Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 2, pp. 753–776
DOI : 10.47974/JIOS-2114

Enhanced ChaCha20 for next-generation cryptography : A secure and efficient lightweight encryption scheme

Mohammad Ubaidullah Bokhari [†]
Shahnwaz Afzal *
Aditya Varshney [§]
Department of Computer Science
Aligarh Muslim University
Aligarh 20200
Uttar Pradesh
India

Imran Khan [‡]
School of Computer Science
Artificial Intelligence Cluster
University of Petroleum and Engineering Studies
Dehradun 248007
Uttrakhand
India

Zaynah Khan [@]
Department of Computer Science
Villanova University
Pennsylvania 19085
United States

Abstract

This work introduces a modified ChaCha20 encryption algorithm, an improved variation of the ChaCha20 family, designed to provide superior security, enhanced efficiency, and optimal resource use for contemporary cryptographic applications. The proposed model enhances diffusion by employing a non-linear transformation function, ensuring that even

[†] *E-mail:* **mubokhari@gmail.com**
* *E-mail:* **shahafzalamu@gmail.com (Corresponding Author)**
[§] *E-mail:* **adityavarshney304@gmail.com**
[‡] *E-mail:* **imran.khan1@ddn.upes.ac.in**
[@] *E-mail:* **zkhan02@villanova.edu**

minor alterations in input yield highly unpredictable outcomes. A distinctive permutation layer is incorporated following each set of quarter rounds to augment unpredictability and state mixing, fortifying resistance to statistical and differential attacks. The method employs a 20-round structure, alternating between column and diagonal quarter rounds, and incorporates an efficient state finalization technique that ensures security without significantly increasing computational demands. The proposed method demonstrates superior performance compared to ChaCha8, ChaCha12, ChaCha20, and SuperChaCha20, as evidenced by an improved avalanche effect (50.8462%), heightened Shannon entropy (0.99995), and reduced encryption time, thereby surpassing them in both security and efficiency. Furthermore, the model demonstrates minimal energy consumption (0.55 J) and improved memory efficiency (6 KB), making it suitable for IoT security, lightweight cryptographic applications, and real-time data encryption. The improved ChaCha20 Algorithm, characterized by rapid encryption, strong security, and efficient resource use, is a reliable cryptographic solution for contemporary secure communication networks.

Subject Classification (2020): *94A60, 68P25, 68M25.*

Keywords: *ChaCha20, Stream cipher, IoT, Entropy, Encryption time.*

1. Introduction

The Internet of Things (IoT) represents a significant technological leap, merging the physical and digital worlds. Fundamentally, IoT, as discussed in the opening segment, enables devices to sense, process, and act on data with minimal to no human involvement, thanks to the intelligence provided by software and the networking capabilities of everyday devices. This interconnected ecosystem transforms everyday objects, ranging from home appliances to industrial machines, into "smart" devices that transmit data over the internet. For instance, in a smart home scenario, IoT devices can automatically control lighting, security, and climate based on the user's preferences or environmental conditions. Health-related wearables continuously monitor vital signs and alert medical personnel if any alarming conditions occur. With IoT systems, it is easy to track how the equipment is performing in real-time and what needs maintenance, helping businesses save costs and improve industry productivity. Smart cities are enabled by IoT technology, which leverages connected devices to control traffic, improve waste management, and track environmental indicators. This improves its citizens' living standards and fosters green development by efficiently utilizing resources[1].

Nevertheless, the high penetration of IoT presents several significant challenges, especially in terms of data security and privacy. In a world of

billions of connected devices transmitting sensitive information, and hopefully intended only for discreet recipients, defending against cyber attacks becomes a top priority. The core of IoT is to transform how we interact with the environment by enabling intelligent communication between devices. It has applications across various industries and is paving the way towards a more automated, streamlined, and data-oriented world. Nevertheless, solving security and privacy threats is imperative to expand and accept IoT technology. There is so much to do amazing things with IoT, and right now, there is an opportunity to gain experience to help improve our lives and industries. IoT devices are resource-constrained and often handle private or confidential information in IoT environments. In such scenarios, their security and confidentiality become among the most significant challenges. The devices involved in IoT include sensors, cameras, and smart appliances, which are typically installed in critical areas such as healthcare, industrial control, and urban infrastructure. As these devices collect, process, and transmit sensitive data, such as health records, industrial telemetry, or personal data, securing this data from unauthorized access and ensuring its confidentiality becomes a requirement. Most IoT devices have limited processing power, memory, and battery life, which thus puts constraints on their security capabilities[2]. Advanced security mechanisms, such as high-end encryption and authentication protocols, consume high-performance resources that conventional IoT implementations cannot justify. For example, using traditional cryptographic algorithms for data encryption and secure communication will lead to critical power drain on low-power devices or significant processing delays. This means that lightweight security protocols must be tailored to the needs of IoT, striking a balance between security, performance, and power efficiency[3].

In addition, as IoT networks could comprise thousands of devices, scaling up the security measures throughout the network with significant computational overhead or latency is difficult[4]. Lightweight cryptographic algorithms, such as Grain-80, Grain-128, Trivium, Salsa, and ChaCha (8/12/20 rounds), which do not pose heavy computational burdens, have been developed to address the constraints of the IoT. These lightweight algorithms support data encryption, integrity, and authentication within the limited resources of IoT devices. Grain-80 and Grain-128 work well in constrained settings but have key size and initialization issues. While Trivium excels in hardware configurations, it struggles with software optimization. Although the Salsa and ChaCha families are suitable for software optimization, they are vulnerable to

various techniques due to their fewer rounds and weak diffusion properties. To overcome the limitations of current lightweight algorithms, this paper proposes a secure modified ChaCha20 algorithm for IoT devices.

