



Expeditiousness Serpent Using CTR Mode and Logistic Map

Huwaida T. Elshoush  

Computer Science Department
Faculty of Mathematical Sciences and Informatics
University of Khartoum, Sudan
htelshoush@uofk.edu

Batool H. Abdallah

Computer Science Department
Faculty of Mathematical Sciences and Informatics
University of Khartoum, Sudan
Batoolhussein3@yahoo.com

Duaa M. Ahmed

Computer Science Department
Faculty of Mathematical Sciences and Informatics
University of Khartoum, Sudan
duaatom@icloud.com

Abdalmajid A. Ishag

Computer Science Department
Faculty of Mathematical Sciences and Informatics
University of Khartoum, Sudan
Majid.fms010@gmail.com

Sittana O. Affi

Applied Mathematics Department
Faculty of Mathematical Sciences and Informatics
University of Khartoum, Sudan
Sittanams@hotmail.com

Received September 2022; revised November 2022

ABSTRACT. *This paper presents an approach to enhance the execution performance and security of the Serpent algorithm. Our proposed method $Serpent_{CTR-LogisticMap}$ used CTR encryption mode with multi-threading to allow the parallel execution of the algorithm. In addition, we made a function which generates the encryption key automatically using Logistic map. This adds more robustness to the algorithm because of the intricate of the key. The proposed method was implemented using Python 3.9, and the comprehensive experiments clearly show that the proposed $Serpent_{CTR-LogisticMap}$ algorithm achieved superior reduction rate in execution time of up to 91%. From the security facet, the experimental results demonstrated the effectiveness of the proposed method against brute force attacks. Moreover, NIST Statistical Test Suite was used to measure the randomness of the proposed method, and the results affirms its efficacy. In particular, compared to prevailing schemes, it proclaimed its effectiveness.*

Keywords: Serpent, Logistic Map, Parallel Computing, Multithreading

1. **Introduction.** Encryption is an ancient technology for securely transmitting and receiving data. The encryption provides data confidentiality, anonymity and authentication, integrity to secure data from attackers.

In January 1997, the National Institute of Standards and Technology (NIST) announced the development of a Federal Information Processing Standard (FIPS) for Advanced Encryption Standard (AES). It was the beginning of an effort to replace the Data Encryption Standard (DES). Call for AES candidate algorithms resulted in 15 candidate algorithms. Five of them have been selected as AES finalists. Serpent was one of them [1–3].

Serpent is a symmetric block cipher, where a 128-bit block is ciphered utilizing a 256-bit key using 32 different rounds. The first 31 rounds are identical, consisting of the same sequence of elementary operations, while the last round differs only in the key schedule. Instead of mixing a single key like in the first 31 rounds, an additional key is mixed in the last round. Hence, 33 round keys are required in the whole process that are generated from the external key [4–9]. Figure 1(a) represents the Serpent encryption process.

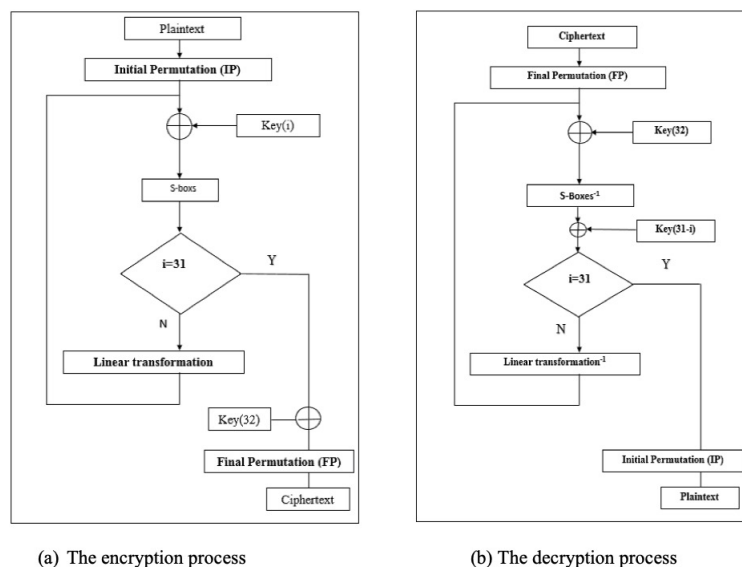


FIGURE 1. The Serpent Algorithm

The algorithm consists of three basic functions, namely Initial Permutation (IP), 32 Round function and Final Permutation (FP). IP is applied on plain-text in order to rearrange the bits. It is given by $(i * 32) \bmod 127$. Applying this to a plain-text, produces a data block B_0 . The round function is performed 32 times on B_i . The algorithm defines eight S-boxes (S_i). Inside these rounds, each data block B_i is mixed with a sub key K_i (i.e. taking XOR), then $B_i \oplus K_i$ is passed through $S_{i \bmod 8}$ which is one of the eight S-boxes. After this, a linear transformation to the $S_i(B_i \oplus K_i)$ is applied to get B_{i+1} , where $i = 0, 1, 2, \dots, 30$. In the 32nd round (i.e. last round), a 33rd key is XOR-ed instead of applying a linear transformation (LT). Now the final permutation $(i * 4) \bmod 127$ is applied to get the cipher text. The whole process is described shortly as [8–12]:

$$\begin{aligned}
 B_0 &= IP(p) \\
 B_{i+1} &= LT(S\text{-box}_i \bmod 8(B_i \oplus K_i)) \\
 \text{where } i &= 0, 1, 2, \dots, 30 \\
 B_{32} &= S_7(B_{31} \oplus K_{31} \oplus K_{32}) \\
 C &= FP(B_{32})
 \end{aligned}$$

