ISSN: 2182-2069 / E-ISSN: 2182-2077

Lightweight Image Encryption Using Chacha20 and Serpent Algorithm

Hayder Najm[1*], Mohammed Salih Mahdi[2], and Wijdan Rashid Abdulhussien[3]

[1*]Department of Computer Techniques Engineering, Imam Alkadhim University College, Baghdad, Iraq. haidernajem@iku.edu.iq, https://orcid.org/0000-0001-9722-4542

[2]Business Information College, University of Information Technology and Communications, Baghdad, Iraq. mohammed.salih@uoitc.edu.iq, https://orcid.org/0000-0003-3177-5469

[3]Department of Information Technology, University of Thi-Qar, Baghdad, Iraq. wijdan_rashid@utq.edu.iq, https://orcid.org/0000-0001-6804-5553

Received: July 19, 2024; Revised: August 28, 2024; Accepted: September 30, 2024; Published: November 30, 2024

Abstract

Data security is prominent today primarily due to the numerous applications of digital images in the contemporary world. Innovations in lightweight encryption are being perceived as the solutions that would allow enhancing the levels of protection while incurring minimal impact on the size of the data and the speed of the processes. The lightweight encryption of images having lower complexities is a good choice in different suites, namely, in cloud computing systems and using social networks for communication. All users will not have to worry about their data being hacked when sharing the images. This paper presents a new lightweight image encryption technique that utilizes the ChaCha20 keys generator with the 16-round Serpent algorithm to provide a lightweight encryption process. The security analysis findings indicate that the suggested cipher is capable of withstanding both known and chosen plaintext attacks. Additionally, it exhibits a high degree of unpredictability, a significant level of security, and a strong sensitivity to key variations. The assessment of the encryption methods is done with the assistance of equipment through histogram analysis, entropy calculation, correlation analysis, mean squared error measurement, peak signal-to-noise ratio calculation, unified average change intensity assessment, normalized pixel difference rate analysis, and structural similarity index. As stated by experimental results, the proposed encryption method performs very well when adopted by various measuring instruments, the value of the information entropy equals to 7.98, which is nearly equivalent to the perfect value of 8. In addition, the value obtained for the normalized pixel difference rate is greater than 99.55%, with the unified average changing intensity equal to 30.11. As it has been mentioned, the decryption process can work very well.

Keywords: Data Security, Image Encryption, Lightweight, Block Cipher, Stream Cipher, ChaCha20, Serpent.

*Corresponding author: Department of Computer Techniques Engineering, Imam Alkadhim University College, Baghdad, Iraq.

1 Introduction

Due to the acceleration of technological advancement in recent decades, individual privacy and information security have transformed into an individual's responsibility (Ullah et al., 2022). Information security can also be useful for ensuring the preservation of information in the world of the Internet, as the latter is numerous today and many sensitive data are transferred through it (Bahrami & Naderi, 2012). The constant influence and main connecting element in digital media is privacy and security questions which became evident by hacking cases of the last decade which involved unauthorized use and exploration of the material culminating in incessant leaks and changes (Mahdi et al., 2020).

In the recent past, several modes of encryption have been developed to enable images to be well-stored yet secure. Nevertheless, the methods frequently share the following structure: the permutation and propagation phases of decryption (Salman et al., 2019). Out of these, most algorithms respond to two; flexibilityy and inadequate security. However, these encryption methods were not effective for images that possessed certain parameters such as high retention rates, high level of redundancy, and the existence of a robust relationship between the pixels. Additionally, the size of the images is invariably larger than objects that require filtering (Mahdi & Hassan, 2018). This means that an extra process is added to the image data and this leads to more computational time and a high utilization of the CPU This makes the encryption and decryption time very slow in real-time transactions and sometimes images take a long time (Najm, 2021).

Due to current technological progress in social media platforms, a large number of photographs are shared online regularly. Ensuring the protection of these photographs from unauthorized access is of utmost importance for authorized individuals and different fields to address their specific security concerns (Leu, 2011). Biology, military sciences, banking, and online shopping are among several fields worth mentioning (Tian & Su, 2022). In such instances, the information conveyed through visuals differs significantly from the information conveyed through text. Images exhibit increased data redundancy and superior information coverage, along with a heightened correlation between neighboring pixels (Peng et al., 2004). In addition, they necessitate real-time and resilient security measures for communication (Anastasiia et al., 2024). Therefore, a variety of methods are employed to protect sensitive image data that is being communicated across an unsecured communication connection. The expansion of social networks has led to an increase in data exchange, resulting in changes to the communication infrastructure (Hassan et al., 2022).

Cryptosystems can be symmetric algorithms with two categories; stream and block ciphers. The differences between the two groups are found in how they convert normal messages into cipher messages. It is an application of stream cipher using a message that is encrypted on a bit by a bit basis through the secret key generator. The decryption operation is carried out using the same technique of encryption and secret key generator (Farhan & Mahdi, 2014). There are many ways in which the stream cipher algorithm can be used to generate the encryption key that will be used to secure messages and some of them include; clock-controlled generators, non-linear combination generators, and shift registers among others. The main difference between a stream cipher and a block cipher is that a stream cipher works on individual bits or bytes of data at the same time that a block cipher transforms the full block of data at the same once. Various stream algorithms, Salsa, Rabbit, and HC128 are used. Different algorithms of block ciphering like RC5 and DES are used (Najm et al., 2021). The other classification is by the usage of a secret key that differentiates between crypto-systems that major in what is known as the private/symmetric key and the other that deals with public/asymmetric key cryptosystems. In a private key cryptographic system, the same key is used for encryption as well as decryption by the sender and recipient respectively. On the other hand in a private key system encryption of the plaintext is obtained with the help of a private key while decryption is carried out with the help of a public key (Mahdi & Hassan, 2018).

The main valuable contribution to the advancing field of secure image transmission and storage, particularly in settings with restricted computational capabilities, such as cloud computing and social media platforms by presenting lightweight encryption using a combination of the ChaCha20 key generator and the 16-round Serpent algorithm. This method is specifically designed to meet the security needs of contemporary digital image applications, with a focus on minimizing computational complexity, resource consumption, efficiency, dependability, resilience, and nonlinearity, which is extraordinary (Kalinin et al., 2024). The goals of the present study consist mainly of achieving a higher degree of protection by revolving around two main aspects.

