section focuses on discussing related works that have contributed significantly to shaping the research's objectives and perspectives. Following this, the third section provides a comprehensive explanation of the RC5 algorithm, elucidating its principles and operational mechanisms. Subsequently, the fourth section presents a detailed account of the proposed enhancement of the RC5 algorithm, outlining the modifications and strategies aimed at improving its performance. In the fifth section, the paper delves into the results obtained from the implementation of the enhanced algorithm, accompanied by a thorough performance evaluation. Finally, the paper concludes by summarizing the key findings and insights drawn from the research, providing closure to the study.

This study undertakes a comprehensive review of the existing literature focusing on the adaptation and refinement of the RC5 block cipher algorithm. A plethora of research endeavors has been dedicated to exploring various methods aimed at enhancing both the encryption time and security levels associated with RC5. One notable contribution, highlighted in reference [5], introduces an enhanced RC5 algorithm that seeks to bolster its encoding speed by leveraging a straightforward yet remarkably effective technique involving the incorporation of random numbers followed by their integration into the key expansion phase. This enhancement strategy encompasses the iterative generation of random numbers to be added to the generated key, a process reiterated for two consecutive rounds, along with the introduction of supplementary blocks and bitwise operations aimed at further fortifying the algorithm's performance. Empirical findings gleaned from experimentation underscore the substantial efficacy of the upgraded RC5 algorithm, evidencing significant improvements in both encryption and decryption performance, surpassing the conventional RC5 approach by remarkable margins of 75.73% and 42.05%, respectively.

Moreover, another notable study, referenced as [12], delves into the practical implementation of the Enhanced RC5 (ERC5) algorithm within the context of an RFID-based payment system deployed at the DMMMSU-NLUC Fast Food Centre, tasked with safeguarding sensitive information within the system. The enhancement process unfolds across three pivotal phases: Key Expansion, Encryption, and Decryption. Results stemming from this implementation endeavor showcase the successful integration of the ERC5 algorithm into the RFID-based payment system, demonstrating marked enhancements over the traditional RC5 in terms of cryptographic efficiency and decryption speed, boasting improvements of 75.73% and 42.05%, respectively. Furthermore, ERC5 emerges as a more resource-efficient alternative compared to its predecessor, classic RC5, exhibiting notably higher encryption and decryption percentages, registering at 316.90% and 77.04%, respectively. These findings collectively underscore the transformative potential of enhanced RC5 variants in significantly augmenting the cryptographic capabilities and operational efficiency of security-sensitive systems and applications.

A study by [19] introduced two suggested improvement techniques (ERC5 and ERSA) in a parallel environment to provide quick and effective picture encryption and decryption. The RC5 technique has been enhanced to use four distinct cipher keys and four cores to encrypt images, as opposed to just one key and one core. Improvement Instead of utilizing two prime numbers, one public key, one private key, and one core, the RSA algorithm (ERSA) encrypts keys ERC5 using four prime numbers, two public keys, two private keys, and four cores. This method Less time than the RSA and RC5 methods initially required roughly 18 seconds. Acceleration of the suggested system using a quad-core parallel environment Approximately 105.16% higher than the acceleration of primitive sequence-based (RC5, RSA) algorithms in the Core environment.

According to [20], the proposition of Format-Preserving Encryption (FPE) based on the RC5 block cipher algorithm emerges as a solution aimed at safeguarding data integrity while simultaneously preserving the original formatting of the data. This innovative approach seeks to overcome the limitations encountered in existing systems, particularly those relying on AES FPE, by accommodating all forms of data formats without imposing a significant overhead on execution time. In contrast to AES FPE, the proposed format-preserving RC5 algorithm demonstrates notably faster execution times, thus offering a compelling alternative for applications requiring efficient data protection mechanisms while maintaining data integrity.

Similarly, reference [21] presents an innovative adaptation of the RC5 method tailored specifically to enhance security measures and expedite encryption processes, particularly for image data. This novel iteration introduces a new operation denoted as (#), which supplants the conventional XOR procedure within the algorithm's sixteen rounds, alongside an additional security layer incorporating two distinct keys. The outcomes of this refinement showcase remarkable efficacy in bolstering security protocols and reducing encryption times, all while preserving a diminished Peak Signal-to-Noise Ratio (PSNR) and minimizing the time discrepancy between original and encrypted images, thus demonstrating time-efficient encryption practices without compromising on security or image quality.

Furthermore, in a study by [1], the introduction of two distinct RC5 hardware architectures geared towards minimizing latency emerges as a significant contribution. The first architecture, denoted as RC5-1b, adopts a bit-serial design methodology, achieving noteworthy improvements in Transactions Per Second (TPS) compared to minimal bit-serial AES and SIMON implementations. Conversely, the second architecture, RC5-8b, employs an 8-bit design strategy featuring dual Data-Direct Memory (DDR) units, demonstrating substantial enhancements in TPS compared to the smallest 8-bit AES LED implementations. Through this comparative analysis, it is inferred that RC5 presents a favorable choice for DDR-based block ciphers, boasting low latency and optimal utilization of hardware resources, thereby emphasizing its suitability for applications demanding efficient and low-latency cryptographic solutions. In a study by [22], the novel HEVC video encoding method Sensitive portions of the video are encrypted using RC5, to realize low latency. The average time needed for videos with low quality drops by 0.7 seconds, while that of videos with high resolution drops by 19 seconds. The research aims to achieve real-time systems with low complexity and quick encoding times.

In a research by [23], a novel method is proposed for organizing keys hierarchically to bolster the security of a variety of Wireless Sensor Networks (WSNs) operating under the Hybrid Energy-Efficient Distributed (HEED) routing protocol. The incorporation of Cipher Block Chaining-Rivest Cipher 5 (CBC-RC5) within this method ensures the fulfillment of cryptographic objectives, particularly confidentiality. To assess the efficacy of the newly proposed approach, a comprehensive comparison is conducted against existing methods, including Dynamic Secret Key Management (DSKM) and Smart Security Implementation (SSI), under identical conditions. The findings reveal that the suggested hierarchical key organization approach offers notable advantages in terms of scalability, security, adaptability, and energy efficiency. By minimizing power consumption in both encryption and decryption processes, the proposed method effectively reduces processing time within the CBC-RC5 algorithm. Consequently, this innovative approach demonstrates significant promise in enhancing the security and operational efficiency of WSNs, particularly in resource-constrained environments, by leveraging the robust cryptographic capabilities of CBC-RC5 while optimizing energy utilization and processing time.