Concerning the decryption, the inverse S-boxes, the inverse linear transformation and reverse order of sub keys are used. Figure 1(b) represents the whole decryption process of Serpent Algorithm [5][9].

There are different encryption modes for block ciphers. Specifically, in counter mode (CTR) no single block of cipher is dependent upon a previously calculated block [13, 14]. Each block can be calculated independently. Therefore, this could be paralleled by splitting the cipher into chunks for each individual thread to encrypt or decrypt. Hence, speeding up the process.

The idea of parallelism was suggested by Pendli et al [15] to enhance the execution time of AES algorithm on multiple cores processors. Their experimental results achieved considerable reduction in time, and hence was recommended to be used with similar algorithms. Nagendra et al [16] also proposed enhancing AES performance using parallelism.

The commencement of this work is precisely to enhance the performance and security of Serpent. Ergo, the proposed approach improves the execution time of Serpent by using multithreading to accelerate the speed of encryption and decryption processes using CTR mode [8]. Moreover, the proposed approach enhances the security by generating different keys for each block using Logistic Map. The execution performance of the proposed method is then compared with the traditional Serpent and some recent related works.

Our contribution can be summed up as follows:

- Expediting the Serpent by dividing the input into blocks and further producing Logistic Map chaos-based sub-keys for every block, with parallel CTR mode implementation.
- Strengthen the Serpent from a security aspect, by creating Logistic Map chaos-based sub-keys for each block which clearly adds more strength to the algorithm due to the complexity of the sub-keys. Besides, using a different key for every distinct block conceals plaintext patterns.
- Testing the proposed method manifests its efficacy by being speedy and secure.
- Moreover, our proposed method gave superior results compared to the state-of-the-art schemes.

The remainder of the paper is structured as follows: the recent schemes in enhancing the performance and security of the Serpent are examined in the next section. Section 3 elucidates the proposed method. The experimental results and analysis are represented in section 4. Finally, section 5 concludes the paper and recommends some future work.

2. Related Work. Here are some improvements of Serpent from functionality aspect.

Early in 2016, Singh et al [17] suggested an algorithm named Block Key Generation (BKG) that generated multiple keys for every block from single user defined key. Hence, each block has a different key and is not depending on the output of the previous blocks. This eliminates any data pattern and lets each block to run independently.

Kadhim et al [18] suggested a new approach for Serpent algorithm by depending on modifying the process and round as the traditional Serpent has more time and simple complexity. This algorithm is used to encrypt wave media. Their paper has a new structure that has many techniques, such as Mix Column, S-box, IP, and IP-1, in order to increase the complexity. As a result, they found that the consumed execution time for Serpent is more than traditional block ciphers because the S-boxes need more time, so they present a new idea for S-box which consumes less time compared with the original S-box. Also, the new structure of Serpent can be used to encrypt text and sound files (Multi input). In addition, the complexity of Serpent can increase by using new functions in new structures. Versions of the S-box instructions are presented, requiring only 5 registers and also utilizing parallelism.

To enhance the performance of Serpent, Elkamchouchi et al [19] used chaotic mapping and cycling group instead of S-box. The number of rounds is hence reduced to 10. Moreover, from security facet, chaotic map adds complexity to the key, hence strengthen Serpent more.

Farooq et al [12] improved the Serpent Algorithm computationally and algebraically making it compatible in different usages like Rijndael Algorithm. Their method uses 4×4 S-box, which consists of bytes instead of nibbles, and is constructed through the multiplicative group of finite commutative chain rings. Furthermore, all the operations in the work coincides with the operations of the commutative chain ring. They modified the Serpent algorithm by using different S-boxes and the cipher in a way that it satisfies the AES requirements and is even better than the Serpent algorithm. The modification occurs in the form that it become 31% less complex and faster than the original algorithm.

Yousif [20] propound modifying Serpent algorithm by replacing the static permutation and substitution with dynamical properties using logistic chaos map and standard map. The proposed dynamic initial permutation (IP), final permutation (FP), substitution box (S-box), and the generated round keys give the best randomness compared with classical Serpent algorithm and also can reduce the number of rounds and time usage. Their method has sensitivity to any change in the key since it uses chaotic map in the key round generation. And also, is more robust than classical Serpent since it uses another chaotic map to create dynamic permutation and substitution.