- Innovative Integration of Encryption Algorithms: The implementation of the ChaCha20 key generator along with the Serpent algorithm has been done successfully to enhance the existing lightweight encryption scheme for better security. This combined response consumes the speedy and secure key generation of ChaCha20 and the powerful crypto of Serpent.
- Evaluation of Performance Using Several Measures: The suggested approach underwent thorough testing using various metrics, such as histogram analysis, entropy calculation, correlation analysis, mean squared error (MSE) measurement, peak signal-to-noise ratio (PSNR) calculation, unified average change intensity (UACI) assessment, normalized pixel difference rate (NPCR) analysis, and structural similarity index (SSIM). These outcomes suggest the method's effectiveness in safeguarding image data with minimal distortion and maintaining high structural similarity in encrypted images. The analysis determines that the suggested plan is exceptionally secure, efficient, and practical for instantaneous communications.

The rest sections of the article have been organized as: Section 2 includes related works while sections 3 and 4 present ChaCha20 stream cipher and Serpent algorithm respectively. Section 5 goes deeper in explaining the proposed methodology. In section 6, we talked about the experiments and the security analysis. In section 7, we concluded the study.

2 Related Works

Various related works on lightweight image encryptions are discussed and elucidated. The security performance of the analysis in image encryption schemes is discussed and studied in many research.

Tariq (Shah et al., 2020), presented a modified version of the SERPENT method that incorporates time-diminishing techniques. The modification involves using chain ring-based substitution boxes (S-boxes) that operate on 8-bit vectors instead of 4-bit ones. The chain ring's multiplicative substructures have numerous generators, resulting in improved algebraic complexity of the cipher when utilizing chain ring-based S-boxes. In addition, the approach is employed in a digital RGB image encryption application where ring operations are done along the chain. The digital image evaluations indicate that the suggested system requires less time (specifically, 8.2 microseconds for encryption and 5.8 microseconds for decryption of a 128-bit block) compared to the already prominent RGB image encryption algorithms (Bashier & Jabeur, 2021). Moreover, the analysis of the proposed RGB image encryption system reveals that it exhibits strong resilience against statistical and differential attacks.

Azeez et al., (2023), presented an image encryption method comprising three encryption layers: Bit-shift encryption, chaos-based encryption, and stream encryption. The chaos algorithm employed is

Arnold's chaotic map, whilst the stream cipher technique utilized is RC4. Each layer possesses distinct cryptographic features to enhance the security of picture encryption. Cryptology encompasses the attributes of permutation, confusion, diffusion, and substitution. The suggested encryption approach protects images from statistical and differential assaults. Statistical analysis (entropy, avalanche effect, and histogram), differential analysis (UACI and NPCR), visual analysis (PSNR and SSIM), and bit error ratio are used to test encryption methods.

Ali & Ressan, (2016) has provided an algorithm of image protection that is based on the Serpent block cipher algorithm in a Feistel network. This choice is made based on increased difficulty for the attackers or any unauthorized person to find the actual images. The bigger block size is 512 bits and uses more rounds and linear transformation functions than the standard 128 bits; thus, it is more secure. The correlation coefficient in the new form of the serpent method reduces to a level that is still below that represented by the hegemonic serpent algorithm. The correlation coefficient ranging from the plain image and cipher image in grey level has the value 0.0023 in the modified serpent. In the conventional model, this takes place in the traditional serpent where the input of a 64*64-pixel bitmap image results in 0.0814.

Azeez et al., (2021) a new way to secure images has been suggested that uses both stream ciphers and block ciphers and is based on several chaotic maps. Some of the good things about these maps are that they are ergodic, they mix well, and they are sensitive to starting conditions and system parameters. The picture is split into two smaller pictures at the start. Next, stream and block cipher methods are used to encrypt these two sub-images. Then, the two sub-images are put together to make the encrypted picture. For this method, secret keys are the standard map system parameter, the standard map initial values, the sine map initial values, and the tent map initial values. The key space size is 1075 if the accuracy is 10 to 15. Because of this, the key area is big enough that brute-force attacks can't work.

Hussein et al., (2020) A new approach to assess the performance of three image encryption techniques namely RC5, Chaotic, and Permutation has been adopted. This entailed evaluating the performance of each algorithm based on five parameters that are PSNR, Entropy, Correlation, NPCR, and UACI. The results of these metrics were then used to feed a Fuzzy Logic System (FLS) to decide on the efficiency of the various encryption methods. The data show that the maximum success was achieved by using the RC5 method. Therefore, the integration of FL quality assessment for image encryption algorithms adds a new dimension for putting up analytical comparisons among the methods implemented. For future work, it is suggested to consider the analysis of other methodologies and carry out the investigation of their efficiency when used in the context of an FL system with more detail.

3 Chacha20 Stream Cipher Algorithm

Security is another category of protection implemented in data and the ChaCha20 is among those algorithms that have topped the list of successful encryption schemes. ChaCha20 is a set of stream cipher algorithms highly effective that may be advised to be incorporated into some methods of encryption. ChaCha20 is an encryption algorithm that is the same for both encryption and decryption with a 256-bit key (Yadav et al., 2016). The ChaCha20 is another encryption algorithm for a computer that is considered to be both fast and secure. It is designed to offer protection against already identified forms of attack such as differential cryptanalysis, and linear cryptanalysis. Moreover, the computation of the proposed method is highly parallelizable because it can fit perfectly into multi-core processors as well as into other forms of high-performance computing architectures (Minglin & Junshuang, 2011).