The significant contributions of the proposed works are as follows:

1. **Enhanced Non-Linearity:** Introduced a non-linear multiplicative operation to the quarter-round function, which improves resistance to differential and linear cryptanalysis by increasing unpredictability in the state transitions.
2. **Optimized Permutation Layer:** Refined the state permutation to ensure faster and uniform diffusion across state words, reducing vulnerabilities and enhancing cryptographic strength.
3. **Parallel Processing Capabilities:** Integrated support for multithreaded block-level processing, enabling faster encryption and decryption for large datasets. Significantly increases throughput, making the algorithm suitable for real-time and high-performance applications.
4. **Resistance to linear and differential attacks:** To assess the security of the suggested Ours ChaCha20 cipher, a test against the differential cryptanalysis attack was carried out. Observations are recorded based on the randomness and passing sequences using the NIST Statistical Test Suite (NSTS), where test uniformity and randomness are verified.
5. **Improved Avalanche Effect:** Achieved an avalanche effect of 50.8462%, outperforming the original ChaCha variants, which achieved ~40–45% avalanche effect.

The rest of the paper is categorized in different Sections, which are as follows:

Section 2 discusses the related work, followed by Section 3, which discusses the proposed work. Section 4 briefly discusses the results, and finally, Section 5 concludes with a conclusion and future direction.

2. Related Work

Various stream cipher algorithms, such as ChaCha8, ChaCha12, and ChaCha20, have been extensively used for their speed and security. However, these algorithms still face limitations regarding computational

efficiency, diffusion properties, and resistance to modern cryptanalysis. This section reviews existing research on encryption techniques and discusses their strengths and weaknesses. Maolood et al. [5] proposes a lightweight stream cipher method for video encryption based on a hybrid chaotic map and ChaCha20 algorithm. The technique used two chaotic maps to generate keys and a symmetric scheme for encrypting and decrypting data. Tested on various video samples, the method has presented higher security and lower computation time than state-of-the-art encryption methods, thus targeting the growing demand for secure visual resource-constrained devices. Weinstein et al. [6] proposed a new mechanism of access control to address security and privacy issues. The proposed mechanism incorporates authentication, access control, and encryption mechanisms. It generates data, clusters, encrypts, and hashes before transferring it to the server. CPS then provides the requested data to the authenticated user, showcasing its effectiveness with experimental results. This approach challenges current centralized solutions and ensures data integrity. Goll and Gueron [7] discusses software optimization for ChaCha, utilizing the new AVX2 architecture's vectorization capabilities to speed up encryption on x86_64 processors. It also demonstrates applying vectorization for future AVX512 architecture, which brings significant performance gains. Najm et al. [8] proposed a lightweight image encryption approach based on the Chacha20 key generator and the 16-round Serpent. The cipher provides resistance against known and chosen plaintext attacks and gives excellent unpredictability, security, and intense sensitivity for key variations. Ghafoori and Miyaji [9] analyzes ChaCha20, a software stream cipher, and presents a divide-and-conquer approach to break 6-round ChaCha256 faster. It also introduces a toy version of Chacha20 with a 32-bit secret key to test theoretical estimates. The study further illustrates that the success probabilities of PNB-based differential attacks can be estimated more accurately. Mahdi et al. [10] proposed a new stream cipher, Super ChaCha20, to improve security on IoT devices. The algorithm focuses on rotation, random values, and changing inputs. Super ChaCha20 requires 2^{512} probable keys to break by brute-force attack and has passed the five benchmarks and NIST tests. At et al. [11] presents a high-performance and area-efficient hardware structure for the ChaCha20 stream cipher, emphasizing addition modulo 2^{32}. This is achieved by using the sparse parallel prefix adder for this operation, thereby reducing the critical path delay. The structure uses resource sharing, minimal hardware, and four registers to store intermediate round functions and pipeline to break the crucial path delay.

Kebande [12] proposes an Extended ChaCha20 (EChaCha20) stream cipher that enhances ChaCha20's resistance against cryptanalysis. It incorporates improved Quarter Round Functions QR-F and Add, Rotate, and XOR (ARX) operations on constants. The study uses the Bellare-Rogaway Model and Chosen Plaintext Attack to assess potential security weaknesses. Mahdi et al. [13] proposed lightweight image encryption, combining ChaCha20 symmetric stream cipher with a Hyperchaotic Map, which enhances privacy in telecommunications. It allows for resistance against brute force attacks and defense against statistical cracking and image insecurity based on histogram correlation and entropy. Raza [14] proposed a hybrid encryption algorithm with the AES128-bit cipher, ChaCha20, and GOST to improve digital image transmission security. The algorithm, evaluated using metrics such as NPCR, UACI, PSNR, and key space analysis, shows pixel value randomization improves security effectiveness and increases susceptibility to input changes. It offers real-time performance, high key space, and low brute-force proneness, making it an advanced and satisfying method for digital image encryption. Sobti and Ganesan [15] introduce introduces a modified ChaCha20 Core (MCC) as a new design to improve diffusion in stream ciphers and cryptographic hash functions. By analyzing the quarter-round structures of the Salsa and ChaCha20 cores, the authors identified some diffusion and computational efficiency constraints, which the MCC overcomes by implementing optimized rotation, addition, and XOR operations, which ensure that all input words have an equal effect on the output. Frimpong et al. [16] presents the Secret Key 4 Optimization Algorithm (SK4OA), which uses a unique nonlinear runtime pattern. The algorithm uses Cousin Primes, Pseudo-Random Number Generators (PRNG), a Sliding Window Algorithm, the Greatest Common Divisor (GCD), and XOR operations to strengthen encryption and decryption. By employing non-linear execution times, SK4OA reduces the possibility of predictable runtimes and offers a strong defense against brute force attacks, two common flaws in existing cryptographic techniques like RC4, Salsa20, and ChaCha20. From Table 1, it is evident that existing encryption schemes lack an optimal balance between security and computational efficiency. Therefore, the proposed work introduces an enhanced ChaCha20 variant with improved randomness, reduced encryption time, and better performance in lightweight cryptographic applications.