Zagi et al [21] proffer to improve and support the confidentiality of data while adhering to the external structure of the standard algorithm, relying on designing a new approach to the key generation function because the sobriety of block cipher relies on the use of a strong and unique key. Several functions are used, such as Gost external structure, with a combination of Shift \lll , AES key schedule, and MD5. The developed Serpent algorithm aims to preserve the general structure and main layers of the standard algorithm. The results of the modified algorithm appear more complicated and random and enhance the spread and confusion properties of the data. Their new method is used to encrypt all types of data. The diffusion and confusion property of the bits of the modified Serpent was tested using five metrics: five basic statistical tests, the NIST test set, coding runtime, brute force attack and analytical attack. Concerning the NIST test, the result sub-keys passed all the test and achieved a full spread for bits.

Recently in 2022, Elshoush et al [9] suggested improving Serpent by running it in parallel using EBC mode and further a key is generated for every block using Lorenz 96 chaotic map. Their method achieved a reduction in execution time by 53.2% compared to traditional Serpent, whilst our proposed method attained up to 91% using five threads.

3. The Proposed Expeditiousness Serpent_{CTR-LogisticMap}

3.1. Automated Key Generation Using Logistic Map. Chaos in dynamical systems has been investigated over a long period of time. With the advent of fast computers, the numerical investigations on chaos have increased considerably over the last two decades and by now, a lot is known about chaotic systems [22–26]. One of the simplest and most transparent system exhibiting order to chaos transition is the logistic map [27].

The logistic map is a polynomial mapping (equivalently, recurrence relation) of degree 2, often cited as an archetypal example of how complex, chaotic behavior can arise from very simple non-linear dynamical equations. The map was popularized in a 1976 paper by the biologist Robert May, in part as a discrete-time demographic model analogous to the logistic equation written down by Pierre François Verhulst. Mathematically, the logistic map is written as in equation 1 [27, 28]:

$$x_{i+1} = \mu x_i (1 - x_i) \quad (1)$$

where x_n is a number between zero and one that represents the ratio of the existing population to the maximum possible population.

This nonlinear difference equation is intended to capture two effects [27]:

- reproduction where the population will increase at a rate proportional to the current population when the population size is small.
- starvation (density-dependent mortality) where the growth rate will decrease at a rate proportional to the value obtained by taking the theoretical "carrying capacity" of the environment less the current population.

In the proposed expeditiousness $\text{Serpent}_{CTR-LogisticMap}$, we applied an auto generation of the encryption key using the logistic map function using equation 2:

$$X_{i+1} = R \times X_i \times (1 - X_i) \quad (2)$$

which receives two initial values x and r , that are generated using a random number generation function as shown in algorithm 1:

Algorithm 1: Generating encryption key using Logistic Map

Output: Key: Encryption key

```

1 Function LogisticMap(key, blockno):
2    $x \leftarrow \text{randomnumber}(0, 1)$  ; // generating the initial value x
3    $r \leftarrow \text{randomnumber}(0, 4)$  ; // generating the initial value r
4   for  $i$  of  $x$  do
5      $x_i \leftarrow r * x(1 - x)$  ; // Logistic map
6      $key \leftarrow \text{converkeytohexa}$  ; // key=(x*10pow16)/256
7   Return: Key

```

3.2. Encryption using the Proposed $\text{Serpent}_{CTR-LogisticMap}$. The proposed expeditiousness $\text{Serpent}_{CTR-LogisticMap}$ uses CTR encryption mode. In the CTR encryption mode, blocks are independent of one another, once the initial vectors have been generated, both encryption and decryption of blocks can be done in parallel. Furthermore, the CTR mode is considered to be very secure and efficient for most purposes. So we made a new function which encrypt and decrypt the plain-text and cipher text respectively using the method of the CTR encryption mode. In addition, to boost the parallelism, we used threads to get more efficiency from the modified algorithm. However, one problem with the CTR is that a synchronous counter needs to be maintained at both receiving and sending ends. Losing track of this counter could lead to incorrect recovery of plain-text. Algorithm 2 demonstrates the encryption process using CTR [29]. The proposed method split-up the input data into fixed-length blocks according to the Serpent block size, which is 128 bit. Finally, the ciphered blocks are united together to form the final cipher text.

3.3. Decryption using the Proposed $\text{Serpent}_{CTR-LogisticMap}$. Algorithm 3 presents the steps taken to decrypt a cipher text using the proposed $\text{Serpent}_{CTR-LogisticMap}$ with multithreading. As in algorithm 2, the proposed method also split the cipher text data into fixed 128 bit blocks, and then finally, the produced blocks are merged together to form the final plaintext.