ChaCha20 is a cryptographically secure stream cipher. It processes data in 64-byte blocks using 20 rounds of an ARX (addition, rotation, exclusive or) iterative function keyed by a 256-bit key and an 8-byte (64-bit) nonce. ChaCha20 begins by initializing the ChaChaState from the key, the nonce, and an initial counter, which increments after each block is produced. It then runs through the main part of the function, the core. The core updates the state of the cipher, processes it through 20 rounds of quarter-rounds, and returns the processed part of the state as the keystream. ChaCha20 is essentially symmetric: the key, nonce, and "position", which controls the counter, are all aspects of the same state, and given these inputs, the algorithm determines the key, nonce, and position (Mahdi et al., 2021). To produce the keystream, the following happens: first, the "key setup" phase, the input is used to initialize the ChaCha state. The "core" of the cipher processes the state to generate the keystream, then the "counter and nonce" are updated so that each block has a distinct nonce when used as a hash or message authentication code. The key setup is first split into 4 cases comparing and setting counter value based on the "constant - k". In the following description, the counter value after each step is illustrated. These values are variable: k=length of constant, p=end of constant, i=iterations. ChaCha20 output is a keystream consisting of 64 bytes or 16 unsigned 32-bit integers produced by making a fixed point of the ChaCha core/quarterround "function" 2 k64 times reusing the counter/nonce component each time. ChaCha20 is essentially a variant, or crackdown "crux", of the Salsa20 symmetric stream cipher. Salsa20 and ChaCha20 share the same two main branches - the mixing of the block, and of the diagonal - based on Add-Rotate-XOR (ARX) operations involving only addition, rotation, and the bitwise exclusive or operation (and negation, addition with its inverse). ARX is known as a fast and secure type of cryptographic arithmetic and operations based on ARX have both matching speeds and security. ChaCha20 mixes its state with a "quarterround" operation that recombines the state, in a way similar to a Secure Hash Algorithm (SHA) or Sponge function shown in Table 1. ChaCha20 reduces the amount of reuse from the mixing step and state size, increasing the number of total rounds to achieve the same security ideal. In addition, ChaCha20 uses a 96-bit nonce, whereas Salsa20 uses a 64-bit nonce shown in Table 2 (Najm, 2021; McLaren et al., 2019).

Table 1: ChaCha20 Inputs (Nir & Langley, 2018)

Cons1	Cons2	Cons3	Cons4
Key1	Key2	Key3	Key4
Key5	Key6	Key7	Key8
Block-counter1	Block-counter2	Nonce1	Nonce2
⬇			
x0	x1	x2	x3
x4	x5	x6	x7
x8	x9	x10	x11
x12	x13	x14	x15

Table 2: ChaCha20 Quarter Round Function (Nir & Langley, 2018)

Column-Form
QR(x0,x4,x8,x12)
QR(x1,x5,x9,x13)
QR(x2,x6,x10,x14)
QE(X2,X7,X11,X15)
⬇
Row-Form
QR(x0,x5,x10,x15)
QR(x1,x6,x11,x12)
QR(x2,x7,x8,x13)
QR(x3,x4,x9,x14)

4 Serpent Algorithm

The serpent is a symmetric key block cipher that was a candidate for AES and stood a second to Rijndael. The Serpent encryption algorithm is (Anderson et al., 1998). The block size of Serpent is always 128 bits, but the key sizes are variable: 128, 192, or 256 bits as shown in Figure 1. The cipher is cryptographic and is made of a 32-round substitution/permutation network. There is therefore a direct correlation between the size of an operation to the number of 32-bit words, of which the block size is four 32-bit words. Each time the iterations are performed, one of the eight 4-bit to 4-bit S-boxes is applied in a 32-times cycle (Schneier et al., 2000). The Serpent algorithm was developed to specifically allow all operations to be conducted at the same time using 32-bit slices. This approach maximizes the parallelism and at the same time takes full advantage of the cryptanalysis done on DES (Anderson et al., 2000).

Serpent was quite conservative when it came to security by selecting a safety margin. The designers believed that 16 rounds provide sufficient security to protect against existing knowledge of attack mechanisms; however, 32 rounds are added as a measure of possible progress in cryptanalysis (Ali, 2010). This algorithm for cipher can be explained as (Biham et al., 2001; Izevbizua, 2015; Weinstein et al., 2012; Najm et al., 2020; Elkamchouchi et al., 2018).

- **Key Schedule:** The key schedule is the main component of the Serpent algorithm. It is primarily responsible for generating round keys from the original key. Before the encryption process, the key schedule generates the round keys so that they can be used to encrypt each plaintext. The key schedule is also responsible for the decryption process: since Serpent uses the concept of key whitening to terminate the encryption process, the key schedule must generate the round keys so that they can be used to nullify the previous whitening operation through decrypting.
- **S-Boxes:** The substitution boxes are called S-Boxes as they perform the substitution part in the substitution-permutation networks. So, a good substitution box is the heart of almost all encryption algorithms. It usually adds non-linearity, avalanche effect, and confusion to the overall cryptographic scheme. The substitution box should be strong enough to ensure high confusion in the encryption algorithm.
- **Linear Transformation (LT):** Some of the basic building blocks used in the Serpent algorithm are linear transformations, which have a key property that after these operations, the correlations between the bits are "spread out" by a process of mixing the bits in every n-bit word. This, in turn, implies that after multiple rounds of the cipher, the correlations between the input and output bits appear to be extremely weak. This appears to be similar to a random transformation, which, in turn, implies that block statistical attacks should fail.

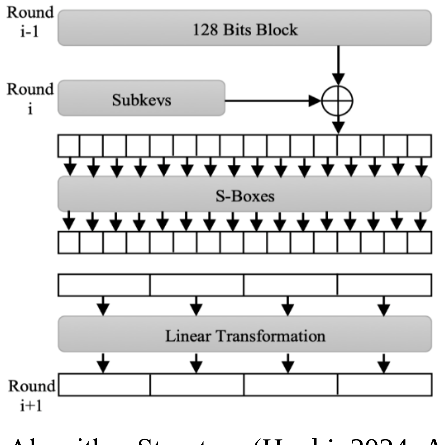


Figure 1: Serpent Algorithm Structure (Hoobi, 2024; Arman et al., 2024)

5 Proposed Methodology

Lightweight image encryption techniques are part of the mechanisms of data security and protection as they are developing with the increase of problems in the sphere of cyber-security. The system of encryption is a critical factor in modern information protection, which witnesses a growing need for the origin of fast and efficient methods for encryption. Out of all these techniques, ChaCha20 and Serpent are examples of developing solutions and they can be applied for an added degree of protection. The ChaCha20 algorithm has great speed and efficiency in data processing, rendering it highly suitable for applications demanding great such as real-time communications. It relies on an inflow encryption method, allowing it to operate effectively on multiple systems. The Chacha20 algorithm shows that it is one of the conspicuous key generators that can be employed in various applications hence the relative speed and efficiency. On the other hand, the serpent algorithm is one of the strongest of the corresponding encryption algorithms, with a high level of safety provided by its design based on several rounds of substitution and mixing processes.

