ICIC Express Letters
Volume **18**, Number **1**, January **2024**

ICIC International ©2024 ISSN 1881-803X
pp. **17–25**

HYBRID CRYPTOGRAPHY BASED ON MODIFIED SALSA20 – GOST ALGORITHMS AND MULTIPLE CHAOTIC KEY LEVELS

Mohammed Abdullah Taha[1], Mostafa Satea Alhamdany[2,3,*] and Saif Ali Abd Alradha Alsaidi[4]

[1]Ministry of Education, Babylon Education Directorates
Baghdad Street, Babylon 51001, Iraq
cs.19.54@grad.uotechnology.edu.iq

[2]Department of Computer Techniques Engineering
AlSafwa University College
Al Hur Highway Karbala, Karbala 56001, Iraq
*Corresponding author: mustafa.sateaa@alsafwa.edu.iq

[3]Department of Building and Construction Technology Engineering
Al-Mussaib Technical College
Al-Furat Al-Awsat Technical University
Baghdad Street, Babylon 51006, Iraq
*Corresponding author: mustafa.satea@atu.edu.iq

[4]College of Education for Pure Science
Wasit University
Alkut Street, Wasit 52001, Iraq
salsaidi@uowasit.edu.iq

Received January 2023; accepted April 2023

Abstract. *Cryptography is the practice of securing communication and information through the use of mathematical algorithms. Block ciphers use the Substitution-Permutation and Feistel network structure to encrypt information, while stream ciphers use a one-time key. GOST is a symmetric key encryption algorithm, which means it uses the same key for both encryption and decryption. The strength of the encryption depends on the key size and the complexity of the algorithm, and a longer key size and more complex algorithm will generally provide stronger encryption. This study suggests a hybrid encryption method based on the GOST block cipher, the modified Salsa (20) stream cipher, and multiple chaotic key levels to provide adequate security with random hardness as high as the NIST standard tests and to change the key schedule as security operations. The simple key scheduling of the GOST algorithm is a weakness, which can sometimes be the weak spot of the related key cryptanalysis method. The proposed solution, on the other hand, uses a new hybrid algorithm based on a modified version of Salsa (20), GOST, and multiple chaotic keys to get the best combination and stronger security. Because the brute-force attack needs 2256 likely keys to break a key, which is inconvenient, it should not be used in this situation.*

Keywords: Cryptography, Stream cipher, Block cipher, Salsa (20), GOST, Chaotic system

1. **Introduction.** Numerous experiments have been conducted to discourage contact-based data access by unauthorized parties. In most technical approaches, flawless safety electronics systems have not yet been realized. Even if the data is incredibly secure, it is always feasible for unauthorized persons to decipher them [1]. Various encryption techniques are available to mitigate this problem. Rapid advances in electronic technology have led to more efficient data transfer employing microcontrollers and computer-based systems in recent years [2-4]. Modern cryptography research reveals that the underlying

DOI: 10.24507/icicel.18.01.17

characteristics of chaotic systems reveal a strong connection between chaos and cryptography [5,6] based on their fundamental similarities. The characteristics of chaotic systems were speed, distortion, reactivity, and aperiodicity. Due to chaotic behavior and dependence on initial conditions and settings, point-of-view-based encryption for communication is increasingly important [7]. The complexity of data encryption is indicative of the importance of data secrecy. In recent decades, researchers have discovered a fascinating connection between cryptography and chaos. Many characteristics of chaotic systems are similar to those of confusion/diffusion with a single change in dynamics represented by a plain text key/secret key [8,9]. These characteristics include ergodicity, sensitivity to initial conditions, mixing qualities, determinism, easy-to-use dynamics, and structural complexity [10]. Originally introduced by Matthews as a stream cipher in 1989 [11] based on the chaotic 1D algorithm, the name "chaos" has a long history in the field of cryptography. The procedure is extremely safe and resistant to cryptographic assaults due to the essential features of chaotic systems. Due to their effectiveness in creating confusion-based diffusion outcomes, many chaos-based encryption algorithms have been investigated and implemented in any cryptosystem.