Algorithm 2: Encryption process using CTR

```

Input:  $P, K, IV, NT, E - IV$  // P=planetext; K=Encryption key;
          IV=initial-vector; E-IV=encrypted IV NT=number of threads
Output:  $C, K$ , // LC=cipher-text ; K=Encryption key
1 Function Encrypt-CTR( $R, IP, SIP, PNO$ ):
2    $nonce \leftarrow IV$ 
3   Function Split-blocks( $P, NT$ ):
4     ReturnListofblocks
5     for each block do
6       encrypt( $K, IV$ ) ReturnE - IV
7       Xor (block, E-IV) ReturnC
8       for each  $C$  do
9         | join ( $c$ )
10      end
11     end
12   Return : LC
13 End Function

```

Algorithm 3: Decryption process using CTR

```

Input:  $C, K, IV, NT, d - IV$  // C=cipher-text; K=Encryption key;
          IV=initial-vector; NT=number of threads; D - IV=decrypted IV
Output:  $C, K$ , // LP=Plain-text ; K=Encryption key
1 Function Decrypt-CTR( $R, IP, SIP, PNO$ ):
2    $nonce \leftarrow IV$  Function Split-blocks( $C, NT$ ):
3     | ReturnListofblocks
4     for each block do
5       dncrypt( $K, IV$ ) ReturnE - IV Xor (block, E-IV) ReturnP
6       | join ( $P$ )
7     end
8     end
9     Return:  $LP$ 
10 End Function

```

4. Experimental Results and Analysis. This section presents series of experiments to evaluate the performance and demonstrating the efficiency of the proposed expeditiousness $Serpent_{CTR-LogisticMap}$ in relation to the execution time, and comparing its performance with the traditional Serpent. From the security aspect, the key space of the proposed method has been investigated. Moreover, comparisons to related schemes are also conducted.

4.1. Preliminaries. The proposed $Serpent_{CTR-LogisticMap}$ and traditional Serpent were implemented using Python 3.9. All the experimental results were tested on a laptop with Windows 64bit OS with 4 GB RAM, Core i5 processor with 2.20 GHz speed.

4.2. Performance testing. Performance testing is conducted to know the proposed $Serpent_{CTR-LogisticMap}$ performance compared to the traditional Serpent and some prevailing recent work. Effectively, the system is tested under multiple and different environments and it meticulously checks the time taken by the proposed method to respond

under varying block sizes and varying number of threads. Hence, it shows how it performs in terms of responsiveness and stability when implemented under differing conditions.

4.2.1. *The Encryption/Decryption Process Performance Using Two Threads.* Here, the Proposed Serpent_{CTR-LogisticMap} performance is tested and compared with the traditional Serpent using varying number of blocks and two threads. Table 1 and figure 2 clearly show the efficacy of the proposed method over the traditional Serpent in terms of encryption execution time. Likewise, table 2 and figure 3 presents the performance comparison of the decryption process using the proposed method and the traditional Serpent, affirming the performance efficiency of the proposed method.

# of blocks	Traditional Serpent	Proposed Serpent _{CTR-LogisticMap}	Time Execution Reduction rate in %
500000	17697.33	15432.955	13%
1000000	34688.72	29873.57	14%
1500000	60182.26	45884.81	24%
2000000	66056.65	61993.94	6%

TABLE 1 Encryption Execution Time for the Proposed Serpent_{CTR-LogisticMap} Compared with Traditional Serpent Using Two Threads

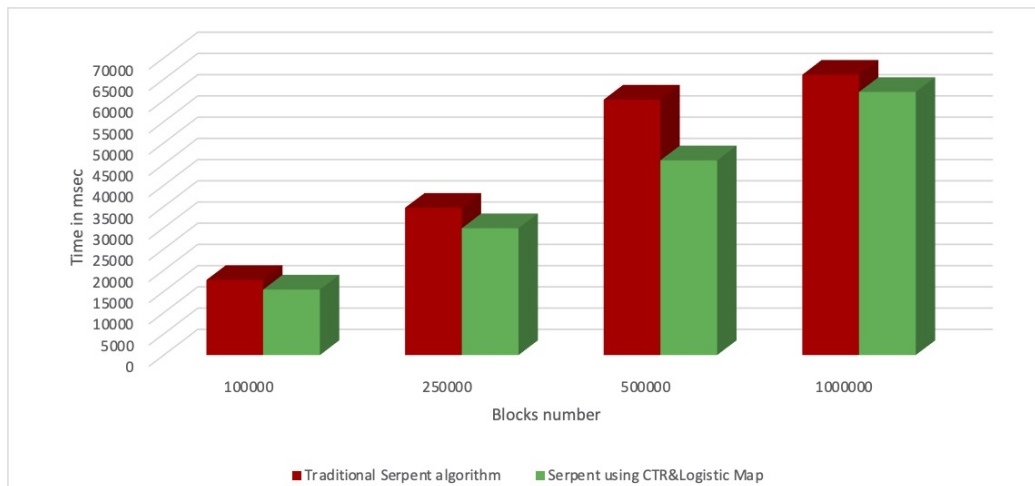


FIGURE 2. The Encryption Process Performance of the Proposed Serpent_{CTR-LogisticMap} Compared with Traditional Serpent Using Two Threads