In this proposed system, we will see how we can establish how to develop the lightweight image algorithm by considering the ChaCha20 key generator so that 16 rounds of serpent encryptions are conducted enabling its applicability in environments with limited resources. The process begins with the generation of keys by using ChaCha20. The operation of the key generator will be random and the output will be sequences that will be stringent to be used as inputs key cipher to the serpent algorithm. Thereafter the key acquired is disassembled into components to be input to the Serpent stages, as shown in Figure 2. These generated keys are used in performing sixteen rounds of serpent encryption, which makes it safe. This encryption process needs an input of plaintext data from 128 bits going through around 16 before the data is put in the circular processing function, The IP function processes the 128 by combining the data. In terms of the characteristics of statistics, the subsequent phase involves circular data processing, which necessitates the utilization of keys generated by the ChaCha20. The XOR function is employed between the first derived key and the 128 obtained from the function (IP) during the subsequent processing. The result of the previous task is divided into four segments, each of which is a length of 128. These segments are then submitted to a function (alternatives) that processes the data using tables (S-boxes) as shown in Figure 3. Subsequently, the round guest's condition is activated, which determines the size of the round. The serpent algorithm also has a weakness connected with work on keys, being one of the most crucial in cases of working with private keys. The power of the key determines the efficiency of the system and if the key is weak or compromised by hackers, then the whole system is vulnerable. If keys are weak or easy to predict, the ability to decrypt data is unfortunately possible, threatening the confidentiality of information. Therefore, the choice of random and complex keys becomes an absolute necessity. Nevertheless, the proposed method resolved this issue by transferring the serpent's keys to the ChaCha20 stream, thereby ensuring a more secure combination and the correct combination. The system's encryption rate is significantly quicker, and energy consumption is now reduced. Our proposal is intended to be utilized for applications where high security is needed with low-cost devices because it is immune to virtually all of the cryptanalytic attacks that are characteristic of block and stream ciphers.

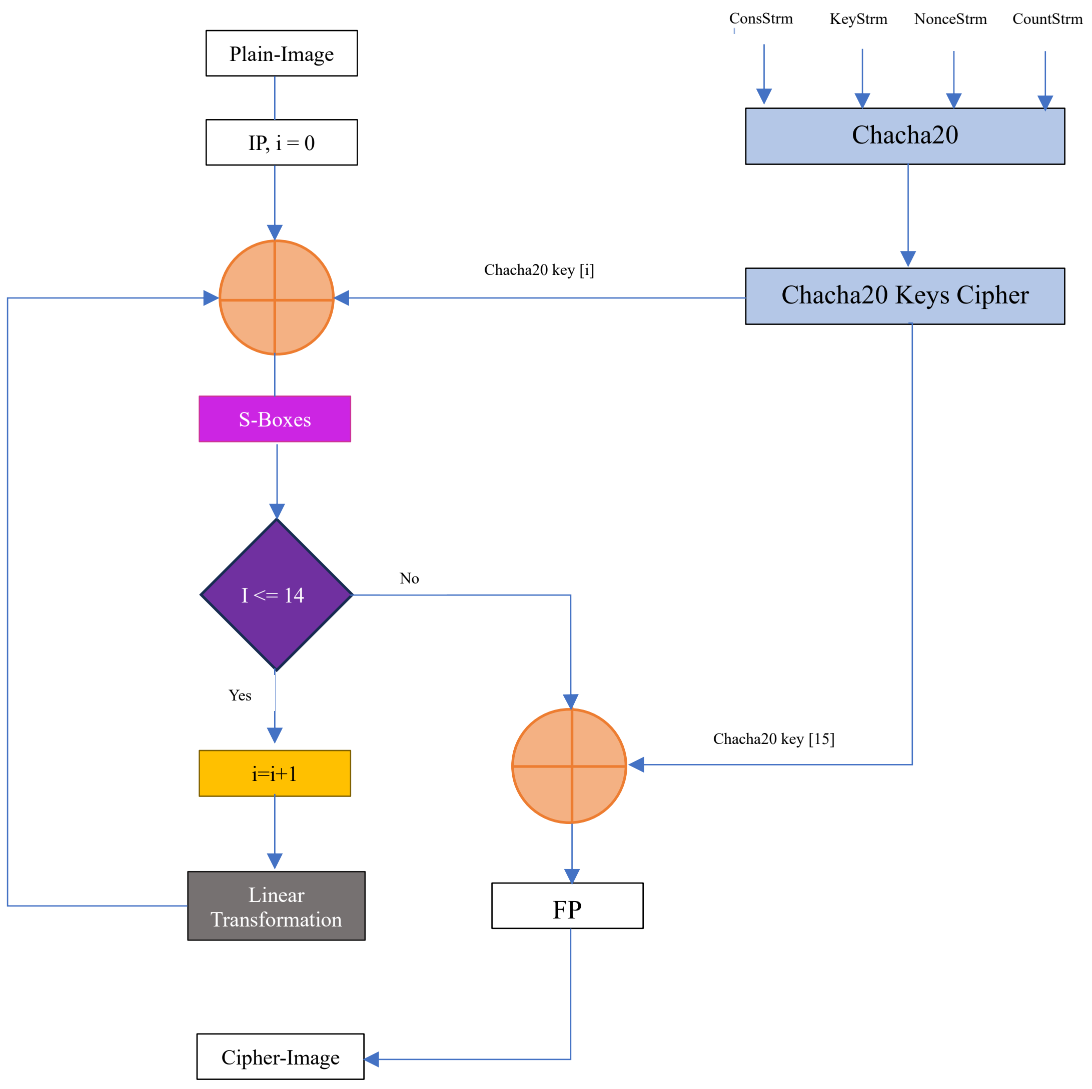
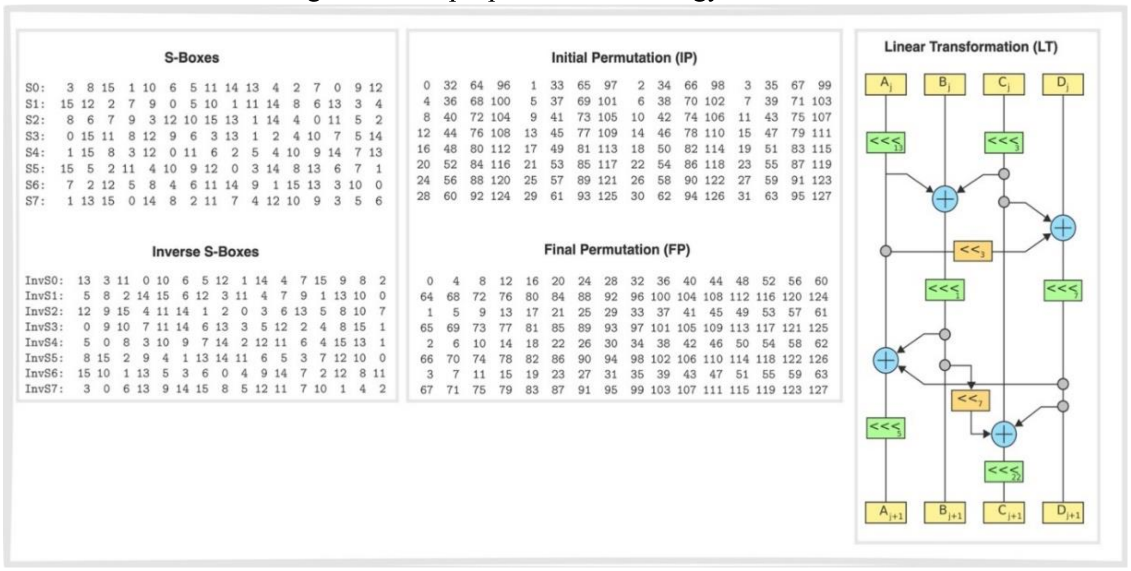


