Al-Furat Journal of Innovations in Electronics and Computer Engineering (FJIECE)
ISSN -2708-3985

Reinforcing RC5: Dynamic S-Box Generation and Magic Square-Based Key Expansion

Zainab Sahib Dhahir[1*]

Al-Furat Al-Awsat Technical University (Technical Institute of Babylon, Department of Technical Computer Systems, Babil, 51015, Iraq). E-mail: zainab.dhahir@atu.edu.iq

*Corresponding author E-mail: zainab.dhahir@atu.edu.iq

https://doi.org/10.46649/fjiece.v3.2.35a.14.6.2024

Abstract. *The ever-increasing and ever-demanding growth of the field of cryptanalysis imposes a continuous requirement for algorithm modifications. This paper introduces modifications in widely-used RC5 block cipher to cater to the increasing security demands, i.e. dynamic S-box generation and magic square-based key expansion. The main purpose is to improve the complexity and linearity of the algorithm as a result making it resistant to newly-devised attacks. A small (3 × 3) magic square is taken for generating dynamic S-boxes and constants to add unpredictable randomness during the key expansion phase. SHA-256 hash function is used for S-boxes generation meanwhile properties of magic square are exploited for constants derivation purpose. Proposed changes resulted in improved security against differential, linear, and algebraic attacks respectively meanwhile efficiency is still sustained so that this modified version can be well-suited for recent applications that necessitate strong encryption services.*

Keywords: *RC5 block ciphers; magic square; S-boxes; key expansion; security.*

1. INTRODUCTION

Enhancements, to encryption algorithms are necessary as cryptanalysis methods become more advanced. Incorporating S box generation and key expansion based on squares can strengthen the widely used RC5 block cipher, known for its straightforwardness and efficiency. This study presents the improvements showing how they boost the security of RC5, against attacks [1, 2]. Firstly, RC5, the algorithm created in 1994 by Ronald Rivest, is famous for being highly adaptable and secure owing to its use of data rotations and adjustable parameters but simplistic design [3] [4]. Thus easy implementation and efficiency, so many security protocols and applications incorporate RC5. Smart cards and embedded devices are where it excels i.e., because of their limited resources [5] [6].

Although a large scale of difference through a range of different outputs shows that this model is not strong enough for commercial purposes, the advantages of the already existing classic RC5 method help for a good foundation can take advantage but it still needs some more effort to make use of biometrics. This paper proposes two basic changes made to the RC5 cipher algorithm: Dynamic S-box construction / Magic square-based constants for key expansion.

Dynamic S-boxes help increase the security of block ciphers by offering non-linearity and confusion to cryptographic algorithms [7] [8] [9]. These S-Boxes are dynamically generated with the help of several techniques such as chaotic maps, DNA computing, and also key-dependent mechanisms in such a way that each encryption produces a unique S-Box while depending on certain parameters or keys in the cipher process against the known conventional methods [10]. Incorporating created S boxes boosts the system's randomness improving its defense against differential attacks. These S boxes uphold characteristics such, as bijection, nonlinearity meeting the strict avalanche criterion, and ensuring the independence of output bits for strong encryption. The integration of S boxes, in generation significantly enhances the security of block ciphers by adding elements of unpredictability and intricacy to the encryption procedure [6]. Magic squares are crucial, in cryptography in strengthening encryption algorithms. Experts have come up with encryption techniques that leverage squares to improve security and effectiveness, in image encryption [11, 12, 13]. These techniques use squares of sizes that are chosen dynamically to mix up images. Additionally, they create random data streams to enhance the effects of confusion and diffusion making the plaintext more sensitive, for security measures. The study of quantum squares in quantum measurements has revealed their features and relationship, to quantum Latin squares [14, 15].

Authors are using the characteristics of squares to improve encryption algorithms and enhance security measures, against hackers. This method aims to strengthen methods by increasing the intricacy and unpredictability of the encryption process with S Box generation. Every encryption session produces an S Box created using hash functions making it harder for attackers to break through the cipher due, to the versions that prevent pre-calculation or anticipation of the S Box. As a result, encryption becomes more secure [5, 16].

The research proposes a method, for expanding the RC5 key by substituting the constants P and Q with values derived from a special mathematical square known as a magic square. The features of the magic squares include equal sum totals of the numbers in each row, column, and diagonal. Leveraging these properties enables the creation of complex constants that enhance the non-linearity and unpredictability of the expansion process. This approach bolsters the strength of the schedule and guarantees that each encryption key generates unique constants thereby improving overall security, in the RC5 algorithm. Here's a summary of the paper's contributions:

1. Combined dynamic S-box generation with magic square-based key expansion into one cohesive cryptographic framework.
2. Showed notable enhancements in the cryptographic strength of the RC5 algorithm.
3. Offered a thorough approach to bolster the RC5 block cipher's defense against contemporary cryptanalytic methods.