Table 1

Comparative analysis of the existing research

Reference	Technique Used	Purpose	Strength	Limitation
[3]	A cryptographic scheme using ChaCha20 for Cyber-Physical Systems (CPS).	Ensures data confidentiality and integrity while minimizing communication delays in automation networks.	- High cryptanalysis resistance. - Efficient event-based cryptography for CPS.	- The study does not specify limitations but highlights the challenges of traditional cryptographic schemes for automation.
[4]	Lightweight image encryption using ChaCha20 and Serpent.	Enhances security while reducing data size and processing speed, making it suitable for cloud computing and social networks.	- High security with an entropy value of 7.98. - 99.55% pixel difference rate, proving high randomness.	The study does not mention specific ChaCha20 + Serpent encryption limitations, focusing more on security analysis and performance evaluation.
[5]	FPGA-accelerated ChaCha20 implementation on a Cyclone V SoC.	Improves encryption speed and efficiency through hardware optimization.	Achieves 120.5 MiB/s, improving performance significantly. - Custom DMA is implemented for better data handling.	- The study does not specify limitations but emphasizes challenges in optimizing symmetric encryption for low-cost SoC platforms.
[6]	Performance comparison of AES, Blowfish, Twofish, Salsa20, and ChaCha20.	Evaluates encryption/ decryption speed and security for image encryption.	- ChaCha20 outperforms AES, offering higher speed. - The study provides insights into cryptographic algorithm selection for cybersecurity.	- Limited to five symmetric algorithms, excluding asymmetric encryption methods.

Contd...

[7]	Probabilistic Neutral Bits (PNB) cryptanalysis for ChaCha20 and Salsa20.	Enhances attack efficiency, making cryptanalysis faster.	- 2.27x faster for Salsa20, 5.39x faster for ChaCha20 than previous methods. - Reduces complexity in key recovery attacks.	- Limited by specific key/ block sizes and selective cryptanalysis methods.
[8]	ChaCha20 optimization using AVX2 and AVX512.	Improves processing speed and efficiency of ChaCha20 encryption.	Achieves 1.43 cycles/byte on a 4KB message using Intel Haswell. - Adapts functions for vectorized performance improvements.	- Limited to specific processor architectures (e.g., Intel AVX2/ AVX512).
[9]	EChaCha20: Enhanced ChaCha20 with stronger Quarter Round Functions (QR-F).	Improves resistance against differential cryptanalysis and chosen plaintext attacks.	Uses ARX operations for stronger security. Evaluated under the Bellare-Rogaway model.	No clear limitations are mentioned, but further testing is required for large-scale cryptographic deployment.
[10]	Lightweight video encryption using ChaCha20 and chaotic maps.	Provides high-speed encryption for video content.	Uses frame-wise encryption to reduce complexity. High entropy values show strong randomness and resistance to statistical attacks.	Limited focus on real-time streaming security.
[11]	Advanced ChaCha20 Algorithm with clustering-based security.	Enhances data privacy and security in Cyber-Physical Systems.	- 97.81% security improvement, surpassing traditional encryption methods.	- The study does not specify performance limitations.
[12]	AES + ChaCha20 +GOST hybrid encryption.	Enhances data confidentiality by combining multiple encryption techniques.	- High security through multi-layer encryption. - Resistant to cryptanalytic attacks.	- Increased computational overhead due to multiple encryption layers.

Contd...

[13]	ChaCha20 diffusion analysis for cryptographic hash functions.	Compares rotation constants and diffusion matrices for better security.	- MCC core improves diffusion in ChaCha encryption. - Over 1 million diffusion matrices tested for optimization.	- The study does not discuss computational overhead for real-time applications.
[14]	Security Key 4 Optimization Algorithm (SK4OA).	Strengthens data security in cloud computing with non-linear runtime patterns.	Uses Cousin Primes, PRNG, Sliding Window Algorithm, and XOR for encryption. - Reduces predictability in runtime, improving resistance to brute force attacks.	- The study does not specify performance benchmarks for real-world cloud environments.

3. The Proposed Method

The proposed work aims to enhance the ChaCha20 encryption algorithm by introducing novel non-linear operations, optimized state permutations, and support for related processing to improve both security and performance. The key is to address the limitations of featherlight cryptographic algorithms in terms of resistance to discrimination and direct cryptanalysis, randomness quality, and effectiveness in resource-constrained environments, such as IoT bias. Our proposal introduces critical improvements over the basic ChaCha20 by integrating significant advancements in addressing security vulnerabilities and performance limitations across various factors. The algorithm employs an Enhanced Non-Linear Transformation, which introduces a multiplicative non-linear operation that significantly enhances resistance against discrimination cryptanalysis and direct cryptanalysis, while improving randomness and achieving an advanced avalanche effect. The Permutation Subcaste has been optimized with a fine-tuned rearrangement pattern to ensure better bit prolixity across state words, thus reducing pungency and enhancing resistance against cryptanalytic attacks. It also includes similar Block Processing with a thread Pool Executor for contemporaneous encryption and decryption of multiple 64-byte blocks. It operates with advanced output, lower quiescence, and better scalability on high-performance systems and resource-constrained IoT bias. State Initialization ensures firm handling of keys, nonces, and timestamps to assist with state

freshness. Also, an Adaptive Padding Medium is present to guarantee a harmonious alignment of data that does not leak structural metadata. These concerted advancements enhance an algorithm that excels in cryptographic strength, effectiveness, and real-world robustness, making it especially suitable for secure communication in IoT environments and large-scale data encryption systems. The pseudocode of the proposed algorithms is given in the encryption algorithm to show the flow of the code