# of blocks	Traditional Serpent	Proposed Serpent _{CTR-LogisticMap}	Time Execution Reduction rate in %
500000	15872.51	14394.5	9%
1000000	30840.49	27791.02	10%
1500000	50218.26	42612.92	15%
2000000	61638.85	57570.13	7%

TABLE 2 Decryption Execution Time for the Proposed Serpent_{CTR-LogisticMap} Compared with Traditional Serpent Using Two Threads

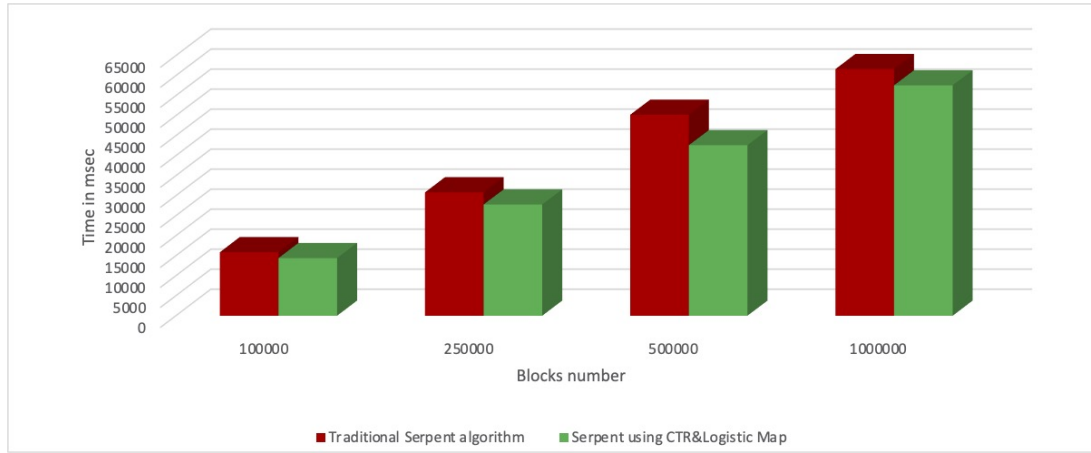


FIGURE 3. The Decryption Process Performance of the Proposed Serpent_{CTR-LogisticMap} Compared with Traditional Serpent Using Two Threads

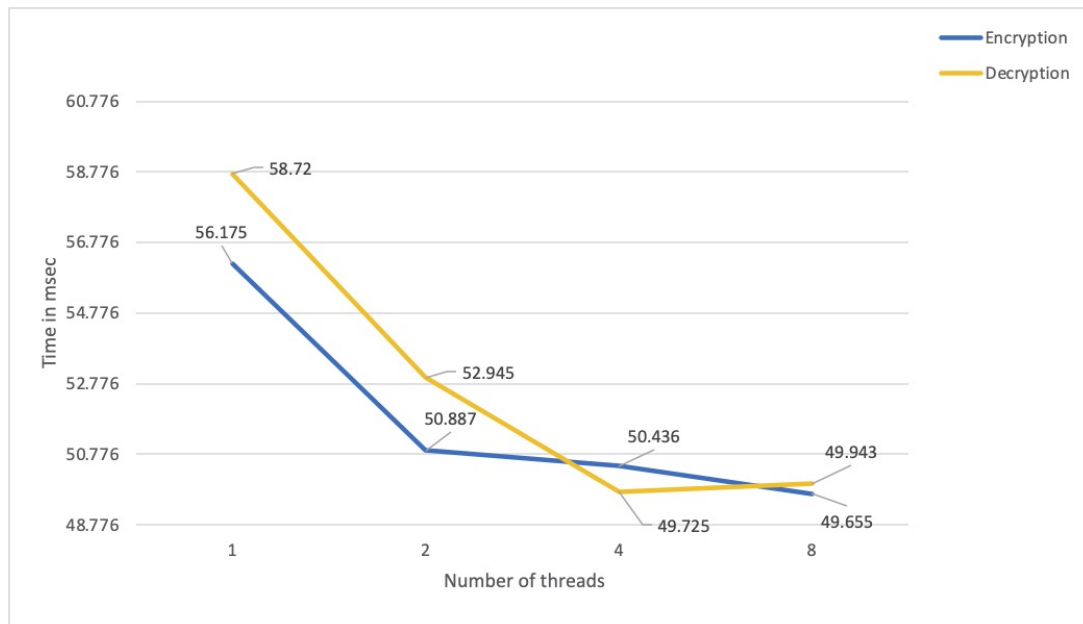


FIGURE 4. The Encryption and Decryption Execution time of the Proposed Serpent_{CTR-LogisticMap} Using Different Number of Threads

4.2.2. *The Performance of the Proposed Serpent_{CTR-LogisticMap} Using Varying number of Threads.* The graph of figure 4 presents the effect of varying the number of threads (from 1 up to 8 threads) in the encryption and decryption execution time for the proposed Serpent_{CTR-LogisticMap}. Blatantly, the more number of threads used, the less is the execution time, and hence the better is the performance.