2. LITERATURE REVIEW

Experts have improved the RC5 algorithm by concentrating on enhancing both its security and effectiveness. They have implemented techniques like adjusting rotations customizing data permutations and creating S boxes dynamically. The combination of S boxes and constants derived from squares within a unified cryptographic system has not received thorough exploration [2, 3]. This study builds upon research enhancing the encryption capabilities of RC5. Various modifications, to the RC5 algorithm have been suggested over time to enhance its safety and effectiveness as illustrated in Table (1). An efficient method involves integrating rotations and permutations based on data boosting the ability of algorithms to withstand linear cryptanalysis. Research indicates that the initial RC5 algorithm was susceptible, to attacks prompting scholars to investigate changes, for enhanced security [17]. The authors of the AES algorithm explored the impact of incorporating S boxes on improving the security and effectiveness of encryption methods in a research paper [18]. The development of the dynamic S-Box and the secret key generation mechanism is discussed in the works referenced as [19], [20], and [22]. In sources [20] and [21], a key-dependent dynamic S-box mechanism is discussed. This mechanism involves dynamic permutations operating on S-boxes that share the same properties. The research cited in [22] presents a method of using dependent S boxes to enhance the security of ciphers, against different types of attacks. Moreover, the study discussed in [23] utilizes Elliptic Curve Cryptography to develop S boxes enhancing the strength of encryption. In [24] explored the application of squares in crafting keys showcasing their effectiveness, in generating varied and unpredictable key sequences. The distinctive characteristics of squares guarantee the uniqueness and intricacy of each produced value providing added defense against attacks, on scheduling. This approach has been observed to boost the security of encryption algorithms by ensuring that key expansion processes are both reliable and unforeseeable [11] [12].

The RC5 algorithm utilization of S box generation and square-derived constants represents a novel approach, to enhancing its security measures. While past studies have explored the advantages of these methods separately their amalgamation, within a cryptographic system remains largely uncharted territory. The suggested enhancements, incorporating S boxes and magic square-derived constants seek to address the vulnerabilities identified in RC5 implementations. In summary, research shows that utilizing S box generation and constants derived from squares can significantly enhance the security of block ciphers. By implementing these techniques in the RC5 algorithm it offers a solution to combat cryptanalytic attacks. The suggested modifications build upon studies to offer a strong enhancement, to the RC5 encryption method.

Table 1. Summary of Enhancements to RC5 Algorithm by Various Works

Author(s)	**Method Used**	**Results Achieved**
S. M. Kareem and A. M. S. Rahma [11]	Proposes a new AES algorithm using the magic square to enhance security	Replaces Mixcolumn function with magic square of order 6
S. M. Kareem and A. M. S. Rahma [15]	Modification using magic square 3x3 Proposes MSDES algorithm modification.	Additional key created using linear first shift register (LFSR) for high-level security
J. Daemen and V. Rijmen [18]	Dynamic S-box generation in AES algorithm	Emphasized the importance of dynamic S-boxes in increasing the unpredictability and robustness of encryption schemes.
S. Pal, R. Selvanambi, P. Malik, and M. Karuppiah [19]	Dynamic S-Box and secret key generation mechanism The Counter-based mechanism for updating security parameters and keys	Chaotic algorithm using 'Henon Chaotic' maps for image transfer safety
A. Y. Al-Dweik, I. Hussain, M. Saleh, and M. T. Mustafa [20]	Group action of symmetric group Sn and subgroup S2n	Extension of previous work involving group action of S8
Ejaz, I. A. Shoukat, U. Iqbal, A. Rauf, and A. Kanwal [21]	Key-dependent dynamic S-box with dynamic permutations	The Proposed method outperforms existing techniques, highly sensitive to secret keys.
Alasaad and A. Alghafis [22]	The Proposed key-dependent S-box scheme enhances cipher security against attacks.	Manipulating standard S-box with primary key bits
T. Ara, P. G. Shah, and P. M [23]	Focus on generating key-dependent S-Boxes using Elliptic Curve Cryptography.	Efficient for resource-constrained IoT devices, tested against specific criteria
I. M. Alattar and R. S. A. Monem [24]	The proposed cryptography system in the paper is based on known normal magic squares, particularly MS7, to derive equations for rows, columns, and diagonals, ultimately resulting in 16 equations for encryption.	The paper presents successful results, as indicated by the histogram statistics for image 2, showcasing the effectiveness of the proposed cryptographic algorithm.
M. Sahni and D. B. Ojha [25]	The research paper utilizes the concept of an 8x8 Magic Square to generate keys and encrypt data using ORDES.	The research paper presents a security analysis of the Magic Square generalized image and ORDES encryption method, which is based on random number generation and works similarly to a one-time pad.

3. BACKGROUND

Improvement of cryptographic algorithms entails the integration of new methods in the algorithms' design to make them more secure against attacks. The two such methods are dynamic S-box generation and magic square based key expansion, both of which enhance the sophistication as well as the security of the encryption operations.

3.1. Dynamic S-box generation

Dynamic S-box generation is the process of deriving the S-boxes during the encryption process as opposed to having fixed S-boxes. The idea on which dynamic S-boxes are founded comes directly into the confusion and non-linear components into the encipherment algorithm against linear as well as differential attacks.

3.1.1. Hash Function-Based S-Box Creation:

Hash function (like SHA-256) takes the encryption key as an input and produces a specific output. The output from the hash is then divided to make the initial S-box entries. A permutation guarantees that the values in the S-box are unique and that no value is at the same position in two different RCs (this implies that there is no fixed point for any of the F-functions), which increases resistance against differential attacks.

3.1.2. Properties of Dynamic S-Boxes:

Dynamic S-boxes facilitate certain cryptographic features like Bilaterality, Non-Linearity, and also conformity to strict Avalanche Standards. The generated S-boxes depend on the instance of encryption and this approach makes it difficult for pre-computer attack and hence enhances the security of the cryptography.