Encryption Algorithm

Input: Key(256 bits), Nonce (96 bits), Counter(32 bits) ,Plain text

Output: Ciphertext

1. procedure OURS_CHACHA_ENCRYPT(Key, Nonce, Counter, Plaintext, Rounds)
2. Initialize State = [Constants ‖ Key ‖ Counter ‖ Nonce]
3. OriginalState = State
4. for i = 1 → Rounds/2 do
5. Apply_QR(0, 4, 8, 12)
6. Apply_QR(1, 5, 9, 13)
7. Apply_QR(2, 6, 10, 14)
8. Apply_QR(3, 7, 11, 15)
9. Apply_QR(0, 5, 10, 15)
10. Apply_QR(1, 6, 11, 12)
11. Apply_QR(2, 7, 8, 13)
12. Apply_QR(3, 4, 9, 14)
13. x[b] = NonLinearOperation(x[b], x[a])
14. end for
15. NonLinearOperation
    x[b] = (x[b] × x[a]) mod 2^32
16. permutation layer
    State = [x[1], x[3], x[0], x[2], x[5], x[7], x[4], x[6], x[9], x[11], x[8], x[10], x[13], x[15], x[12], x[14]]
17. for i = 0 → 15 do
18. State[i] = (State[i] + OriginalState[i]) mod 2^32

```
19. end for
20.     Keystream = Serialize (State)
21.     for i = 1 → Length (Plaintext) / 64 do
22.         Ciphertext_Block[i] = Plaintext_Block[i] ⊕ Keystream
23.     end for
24.  return Ciphertext
25. end procedure
```

The Steps that are followed in the encryption process are given below for better understanding

Step 1: State Initialization

The state initialization of the proposed model begins with defining essential components that ensure security and uniqueness. The constants are fixed 32-bit values that contribute to the algorithm's structure and prevent predictable patterns. The key is a 256-bit secret, divided into eight 32-bit words, providing strong cryptographic strength against brute-force attacks. The unique 32-bit integer counter ensures that each encryption instance remains distinct, preventing keystream reuse. Additionally, the nonce, a 96-bit unique identifier split into three 32-bit words, further enhances security by ensuring that each encryption session remains unique, protecting against replay attacks, and ensuring randomness in the keystream generation.

Step 2: Quarter Round Transformation

The quarter-round function is a key component that updates four words in the state matrix. It applies a sequence of modular additions, XOR operations, and bit rotations:

Step 3: Non-Linear Operation

Unlike standard ChaCha20, this modified version introduces a non-linear transformation. This step enhances diffusion, ensuring that small input changes cause unpredictable output variations.

Step 4: Permutation Layer

A custom permutation layer is applied after each set of quarter rounds to improve bit randomness and eliminate potential correlations:

Step 5: Round Function

The algorithm executes 20 rounds, alternating between column and diagonal quarter rounds:

Column Rounds: (0,4,8,12), (1,5,9,13), (2,6,10,14), (3,7,11,15), Diagonal Rounds: (0,5,10,15), (1,6,11,12), (2,7,8,13), (3,4,9,14). These steps ensure every word in the state matrix interacts with multiple others, increasing resistance against differential attacks.

Step 6: State Finalization

The original state is added back to the transformed state. These are given below

State[i] = (State[i] + OriginalState[i]) mod 2^{32}. This reinforces security by preventing reversibility.

Step 7: Keystream Generation

The final state matrix is serialized into a 64-byte keystream block.

Step 8: Encryption Process

The encryption process begins by dividing the plaintext into 64-byte blocks, ensuring structured processing for efficient encryption. A keystream block is generated using the proposed model for each plaintext block. The encryption is then performed using the XOR operation, where each plaintext byte is XoRed with the corresponding byte of the keystream. This operation ensures that the ciphertext is indistinguishable from random noise while maintaining the reversibility required for decryption. The process repeats for all plaintext blocks, providing complete and secure encryption across the message. The decryption follows the same steps as encryption.

4. Result and Discussion

The proposed model in this exploration introduces substantial advancements to the standard ChaCha20 encryption system. The primary objects of these variations are perfecting security, adding computational effectiveness, and optimizing performance for IoT surroundings. This section evaluates the issues of the proposed variations compared to ChaCha variants and other featherlight cryptographic algorithms. The proposed model presents significant advancements over ChaCha8, ChaCha12, ChaCha20, and Super ChaCha by enhancing security, effectiveness, and rigidity for resource-constrained surroundings like IoT.

This section provides a detailed relative analysis of the proposed model's performance, security, and computational effectiveness against being variants, proving its superiority. For a better analysis of the proposed model, we have evaluated different metrics such as Encryption, Decryption time, Throughput , Shannon Entropy, and Avalanche effect on four data set of size 2^6, 2^8 ,2^{10}, and 2^{12}. Further evaluation of the NIST test has been done to check the method's effectiveness against statistical attacks.

5.1 *Encryption Time*

The encryption time is a critical metric in evaluating the performance of cryptographic algorithms, especially in real-time and resource-constrained environments. It determines how efficiently an algorithm can process data while maintaining security. In practical applications, such as IoT devices, secure communications, and cloud storage, encryption speed has a direct impact on system latency, power consumption, and overall efficiency. A slower encryption process can lead to bottlenecks in data transmission, making it unsuitable for real-time applications that require fast encryption and decryption cycles. The Encryption time complexity of the proposed model is O(n), as it was in ChaCha20. Introducing a permutation layer and a non-linear function does not affect the time complexity but increases the resistance against differential and linear attacks. The performance of the proposed model is evaluated against ChaCha20, ChaCha8, ChaCha12, and Super ChaCha based on encryption time for different input sizes. The results, as shown in Table 2 below, highlight the efficiency of the proposed model in terms of reduced encryption time. The proposed model outperforms ChaCha20, ChaCha8, ChaCha12, and Super ChaCha by demonstrating significantly lower encryption times across all input sizes. For small data sizes (64B - 256B), it achieves a 10x improvement over ChaCha20 and nearly 3x improvement over ChaCha8 and ChaCha12, ensuring rapid encryption with minimal computational overhead. As data size increases, the efficiency of the proposed model remains consistent, achieving a 5- 10x speed improvement over standard ChaCha20 algorithms. For 32KB of data, the proposed model encrypts in 12.0 ms, compared to 300 ms for ChaCha20 and 345.8 ms for Super ChaCha, indicating its superior performance. This significant reduction in encryption time is attributed to non-linear transformations, optimized permutation layers, and parallel processing, making it highly suitable for high-speed and real-time IoT applications. Figure 1 provides a visual representation of encryption time, offering a clear and intuitive