Another study by [24], the challenges and importance of privacy and security in genomic data storage and management, particularly in the context of utilizing cloud services, are discussed in the provided reference. Emphasis is placed on the necessity for effective solutions, including cryptography, to address concerns related to data tampering and unauthorized access. Insights into the significance of privacy-preserving methods for genomic data are provided by this source, rendering it relevant to the literature review. In reference [25], the proposed algorithm for image encryption using the AES algorithm, combined with error detection using Cyclic Redundancy Check (CRC) to maintain data integrity, is discussed in the provided reference. Emphasis is placed on optimizing the size of redundant bits for error detection,

particularly for compatibility with various computer architectures and software implementation. Different noise ratios are applied to tested images to evaluate their effect on encrypted images, with findings indicating varying impacts based on image size and noise levels. The significance of this research lies in its contribution to enhancing confidentiality in color image transmission over noisy communication channels. Table 1 demonstrates in brief the aforementioned research for RC5 enhancement algorithm.

TABLE I
COMPARATIVE PRESENTATION OF VARIOUS TECHNIQUES FOR RC5 ENHANCEMENT

Ref.	Techniques	Performance Metric	Results evaluation
[5]	The enhanced RC5 (ERC5) algorithm	T-test-performance	ERC5 performs better than the traditional RC5 by 75.73% and 42.05%, respectively in encrypted and decrypted time
[12]	ERC5 algorithm in RFID	T-test- performance	ERC5 is exhibiting extremely high levels of encryption and decryption, at 316.90 and 77.04 percent respectively.
[19]	New approach based on AES and RC5 algorithm cryptosystem.	MSE - Peak Signal to Noise Ratio (PSNR)	105.16% faster acceleration than RSA and RC5 algorithms
[20]	Format preserved RC5 algorithm for data security	Execution Time graphical visualization	The amount of time required by the suggested algorithm is extremely reduced.
[21]	Modifying the RC5 algorithm	$Entropy$- mean square error (MSE) - PSNR- Correlation Value- Encryption Time	Enhance security, significantly reducing encryption and decryption time.
[1]	Two RC5 hardware architectures	latency comparison	a 72% and 33% increase in TPS in (RC5-1b), it showcases a remarkable 69% and 4× boost in TPS in (RC5-8b), resulting in low-latency.
[22]	HEVC RC5-based SE algorithm	PSNR – SSIM- Histograms Test- Correlation Coefficient	The average time decreases for low-quality video by 0.7 seconds and for high-resolution video by 19 seconds.
[23]	a hierarchical key management method for safeguarding heterogeneous WSNs on hybrid energy-efficient distributed (HEED) routing	Energy consumption test	scalability, reliability, flexibility, and energy conservation.
[24]	Homomorphic encryption (HE), Garbled Circuit (GC), and Differential Privacy (DP)	Many methods had been reviewed.	The results were listed in tables.
[25]	AES algorithm, Cyclic Redundancy Check (CRC), CRC64 (Slicing-by-4 algorithm)	Mean Square Error (MSE)	different tested images by added different noise ratios (1% and 5%) of total images size to study the noise effect on the encrypted images.

The summarized findings from Table 1 highlight the superior performance of the RC5 algorithm compared to competing cryptographic algorithms such as RSA, AES, SIMON, and Blowfish, particularly in terms of both speed and security. Despite these advantages, RC5, being a symmetric key cryptosystem, faces challenges in efficiently encrypting large volumes of data due to processing time constraints. To mitigate this issue, leveraging multiple cores presents a viable solution. The subsequent sections of this paper delve into a detailed analysis of RC5 and introduce a novel technique aimed at addressing these time challenges.

II. MATERIALS AND METHOD

The RC5 algorithm, a symmetric key block cipher, forms the basis of this study. It operates on variable-length blocks, with customizable key sizes, utilizing modular arithmetic operations. The Enhanced RC5 (ERC5) algorithm is an extension of RC5, integrating the PKCS#7 padding method to adapt to various data sizes more effectively, enhancing encryption and decryption speeds in resource-constrained environments.

A. RC5 Algorithm

The RC5 algorithm, conceived by Ronald Rivest of RSA Security, stands as a symmetric key block cipher renowned for its simplicity, efficiency, and adaptability. At the heart of its operation lies a few fundamental principles, rendering it a preferred choice for a diverse array of cryptographic applications. RC5 operates on data blocks represented as sequences of fixed-size words, denoted by the parameter ω (typically 16, 32, or 64 bits), influencing the algorithm's block size and key length. Key expansion plays a pivotal role in RC5, where a set of round keys is generated from the original user-provided secret key. This process involves mixing the secret key with a constant value derived from the mathematical constant ϕ (the golden ratio). The algorithm employs a round function to execute encryption and decryption operations on data blocks, usually comprising modular arithmetic operations like addition, bitwise XOR, and rotations. RC5 conducts multiple rounds (typically 12, 16, or 20 rounds) of encryption or decryption operations. During each round, the data block undergoes a sequence of transformations utilizing the round keys generated during the key expansion phase. RC5 supports various block cipher modes, including Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB), enhancing its versatility in cryptographic protocols and applications [24].

Operationally, RC5 encrypts plaintext by dividing it into two halves (A and B), subjecting them to operations such as addition with round keys, bitwise XOR, and circular rotations, culminating in the generation of ciphertext blocks. Decryption, conversely, follows the reverse sequence of operations,

employing the same round keys generated during encryption to revert ciphertext blocks back to plaintext. Flexibility is a hallmark of RC5, permitting parameter customization such as word size, number of rounds, and key length to cater to specific security requirements and performance constraints. This adaptability, combined with its operational efficiency and simplicity, solidifies RC5's position as a prominent and extensively studied cryptographic algorithm in contemporary cryptography [26], [27].