3.2. Magic Square-Based Key Expansion

Magic square key expansion originates from the fundamentals of mathematics and implements magic squares into the key schedule process to augment the quantity of non-linearity and randomness in the encryption algorithm.

3.2.1. *Magic Square Theory*:

A magic square is a grid of numbers in which the total of the numbers along any row, any column, and the two diagonals are equal. The elements in the rows of the magic square are combined to produce numerical elements, which are then further multiplied by primes obtained from the golden ratio. The produced values are then masked to make them occupy not more than 32-bit integers, making it produce different constants for each encryption key. The unique constants generated from the magic square are used to initialize the subkeys array.

4. PROPOSED METHOD

The suggested method combines two improvements to the RC5 block cipher algorithm: dynamic S-box generation and magic square-based key expansion techniques as shown in Fig.1. These enhancements aim to increase the algorithm's cryptographic robustness by introducing additional layers of complexity and non-linearity. The following paragraphs include a detail of the proposed method and its algorithms.

4.1. Dynamic S-Box Generation

Dynamic S-box generation is the process of creating S-boxes during the process of encryption, unlike fixed S-boxes as in Algorithm (1). This approach adds nonlinearity and confusion to the algorithm of encryption, hence improving its resilience to both linear and differential attacks. The generation process undergoes several stages to ensure a unique and unpredictable result.

Algorithm (1): Hash Function-Based S-Box Generation

```
Input: key (string)
Output: sbox (array of 256 integers)

Begin
   hash_output = SHA256(key)
   sbox = First256Bytes(hash_output)

   j = 0      // index variable used in the permutation process of the S-box.
   for i = 0 to 255 do    // iteration over the elements of the S-box.
      j = (j + sbox[i] + i) % 256
      Swap(sbox[i], sbox[j])
   end for

   return sbox
End
```

The input key is hashed using the SHA-256 hash function to produce a 256-bit hash output then the first 256 bytes of the hash output are extracted to initialize the S-box array. A for-loop iterating from 0 to 255 runs over each element of the S-box. The variable j will be updated in each iteration by the expression (j + sbox[i] + i) mod 256, it then swaps the elements of the S-box at indices i and j to produce the output of the algorithm is the permuted S-box. Some properties of dynamic S-boxes are bilaterality, non-linearity, and strict avalanche standards. Since a new S-box is created every time for instances of encryption, it is very hard for pre-computation attacks and hence improves the security in general ways.

4.2. Magic Square-Based Key Expansion

Magic square-based key expansion is a process of introducing nonlinearity and randomness into the encryption algorithm using the mathematical properties of magic squares. A magic square denotes a grid whereby every row, column, and diagonal gives the same sum. Algorithm (2) produces constants using a magic square for key expansion.

Algorithm (2): Producing Constants Using a Magic Square

```
  # Step 1: Initialize Magic Square
  magic_square = [
      [8, 1, 6],
      [3, 5, 7],
      [4, 9, 2]        ]

  # Step 2: Concatenate Numbers in Rows
  P = int(''.join(map(str, magic_square[0])))  # 816
  Q = int(''.join(map(str, magic_square[1])))  # 357

  # Step 3: Multiply by Constants
  P = P * 0x9e3779b9
  Q = Q * 0xb7e15163

  # Step 4: Truncate to 32-bit Integers
  P = P & 0xffffffff
  Q = Q & 0xffffffff

  # Step 5: Return Constants
  return P, Q
```

Where P is derived by taking the numbers of the first line of the magic square (8, 1, and 6) concatenated into one integer, 816. That number is concatenated and multiplied by a constant derived from the golden ratio: 0x9e3779b9. That multiplication result is then truncated at the limitations of a 32-bit integer with a bitwise AND with 0xffffffff. Similar to P, Q this variable is also derived by catenation of the numbers in the second line of the magic square (3, 5, 7) into one integer: 357. The obtained concatenated number is multiplied by another constant, 0xb7e15163, again derived from the golden ratio. As before, the result is fitted into the 32-bit integer by a bitwise AND operation with 0xffffffff. Generating Constants by Magic Square: This constitutes a part of a methodology to enhance the ultralightweight RC5 block cipher algorithm through dynamic generation of S-boxes and key expansion based on a magic square.

3.3. key expansion

In the proposed improved RC5 algorithm, the key expansion process generates subkeys and packs the user key. Hereafter is represented the Algorithm (3) representation for the key expansion according to the above description:

Algorithm (3): Key Expansion

```
 # Step 1: Initialize subkeys array (S) with constant P
  S = [P]
  for i in range(1, 2 * R + 2):
      S.append((S[i - 1] + Q) % (2 ** w))

 # Step 2: Calculate the number of c words in the key
  c = len(key) // (w // 8)
  if len(key) % (w // 8) != 0
      c += 1
 # Step 3: Pack the user key into a list L
  L = [0] * c
  for i in range(len(key) - 1, -1, -1):
      L[i // (w // 8)] = (L[i // (w // 8)] << 8) + ord(key[i])

  Return S, L

# Output: Generate subkeys (S) and packed key (L)
```

The variables P and Q obtained from the previous step are used to initialize the subkeys array and then calculate the number of words in Key (c) based on the word size (*w*). L is an array that stores the packed user key, divided into words of size (*w*). The algorithm returns the generated subkeys array and the packed key array *L*.