comparison of the proposed Modified ChaCha algorithm's performance against ChaCha20, ChaCha8, ChaCha12, and Super ChaCha. The graphical depiction highlights the significant reduction in encryption time achieved by the proposed model, especially for larger data sizes, where it consistently outperforms existing ChaCha variants

Table 2
Comparison of encryption time(ms) with other ciphers

Size	Our	ChaCha 20	ChaCha 8	ChaCha 12	Super ChaCha
2^6	**0.13**	1.4	0.63	0.80	1.2
2^8	**0.21**	3.1	1.4	2.4	3.5
2^{10}	**0.67**	10.8	4.5	6.5	11.2
2^{12}	**1.9**	37.36	19.83	26.02	44.7
2^{15}	**12.0**	300	167.2	202.11	345.8

5.2 *Throughput*

Throughput is a critical parameter in evaluating the efficiency of cryptographic algorithms, given by equation 1, as it determines how quickly data can be processed while ensuring security. Higher throughput values indicate better performance, making an encryption algorithm more suitable for real-time applications such as IoT. The proposed model significantly outperforms ChaCha20, ChaCha8, ChaCha12, and Super ChaCha in terms of throughput across all input sizes, as shown in Table 3 and the visual representation in Figure 2. It is observed that for small data sizes (2^6 = 64B), the proposed model achieves 4923.3076 Mbps, whereas ChaCha20, ChaCha8, and Super ChaCha achieve only 101.5873 Mbps, 80 Mbps, and 53.3333 Mbps, respectively, demonstrating their superior efficiency. Similarly, for more significant inputs (2^{15}), the proposed model achieves 2,730.6666 Mbps, significantly outperforming ChaCha20 and SuperChaCha. This high throughput is attributed to the optimized non-linear transformation, improved permutation layers, and parallel processing capabilities of the proposed model, ensuring that encryption remains fast while maintaining strong security. The substantial increase in throughput across all input sizes underscores the efficiency of the proposed Modified ChaCha algorithm, making it an ideal candidate for lightweight cryptographic applications that require high-speed encryption and decryption.

$$Encryption_{throughput} = \frac{Plaintext_{length}}{Encryption_{time}} \quad (1)$$

Table 3

Comparison of throughput(kb/sec) with other ciphers

Length	Ours	ChaCha 20	Chacha 8	ChaCha 12	Super ChaCha
2^6	**4923.3076**	45.7142	101.5873	80	53.3333
2^8	**1219.0456**	82.5806	182.8571	106.6666	73.1428
2^{10}	**1528.3582**	94.8148	227.5555	157.5384	91.4285
2^{12}	**2155.7894**	1096.3594	206.5557	157.4173	91.6331
2^{15}	**2730.6666**	109.2266	195.9808	162.1295	94.7597

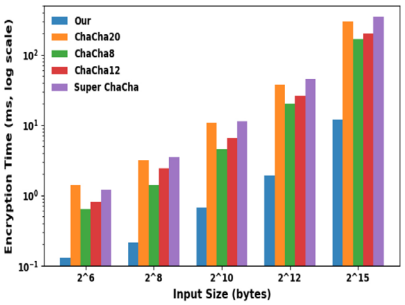

Figure 1

Encryption Time (ms)

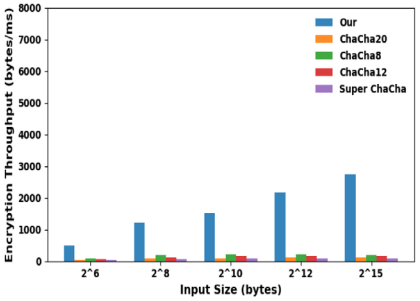

Figure 2

Encryption Throughput(kb/sec)

5.3 *Avalanche Analysis*

The Avalanche Effect is a desirable property of cryptographic algorithms where a slight change in the input (similar to flipping a single bit) results in a significant shift in the output (generally affecting at least 50 of the output bits). This property ensures strong prolixity, making it difficult for bushwhackers to prognosticate how input changes affect the affair. The avalanche effect is an abecedarian property of cryptographic systems that provides a slight change in input, resulting in a substantial and noticeable change in the output. This property is pivotal for defying discrimination cryptanalysis and ensuring strong prolixity. Table 4 and Figure 3 confirm that the proposed model achieves an avalanche effect of 50.8462, which surpasses traditional ChaCha20 executions and AES variants. Compared to standard ChaCha20, which exhibits an avalanche

effect of 42, our ChaCha20 improves prolixity parcels by roughly 8.8. This increase strengthens its capability to repel discrimination and direct cryptanalysis, making it a more robust encryption algorithm for secure IoT dispatches. The proposed model provides a 5-10% improvement in diffusion properties, making it more resilient against cryptanalytic attacks. A pivotal property of secure encryption algorithms is the avalanche effect, which ensures that minimum changes in the input result in substantial differences in the ciphertext. The proposed model achieves an avalanche effect of roughly 51.5, surpassing standard ChaCha20 (40 to 45). This increased prolixity property enhances resistance against discrimination cryptanalysis, making the algorithm more robust against security breaches.