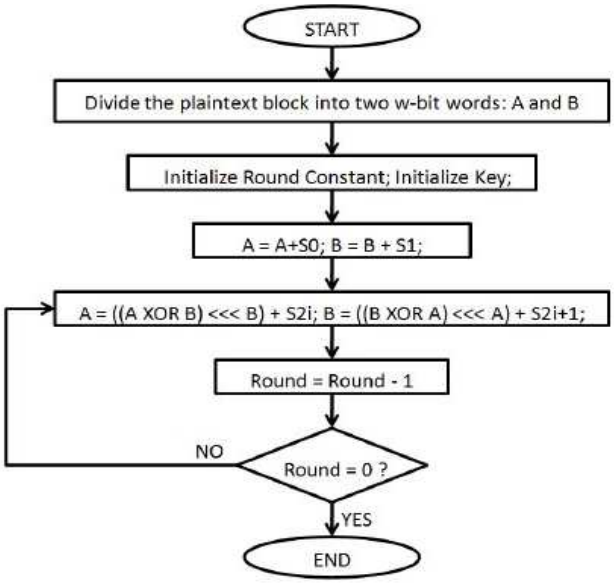

Fig. 1 Flow chart of RC5 algorithm [28].

The RC5 algorithm stands out for its ease of use and versatility as an infinite block cipher. Notably, RC5's reliance on rotating data, variable round numbers, and adjustable key sizes further distinguishes it among cryptographic algorithms. Its simplicity, attractiveness, and minimal RAM requirements contribute to its reputation as a faster and safer option across various applications. The structure of the RC5 algorithm is visually depicted in Figure 1.

B. Enhanced RC5 Algorithm

This research endeavors to tackle a significant drawback of the RC5 Algorithm: its relatively slow encryption speed. To address this challenge, the study proposes an enhanced approach aimed at improving the algorithm's efficiency, particularly in terms of encryption time. The overarching goal is to comprehensively evaluate the encryption time across a spectrum of data sizes while harnessing the computational power of multiple cores. Additionally, the study seeks to gauge the performance gains achieved by employing multiple cores for both encryption and decryption processes.

The initial step in this enhancement methodology involves partitioning input files into multiple blocks. Each block's size is meticulously determined to align with the number of threads available on the user's machine, ensuring optimal resource utilization. Subsequently, the encryption process is initiated for each block, and the resulting encrypted data is stored in assembler format for further analysis.

A critical precondition is established at the outset, mandating that the file length must precisely match 0.5 megabytes. Prior to encryption, another pivotal condition is assumed to ascertain whether the size of each block conforms to the requisite number of words. If this criterion is satisfied, the block is subjected to encryption without further ado. However, if additional processing, such as padding, is warranted, the encryption process is duly adjusted to accommodate these requirements. This intricate process flow is meticulously detailed and illustrated in Figure 2, providing a clear visual representation of the proposed encryption enhancement methodology.

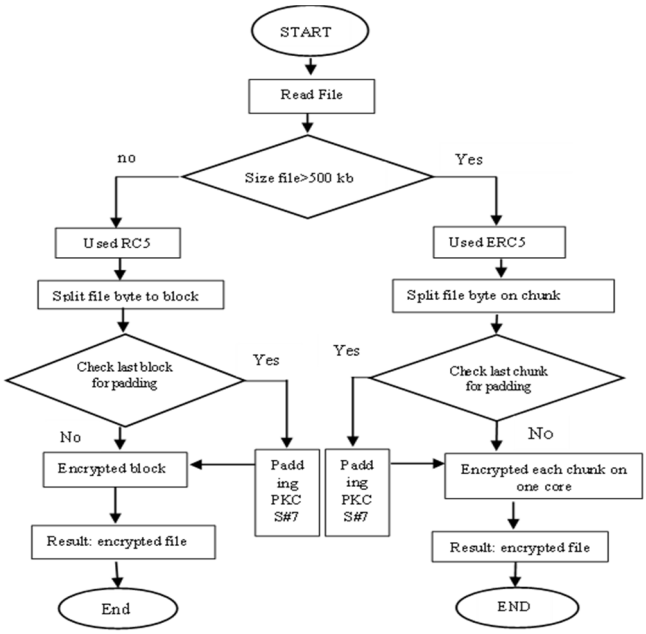

Fig. 2 Flowchart for the standard RC5 and the proposed ERC5 algorithm

In this study, the primary focus revolves around the efficient execution of the encrypted and decrypted functions, which are implemented through threads in both operations. However, the utilization of multiple threads introduces certain overheads, particularly during the thread preparation process. One notable issue is the potential waiting time incurred when threads are engaged in system programming processes. Additionally, the requirement to write results into separate files for each thread incurs additional overhead, as assembling the results into a single main file necessitates reading from multiple files. To mitigate these drawbacks, a modified framework has been developed, comprising the following key steps.

1) Enhanced Encrypted Function: Within the encrypted function, all variables and functions required by the threads for encryption and decryption processes are provided within the chunk itself. This ensures that each thread has access to all necessary resources without the need for sharing or contention with other threads.

2) Dictionary Manager: Instead of dealing with file reading and writing, a Dictionary Manager function from the multiprocessing library in Python is employed. This function facilitates storing results in RAM in a key-value format, thereby eliminating the need to write results to separate files. This significantly reduces overhead associated with file I/O operations.

3) Optimized Chunk Size: Rather than dividing the file into a number of chunks equal to the number of threads, the file is divided into larger chunks. The distribution of work

among threads is then handled by the multiprocessing library based on the available cores. This approach optimizes resource utilization and reduces waiting time by efficiently utilizing the available processing power.

By implementing these modifications, the proposed system effectively minimizes waiting time during thread preparation, streamlines result handling, and optimizes resource allocation. This results in enhanced efficiency and performance of the encryption and decryption processes, ultimately improving the overall performance of the RC5 algorithm.

III. Results and Discussion

In this section, the outcomes of the proposed enhancement approach are meticulously analyzed and scrutinized. Various performance metrics are assessed, including encryption and decryption speeds, throughput, and resource utilization. These results are compared against baseline measurements to ascertain the efficacy of the enhancement methodology. Additionally, the impact of utilizing multiple cores on the overall performance of the RC5 algorithm and ERC5 are thoroughly evaluated. After a number of experimental attempts, the following have been reached:

For larger file sizes, dividing the data into a smaller number of large pieces significantly accelerates both encryption and decryption processes, as demonstrated in Tables 2, 3, and 4. Notably, when the chunk size equals 20, the proposed enhancement exhibits smooth performance, leading to substantial improvements. Specifically, the enhancement achieved improvement rates of 50.90% and 64.18% for audio files of sizes 5000kb and 10000kb, respectively, as depicted in Table 8. Similarly, for image files of the same sizes, improvement rates of 56.84% and 46.97% are observed in Table 9, while a 59.90% enhancement is evident for a 10500kb data file in Table 10. Conversely, smaller files of sizes 1000kb and 2000kb did not benefit from this division, as indicated in Tables 2, 3, and 4.