Figure 2: The proposed Methodology Flowchart



S-Boxes

S0:	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S1:	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S2:	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2
S3:	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S4:	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S5:	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S6:	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S7:	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

Inverse S-Boxes

InvS0:	13	3	11	0	10	6	5	12	1	14	4	7	15	9	8	2
InvS1:	5	8	2	14	15	6	12	3	11	4	7	9	1	13	10	0
InvS2:	12	9	15	4	11	14	1	2	0	3	6	13	5	8	10	7
InvS3:	0	9	10	7	11	14	6	13	3	5	12	2	4	8	15	1
InvS4:	5	0	8	3	10	9	7	14	2	12	11	6	4	15	13	1
InvS5:	8	15	2	9	4	1	13	14	11	6	5	3	7	12	10	0
InvS6:	15	10	1	13	5	3	6	0	4	9	14	7	2	12	8	11
InvS7:	3	0	6	13	9	14	15	8	5	12	11	7	10	1	4	2

Initial Permutation (IP)

0	32	64	96	1	33	65	97	2	34	66	98	3	35	67	99
4	36	68	100	5	37	69	101	6	38	70	102	7	39	71	103
8	40	72	104	9	41	73	105	10	42	74	106	11	43	75	107
12	44	76	108	13	45	77	109	14	46	78	110	15	47	79	111
16	48	80	112	17	49	81	113	18	50	82	114	19	51	83	115
20	52	84	116	21	53	85	117	22	54	86	118	23	55	87	119
24	56	88	120	25	57	89	121	26	58	90	122	27	59	91	123
28	60	92	124	29	61	93	125	30	62	94	126	31	63	95	127

Final Permutation (FP)

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
64	68	72	76	80	84	88	92	96	100	104	108	112	116	120	124
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
65	69	73	77	81	85	89	93	97	101	105	109	113	117	121	125
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
66	70	74	78	82	86	90	94	98	102	106	110	114	118	122	126
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
67	71	75	79	83	87	91	95	99	103	107	111	115	119	123	127

Figure 3: Block Diagram of the Main Functions of the Proposed Encryption

6 Experimental Results and Security Analysis

This section outlines a set of tests conducted to assess the effectiveness and demonstrate the efficiency of the suggested method in the analysis performance by using characteristics of the Serpent 128-bit block cipher and ChaCha20 128-bit keys generator to encrypt and decrypt images through 16-round, which is significantly quicker than the standard Serpent and more reliable. The encryption procedure requires a 128-bit input of plainImage data, which goes through 16 iteration stages (rounds). The deciphering process entails reversing the ciphering process. Energy consumption can be minimized while increasing the system's encryption speed. In the proposed methodology, a set of tests is implemented to assess the security analysis of the schema:

- **Image Histogram:** A lightweight image encryption the Serpent algorithm with ChaCha20, is demonstrated to be effective against statistical attacks through the histogram. Figure 4 shows where the histogram of the encrypted image must be equalized and normalized to encrypt it. The histogram of the original image is characterized by significant peaks and is not uniform.