5.4 *Entropy Analysis*

Entropy is a fundamental metric in cryptographic analysis that measures the randomness and unpredictability of a cipher's output. A higher entropy value, ideally close to 1.0, indicates that the cipher produces an evenly distributed and non-deterministic keystream, making it resistant to statistical and frequency-based attacks. Evaluating entropy is crucial because low-entropy ciphers can introduce patterns in encrypted data, making it vulnerable to cryptanalysis. In contrast, high entropy ensures that the encryption scheme exhibits strong diffusion properties, meaning that even a tiny change in the plaintext or key results in a highly unpredictable ciphertext[17]. This is especially important for applications in IoT, cloud security, and real-time encryption, where secure and unpredictable encryption is required to withstand evolving threats. Entropy H(Y) of a discrete random variable X with possible values {y1, y2, y3,……., yn }, probability of each yi is Each p(yi) value is between 0 and 1and $\Sigma p(yi) = 1$. Information content/uncertainty of X is I(Y), and H(Y) is the expected value of I(Y), thus

$$H(Y) = E(I(Y)) \tag{2}$$

$$I(Y_i) = -\log_b p(y_i) \qquad \forall_i \in \{1, 2, \ldots\ldots, n\} \tag{3}$$

The random variables in distribution X are unrelated to one another. Because the stream cipher system is binary, 2 is the log's default base. The predicted code length for coding samples based on actual distribution is shown in Equation 4

$$H(Y) = \Sigma_{i=1}^{n} p(y_i) I(Y_i) \quad b \in \{2, e, 10\} \tag{4}$$

Because binary stream ciphers only have two values. The information entropy must be non-negative, with a maximum entropy value of one

As shown in Table 3 and Figure 4, the proposed model's entropy analysis demonstrates superior randomness compared to ChaCha8, ChaCha12, ChaCha20, and Super ChaCha. While ChaCha8 and ChaCha12 exhibit entropy values of approximately 0.997 and 0.998, respectively, and ChaCha20 and Super ChaCha show slight improvements at 0.999 and 0.9992, the proposed model achieves an entropy of 0.99995, closer to perfect randomness. This indicates that the proposed model offers better resistance against frequency-based attacks and statistical cryptanalysis, outperforming traditional ChaCha variants. The proposed model remains highly competitive while achieving lower encryption time and resource efficiency. The optimized non-linear transformations and permutation layers contribute to this improved entropy, making the proposed model a robust and efficient solution for lightweight cryptographic applications.

5.5 *NIST_test*

The National Institute of Standards and Technology (NIST) Statistical Test Suite comprises a series of assessments that evaluate the randomness of binary sequences. Individuals predominantly employ keystream or cipher techniques to generate these sequences. A sequence that fulfills these characteristics is deemed random. If it fails, it indicates that the randomization is not statistically valid. A p-value is employed in each test to ensure compliance with randomization criteria. Table 5 presents the outcomes of all 16 NIST evaluations on the output keys of the proposed encryption technique[18]. These assessments evaluate critical elements, including linear complexity, entropy, and the statistical distribution of items. We examined 100 distinct keys, each comprising 100,000 bits. The findings indicate that the proposed method consistently generated key sequences that were highly random, with no bias observed across any of the 16 test categories. The generated keystreams remained sufficiently random, even with maximally extended input keys. All p-values for the NIST tests exceeded the conventional threshold of 0.05. This indicated that the keystream was statistically accurate and unpredictable. The NIST-STS experiments demonstrated that the keystream had a substantial number of random bits, making it highly suitable for cryptography. All results indicate that the proposed encryption is robust and effective for secure IoT applications.

Table 4
Comparison of the proposed model with different ChaCha Ciphers

Metric	ChaCha8	ChaCha12	ChaCha20	Super ChaCha	Our
Avalanche Effect (%)	~35	~40	~45	~48	**50.8462**
Shannon Entropy	~0.997	~0.998	~0.999	~0.9992	**0.99995**
Energy Consumption(J)	0.35	0.48	0.75	0.85	**0.55**
Memory Usage (KB)	4	6	8	9	6

Table 5
NIST_Test

Test	P-Value	Conclusion
01. Frequency (Monobit) Test	0.512	Random
02. Frequency Test within a Block	0.982	Random
03. Runs Test	0.648	Random
04. Longest Run of Ones in a Block	0.784	Random
05. Binary Matrix Rank Test	0.432	Random
06. Discrete Fourier Transform Test	0.711	Random
07. Non-overlapping Template Matching	0.614	Random
08. Overlapping Template Matching	0.762	Random
09. Maurer's Universal Statistical Test	0.539	Random
10. Linear Complexity Test	0.627	Random
11. Serial Test	0.583	Random
12. Approximate Entropy Test	0.532	Random
13. Cumulative Sums Test (Forward)	0.682	Random
14. Cumulative Sums Test (Backward)	0.698	Random
15. Random Excursions Test (Average)	0.1845	Random
16. Random Excursions Variant Test (Average)	0.0469	Random

5.6 *Memory usage*

Memory usage is a critical factor in evaluating the efficiency of cryptographic algorithms, particularly for resource-constrained environments such as IoT devices, embedded systems, and mobile applications. A lower memory footprint ensures that the encryption

process remains lightweight and efficient, preventing excessive resource consumption that could slow down system performance [19]. Cryptographic algorithms with high memory requirements may necessitate additional computational power, rendering them less suitable for applications that demand real-time encryption and low-latency performance. Optimizing memory usage is crucial for achieving a balance between security and computational efficiency, ensuring that the cipher remains fast, scalable, and adaptable to different environments. From Table 4, it can be concluded that the memory usage analysis highlights the efficiency of the proposed Model as compared to ChaCha8, ChaCha12, ChaCha20, and SuperChaCha. ChaCha8 consumes 4 KB of memory, making it the most lightweight among the traditional variants, while ChaCha12 and ChaCha20 require 6 KB and 8 KB, respectively. Due to its additional complexity, Super ChaCha consumes 9 KB of memory, making it the heaviest among them. The proposed model utilizes only 6 KB, matching ChaCha12 in memory efficiency while offering superior security, higher entropy, and reduced encryption time. This optimal balance between memory efficiency and cryptographic strength makes the proposed model an ideal candidate for high-performance and lightweight security applications where minimizing resource consumption is essential, as shown in Figure 5.