TABLE II
ENCRYPTION TIME FOR AUDIO FILES FOR SINGLE AND MULTI-CORE (16 THREAD).

File name	File Size (kb)	No. chunk	Time(sec)/ single core	Time(sec)/ multi core
Audio1.mp3	1000	7	0.7129194736480713	0.5310373306274414
		20	0.6719334125518799	1.0443308353424072
Audio2.mp3	2000	7	1.4054770469665527	0.7903072834014893
		20	1.4010844230651855	1.1030058860778809
Audio3.mp3	5000	7	3.5060722827911377	2.1484169960021973
		20	3.451677083969116	1.6945405006408691
Audio4.mp3	10000	7	7.193301439285278	19.53294801712036
		20	6.934879541397095	2.5319912433624268

TABLE III
ENCRYPTION TIME FOR IMAGE FILES FOR SINGLE AND MULTI-CORE (16 THREAD).

File name	File Size(kb)	No. chunk	Normal encryption	Multi encryption
image1.jpg	2000	7	1.505091905593872	1.1136949062347412
		20	1.4239342212677002	1.1237168312072754
image2.png	3000	7	1.866837739944458	1.156327247619629
		20	2.043825626373291	1.15238618850708
image3.jpg	5000	7	3.735459089279175	2.282346248626709
		20	3.5905044078826904	1.549600601196289
Image4.tiff	10000	7	7.283079147338867	17.679017066955566
		20	7.187985897064209	2.5173842906951904

TABLE IV
ENCRYPTION TIME FOR DATA FILES FOR SINGLE AND MULTI-CORE (16 THREAD)

File name	Text file size (kb)	No. chunk	Classic (sec)	Enhanced(sec)
File1.pdf	800	7	0.9380898475646973	0.8603897094726562
		20	0.5765552520751953	1.0760319232940674
File2.pdf	1000	7	1.3273088932037354	0.9134235382080078
		20	0.7841880321502686	1.0756402015686035
File3.pdf	1500	7	1.9120330810546875	1.2361254692077637
		20	1.192474365234375	1.116013765335083
File4.pdf	10500	7	11.804539680480957	35.172152519226074
		20	7.532015800476074	2.9433376789093018

Conversely, for smaller file sizes, dividing the data into a larger number of smaller pieces yields favorable results. Tables 2, 3, and 4 illustrate that files of sizes 1000kb and 2000kb experience notable improvement rates of 25.51% and 43.76% in audio files, respectively, as shown in Table 8. Similarly, for image files of the same sizes, speedup improvements of 26.00%, 38.05%, and 38.90% are observed. However, larger files, such as 10000kb and 10500kb, did not benefit from this division, as it led to a significant increase in encryption and decryption times, as evidenced in Tables 8, 9, and 10, respectively.

After conducting numerous experimental trials, it has been deduced that the optimal strategy depends on the size of the file being processed. For larger file sizes, dividing the file into larger pieces proves to be more advantageous, leading to expedited encryption and decryption times. This observation is substantiated by the findings presented in below tables, particularly for files sized at 5000kb, 10000kb, and 10500kb. Conversely, for smaller file sizes, a smaller number of pieces

is preferred. This is because smaller files inherently require less processing time and dividing them into more pieces does not significantly reduce the encryption time. This incongruity with the research objective is evident in these tables, where files sized at 1000kb and 2000kb benefit from being divided into 7 pieces, while larger files, such as 10000kb, experience a substantial increase in encryption and decryption time when divided into fewer pieces.

Table (2) illustrates the encryption times for audio files of varying sizes and different piece counts. It is evident that larger files with a higher number of pieces' experience accelerated encryption times, aligning with expectations. However, for smaller files with larger piece counts, there is a notable increase in encryption time. Conversely, smaller files with fewer pieces exhibit a decrease in encryption time. This observation underscores the importance of considering both file size and piece count when optimizing encryption processes.

In Table (3) we note that file is encrypted of image type and of different size. As can be seen from the fact that large-sized and many-piece-counts have been accelerated from encryption time, and small-sized filaments also accelerated, but with tiny proportions. Only file 10000KB and cut-off number = 7 decreased encryption time.

In Table 4, the processing of text files of varying sizes is presented. Interestingly, the 10500kb file, when divided into numerous parts, exhibited quicker encoding times with the enhanced algorithm compared to the baseline. Conversely, smaller files proved to be inefficiently split into multiple parts, but benefited from fewer divisions. When decoding the results, two scenarios were considered: when the number of pieces is 7 and 20, respectively.

In Table 5, decryption results for audio files of varying sizes are presented. It is observed that larger files, such as those sized at 5000kb and 10000kb, demonstrate improved decryption times when split into a larger number of pieces compared to smaller files of 1000kb and 2000kb. This trend highlights the importance of optimizing the division of files into pieces based on their respective sizes to achieve efficient decryption performance.

TABLE V
DECRYPTION TIME FOR AUDIO FILES FOR SINGLE AND MULTI-CORE (16 THREAD).

File name	File Size (kb)	No. chunk	Classic(sec)	Enhanced(sec)
Audio1.mp3	1000	7	0.6048743724822998	0.5106022357940674
		20	0.631829023361206	1.0198681354522705
Audio2.mp3	2000	7	1.26908278465271	0.837064266204834
		20	1.2436838150024414	1.068835973739624
Audio3.mp3	5000	7	1.26908278465271	1.7774937152862549
		20	3.084986925125122	1.6945405006408691
Audio4.mp3	10000	7	6.295373916625977	19.858016967773438
		20	6.153172969818115	2.5319912433624268

TABLE VI
DECRYPTION TIME FOR IMAGE FILES FOR SINGLE AND MULTI-CORE (16 THREAD).