3.4. Mixing with S-box

Algorithm (4) illustrates mixing with the S-box within the RC5 block cipher algorithm enhancement.

Algorithm (4): Mixing with S-box

```
Input: ( Subkeys Array S, Packed Key Array L, S-box SBOX, Word size w ).
Output: ( Mixed Subkeys Array S, Mixed Packed Key Array L ).

#Step 1:
   I = J = 0
   A = B = 0
#Step 2:
   for k in range(3 * max(len(S), len(L))):
      A = S[I] = (S[I] + A + B) % 2**w
      A = SBOX[A % len(SBOX)]
      B = L[J] = (L[J] + A + B) % 2**w

      B = SBOX[B % len(SBOX)]

      I = (I + 1) % len(S)

      J = (J + 1) % len(L)

#Step 3:
   return S, L
```

A and B: These are intermediate variables used in the mixing process to store temporary values. They are initialized to 0 at the start of the algorithm and are updated and then substituted using the S-box.

3.4.Enhanced RC5 Algorithm Implementation:

The proposed method combines dynamic S-box generation with magic square-based key expansion to enhance the RC5 block cipher algorithm. In this section, the steps to implement the enhanced RC5 algorithm are illustrated in Algorithm (5).

Algorithm (5): Enhanced RC5 with Dynamic S-Box Generation and Magic Square-Based Key Expansion

```
Input: Secret Key K, Word Size w, Number of Rounds R, Subkey array initialized using dynamic S-box and magic square-based key expansion S).
 Output: Encrypted/Decrypted Text
#step 1:
Dynamic S-Box Generation uses Algorithm (1).
#step 2:
Magic Square-Based Key Expansion uses Algorithm (2).
#step 3:
Key Expansion use Algorithm (3).
#step 4:
Mixing with S-Box uses Algorithm (4).
#step 5:
Encryption Process:
For  I  in  range(1, R + 1):
    A = (A ^ B)
    A = ((A << (B % w)) | (A >> (w - (B % w)))) % (2 ** w)
    A = (A + S[2 * i]) % (2 ** w)
    B = (B ^ A)
    B = ((B << (A % w)) | (B >> (w - (A % w)))) % (2 ** w)
    B = (B + S[2 * i + 1]) % (2 ** w)
End for
#step 6:
Decryption Process:
For  I  in range(R, 0, -1):
    B = (B - S[2 * i + 1]) % (2 ** w)
    B = ((B >> (A % w)) | (B << (w - (A % w)))) % (2 ** w)
    B = (B ^ A)
    A = (A - S[2 * i]) % (2 ** w)
    A = ((A >> (B % w)) | (A << (w - (B % w)))) % (2 ** w)
    A = (A ^ B)
End for
```

Here A, B: Registers holding the plaintext or ciphertext during encryption and decryption processes. Fig. 1 shows the steps of the proposed method.

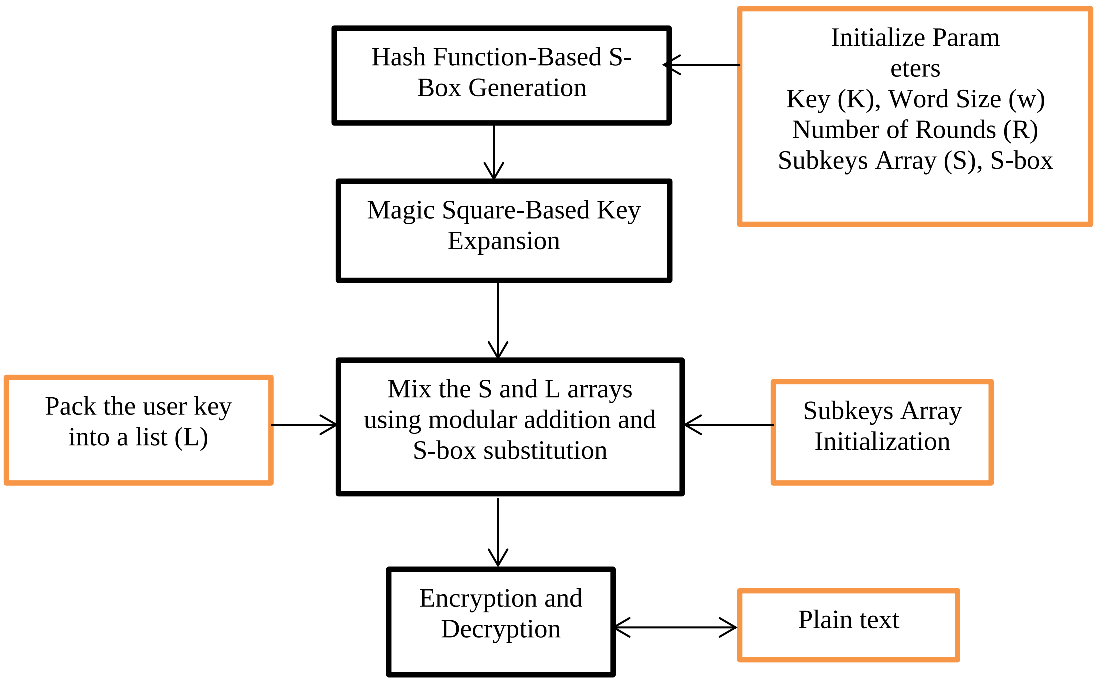
Hash Function-Based S-Box Generation
Initialize Param eters Key (K), Word Size (w) Number of Rounds (R) Subkeys Array (S), S-box
Magic Square-Based Key Expansion
Pack the user key into a list (L)
Mix the S and L arrays using modular addition and S-box substitution
Subkeys Array Initialization
Encryption and Decryption
Plain text