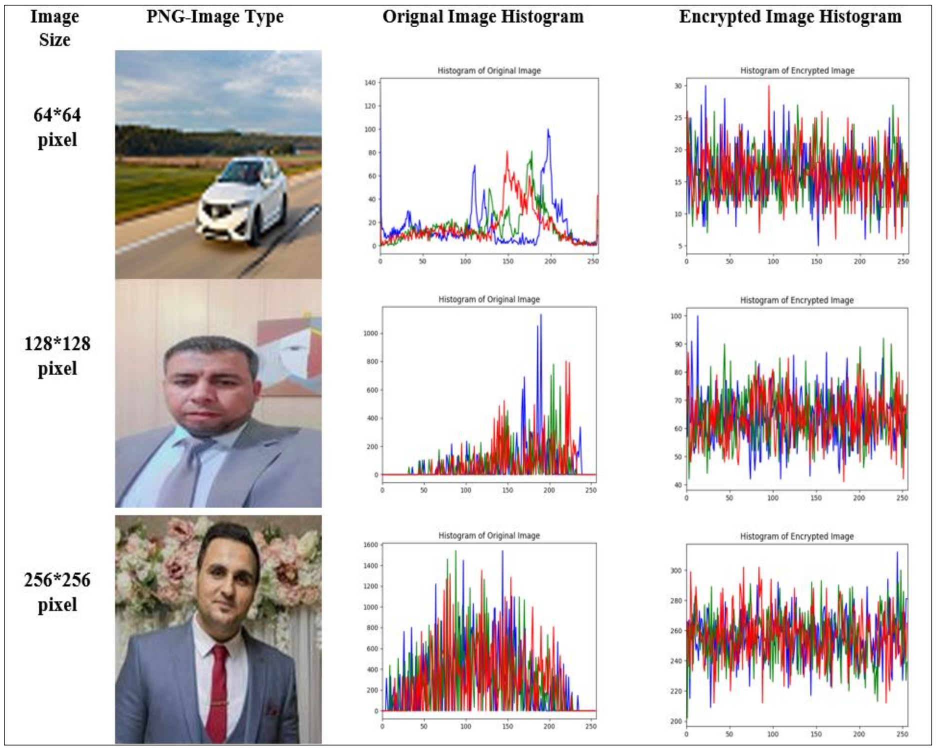


Figure 4: Histogram Measurement of three Samples of Images

- **Entropy:** One of the primary characteristics that establish the admissibility of the suggested system to contain image insecurity is its entropy, which is based on the Serpent algorithm with ChaCha20. This is illustrated in Table 3 below where the entropy value presented is almost equivalent to the acceptable limit of eight.

Table 3: Entropy Measurements of Three Samples of Images

Image Size	Entropy of Encrypted
64*64 pixel	7.986609955728056
128*128 pixel	7.995167091718644
256*256 pixel	7.998860043678963

- **Correlation:** The correlation between the proposed lightweight image encryption, utilizing the Serpent algorithm, and ChaCha20 affirms the robustness and resilience of the recommended lightweight image encryption. Figures 5, 6, and 7 illustrate the correlation between the two neighboring pixels of the plain image and the encrypted image. This correlation is observed both horizontally, vertically, and diagonally. The plain-image pixels are close to a critical value of 1, while the encrypted image pixels are close to a critical value of 0.

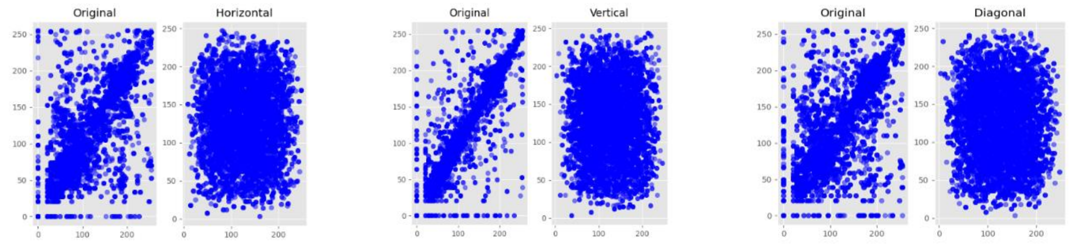


Figure 5: Sample 1 Shows how the Images (Original, Encrypted) are Correlated with Three Types (Horizontally, Vertically, and Diagonally)

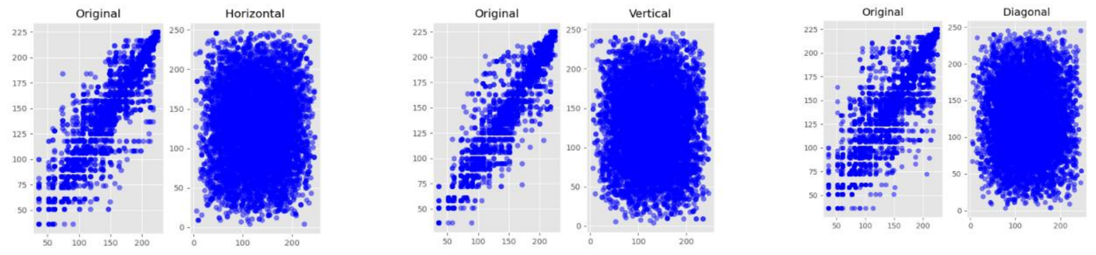


Figure 6: Sample 2 Shows How the Images (Original, Encrypted) are Correlated with Three Types (Horizontally, Vertically, and Diagonally)

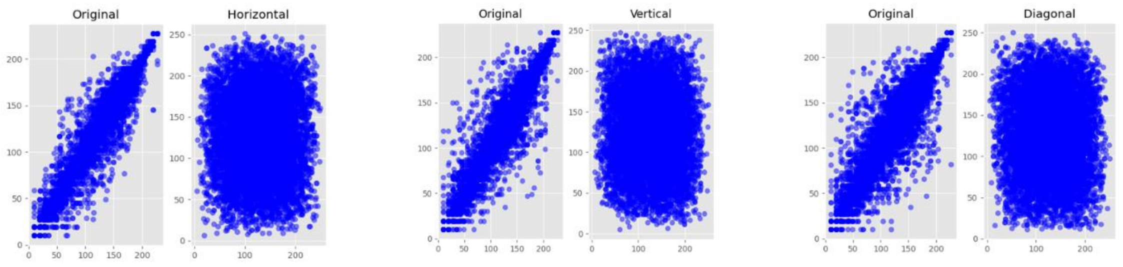


Figure 7: Sample 3 Shows how the Images (Original, Encrypted) are Correlated with Three Types (Horizontally, Vertically, and Diagonally)

- **MSE and PSNR:** are important in the analysis of lightweight image encryption. Differences between MSE and PSNR are described in the following: MSE quantifies distortion, while PSNR offers a different perspective on image quality. It has been the essence of optimizing the encryption algorithms that retain the image quality and the same time can offer security in a world that is fast going digital. Table 4 calculates the MSE and PSNR for the 3 samples of images.

Table 4: MSE and PSNR Measurements of Three Samples of Images

Images	**MSE**	**PSNR**
Sample 1	105.335693359375	27.9050480257545
Sample 2	105.00640869140625	27.918645553753144
Sample 3	105.38191731770833	27.903142650091457

- **NPCR, UACI, and SSIM:** NPCR and UACI are the basic parameters that are commonly employed to measure the ability of a cipher against differential types of attacks. These are also used in the **determination** of the sensitivity of the ciphertext using NPCR and UACI. This means that a good NPCR value should be as close to 100% as possible. The opposition is true in the value of the UACI in which a small value suggests a good performance. SSIM has become accepted as one of the main prominent equipment for quantifying visual image quality. In this particular scenario, the objective is to get the lowest feasible SSIM values by appropriate image encryption. Table 5 calculates the NPCR, UACI, and SSIM for the 3 samples of images. Table 6 illustrates the security analysis comparison with previous work.