File name	File Size(kb)	No. chunk	Classic(sec)	Enhanced(sec)
image1.jpg	2000	7	1.3583087921142578	1.077310562133789
		20	1.2845025062561035	1.1402215957641602
image2.png	3000	7	1.866837739944458	1.115419864654541
		20	1.7862679958343506	1.1090426445007324
image3.jpg	5000	7	3.5318500995635986	2.0596723556518555
		20	3.245405912399292	1.6482517719268799
image4.tiff	10000	7	6.325125455856323	18.638334274291992
		20	6.41200852394104	2.5004501342773438

TABLE VII
DECRYPTION TIME FOR DATA FILES FOR SINGLE AND MULTI-CORE (16 THREAD).

File name	Text File Size(kb)	No chunk	Classic (sec)	Enhanced(sec)
File1.pdf	800	7	1.0161395072937012	0.7819924354553223
		20	0.4986586570739746	1.071523666381836
File2.pdf	1000	7	1.2481083869934082	0.8989977836608887
		20	0.6483018398284912	1.0348796844482422
File3.pdf	1500	7	1.6352155208587646	1.1935415267944336
		20	0.98117995262146	1.1402912139892578
File4.pdf	10500	7	11.542507886886597	35.368311405181885
		20	6.716280460357666	2.6931724548339844

In Table 6, decryption results for image files of various sizes are displayed. It is evident that larger files, such as those with sizes of 5000kb and 10000kb, experienced reduced decryption times when divided into a greater number of pieces. Conversely, smaller files exhibited benefits from fewer pieces. This underscores the importance of tailoring the division of files into pieces based on their specific sizes to optimize decryption efficiency.

Table 7 shows the decryption results for data files of varying sizes are presented. It is apparent that larger files derive benefits from being split into more pieces, resulting in reduced decryption times.

Conversely, for smaller files, accelerating decryption is achieved with fewer chunks. This highlights the importance of appropriately dividing files based on their sizes to optimize decryption efficiency. Both audio and text files of larger sizes

have demonstrated benefits from being divided into multiple pieces, resulting in reduced encryption times. Conversely, smaller files in both categories have shown improvements when divided into fewer pieces. Similarly, image files have exhibited considerable improvements in encryption time for larger files, with smaller but noticeable improvements observed for smaller files. The same trend is observed in decryption, where dividing files into fewer pieces is beneficial for reducing time, although the extent of improvement varies across file types.

Two sets of classes and programs have been meticulously developed: one for the classic RC5 algorithm and another for the enhanced version. The experimentation was conducted on a laptop equipped with an 11th Gen Intel(R) Core (TM) i7-11800H @ 2.30GHz processor and 16.0 GB of RAM. Both sets of programs were implemented in the Python programming language. The primary objective of this study is to assess the encryption speed of each algorithm across various file sizes and media types, with the aim of maximizing their performance. To quantify the performance, the throughput of each encryption scheme is calculated by dividing the total plaintext size in kilobytes by the total encryption time in seconds. It is worth noting that the power consumption of the encryption method tends to decrease as the throughput increases [23]. In the following subsections the performance evaluation of encryption/decryption with the throughput are discussed in detail.

A. Encryption and Decryption performance

To evaluate the encryption and decryption performance of both algorithms, Eq (1) is employed, which takes into account the implementation of the RC5 algorithm in both single and multi-core processes [12]. Additionally, the PKCS#7 padding scheme is utilized to check the last block of the file [29].

$$\text{Performance} = ((r - p) / r) * 100 \quad (1)$$

where:
r = performance for algorithm in single core.
p = performance for algorithm in multi-core.
The simulation made use of a 64-bit size password XORed with a 128-bit key size and 12 round.

B. Throughput

The throughput of the RC5 algorithm can be calculated by dividing the total plaintext size (in kilobytes) by the total encryption or decryption time (in seconds). This yields a measure of the amount of data processed per unit of time. The formula for throughput (T) can be expressed as [30]:

$$T=(S/t) \quad (2)$$

where: *S* is the total plaintext size in kilobytes.
t is the total encryption or decryption time in seconds.
Throughput is typically measured in kilobytes per second (KB/s).

In Table (8), the encryption of audio files is presented across various sizes and numbers of pieces. The table illustrates that encryption speed is notably improved for larger documents, while smaller files benefit from being split into smaller pieces. Notably, the 5000kb file demonstrates benefits from both types of partitions. Additionally, a comparison of encryption speeds for audio files between the classic RC5 algorithm and the enhanced RC5 algorithm is provided in the table below and Figure 3.

TABLE VIII
AUDIO FILES ENCRYPTION TIME AND SPEED PERFORMANCE

File name	File size (kb)	chunk No.	time(sec)/ single core	time(sec)/ multi-core	speedup rate
Audio1.mp3	1000	7	0.7129	0.5310	25.51%
		20	0.6719	1.0443	-55.42%
Audio2.mp3	2000	7	1.4054	0.7903	43.76%
		20	1.4010	1.1030	21.27%
Audio3.mp3	5000	7	3.5060	2.1484	38.72%
		20	3.4516	1.6945	50.90%
Audio4.mp3	10000	7	7.1933	19.5329	171.54%
		20	6.9348	2.4834	64.18%

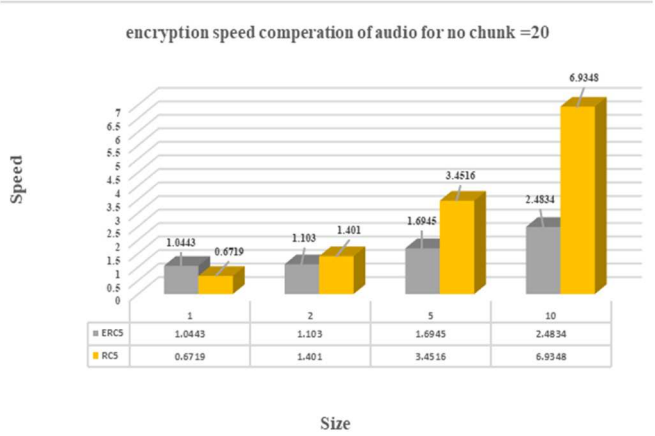

Fig. 3 Encryption speed comparison for audio file

Table 8 compares encryption speed rates between Enhanced RC5 and Classic RC5 across various file sizes and chunk counts. Enhanced RC5 exhibits faster encryption, especially for small files with low chunk counts and large files with high chunk counts. For example, for large files (5000kb and 10000kb) with 20 chunks, Enhanced RC5's average encryption time reduces significantly compared to Classic RC5. Similarly, for small files (1000kb and 2000kb) with 7 chunks, Enhanced RC5 demonstrates quicker encryption. Overall, Enhanced RC5 proves more efficient in encrypting audio data compared to Classic RC5. The encryption speed comparison between the classic RC5 algorithm and the enhanced RC5 algorithm for image files is presented in the following table and Figure 4.