Fig.1. Diagram of the proposed method

4. RESULTS AND DISCUSSION

The description of the experimental setup, including hardware and software specifications with the following detailed information: for hardware (Processor: Intel Core i7-9700K (8 Cores, 3.6 GHz), RAM: 16 GB DDR4, Storage: 512 GB SSD and Graphics: NVIDIA GeForce GTX 1660, 6 GB. For software Python 3.9 language used, Jupyter Notebook: Version 6.3.0 and Google Colab: For running and testing Python code in a cloud-based Jupyter environment with GPU support.

Initialization Parameters:

Key (K): "SecretKey1234567", a 128-bit key is used for the encryption process.

The key is hashed using SHA-256 to generate a dynamic S-box.

Word Size (w): 32 bits

Number of Rounds (R): 12

Subkeys Array (S): Initialized based on the word size and number of rounds.

At first, the key is used as input to algorithm (1), in this case, "SecretKey1234567", generates a SHA-256 hash of the input key. This produces a 256-bit hash output. The first 256 bytes of the hash output are used to initialize the S-box. However, since SHA-256 outputs 32 bytes (256 bits), we need to modify this step. Instead, we'll use the hash output to seed the permutation process. The function then permutes the initialized S-box using a simple pseudo-random permutation algorithm. This step ensures the uniqueness and distribution of values in the S-box. After executing Algorithm (1) the generated S-box with the key "SecretKey1234567" has the following first 10 elements:

229, 210, 185, 114, 111, 23, 231, 49, 88, 73

Using this function can generate a unique S-box based on any given key, ensuring a high level of security due to the SHA-256 hashing and subsequent permutation process.

Secondly, the constants (P and Q) are generated using algorithm (2) by defining a (3x3) Magic Square and then concatenate numbers in the rows (First row: 816 Second row: 357) which 816 is multiplied by 0x9e3779b9 (which is a constant derived from the golden ratio) and 357 is multiplied by 0xb7e15163 (another constant derived from the golden ratio). The results of these multiplications are then truncated to 32-bit integers using the bitwise AND operation with (0xffffffff). These are the hexadecimal representations of the constants derived from the magic square.

P: 0xa8f1f8d0

Q: 0x6c23d0d5

The key_expansion is a critical part of the RC5 encryption algorithm, which generates an expanded key schedule from a given key. This schedule is used to perform the encryption and decryption operations. The output of key expansion algorithm (3) subkeys and the packed key:

Subkeys (S):
Displaying the first 10 elements of the subkey list for brevity:

```
[2834430160, 353749413, 2168035962, 3982322511 1501641764, 3315928313 835247566, 2649534115
168853368, 1983139917]
```

Packed Key (L):
The packed key represents the input key in a format suitable for the RC5 algorithm:

```
[1919116627, 1699443813, 858927481, 926299444]
```

The algorithm (4) is used to integrate dynamic S-box generation into the RC5 key expansion. Algorithm (4) successfully mixes the provided subkey array S and the packed key array L using the S-box (sbox) and the word size (w). The output displays the first 10 elements of the mixed subkey array S and the entire mixed packed key array L.
Mixed Subkeys (S):

```
[2834430160, 353749413, 2168035962, 3982322511, 1501641764, 3315928313, 835247566,
2649534115, 168853368, 1983139917]
```

Mixed Packed key (L):

[6763, 6761, 6822, 7005]

The encryption and decryption process has been tested on previous results to initialize the key scheduling array. The plaintext block (0x12345678, 0x9abcdef0) substituted as A, and B are encrypted and then decrypted using algorithm (5). The output of the encryption and decryption process:

Plaintext: (305419896, 2596069104)

Ciphertext: (2715026565, 2004720599)

Decrypted text: (305419896, 2596069104)

The above implementation demonstrates the steps involved in enhancing the RC5 algorithm with dynamic S-box generation and magic square-based key expansion. The test results verify that the encryption and decryption functions work correctly with the given key schedule array. The time consumed by the proposed approach tested in Python is (0.0003 seconds) and then testing the same plain text, Key on classical RC5 algorithm takes the same time, that proves the proposed approach RC5 with dynamic S-box generation and magic square-based key expansion doesn't affect the execution time but in another side, it improves the algorithm by increasing non-linearity and the secrecy as a result.

To demonstrate the feasibility obtained from the proposed method, a calculation will be made of time and space complexity introduced by the dynamic S-box generation and the magic square-based key expansion techniques. Using SHA-256 to hash the key takes a constant time, $\mathcal{O}(1)$, since the length of the input key is fixed. In permutation iterating over 256 elements to permute them results in a time complexity of $\mathcal{O}(256)$ which simplifies to $\mathcal{O}(1)$ as it is a constant then the overall complexity for dynamic S-box generation is $\mathcal{O}(1)$ and the same for magic square initialization. The maximum complexity for filling the subkeys array based on constants P and Q is $\mathcal{O}(\max(R, k))$ moreover, the encryption and decryption process consumes about $\mathcal{O}(\mathrm{R})$ as time complexity As a result the overall time complexity is $\mathcal{O}(R + k)$.