4.2.3. *Performance Results of the Proposed Serpent_{CTR-LogisticMap} Using Five number of Threads.* Table 3 illustrates the execution time for the proposed Serpent_{CTR-LogisticMap} compared to the traditional Serpent algorithm using varying number of blocks with five threads. Patently, the performance test revealed that the proposed method is more efficient for both encryption and decryption processes. Moreover, the execution time reduction rate reaches up to 91%, hence asserting the effectiveness of the proposed method.

# of blocks	Traditional Serpent		Proposed Serpent _{CTR-Logisticmap}		Execution Time Reduction rate in %	
	Enc	Dec	Enc	Dec	Enc	Dec
100	3.760	3.496	3.337	2.939	11%	16%
1000	36.431	33.365	32.672	30.213	10%	9%
10000	354.66	330.47	339.90	324.95	4%	2%
100000	3469.41	3291.66	336.88	309.51	90%	91%

TABLE 3 Execution time Comparison of the Proposed Serpent_{CTR-LogisticMap} Compared with Traditional Serpent Using Five Threads

Its worth noted that execution time reduction rate is much better using five number of threads than using only two threads as shown in tables 2 and 3. Ergo, using more number of threads in parallelism, the superior is the performance.

4.3. Security Analysis.

4.3.1. *Key Space Analysis.* From the security aspect, the key space of the proposed method is analyzed and is given by the following equation 3:

$$n \times 2^{256} \quad (3)$$

where n is the number of blocks.

The traditional serpent, on the other hand, has a key space of only 2^{256} , which is incomparable to our proposed method. Furthermore, the more number of threads used, the better execution time as demonstrated in figure 4.

4.3.2. *The Security of the Logistic Map.* The proposed method enhances the security by generating sub keys for each block using the Logistic Map, which is chaos-based. Chaotic systems are well suited to encryption and secure transmission because of its characteristics of being unpredictable, random, ergodic and high sensitive to preliminary conditions [8]. Hence, adding complexity to the Serpent. Moreover, all block keys can be generated prior to the initiation of the Serpent, hence encrypting each block in parallel and thus hide plaintext patterns. Furthermore, running the enhanced Serpent in parallel CTR mode makes the algorithm expeditious.

4.3.3. *The Randomness of the Proposed Method Serpent_{CTR-LogisticMap}.* The randomness property of the proposed Serpent_{CTR-LogisticMap} is tested using the Statistical Test Suite (STS) recommended by the NIST [30], which ensures that the output is statistically indistinguishable when a random output is used. This NIST suite consists of 15 distinct tests, where a probability (P-Value) is calculated for each test. A method passes any specific test if the P-Value is in the range $0.01 \leq P \leq 1$. As specified in the guidelines of the NIST, the null hypothesis is that the sequence being tested is random. The tests have been conducted on a significance level of 0.01, which signifies that the probability of rejecting the null hypothesis while it is true is 0.01 [30]. Figure 5 manifests that the proposed method passed all tests successfully, and thus affirms the efficacy of the proposed method.

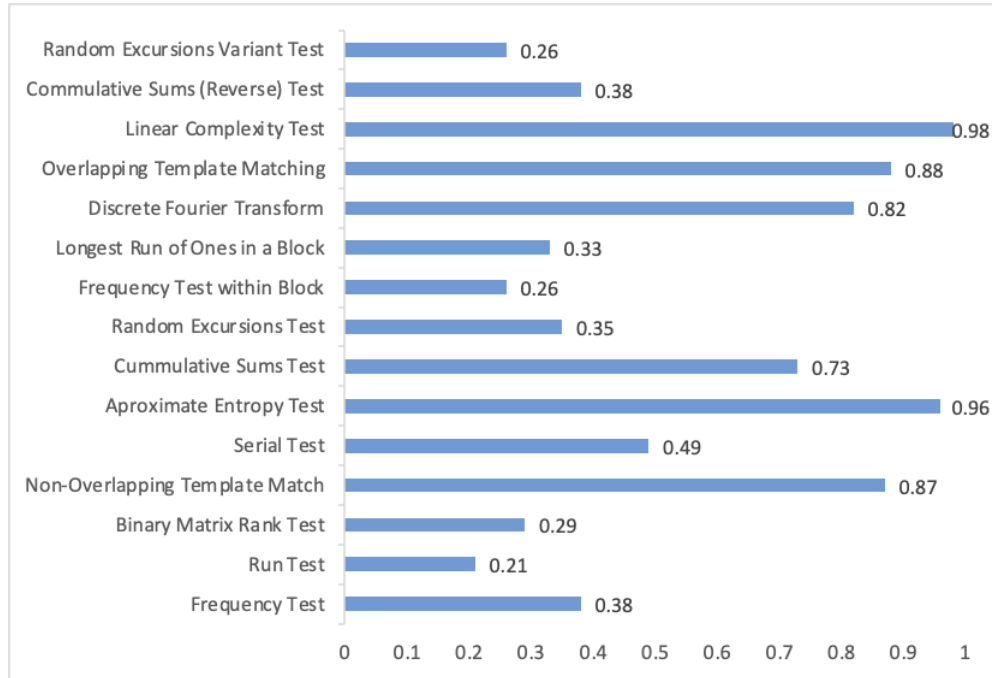


FIGURE 5. Examining the randomness of the Proposed Serpent_{CTR-LogisticMap} using NIST statistical test suite