Table 5: NPCR, UACI, and SSIM Measurements of Three Samples of Images

Images	**NPCR**	**UACI**	**SSIM**
Sample 1	99.59309895833333	30.11019837622535	0.004979623522399958
Sample 2	99.65006510416667	31.270584405637216	0.01723325219845066
Sample 3	99.69075520833333	29.475911458333243	0.017768900425535278

Table 6: The Proposed System's Security Analysis in Comparison to Previous Work in the Same Field

No. of References	**Entropy**	**Correlation**			**MSE**	**PSNR**	**NPCR**	**UACI**	**SSIM**
		H	**V**	**D**					
[13]	7.999	0.007	0.007	-0.0034	106.30	7.865	99.60	30.53	0.0103
[37]	7.997	/	/	/	/	8.3925	99.58	31.01	0.0069
[15]	7.234	0.00734	0.00582	0.00546	/	/	/	/	/
[16]	7.999	0.00049	0.00062	/	/	/	99.61	33.47	-
[17]	7.949	0.02785			/	43.294	99.93	57.12	/
Proposed	7.986	-0.01054	-0.01147	-0.01260	105.335	27.905	99.59	30.11	0.0049

7 Conclusion

A new lightweight image encryption depends on the ChaCha20 keys generator and the 16-round Serpent algorithm is presented in this study. The security analysis results show that the suggested system can protect data against any types of plaintext attacks, and produce a good level of randomness, considerable security margin, and a high key sensitivity.

About what was stated and analyzed in the context of this paper, the creation of a lightweight image encryption depends on the ChaCha20 key generator can be considered a meaningful development of modern cyber security. This option combines high security with the Serpent algorithm and utilizes the speed of ChaCha20, which is quite suitable for systems with a limited amount of resources. As the trends toward the use of effective algorithms are rising in multiple scenarios, this new algorithm on the map will most probably become one of the frequently employed selections.

Also, security threats are still present and constantly changing, so there is always a need to invent the principles of encryption and improve them. Thus, performance audits and security requirements ought to be followed to make certain that the solution will continue to be useful and protective in the future.

References

[1] Ali, Y. H. (2010). Proposed 256 bits RC5 Encryption Algorithm Using Type-3 Feistel Network. *Engineering and Technology Journal, 28,* 2337-2352. https://doi.org/10.30684/etj.28.12.5

[2] Ali, Y. H., & Ressan, H. A. (2016). Image encryption using block cipher based serpent algorithm. *Engineering and Technology Journal*, *34*(2), 278-286. https://doi.org/10.30684/etj.34.2B.10

[3] Anastasiia, H., Oleksandr, K., Yuliia, N., Alla, N., Veronika, S., & Tetiana, D. (2024). Management of Strategies for Shaping the Innovative and Investment Potential of Enterprises as a Factor Ensuring Their Economic Security. *Indian Journal of Information Sources and Services*, *14*(3), 16–22. https://doi.org/10.51983/ijiss-2024.14.3.03

[4] Anderson, R., Biham, E., & Knudsen, L. (1998). Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, *174*, 1-23.

[5] Anderson, R., Biham, E., & Knudsen, L. (2000). Serpent and smartcards. In *Smart Card Research and Applications: Third International Conference, CARDIS'98, Louvain-la-Neuve, Belgium, September 14-16, 1998. Proceedings 3* (pp. 246-253). Springer Berlin Heidelberg. https://doi.org/10.1007/10721064_23

[6] Arman, M. S., Al Mamun, S., & Jannat, N. (2024, April). A modified AES based approach for data integrity and data origin authentication. In *2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICAEEE62219.2024.10561750

[7] Azeez, R. A., Abdul-Hussein, M. K., Mahdi, M. S., & ALRikabi, H. T. S. (2021). Design a system for an approved video copyright over cloud based on biometric iris and random walk generator using watermark technique. *Periodicals of Engineering and Natural Sciences (PEN)*, *10*(1), 178-187. https://doi.org/10.21533/pen.v10i1.2577

[8] Azeez, R. A., Jamil, A. S., & Mahdi, M. S. (2023). A Partial Face Encryption in Real World Experiences Based on Features Extraction from Edge Detection. *International Journal of Interactive Mobile Technologies*, *17*(7), 69-81. https://doi.org/10.3991/ijim.v17i07.38753

[9] Bahrami, S., & Naderi, M. (2012). Image encryption using a lightweight stream encryption algorithm. *Advances in Multimedia, 2012*(1), 767364. https://doi.org/10.1155/2012/767364

[10] Bashier, E., & Jabeur, T. B. (2021). An Efficient Secure Image Encryption Algorithm Based on Total Shuffling, Integer Chaotic Maps and Median Filter. *Journal of Internet Services and Information Security, 11*(2), 46-77. https://doi.org/10.22667/JISIS.2021.05.31.046

[11] Biham, E., Dunkelman, O., & Keller, N. (2001, April). Linear cryptanalysis of reduced round serpent. In *International Workshop on Fast Software Encryption* (pp. 16-27). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45473-X_2

[12] Elkamchouchi, H. M., Takieldeen, A. E., & Shawky, M. A. (2018, March). A modified serpent based algorithm for image encryption. In *2018 35th National Radio Science Conference (NRSC)* (pp. 239-248). IEEE. https://doi.org/10.1109/NRSC.2018.8354369

[13] Farhan, A. K., & Mahdi, M. S. (2014). Proposal dynamic keys generator for DES algorithms. *islamic college university journal*, *29*, 25-48.

[14] Hassan, N. F., Al-Adhami, A., & Mahdi, M. S. (2022). Digital speech files encryption based on Hénon and gingerbread chaotic maps. *Iraqi Journal of Science*, 830-842. https://doi.org/10.24996/ijs.2022.63.2.36

[15] Hoobi, M. M. (2024). Multilevel Cryptography Model using RC5, Twofish, and Modified Serpent Algorithms. *Iraqi Journal of Science*, *65*(6), 3434-3450. https://doi.org/10.24996/ijs.2024.65.6.37

[16] Hussein, M. K., Hassan, K. R., & Al-Mashhadi, H. M. (2020). The quality of image encryption techniques by reasoned logic. *Telkomnika (Telecommunication Computing Electronics and Control)*, *18*(6), 2992-2998. https://doi.org/10.12928/TELKOMNIKA.v18i6.14340

[17] Izevbizua, P. O. (2015). Data security in the cloud using serpent encryption and distributed steganography. *European Scientific Journal*, *11*(18), 5845.