TABLE IX
ENCRYPTION SPEED COMPARISON FOR IMAGE FILE.

File name	File size (kb)	No. chunk	Classic (sec)	Enhanced (sec)	Speedup rat
Image1.jpg	2000	7	1.5050	1.1136	26.00%
		20	1.4239	1.1237	21.08%
Image2.png	3000	7	1.8668	1.1563	38.05%
		20	2.0438	1.1523	43.61%
Image3.jpg	5000	7	3.7354	2.2823	38.90%
		20	3.5905	1.5496	56.84%
Image4.tiff	10000	7	7.2830	17.6790	142.74%
		20	7.1879	2.5173	64.97%

In Table 9, we see that the enhanced RC5 algorithm achieves higher efficiency in encrypting image data compared to the classic RC5 algorithm. The encryption time decreases for both large files with high chunk counts and small files with low chunk counts. For instance, for large files (5000kb and

10000kb) with 20 chunks, the average encryption time decreases significantly from 10.7784 seconds to 4.0669 seconds in the enhanced RC5 algorithm. Similarly, for small files (1000kb and 2000kb) with 7 chunks, the average encryption time decreases from 3.3718 seconds to 2.2699 seconds. Overall, the throughput also improves in the enhanced RC5 algorithm across both file sizes and chunk counts.

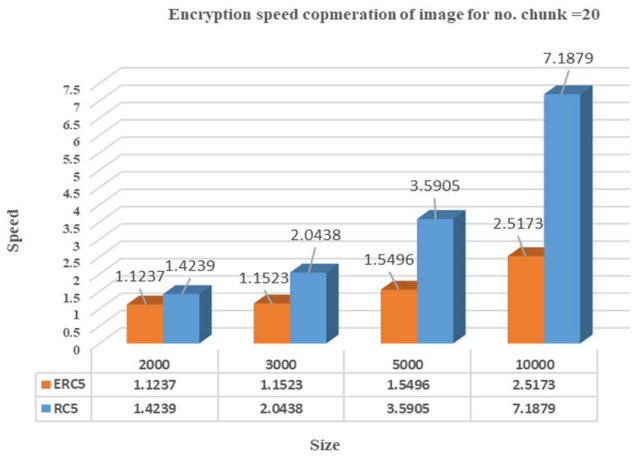

Fig. 4 Encryption speed comparison for image file.

Table 10 and Figure 5 present the encryption speed comparison between the classic RC5 algorithm and the enhanced RC5 algorithm for image files. The enhanced RC5 algorithm consistently demonstrates faster encryption speeds across different file sizes and chunk counts compared to the classic RC5 algorithm. Notably, encryption times decrease for both large files with high chunk counts and small files with low chunk counts, leading to improved throughput in the enhanced RC5 algorithm. Overall, these findings highlight the enhanced efficiency of the RC5 algorithm in encrypting image data.

TABLE X
ENCRYPTION SPEED COMPARISON FOR FILE DATA TYPE

File name	Text File Size (kb)	No. chunk	Classic (sec)	Enhanced (sec)	Speedup rate
File1.pdf	800	7	1.0161	0.7819	23.04%
		20	0.4986	1.0715	114.90%
File2.pdf	1000	7	1.2481	0.8989	27.97%
		20	0.6483	1.0348	-59.61%
File3.pdf	1500	7	1.6352	1.1935	27.01%
		20	0.9811	1.1402	-16.21%
File4.pdf	10500	7	11.5425	35.3683	206.41%
		20	6.7162	2.6931	59.90%

In Table 10, the enhanced RC5 algorithm shows increased enhancement rates and reduced encryption times, particularly for large files with high chunk counts and small files with low chunk counts. For instance, for a large file (10500kb) with 20 chunks, encryption time decreases from 6.7162 seconds in the classic algorithm to 2.6931 seconds in the enhanced RC5 algorithm, resulting in a higher throughput of 3,898.85 KB/s compared to 1,563.38 KB/s in the classic algorithm. Similarly, for small files (800kb and 1000kb) with 7 chunks, encryption time decreases from 2.2642 seconds to 1.6808 seconds, leading to an improved throughput of 1,070.91 KB/s compared to 794.98 KB/s in the classic algorithm. These results affirm the enhanced efficiency of the RC5 algorithm in encrypting image data.

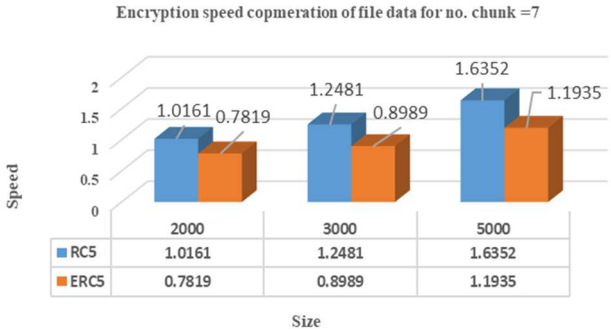

Fig. 5 Encryption speed comparison for file data.

Table 5 and Figure 6 illustrate the decryption speed comparison between the classic RC5 algorithm and the enhanced RC5 algorithm. The decryption process is crucial in assessing the efficiency and performance of cryptographic algorithms, especially in real-world applications where data integrity and security are paramount. In Table 5, we observe the decryption times for various file sizes and chunk counts for both the classic and enhanced RC5 algorithms. The results demonstrate that the enhanced RC5 algorithm consistently achieves faster decryption speeds compared to the classic RC5 algorithm across different scenarios.