Hybrid encryption methods combine two or more encryption techniques to enhance the security of the overall encryption process. The use of multiple encryption techniques can help to overcome the weaknesses of individual encryption methods and provide greater protection against attacks. By combining the GOST block cipher, the modified Salsa (20) stream cipher, and multiple chaotic key levels, a hybrid encryption method can be created that provides a high level of security against attacks. The use of multiple encryption techniques and chaotic key levels can make it more difficult for an attacker to decrypt the encrypted data, even if they manage to break one of the encryption techniques. Overall, exploring hybrid encryption methods based on the GOST block cipher, the modified Salsa (20) stream cipher, and multiple chaotic key levels can lead to the development of more secure encryption algorithms that can protect sensitive data and information from unauthorized access. The goal of this research was to provide a new, lightweight cryptographic algorithm based on the Modified SALSA20 – GOST Algorithms and the 5D Chaotic System. It is easy to implement in both software and hardware and can provide robust security at a fast encryption rate. The proposed encryption algorithm is designed to be used in low-cost devices that require high security, as it is resistant to most of the cryptanalytic attacks commonly encountered by block ciphers and stream ciphers. In this study, we conduct a series of tests to prove that our approach works. The proposed scheme security and timing are demonstrably inferior, as shown by extensive statistical research and trials [12,13]. Section 2 and its subsections of the following paper describe the research methodology. The results are described in Section 3. In the end, Section 4 provides a conclusion.

2. **Research Methodology.**

In this section, a comprehensive assessment of the suggested system and its constituent components is provided. Specifically, the functioning of Salsa, GOST, and Multiple Chaotic Key Generator is scrutinized in detail.

2.1. **Salsa (20) stream cipher.**

The term "Salsa (20) wisdom" pertains to the counter-encryption mode utilized in the stream cipher process. The input of Salsa comprises 512 bits, wherein each 32-bit segment is a seed. These bits constitute the elements of the subsequent matrix [14]:

- The total size of the key is 256 bits (K1, . . . , K8).
- The total nonce size [n1, n2] and constants [c1, c2, c3, c4] are 192 bits. The block message counter (b1, b2) has a total size of 64 bits.

Consequently, three adaptable methodologies are utilized to amalgamate the Salsa (20) input, generating a collection of 512 bits representing the keystream. This is achieved through the XOR operation between the original data and the Salsa (20) encrypted data. The three techniques employed are Add, XOR, and Rotation, which involves the rotation of 32 bits. The Quadratic Relationship Function (QRF) forms the nucleus of the dual function represented by the three procedures. The update of data rows and columns is executed as demonstrated in Figure 1.

Column form	Row form
QR(X0. X4. XS. X12)	QR(X0. X1. X2. X3)
QR(X5. X9. X13. X1)	QR(X5. X6. X7. X4)
QR(X10. X14. X2. X6)	QR(X10. X11. X8. X9)
QR(X15. X3. X7. X11)	QR(X15. X12. X13. X14)

FIGURE 1. Salsa (20) four-quarter functions [11]

Four 32-bit values are provided as input to QRF, and four 32-bit values are returned using the three techniques depicted in Figure 2.

$b = b \oplus [(a + d) << 7]$
$c = c \oplus [(b + a) << 9]$
$d = d \oplus [(c + b) << 13]$
$a = a \oplus [(d + c) << IS]$

FIGURE 2. Salsa (20) procedures for four-quarter functions [15]

A new operation will be applied between the most recently changed matrix and the first input of salsa seed input during the Salsa (20) final processing step [15]. The 96-word procedure has been performed in each of Salsa's twenty rounds, with the first altering requiring 48 words, and the second 48 words. The calculation for the 48-word operation involves multiplying the 16-word operation by 3 (Addition, XOR, and Rotation). The total number of operations for 10 rounds is 960 words. Encryption requires 960 words of work at the end of the Salsa (20), plus 16 words of work to round out the total [15].

2.2. **GOST block cipher.** In the Soviet Union and Russia, the GOST block cipher was the de facto cipher standard. In Russia, it is the foundation of the country's most secure data system. The Feistel scheme is followed by the GOST algorithm, which is a symmetric block cipher. A 256-bit key is used to convert unencrypted 64-bit data blocks into encrypted 64-bit data blocks [16]. Function F transmits plain text to three cryptographic processes on the right side of each cycle of plain text messages:

- Adding the data and the subkey module 232;
- Substitution of data by using secret S boxes;
- Left cyclical shift by 11 positions.

This is done by combining the F function's output with that of the plaintext modulo 2, then swapping the right and left sides for the following round [17].