[18] Kalinin, O., Gonchar, V., Abliazova, N., Filipishyna, L., Onofriichuk, O., & Maltsev, M. (2024). Enhancing Economic Security through Digital Transformation in Investment Processes: Theoretical Perspectives and Methodological Approaches Integrating Environmental

Sustainability. *Natural and Engineering Sciences*, *9*(1), 26-45. https://doi.org/10.28978/nesciences.1469858

[19] Leu, F.Y. (2011). Guest Editorial: Emerging Security Technologies and Applications. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2*(3), 1-3.

[20] Mahdi, M. S., & Hassan, N. F. (2018). Design of keystream Generator utilizing Firefly Algorithm. *Journal of Al-Qadisiyah for computer science and mathematics*, *10*(3), 91–99. https://doi.org/10.29304/jqcm.2018.10.3.441

[21] Mahdi, M. S., Azeez, R. A., & Hassan, N. F. (2020). A proposed lightweight image encryption using ChaCha with hyperchaotic maps. *Periodicals of Engineering and Natural Sciences, 8*(4), 2138-2145.

[22] Mahdi, M. S., Hassan, N. F., & Abdul-Majeed, G. H. (2021). An improved chacha algorithm for securing data on IoT devices. *SN Applied Sciences*, *3*(4), 429. https://doi.org/10.1007/s42452-021-04425-7

[23] Mahdi, M., & Hassan, N. (2018). A suggested super salsa stream cipher. *Iraqi Journal for Computers and Informatics, 44*(2), 1-6.

[24] McLaren, P., Buchanan, W. J., Russell, G., & Tan, Z. (2019). Deriving ChaCha20 key streams from targeted memory analysis. *Journal of Information Security and Applications*, *48*, 102372. https://doi.org/10.1016/j.jisa.2019.102372

[25] Minglin, Y., & Junshuang, M. (2011, January). Stream ciphers on wireless sensor networks. In *2011 Third International Conference on Measuring Technology and Mechatronics Automation* (Vol. 3, pp. 358-361). IEEE. https://doi.org/10.1109/ICMTMA.2011.660

[26] Najm, H. (2021). Data authentication for web of things (WoT) by using modified secure hash algorithm-3 (SHA-3) and Salsa20 algorithm. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(10), 2541-2551.

[27] Najm, H., Hoomod, H. K., & Hassan, R. (2020). A proposed hybrid cryptography algorithm based on GOST and salsa (20). *Periodicals of Engineering and Natural Sciences (PEN)*, *8*(3), 1829-1835.

[28] Najm, H., Hoomod, H., & Hassan, R. (2021). *A New WoT Cryptography Algorithm Based on GOST and Novel 5d Chaotic System*. International Association of Online Engineering. https://www.learntechlib.org/p/218922/

[29] Nir, Y., & Langley, A. (2018). *ChaCha20 and Poly1305 for IETF Protocols*, no. rfc8439.

[30] Peng, X., Zhang, P., & Cai, L. (2004). Information security system based on virtual-optics imaging methodology and public key infrastructure. *Optik*, *115*(9), 420-426. https://doi.org/10.1078/0030-4026-00386

[31] Salman, D., Azeez, R., & Abdul-hossen, A. (2019). Build Cryptographic System from Multi-Biometrics using Meerkat Algorithm. *Iraqi Journal for Computers and Informatics*, *45*(2), 1-8. https://doi.org/10.25195/ijci.v45i2.46

[32] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N., & Stay, M. (2000). The Twofish team's final comments on AES Selection. *AES round*, *2*(1), 1-13.

[33] Shah, T., Haq, T. U., & Farooq, G. (2020). Improved SERPENT algorithm: design to RGB image encryption implementation. *IEEE Access*, *8*, 52609-52621. https://doi.org/10.1109/ACCESS.2020.2978083

[34] Tian, P., & Su, R. (2022). A Novel virtual optical image encryption scheme created by combining chaotic S-Box with double random phase encoding. *Sensors*, *22*(14), 5325. https://doi.org/10.3390/s22145325

[35] Ullah, A., Shah, A. A., Khan, J. S., Sajjad, M., Boulila, W., Akgul, A., ... & Ahmad, J. (2022). An efficient lightweight image encryption scheme using multichaos. *Security and Communication Networks*, *2022*(1), 5680357. https://doi.org/10.1155/2022/5680357

[36] Weinstein, D., Kovah, X., & Dyer, S. (2012). *SeRPEnT: Secure remote peripheral encryption tunnel*. Technical Report MP120013, The MITRE Corporation.

[37] Yadav, P., Gupta, I., & Murthy, S. K. (2016, March). Study and analysis of eSTREAM cipher Salsa and ChaCha. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)* (pp. 90-94). IEEE. https://doi.org/10.1109/ICETECH.2016.7569218

Authors Biography

Hayder Najm, is received his BSc degree in computer science from the University of Technology in 2011, received his MSc degree in computer science from the University of Technology in 2014, and received his PhD degree in computer science from the University of Technology in 2022. He is currently working at Imam Al-kadhim Unversity College (IKU), Computer Technique Engineering Department, Wasit, Iraq.

Mohammed Salih Mahdi, is currently Asst. Prof. Dr. in Business Information College, University of Information Technology and Communications, Iraq. His BSc degree in hiding data in 2010 and his MSc degree in a security of cloud computing in 2012 and his PhD degree in a security of IoE in 2019 from Computer Science department, University of Technology, Iraq.

Wijdan Rashid Abdulhussien, is a lecturer at the College of Computer Science and Mathematics - University of Thi Qar, Educational qualifications: Bachelor's degree with first class honors in Computer Science - University of Thi Qar - College of Science, Master's degree in Computer Science - University of Basra - College of Science, PhD in Computer Science - University of Technology.