4.4. Comparison with State-of-the-Art. The proposed Serpent_{CTR-LogisticMap} is further compared in terms of the reduction in execution time with the works of Pendli et al [15], Rahmah et al [21] and Elshoush et al [9]. Obviously, as illustrated in table 4, it is blatant that the proposed method prove superior to the prevailing enhanced Serpent schemes by attaining a reduction rate of up to 91% using five threads, whereas [9] achieved 48.1 - 53.2%, [15] achieved a reduction of 40 - 45% and [21] a 20.6 - 23.8% reduction in execution time.

TABLE 4 Comparison of the Time Execution Reduction Rate in % for different Serpent Enhanced Schemes

Reference	Algorithm	Time Execution Reduction rate in %
Pendli et al [15] 2016	AES	40 - 45%
Rahmah et al [21] 2020	Serpent	20.6 - 23.8%
Elshoush et al 2022 [9]	Serpent	48.1 - 53.2%
Proposed Method	Serpent	up to 91%

5. Conclusion. A new method to expedite the Serpent algorithm is proposed. The proposed Serpent_{CTR-LogisticMap} method uses CTR encryption mode enabling the parallel execution of the data blocks. Specifically, distinct sub-keys were produced for every different block using random key generator through the Logistic Chaotic Map, hence concealing plaintext patterns. Moreover, adding more security due to the complexity of the keys. Furthermore, the sub-keys may be generated prior to the execution of the algorithm, ergo speeding it up. In addition to the Logistic Map randomness key generation, an exceedingly large key space contributes to higher security and assured the strength against brute force attacks. Moreover, NIST Statistical Test Suite was used to evaluate the randomness of the proposed method, and it passed all tests successfully ratifying its

efficacy. The proposed method was tested with respect to execution time and the experimental results accomplished affirmative results in reduction rates of up to 91%. Ergo, the effectual results of the proposed method compared to the traditional Serpent gave reassurance of its efficacious and affirms its superiority to existing schemes. Therefore, the proposed Serpent_{CTR-LogisticMap} proclaimed its effectiveness.

For future work, we recommend to implement the proposed method in encrypting images, as it is reckon to attain great results.

REFERENCES.

- [1] D. A. Osvik, "Speeding up Serpent," presented at the *AES Candidate Conf.*, pp. 317–329, Citeseer, 2000.
- [2] K. Kabilan and M. Saketh, and K. K. Nagarajan, "Implementation of Serpent Cryptographic Algorithm for Secured Data Transmission," *IEEE*, pp. 1-6, 2017.
- [3] R. Anderson and E. Biham, and L. Knudsen, "The Case for Serpent," presented at the *AES Candidate Conf.*, 2000.
- [4] S. Simha, Prathibha and Prof. H. Priya, "Enhancing Cloud Security with the Implementation of Serpent Encryption Algorithm," *Imperial Journal of Interdisciplinary Research*, vol. 3, no. 5, 2017.
- [5] H. R. Zagi and A. T. Malood, "A Novel Serpent Algorithm Improvement By The Key Schedule Increase Security," *Tikrit Journal of Pure Science*, vol. 25, no. 6, 2020.
- [6] M. Tayel, G. Dawood, and H. Shawky, "A Proposed Serpent-Elliptic Hybrid Cryptosystem For Multimedia Protection," presented at the *2018 Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI)*, *IEEE*, pp. 387-391, 2018.
- [7] Y. H. Ali, and H. A. Rissan, "Image Encryption Using Block Cipher Based Serpent Algorithm," *Engineering and Technology Journal*, vol. 34 (2 Part (B) Scientific), pp. 278–286, University of Technology, 2016.
- [8] H. T. Elshoush, B. M. Al-Tayeb, and K. T. Obeid, "Enhanced Serpent algorithm using Lorenz 96 Chaos-based block key generation and parallel computing for RGB image encryption," *PeerJ Comput. Sci.*, 2021. 7:e812 <http://doi.org/10.7717/peerj-cs.812>
- [9] H. T. Elshoush, K. T. Obeid, and M. M. Mahmoud, "A Novel Approach to Improve the Performance of Serpent Algorithm using Lorenz 96 Chaos-based Block Key Generation," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 13, no. 1, Mar. 2022
- [10] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A candidate block cipher for the Advanced Encryption Standard," *Página oficial do SERPENT*, disponível em <http://www.cl.cam.ac.uk/~rja14/serpent.html>, 2005.
- [11] M. Naeemabadi, B. S. Ordoubadi, A. M. Dehnavi and K. Bahaadinbeigy, "Comparison of Serpent, Twofish and Rijndael encryption algorithms in teleophthalmology system," *Advances in Natural and Applied Sciences*, vol. 9, no. 4, pp. 137-149, Apr. 2015.
- [12] T. Shah, T. ul Haq, and G. Farooq, "Serpent Algorithm: An Improvement by 4×4 S-Box from Finite Chain Ring," in *Proc. 2018 Int. Conf. on Applied and Engineering Mathematics (ICAEM) IEEE*, 2018, pp. 1-6.
- [13] A. Altigani, M. Abdelmagid, and B. Barry, "Analyzing the Performance of the Advanced Encryption Standard Block Cipher Modes of Operation: Highlighting the National Institute of Standards and Technology Recommendations," *Indian Journal of Science and Technology*, vol. 9, no. 28, 2016.