For instance, consider the decryption time for large files (5000kb and 10000kb) with 20 chunks. In the classic RC5 algorithm, the decryption time is notably higher compared to the enhanced RC5 algorithm. Similarly, for small files (1000kb and 2000kb) with 7 chunks, the enhanced RC5 algorithm exhibits quicker decryption times compared to the classic RC5 algorithm. Figure 6 provides a visual representation of the decryption speed comparison, allowing for a more intuitive understanding of the performance differences between the two algorithms across different file sizes and chunk counts.

TABLE XI
DECRYPTION SPEED COMPARISON FOR AUDIO FILE

File name	Text File Size(kb)	No. chunk	Classic (sec)	Enhanced (sec)	Speedup rate
Audio1.mp3	1000	7	0.6048	0.51060	15.57%
		20	0.6318	1.0198	- 61.41%
Audio2.mp3	2000	7	1.2690	0.8370	34.04%
		20	1.2436	1.0688	14.05%
Audio3.mp3	5000	7	3.0923	1.7774	42.52%
		20	3.0849	1.6945	45.07%
Audio4.mp3	10000	7	6.2953	19.8580	-215.44
		20	6.1531	2.5319	58.85%

In Table 11, the decryption speed comparison between Enhancement RC5 and Classic RC5 is presented for various file sizes and chunk counts. Across all scenarios, we observe an enhancement in decryption speed and a decrease in encryption time, particularly for small files with low chunk counts and large files with high chunk counts.

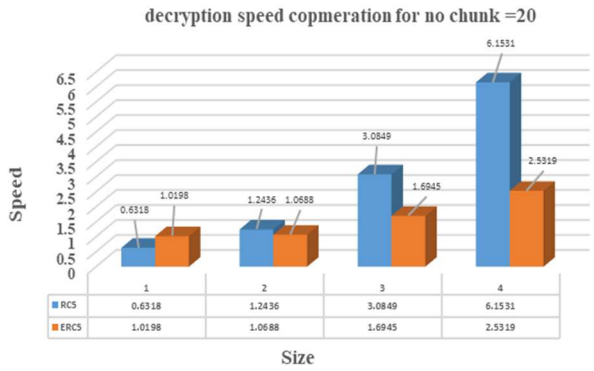

Fig. 6 Decryption speed comparison for audio file.

Figure 6 depicts the decryption speed comparison for audio files. It likely presents a visual representation of the decryption times for different scenarios, such as various file sizes and possibly different chunk counts, comparing the performance of the classic RC5 algorithm with the enhanced RC5 algorithm. The graph may show how the decryption times vary across these different scenarios, highlighting any improvements or differences between the two algorithms.

IV. Conclusion

This study proposes an efficient enhancement technique, the ERC5 algorithm, aimed at accelerating encryption speeds. By leveraging multiple cores for computation instead of relying solely on a single core, the ERC5 algorithm demonstrates significant improvements in encryption and decryption times across various data types and sizes. The algorithm covers audio, image, and file data types, ensuring comprehensive applicability. Additionally, the integration of the PKSC#7 padding algorithm facilitates data block padding before encryption, ensuring optimal encryption performance. Evaluation metrics such as encryption time and throughput attest to the enhanced efficiency of the ERC5 algorithm. Experimental results highlight the superiority of ERC5 over the Classic RC5 algorithm. Notably, encryption improvement rates reach 50.90% and 64.18% for audio files, 56.84% and 46.97% for image files, and 59.90% for large files with multiple pieces in the data file category. Moreover, even for smaller files, significant enhancements are observed, with encryption improvement rates of 25.51% and 43.76% for audio files, 26.00%, 38.05%, and 38.90% for image files, and 27.97% for data types. Overall, the enhanced RC5 algorithm demonstrates higher encryption average times and superior throughput compared to the classic RC5 algorithm, validating its effectiveness in achieving faster encryption speeds across diverse data types and sizes.

References

[1] Y. A. Birgani, S. Timarchi, and A. Khalid, "Ultra-lightweight FPGA-based RC5 designs via data-dependent rotation block optimization," Microprocess. Microsyst., vol. 93, p. 104588, 2022. doi:10.1016/j.micpro.2022.104588.

[2] P. Velmurugan, S. S. Sridhar, and E. Gotham, "An advanced and effective encryption methodology used for modern IoT security," *Mater. Today Proc.*, vol. 81, pp. 389–394, 2023. doi:10.1016/j.matpr.2021.03.424.

[3] M. Y. Hasan and D. J. Kadhim, "Efficient energy management for a proposed integrated internet of things-electric smart meter (2IOT-ESM) system," *J. Eng.*, vol. 28, no. 1, pp. 108–121, 2022, doi:10.31026/j.eng.2022.01.08.

[4] M. M. Abed and M. F. Younis, "Developing load balancing for IoT - Cloud computing based on advanced firefly and weighted round robin algorithms," *Baghdad Sci. J.*, vol. 16, no. 1, pp. 130–139, 2019, doi:10.21123/bsj.2019.16.1.0130.

[5] E. B. Villanueva, R. P. Medina, and B. D. Gerardo, "An enhanced RC5 (ERC5) algorithm based on simple random number key expansion technique," in *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, IEEE, 2018, pp. 134–138, doi:10.1109/iscaie.2018.8405458.

[6] Y. M. Hameed and N. H. M. Ali, "Enhanced RC5 Key Schedule Using One-Dimensional Cellular Automata for Audio File Encryption," *Iraqi J. Sci.*, pp. 388–401, 2019, doi: 10.24996/ijs.2019.60.2.19.

[7] R. Shahzadi, S. M. Anwar, F. Qamar, M. Ali, and J. J. Rodrigues, "Chaos based enhanced RC5 algorithm for security and integrity of clinical images in remote health monitoring. IEEE Access, 7: 52858-52870." 2019, doi: 10.1109/access.2019.2909554.

[8] A. Soboń, M. Kurkowski, and S. Stachowiak, "Complete SAT based Cryptanalysis of RC5 Cipher," *J. Inf. Organ. Sci.*, vol. 44, no. 2, pp. 365–382, 2020, doi: 10.31341/jios.44.2.10.

[9] T. Zhovnovach, A. Sagun, V. Khaidurov, H. Martyniuk, and T. Scherbak, "Modification of RC5 Cryptoalgorythm for Electronic Data Encryption Systems.," *Ukr. Sci. J. Inf. Secur.*, vol. 25, no. 3, 2019.19, doi: 10.18372/2225-5036.25.14458.