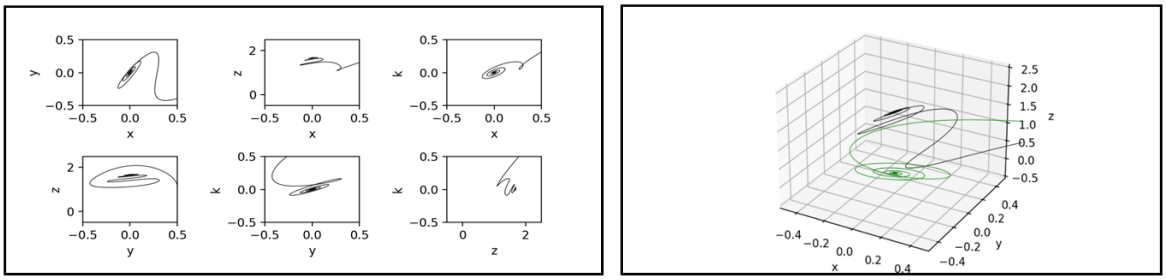
The algorithm has a total of 32 iterations. The right and left sections of the encryption are not swapped in the final round [17]. GOST systems convert a 4-bit input to a 4-bit output using 8 S-boxes. Unlike some other encryption algorithms, GOST does not specify the values to be used in these S-boxes, allowing for flexibility in implementation. The Secret Key is a series of eight 32-bit phrases: K1, K2, K3, K4, K5, K6, K7, and K8. One of these 32-bit phrases is used in every encryption round as a round subkey [17,18]. The order is straight from round 1 to round 24; the reversed order is used for round 25 and

round 32 [18]. The S-boxes accept a 4-bit input and a 4-bit output is generated. Eight 4×4 S-boxes are used to replace the S-box in the round function. When using GOST, parties must use the same S-boxes if they want their conversations to be safe. S-boxes may be kept confidential for additional safety purposes [13-16].

2.3. **Multiple chaotic key generator.** In this section, the chaotic 5D system has more complex dynamic features than the chaotic systems of lower dimensions, while the chaotic systems of lower dimensions are unique nonlinear dynamical systems. If you want to know for sure whether your system is chaotic or not, you can use a number called the Lyapunov exponent. When both the exponent of chaos and the Lyapunov exponent are positive, we say that the system is hyperchaotic. The equations underlying the suggested system are as follows [17-19]:

$$\begin{aligned} \text{xi+i} &= -\text{r} \cdot \text{xi} + \text{b} \cdot \text{yi} \cdot \text{ki} - 2.5 \cdot \text{s} \cdot \text{pi} \\ \text{yi+i} &= -\text{q} \cdot \text{yi} - \text{s} \cdot \text{xi} \cdot \text{zi} + \text{r} \cdot \text{xi} - \text{u} \cdot \text{pi} \\ \text{zi+i} &= 2 \cdot \text{zi} \cdot \text{xi} \cdot \text{yi} - 1.1 \cdot \text{r} \cdot \text{pi} - \text{q} \cdot \text{ki} \\ \text{ki+i} &= \text{r} \cdot \text{xi} + \text{s} \cdot \text{yi} - \text{u} \cdot \text{ki} \\ \text{pi+i} &= \text{b} \cdot ((\text{xi} + \text{ki})/\text{zi}) + \text{r} \cdot \text{yi} \end{aligned} \tag{1}$$

Here, x, y, z, k, and p are the states of the system, and b, r, s, u, and q are constants that stay the same. When $\text{b} = 0.001$, $\text{r} = 0.7$, $\text{s} = 0.5$, $\text{u} = 1.9$, and $\text{q} = 0.2$, and the initial state is $\text{x0} = 2.1$, $\text{y0} = 0.5$, $\text{z0} = 1.1$, $\text{k0} = 1.1$, and $\text{p0} = 0.1$, the system behaves chaotically, and the Lyapunov exponents are $\text{LE1} = 0.04187490335084436$, $\text{LE2} = 0.056182990598499266$, $\text{LE3} = 4$. Chaos attractors are shown in Figure 3.



(a) System acts with a two-dimensional perspective (xy, xz, xk, yz, yk, and zk).

(b) System acts with three-dimensional view (xyz).

FIGURE 3. Multiple chaotic key generators [22]

2.4. **Modified SALSA20 Algorithm (MSA).** The Modified SALSA20 Algorithm (MSA) core is built with two arrays of 16 words (work in parallel) to produce 16-byte blocks, each array containing 16-byte of the constant word, 32-byte of a keyword that is generated by a 5-D hybrid chaos system, 16-byte of input data word (IR0) and 16-byte of the counter word 22, 23, 24. The data entering the MSA is 128-bits (IR0). To obtain more complexity, the proposed idea is to XOR the IR0 with 128-bits from the generated chaos key. The MSA is a continuous stream of three simple 128-bit word transactions (128-bit addition, 128-bit exclusive, or 128-bit rotation regular). The 16-word MSA array is illustrated in Table 1.