- [14] D. Bujari and E. Aribas, "Comparative Analysis of Block Cipher Modes of Operation," presented at the *Int. Advanced Researches & Engineering Congr.*, 2017 <http://iarec.osmaniye.edu.tr/>. Osmaniye/TURKEY 16-18 November 2017.
- [15] V. Pendli, M. Pathuri, S. Yandrathi, and A. Razaque, "Improving performance of Advanced Encryption Standard algorithm," in *Proc. Mobile and Secure Services (MobiSecServ), 2016 2nd Int. Conf. on*, IEEE, 2016, pp. 1-5.
- [16] M. Nagendra and M. C. Sekhar, "Performance improvement of Advanced Encryption Algorithm using parallel computation," *Int. Journal of Software Engineering and Its Applications*, vol. 8, no. 2, pp. 287-296, 2014.
- [17] H. Singh and P. Singh, "Enhancing AES using Novel Block Key Generation Algorithm and Key Dependent S-boxes," *Cyber-Security and Digital Forensics*, vol. 30, 2016.
- [18] Prof. A. F. Kadhim, S. Hassin, and G. Ali, and I. Ahmed, "New Approach for Serpent Block Cipher Algorithm Based on Multi Techniques," *Iraqi Journal of Information Technology*, Iraqi Association of Information, vol. 7, no. 3, pp. 1-13, 2017.
- [19] H. M. Elkamchouchi, A. E. Takieldeem, and M. A. Shawky, "A Modified Serpent Based Algorithm for Image Encryption," in *Proc. National Radio Science Conf. (NRSC), 2018 35th*, IEEE, vol. 25, no. 6, pp. 239-248, 2018.
- [20] I. A. Yousif, "Proposed a Permutation and Substitution Methods of Serpent Block Cipher," *Ibn AL-Haitham Journal For Pure and Applied Science*, vol. 32, no. 2, pp. 131-144, 2019.
- [21] H. R. Zagi and A. T. Maalood, "A Novel Serpent Algorithm Improvement by the Key Schedule Increase Security," *Tikrit Journal of Pure Science*, vol. 25, no. 6, pp. 114-125, 2020.
- [22] R. Matthews, "On the Derivation of a "Chaotic" Encryption Algorithm," *Cryptologia*, Taylor & Francis, vol. 13, no. 1, pp. 29-42, 1989.
- [23] Z. Lin, G. Wang, X. Wang, and S. Yu, and J. Lü, "Security Performance Analysis of a Chaotic Stream Cipher," *Nonlinear Dynamics*, Springer, pp. 1-15, 2018.
- [24] K. Audhkhasi, "Chaos Based Cryptography," *CiteSeerX Scientific Literature Digital Library and Search Engine*, 2009.
- [25] L. Kocarev, "Chaos-based Cryptography: A Brief Overview," *IEEE Circuits and Systems Magazine*, IEEE, vol. 1, no. 3, pp. 6-21, 2001.
- [26] D. Xiao, X. Liao, and S. Deng, "One-way Hash Function Construction Based on the Chaotic Map with Changeable-parameter," *Chaos, Solitons & Fractals*, Elsevier, vol. 24, no. 1, pp. 65-71, 2005.
- [27] S. C. Phatak and S. S. Rao, "Logistic map: A possible random-number generator," *Physical Review E*, vol. 51, no. 4, pp. 3670, 1995.
- [28] Lin You, Jiawan Wang and Bin Yan, "A Secure Finger Vein Recognition Algorithm Based on MB-GLBP and Logistic Mapping," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 7, No. 6, pp. 1231-1242, November 2016
- [29] H. Lipmaa, and P. Rogaway, and D. Wagner, "CTR-mode encryption," *Citeseer. MD*, vol. 39, pp. 114-125, 2000.
- [30] M. M. Mahmoud and H. T. Elshoush, "Enhancing LSB Using Binary Message Size Encoding for High Capacity, Transparent and Secure Audio Steganography—An Innovative Approach," *IEEE Access*, vol. 10, pp. 29954-29971, 2022, doi: 10.1109/ACCESS.2022.3155146.