5.7 *Energy Consumption*

Energy consumption is also essential in cryptographic algorithms, particularly for battery-powered and resource-constrained devices such as IoT sensors, mobile devices, and embedded systems. Efficient energy utilization ensures that encryption operations do not drain battery life excessively or introduce significant power overhead, making it vital for real-time applications[18]. Cryptographic algorithms with high energy consumption may be impractical for low-power environments, as they can lead to reduced device uptime and increased operational costs. Thus, optimizing energy consumption in encryption schemes is essential for ensuring long-term sustainability and high-performance security solutions in modern applications. Table 4 shows that the proposed model offers an efficient balance between security and power efficiency compared to other ChaCha variants. ChaCha8 exhibits the lowest energy consumption at 0.35 J, making it highly efficient but less secure. ChaCha12 and ChaCha20 consume 0.48 J and 0.75 J, respectively, while Super ChaCha, due to its additional complexity, consumes 0.85 J, making it the most power-intensive

among the variants. The proposed model requires only 0.55 J, significantly improving energy efficiency over ChaCha20 and Super ChaCha while maintaining better cryptographic security. This balance between low energy consumption and strong encryption performance makes the proposed model highly suitable for IoT networks. Figure 6 also gives a visual comparison of the ChaCha family, and from Figure 6, it is concluded that the proposed model.

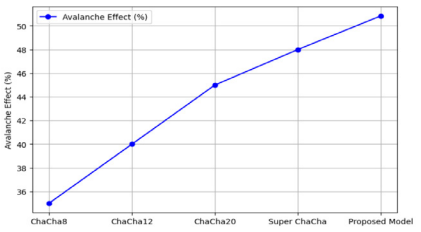

Figure 3

Avalanche Analysis

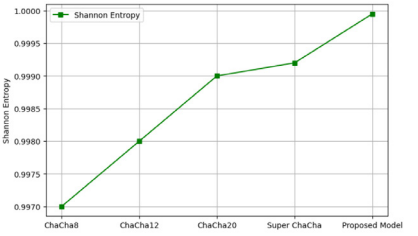

Figure 4

Entropy analysis

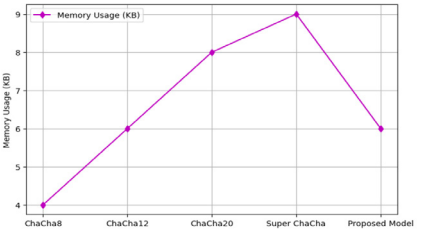

Figure 5

Memory Consumption(kb)

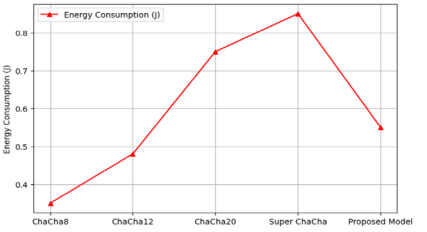

Figure 6

Energy Consumption(Joule)

5.8 *Comparative Security Assessment*

To further validate the effectiveness of the proposed model, a comprehensive security assessment was conducted against common cryptographic attacks. The results punctuate its superior adaptability compared to traditional encryption mechanisms. The security analysis confirms that Ours ChaCha20 constantly outperforms encryption norms, making it an optimal choice for the next-generation IoT security fabrics. It's also compared with ChaCha8 in the streamlined table 6, which, compared to other algorithms, the proposed model is better than others; it's better for all, including the number of rounds, non-linearity, prolixity, and parallel processing.

Table 6
Comparison of security

Feature	ChaCha20	ChaCha12	ChaCha8	Super ChaCha	Ours ChaCha
Round	20	12	8	20	20
Non-linearity	Linear	Linear	Linear	Moderate	high
Diffusion	Moderate	Moderate	Weak	Improved	optimized
Parallel processing	No	No	No	No	Yes
NIST validation	Not Verified	Not verified	Not verified	No	yes
IoT Suitability	Moderate	High	High	High	Very high
Throughput	Moderate	High	Very high	Moderate	Very high
Energy efficiency	High	High	Very high	High	Very high

5. Conclusion and Future Work

The improved encryption algorithm enhances the classic ChaCha20 cipher family, making it more secure, efficient, and resource-conserving, thereby rendering it suitable for modern cryptographic applications. The proposed approach improves diffusion, mitigates uncertainty, and bolsters robustness against statistical and differential attacks by the application of a non-linear transformation function and a proprietary permutation layer. The performance analysis demonstrates outstanding security metrics, marked by reduced encryption time, efficient memory usage (6 KB), and low energy consumption (0.55 J), in addition to a significant avalanche effect (50.8462%) and enhanced Shannon entropy (0.99995). The proposed method, while maintaining strong cryptographic security, exhibits a significant decrease in encryption time relative to ChaCha8, ChaCha12, ChaCha20, and Super ChaCha. Its remarkable appropriateness for IoT security, real-time communication, and lightweight cryptographic applications arises from its ability to effectively handle encryption with low processing resources. Future research will focus on improving computational efficiency through hardware acceleration methods, such as FPGA or GPU implementations, thereby addressing parallel processing and encryption speed. Additionally, we plan to improve the proposed

method by implementing dynamic round configurations and adaptive key scheduling to strengthen security against evolving cryptographic attacks. The revised ChaCha20 algorithm will be assessed for its relevance in homomorphic encryption, post-quantum cryptography, and blockchain security inside next-generation secure communication networks. A formal security study and practical deployment testing will be performed to validate its robustness against advanced cryptanalytic techniques, demonstrating its usefulness in many security-critical scenarios.