The space complexity is determined by the storage of the S-box, subkeys, and packed key, which is proportional to the key length and the number of rounds: $\mathcal{O}(256 + R + k) \approx (R + k)$. Thus, the proposed approach's complexity is $\mathcal{O}(R + k)$ For both time and space, where R is the number of rounds and k is the length of the key.

Table 2. Complexity Comparison: Classical RC5 vs. Proposed Enhancements.

	Time	**Space**
RC5	$\mathcal{O}(t + b) + \mathcal{O}(r)$	$\mathcal{O}(t) + \mathcal{O}(1)$
Proposed method	$\mathcal{O}(R + k)$+ $\mathcal{O}(\max(R, k))$+ $\mathcal{O}(R)$	$\mathcal{O}(R) + \mathcal{O}(256)$+ $\mathcal{O}(R + k)$

As shown in Table 2 the proposed enhancements add a fixed overhead due to dynamic S-box generation and a proportional increase in key scheduling time due to magic square-based key expansion. While the enhanced RC5 requires additional space for the S-box and mixed key arrays. Thus, the proposed approach introduces additional complexity primarily in the key scheduling phase, making it more resistant to cryptanalytic attacks while maintaining efficient encryption/decryption processes.

To thoroughly evaluate the proposed enhancements to the RC5 block cipher algorithm several security metrics are used as shown in Table 3.

Table 3. Security Metrics Evaluation of Enhanced RC5 Algorithm.

Security metric	
Avalanche Effect	50.88%
Statistical Tests	p-value: 0.553
	Entropy: 7.818

The percentage of bits changed in the cipher text is approximately 50.88%. This indicates that a small change in the input (one bit) significantly changes the output, which is a desirable property for a secure encryption algorithm, demonstrating good diffusion and resistance to certain types of cryptanalytic attacks. The high p-value indicates that the byte distribution of the cipher text is uniform, suggesting good randomness. An entropy value close to 8 (the maximum possible for byte values) indicates high randomness and unpredictability in the cipher text.

The use of SHA-256 for dynamic S-box generation introduces a high degree of non-linearity and complexity, making it challenging to express the algorithm in a simple algebraic form. Each encryption instance produces a unique S-box, complicating the creation of consistent algebraic equations. The key expansion technique using magic square-derived constants adds additional layers of non-linearity and unpredictability. The mathematical properties of magic squares, combined with the multiplicative constants derived from the golden ratio, create complex relationships that are difficult to simplify into solvable algebraic equations. The enhanced RC5 algorithm's resistance to algebraic attacks is significantly improved by the introduced non-linear components. The dynamic S-box generation and magic square-based key expansion techniques increase the complexity and unpredictability of the encryption process, making it highly resistant to algebraic attacks. Table 4 compares the performance of the proposed RC5 enhancements with the original RC5 and other contemporary encryption algorithms in various metrics.

Table 4. Performance Comparison of Encryption Algorithms.

Algorithm	Encryption Time (ms)	Decryption Time (ms)	Avalanche Effect (%)	p-Value	Entropy	Complexity (Time)	Complexity (Space)	Resistance to Cryptanalysis
Original RC5	0.0003	0.0003	45	0.47	7.6	O(t + b + O(r))	O(t + O(1))	Medium
AES	0.0005	0.0005	49	0.51	7.7	O(n^2)	O(n^2)	High
DES	0.0007	0.0007	47	0.5	7.65	O(n^2)	O(n^2)	Medium
Blowfish	0.0006	0.0006	48	0.52	7.7	O(n^2)	O(n^2)	High
Proposed	0.0003	0.0003	50.88	0.553	7.818	O(R+k+O(max(R,k)))	O(R+k+O(256))	High

5. Potential Vulnerabilities

In the previous section, an evaluation of the enhanced approach was presented in terms of security metrics. To support the previous analysis, the potential vulnerabilities will be explained, and how the enhanced approach will mitigate them.

5.1.Linear Cryptanalysis:

- Vulnerability in RC5: RC5's simplicity, its 'worst enemy', could mean that it was open to linear approximations of enough rounds to see it as potentially vulnerable.
- Mitigation by Enhancements: The dynamic nature of S-boxes provides high non-linearity to make it difficult to create linear approximations. Due to the non-deterministic nature of S-boxes, different encryption instances behave differently, thus eliminating one of the ways linear cryptanalysis can work.

5.2. *Differential Cryptanalysis:*

- Vulnerability in RC5: RC5, like other block ciphers, could be susceptible to differential attacks if the differential patterns are predictable.
- Mitigation by Enhancements**:** More so, the Dynamic S-boxes interfere with differential patterns because of their random characteristics in the encryption activity. This high level of confusion added by S-boxes is helpful in the decryption process since it enables the breaking of predictable differential trails.

5.3. Algebraic Attacks:

- Vulnerability in RC5: The simplicity of RC5's operations can sometimes be expressed in algebraic terms, making it a potential target for algebraic attacks.

- Mitigation by Enhancements: The dynamic S-box generation using SHA-256 introduces complex and non-linear equations, making it infeasible to reduce the system to a solvable form. Similarly, the magic square-based key expansion adds further layers of complexity, preventing the formation of simple algebraic equations.

5.4. Side-Channel Attacks:

- Vulnerability in RC5: Standard RC5 does not inherently protect against side-channel attacks.