In the SALSA20 Algorithm (SA), the first step is XOR diagonal, rotate left above diagonal by the last digits from (K1 to K5) times, and store the result in the array which is located on the diagonal words, as seen in Equation (2):

$$\begin{aligned} \text{T}\ (2.1) &= (\text{T}\ (1.1) \oplus \text{T}\ (4.1)) <<< \text{K2} + \text{K5} \\ \text{T}\ (3.2) &= (\text{T}\ (2.2) \oplus \text{T}\ (1.2)) <<< \text{K2} + \text{K4} \end{aligned}$$

$$\begin{aligned} T\ (4.3) &= (T\ (3.3) \oplus T\ (2.3)) <<< K1 + K3 \\ T\ (1.4) &= (T\ (4.4) \oplus T\ (3.4)) <<< K3 + K4 \end{aligned} \tag{2}$$

Table 1. The proposed MSA word array for encryption

Constant word 16 byte	K1 16 byte	K2 16 byte	K3 16 byte
K4 16 byte	Constant word 16 byte	L0 ⊕ K1 16 byte	L0 ⊕ (K5 ⊕ K4) 16 byte
Counter word 16 byte	Counter word 16 byte	Constant word 16 byte	K5 16 byte
K2 ⊕ K4 16 byte	K1 ⊕ K3 16 byte	K1 ⊕ K4 16 byte	Constant word 16 byte

The second step of MSA is XOR, rotate left by the last digits from (K1 to K5) bits on the diagonal and below-diagonal words respectively, and save out coming words in the below-diagonal words, as seen in Equation (3):

$$\begin{aligned} T\ (3.1) &= (T\ (1.1) \oplus T\ (2.1)) <<< K1 + K5 \\ T\ (4.2) &= (T\ (2.2) \oplus T\ (3.2)) <<< K2 + K4 \\ T\ (1.3) &= (T\ (3.3) \oplus T\ (4.3)) <<< K1 + K3 \\ T\ (2.4) &= (T\ (4.4) \oplus T\ (1.4)) <<< K2 + K5 \end{aligned} \tag{3}$$

MSA comes back down each column shifting left by the last digit of (K1 + K4) bits, then the last stage shifts the diagonal words left by the last digits of (K3 + K2) bits, and finally transposes the array. These steps will repeat 5 rounds until access to the final cipher text result.

2.5. **Proposed system encryption.** The proposed system is a new hybrid algorithm designed to be a 256 bits lightweight encryption algorithm using two GOST algorithms combined with the MSA algorithm. The proposed system has been combined with MSA to guarantee better security with a suitable execution time and to obtain more efficiency in encryption. The proposed Modified Lightweight (256-bit) will use different steps to encrypt the 256-bits of the input data using a 512 bits key. In the first step, the 256 bits of data that are entered into the proposed system will be XORing with chaos keys (K1, K2, K3, K4, K5) generated by the key generator system. Another step is to divide the input data block into two parts (L1 192 bits and R1 64 bits). R1 part passed (part of 64 bits) to the GOST algorithm (with 7 rounds) work in parallel with the left side. The result R2 from the GOST algorithm will be XORing with (K1, K2, K3, K4, K5) and then apply the bit permutation operation using the proposed MP layer consisting of an $8 * 8$ array. The set of values entered into this array is based on chaos key generation. Data will be encrypted by the present algorithm (MP-layer). Bit j is shifted to bit position MP(j) as shown in Table 2.

The L1 data block passed into the MSA and chaos keys (K1, K2, K3, K4, K5). The second part of the input block (L1) was encrypted using the MSA stream cipher. 192 bits of the input block (L1) passed to the MSA. The MSA builds an array of 24 words that holds the input data (192-bits), the constant word (128-bits), the chaos keys word (256 bits), and the counter word (128 bits) with 5 rounds. The final result from MSA is that 192 bits of cipher text will be stored in (L2). Then the 192-bits that are stored in (L2) with a 64-bit output of (R3) will be combined and swapped, and the operations repeated until 8 rounds to produce the final result 256-bit cipher block. The two sides work in a parallel processing manner. The general block diagram of the proposed system is illustrated in Figure 4.