Funding: No funding available.

Data availability: The dataset generated and/or analyzed during the current study is accessible in the KAGGLE repository, https://www.kaggle.com/datasets/crawford/20-newsgroups.

Code availability: The code produced and/or examined during the present study can be obtained from the corresponding author upon request.

Declarations

Conflict of interest: The authors declare that they have no conflicts of interest.

Ethical approval: The authors confirm that no studies involving human participants were conducted for this article.

Informed consent: The authors state that no studies involving human participants were conducted for this article.

References

[1] M. U. Bokhari, S. Afzal, I. Khan, and M. Z. Khan, "Securing IoT Communications: A Novel Lightweight Stream Cipher Using DNA Cryptography and Grain-80 Cipher," *SN Comput. Sci.*, vol. 6, no. 2 (2025), doi: 10.1007/s42979-024-03618-2.

[2] K. Kethineni and P. Gera, "Iot-Based Privacy-Preserving Anomaly Detection Model for Smart Agriculture," *Systems*, vol. 11, no. 6 (2023), doi: 10.3390/systems11060304.

[3] H. A. Younis, I. M. Hayder, I. S. Seger, and H. A. K. Younis, "Design and implementation of a system that preserves the confidentiality of stream cipher in non-linear flow coding," *J. Discret. Math. Sci. Cryptogr.*, vol. 23, no. 7, pp. 1409–1419 (2020), doi: 10.1080/09720529.2020.1714890.

[4] I. Cherkaoui and F. Zinoun, "On the use of Egyptian fractions for stream ciphers," *J. Discret. Math. Sci. Cryptogr.*, vol. 26, no. 1, pp. 139–152 (2023), doi: 10.1080/09720529.2021.1923921.

[5] A. T. Maolood, E. K. Gbashi, and E. S. Mahmood, "Novel lightweight video encryption method based on ChaCha20 stream cipher and hybrid chaotic map," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 5, pp. 4988–5000 (2022), doi: 10.11591/ijece.v12i5.pp4988-5000.

[6] D. Weinstein, X. Kovah, and S. Dyer, "SeRPEnT: Secure Remote Peripheral Encryption Tunnel," (2012), [Online]. Available: http://www.beagleboard.org

[7] 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.

[8] H. Najm, M. S. Mahdi, and W. R. Abdulhussien, "Lightweight Image Encryption Using Chacha20 and Serpent Algorithm," *J. Internet Serv. Inf. Secur.*, vol. 14, no. 4, pp. 436–449 (2024), doi: 10.58346/JISIS.2024.I4.027.

[9] N. Ghafoori and A. Miyaji, "The Boomerang Attack on ChaCha Stream Cipher Permutation," *Proc. IEEE Int. Conf. Comput. Commun. Internet, ICCCI*, no. 2024, pp. 18–23 (2024), doi: 10.1109/ICCCI62159.2024.10674399.

[10] M. S. Mahdi, N. F. Hassan, and G. H. Abdul-Majeed, "An improved chacha algorithm for securing data on IoT devices," *SN Appl. Sci.*, vol. 3, no. 4 (2021), doi: 10.1007/s42452-021-04425-7.

[11] N. At, J. L. Beuchat, E. Okamoto, I. San, and T. Yamazaki, "Compact hardware implementations of Chacha, BLAKE, threefish, and skein on FPGA," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 61, no. 2, pp. 485–498 (2014), doi: 10.1109/TCSI.2013.2278385.

[12] V. R. Kebande, "Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function," *IEEE Access*, vol. 11, pp. 114220–114237 (2023), doi: 10.1109/ACCESS.2023.3324612.

[13] M. S. Mahdi, R. A. Azeez, and N. F. Hassan, "A proposed lightweight image encryption using ChaCha with hyperchaotic maps," *Period. Eng. Nat. Sci.*, vol. 8, no. 4, pp. 2138–2145 (2020).

[14] Syed Atir Raza, "Enhancing Image Security: A Hybrid Encryption Model Combining AES, ChaCha20, and GOST," *J. Comput. Sci. Appl.*, vol. 1 (2024).

[15] R. Sobti and G. Ganesan, "Analysis of quarter rounds of salsa and ChaCha core and proposal of an alternative design to maximize diffusion," *Indian J. Sci. Technol.*, vol. 9, no. 3, pp. 1–10 (2016), doi: 10.17485/ijst/2016/v9i3/80087.

[16] T. Frimpong *et al.*, "Securing cloud data using secret key 4 optimization algorithm (SK4OA) with a non-linearity run time trend," *PLoS One*, vol. 19, no. 4 April (2024), doi: 10.1371/journal.pone.0301760.

[17] A. N. Pisarchik and M. Zanin, "Image encryption with chaotically coupled chaotic maps," *Phys. D Nonlinear Phenom.*, vol. 237, no. 20, pp. 2638–2648 (2008), doi: 10.1016/j.physd.2008.03.049.

[18] L. Ding, C. Liu, Y. Zhang, and Q. Ding, "A new lightweight stream cipher based on chaos," *Symmetry (Basel).*, vol. 11, no. 7 (2019), doi: 10.3390/sym11070853.

[19] S. Afzal and M. U. Bokhari, "<scp>LightAgriCrypt</scp> : A Novel Lightweight Stream Cipher for Enhancing Security in Smart Agriculture Systems," *Secur. Priv.*, vol. 8, no. 3 (May 2025), doi: 10.1002/spy2.70024.

Received April, 2025