- Mitigation by Enhancements: While the proposed enhancements do not directly address side-channel attacks, the increased complexity and variability in the encryption process can obscure patterns that might be exploited in such attacks. Additional countermeasures like constant-time implementations can be considered to bolster defense.

5.5. Key Schedule Attacks:

- Vulnerability in RC5: If the key schedule process is predictable, it can be a point of weakness.

- Mitigation by Enhancements: The use of magic square-based constants introduces high entropy and uniqueness in the key schedule. This unpredictability ensures that even with partial knowledge of subkeys, deriving the original key is highly complex and impractical.

New suggested changes to the dynamic S-Box for the RC5 block cipher algorithm and the magic square-based key schedule increases the overall resistance against different kinds of cryptanalytic attacks. Due to the dynamic S-boxes, high non-linearity is maintained as well as unpredictability, which makes linear and differential attacks less efficient. Concerning the key schedule, the magic square-based key expansion generates new and complex constants thus enhancing the key schedule's resistance against key schedule attacks. As a result, these improvements enable the RC5 algorithm to be highly resistant while remaining fast enough to be used in current applications, thus, making it a sound encryption approach.

6. CONCLUSION

This paper has advanced the RC5 block cipher algorithm phenomenally further compared to the work done in literature by including dynamic S-box generation and a key expansion approach grounded on the magic square. Thus, to counter the constantly evolving threats posed by today's cryptanalytic methods, indicated improvements have been incorporated into RC5 in a manner that continues to make the encryption scheme provide a beneficial and secure solution. The dynamic generation of S-boxes employs the SHA-256 hash function to come up with different and extremely strong S-boxes for the different instances of the encryption technique. This method introduces a lot of complexity and non-linearity hence it becomes more resistant to differential and linear cryptanalysis. The above gives rise to the unpredictability of S-boxes which makes RC5 more secure. Moreover, all proposed changes have been put through numerous cryptographic tests. Finally, the results exhibit a forest-fire effect of 50.88% denoting its superior diffusive nature to cascade over nodes in evolving dynamical processes. Tests with p 0.553 and entropy of 7.818 confirm the generation by an enhanced RC5 algorithm that includes better randomness tests than in the conventional version, making ciphertext improve unpredictability results as well. Together, these contributions extend the state of the art in cryptography by providing a proven scalable and deployable technology to strengthen existing cryptographic algorithms against relevant threats.

REFERENCES

[1] H. K. Verma and R. K. Singh, "Enhancement of RC6 block cipher algorithm and comparison with RC5 &amp"; RC6. 2013. Available: https://doi.org/10.1109/iadcc.2013.6514287.

[2] Suresh, Swathi & Varghese, Mary & D, Aju. "An Efficient and Optimized RC5 Image Encryption Algorithm for Secured Image Transmission". International Journal of Imaging and Robotics.2015. 15. 117-125.

[3] R. L. Rivest, "The RC5 encryption algorithm," in Lecture notes in computer science, 1995, pp. 86–96. doi: 10.1007/3-540-60590-8_7. Available: https://doi.org/10.1007/3-540-60590-8_7

[4] E. B. Villanueva, R. P. Medina, and B. D. Gerardo, An enhanced RC5 (ERC5) algorithm based on simple random number key expansion technique. 2018. doi: 10.1109/iscaie.2018.8405458. Available: https://doi.org/10.1109/iscaie.2018.8405458

[5]J. C. N. Vibar, R. P. Medina, and A. M. Sison, "ERC5A – an enhanced RC5 algorithm on bit propagation in the encryption function," 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), Feb. 2019, doi: 10.1109/ccoms.2019.8821685. Available: https://doi.org/10.1109/ccoms.2019.8821685

[6] A. M. S. Et. Al, "Developing RC4 algorithm using S-Box of advanced encryption Standard Cipher," International Journal of Computing and Digital System/International Journal of Computing and Digital Systems, vol. 7, no. 4, pp. 207–214, Jul. 2018, doi: 10.12785/ijcds/070404. Available: https://doi.org/10.12785/ijcds/070404

[7] S. Pal, R. Selvanambi, P. Malik, and M. Karuppiah, "A chaotic system and count tracking mechanism-based dynamic S-Box and secret key generation," International Journal of Mathematical, Engineering and Management Sciences, vol. 8, no. 2, pp. 230–244, Apr. 2023, doi: 10.33889/ijmems.2023.8.2.014. Available: https://doi.org/10.33889/ijmems.2023.8.2.014

[8] A. H. Zahid, M. J. Arshad, M. Ahmad, N. F. Soliman, and W. El-Shafai, "Dynamic S-Box Generation Using Novel Chaotic Map with Nonlinearity Tweaking," Computers, Materials & Continua/Computers, Materials & Continua (Print), vol. 75, no. 2, pp. 3011–3026, Jan. 2023, doi: 10.32604/cmc.2023.037516. Available: https://doi.org/10.32604/cmc.2023.037516

[9] H. R. Shakir, S. A. A. Mehdi, and A. A. Hattab, "A dynamic S-box generation based on a hybrid method of new chaotic system and DNA computing," Telkomnika, vol. 20, no. 6, p. 1230, Dec. 2022, doi: 10.12928/telkomnika.v20i6.23449. Available: https://doi.org/10.12928/telkomnika.v20i6.23449