[10] M. B. Qureshi *et al.*, "Encryption techniques for smart systems data security offloaded to the cloud," *Symmetry (Basel).*, vol. 14, no. 4, p. 695, 2022. doi:10.3390/sym14040695.

[11] J. B. Awotunde, A. O. Ameen, I. D. Oladipo, A. R. Tomori, and M. Abdulraheem, "Evaluation of four encryption algorithms for viability, reliability and performance estimation," *Niger. J. Technol. Dev.*, vol. 13, no. 2, pp. 74–82, 2016. doi:10.4314/njtd.v13i2.5.

[12] E. B. Villanueva, B. D. Gerardo, and R. P. Medina, "Implementation of the Enhanced RC5 (ERC5) Algorithm in an RFID-based Payment Scheme," in *Proceedings of the 2nd International Conference on Business and Information Management*, 2018, pp. 6–10, doi:10.1145/3278252.3278261.

[13] H. Khazaei, C. K. Wei, and S. K. Bin Sulaiman, "The Application of IoT on Consumer Behaviour and Decision-Making Process," *Int. J. Adv. Sci. Comput. Eng.*, vol. 4, no. 3, pp. 188–192, 2022. doi:10.30630/ijasce.4.3.93.

[14] B. A. Hameedi, M. M. Laftah, and A. A. Hattab, "Data Hiding in 3D-Medical Image.," *Int. J. Online Biomed. Eng.*, vol. 18, no. 3, 2022. doi:10.3991/ijoe.v18i03.28007.

[15] N. A. Ali, A. M. S. Rahma, and S. H. Shaker, "Multi-level encryption for 3D mesh model based on 3D Lorenz chaotic map and random number generator.," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, 2022. doi:10.11591/ijece.v12i6.pp6486-6495.

[16] D. M. Satyavathi, B. V. Mala, C. V. Vamsi, C. C. Chiranjeevi, and C. N. Neeraj, "Real-Time Hidden Data Transmission Using Lora," *Int. J. Adv. Sci. Comput. Eng.*, vol. 4, no. 2, pp. 130–137, 2022. doi:10.36713/epra11016.

[17] M. Hadi and M. Najm, "Introduction to ChatGPT: A new revolution of artificial intelligence with machine learning algorithms and cybersecurity," *Sci. Arch*, vol. 4, pp. 276–285, 2023. doi:10.47587/sa.2023.4406.

[18] L. Faridah, A. U. Rahayu, R. N. Shopa, H. Sulastri, N. Hiron, and F. M. S. Nursuwars, "Caribi Mobile Application Based on Radio Frequency Identification (RFID) for Internet of Things (IoT)," *Int. J. Adv. Sci. Comput. Eng.*, vol. 4, no. 3, pp. 203–209, 2022. doi:10.30630/ijasce.4.3.98.

[19] N. A. Taha, Z. Qasim, A. Al-Saffar, and A. A. Abdullatif, "Steganography using dual tree complex wavelet transform with LSB indicator technique," *Period. Eng. Nat. Sci.*, vol. 9, no. 2, pp. 1106–1114, 2021. 1106. doi:10.21533/pen.v9i2.2060.

[20] A. Hussein and T. B. Kareem, "Proposed Parallel Algorithms to Encryption Image Based on Hybrid Enhancement RC5 and RSA," in *2019 International Engineering Conference (IEC)*, IEEE, 2019, pp. 101–106, doi: 10.1109/IEC47844.2019.8950593.

[21] D. Prakash, "FPRC5: Improving the Data Security using Format PreservedRC5 Block Cipher Algorithm in Credit Card Application," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, 2021, pp. 775–785, doi:10.1109/icces51350.2021.9488936.

[22] A. S. Jamil and A. M. S. Rahma, "Image Encryption Based on Multi-Level Keys on RC5 Algorithm," *Int. J. Interact. Mob. Technol.*, vol. 16, no. 17, pp. 101–115, 2022, doi: 10.3991/ijim.v16i17.34335.

[23] O. S. Faragallah, A. I. Sallam, and H. S. El-Sayed, "Visual protection using rc5 selective encryption in telemedicine," *Intell. Autom. Soft*

Comput., vol. 31, no. 1, pp. 177–190, 2022, doi:10.32604/IASC.2022.019348.

[24] R. A. Muhajjar, N. A. Flayh, and M. Al-Zubaidie, "A Perfect Security Key Management Method for Hierarchical Wireless Sensor Networks in Medical Environments," *Electron.*, vol. 12, no. 4, pp. 1–20, 2023, doi: 10.3390/electronics12041011.

[25] H. M. Yousif and S. M. Hameed, "Review of Challenges and Solutions for Genomic Data Privacy-Preserving," *Iraqi J. Sci.*, pp. 4729–4746, 2023. doi:10.24996/ijs.2023.64.9.35.

[26] A. Abidi, A. Sghaier, M. Bakiri, C. Guyeux, and M. Machhout, "Statistical analysis and security evaluation of chaotic RC5-CBC symmetric key block cipher algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 10, pp. 533–538, 2019, doi: 10.14569/ijacsa.2019.0101070.

[27] V. K. Raj, H. Ankitha, N. G. Ankitha, and L. S. Kanthi Hegde, "Honey encryption based hybrid cryptographic algorithm: A fusion ensuring enhanced Security," *Proc. 5th Int. Conf. Commun. Electron. Syst. ICCES 2020*, no. Icces, pp. 490–494, 2020, doi: 10.1109/icces48766.2020.09137849.

[28] H.S. Gill, " Selection of Parameter 'r' in RC5 Algorithm on the basis of Prime Number, Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06 – 08 March, 2014 doi:10.1109/raecs.2014.6799519.

[29] H. Alamleh, M. Gogarty, D. Ruddell, and A. A. S. AlQahtani, "Securing the invisible thread: A comprehensive analysis of BLE tracker security in Apple AirTags and Samsung SmartTags," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2401.13584.

[30] B. A. Buhari,A. A. Obiniyi, k. Sunday, S. Shehu,"Performance Evaluation of Symmetric Data Encryption Algorithms : AES and Blowfish," vol. 6272, pp. 407–414, 2019, doi:10.36348/SJEAT.2019.v04i10.002.