[10] A. Y. Al-Dweik, I. Hussain, M. Saleh, and M. T. Mustafa, "A novel method to generate key-dependent s-boxes with identical algebraic properties," Journal of Information Security and Applications, vol. 64, p. 103065, Feb. 2022, doi: 10.1016/j.jisa.2021.103065. Available: https://doi.org/10.1016/j.jisa.2021.103065

[11] S. M. Kareem and A. M. S. Rahma, "An innovative method for enhancing advanced encryption standard algorithm based on magic square of order 6," Bulletin of Electrical Engineering and Informatics, vol. 12, no. 3, pp. 1684–1692, Jun. 2023, doi: 10.11591/eei.v12i3.4498. Available: https://doi.org/10.11591/eei.v12i3.4498

[12] M. U. Hassan, A. Alzayed, A. A. Al-Awady, N. Iqbal, M. Akram, and A. Ikram, "A novel RGB image obfuscation technique using dynamically generated all order-4 magic squares," IEEE Access, vol. 11, pp. 46382–46398, Jan. 2023, doi: 10.1109/access.2023.3275019. Available: https://doi.org/10.1109/access.2023.3275019

[13] G. De Las Cuevas, T. Netzer, and I. Valentiner-Branth, "Magic squares: Latin, semiclassical, and quantum," Journal of Mathematical Physics, vol. 64, no. 2, Feb. 2023, doi: 10.1063/5.0127393. Available: https://doi.org/10.1063/5.0127393

[14] G. De Las Cuevas, T. Netzer, and I. Valentiner-Branth, "Magic squares: Latin, Semiclassical and Quantum," arXiv (Cornell University), Jan. 2022, doi: 10.48550/arxiv.2209.10230. Available: https://arxiv.org/abs/2209.10230

[15] S. M. Kareem and A. M. S. Rahma, "Development of data encryption standard algorithm based on magic square," Indonesian Journal of Electrical Engineering and Computer Science, vol. 28, no. 1, p. 297, Oct. 2022, doi: 10.11591/ijeecs.v28.i1.pp297-305. Available: https://doi.org/10.11591/ijeecs.v28.i1.pp297-305

[16] D. F. C. Salas, “Enhanced RC5 Algorithm using Parallel Computing for Communication Networks,” Ingeniería Y Ciencia/Ingeniería Y Ciencia, vol. 15, no. 29, pp. 103–125, Jun. 2019, doi: 10.17230/ingciencia.15.29.4. Available: https://doi.org/10.17230/ingciencia.15.29.4

[17] J. Kelsey, B. Schneier, and D. Wagner, “Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES,” in Lecture Notes in Computer Science, 1996, pp. 237–251. doi: 10.1007/3-540-68697-5_19. Available: https://doi.org/10.1007/3-540-68697-5_19.

[18] J. Daemen and V. Rijmen, “The Advanced Encryption Standard process,” in Information security and cryptography, 2002, pp. 1–8. doi: 10.1007/978-3-662-04722-4_1. Available: https://doi.org/10.1007/978-3-662-04722-4_1

[19] S. Pal, R. Selvanambi, P. Malik, and M. Karuppiah, “A chaotic system and count tracking mechanism-based dynamic S-Box and secret key generation,” International Journal of Mathematical, Engineering and Management Sciences, vol. 8, no. 2, pp. 230–244, Apr. 2023, doi: 10.33889/ijmems.2023.8.2.014. Available: https://doi.org/10.33889/ijmems.2023.8.2.014

[20] A. Y. Al-Dweik, I. Hussain, M. Saleh, and M. T. Mustafa, “A novel method to generate key-dependent s-boxes with identical algebraic properties,” Journal of Information Security and Applications, vol. 64, p. 103065, Feb. 2022, doi: 10.1016/j.jisa.2021.103065. Available: https://doi.org/10.1016/j.jisa.2021.103065

[21] Ejaz, I. A. Shoukat, U. Iqbal, A. Rauf, and A. Kanwal, “A secure key dependent dynamic substitution method for symmetric cryptosystems,” PeerJ. Computer Science, vol. 7, p. e587, Jul. 2021, doi: 10.7717/peerj-cs.587. Available: https://doi.org/10.7717/peerj-cs.587

[22] Alasaad and A. Alghafis, Key-Dependent S-box Scheme for Enhancing the Security of Block Ciphers. IEEE, 2019. doi: 10.1109/icspis48135.2019.9045900. Available: https://doi.org/10.1109/icspis48135.2019.9045900

[23] T. Ara, P. G. Shah, and P. M, Dynamic key Dependent S-Box for Symmetric Encryption for IoT Devices. Bangalore, India: IEEE, 2018. doi: 10.1109/icaecc.2018.8479442. Available: https://doi.org/10.1109/icaecc.2018.8479442

[24] I. M. Alattar and R. S. A. Monem, “A block cipher algorithm developed using Magic Square in the seventh Order,” Journal of Physics. Conference Series, vol. 1999, no. 1, p. 012119, Sep. 2021, doi: 10.1088/1742-6596/1999/1/012119. Available: https://doi.org/10.1088/1742-6596/1999/1/012119

[25] M. Sahni and D. B. Ojha, “MAGIC SQUARE AND CRYPTOGRAPHY,” Journal of Global Research in Computer Science, vol. 3, no. 12, Dec. 2012, Available: https://www.rroij.com/open-access/magic-square-and-cryptography-15-17.pdf