TABLE 2. MP-layer table

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MP(j)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
j	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MP(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63
j	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
MP(j)	0	16	32	48	1	17	33	49	2	19	34	50	3	19	35	51
j	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
MP(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55

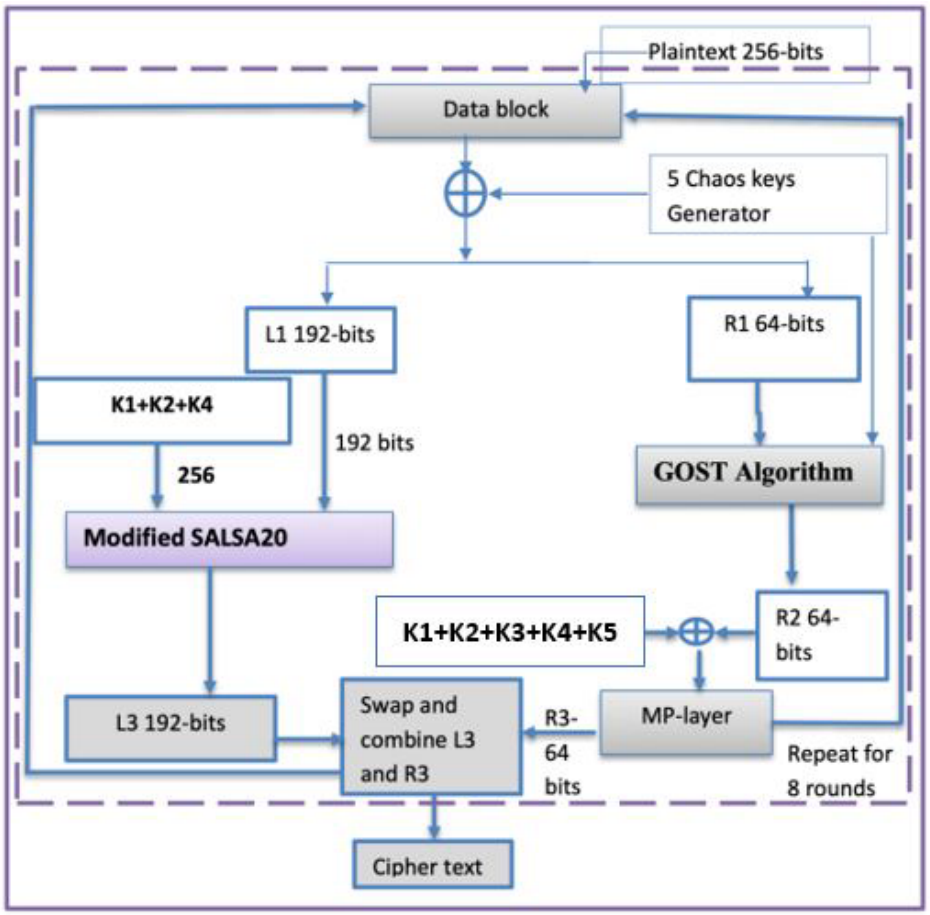


FIGURE 4. The bock diagram of the proposed system

3. **Results.** The GOST and MSA encryption algorithms are merged in the proposed system, which is supposed to be a 256-bit lightweight algorithm. It is possible to encrypt 256 bits of input data using 512 bits of the key. The data will be XORed with chaos keys (K1, K2, K3, K4, K5) created by the key generator system in the first phases. The input data block can also be divided into two halves as an additional step (L1, 192 bits, and R1 64 bits). The GOST algorithm (with 7 rounds) processes R1 (64-bit portions) in parallel with the left side. In the GOST algorithm, the output R2 will be XORed with (K1, K2, K3, K4, K5) and then applied to an 8×8 array to perform the bit permutation operation. This array's values are generated using the chaotic key generator. The proposed system can be used on low-cost devices that need high security. It is resistant to most of the attacks that can be used to break block ciphers and stream ciphers. Table 3 shows that the proposed system offers strong security and dependability.

Table 3. NIST 15 statistical tests of the proposed system

Test	Proposed 32 rounds	Proposed 28 rounds	Proposed 24 rounds	Proposed 20 rounds
Monobit	0.934	0.920	0.907	0.896
Frequency	0.911	0.860	0.854	0.847
Runs	0.979	0.937	0.926	0.913
Longest Run	0.756	0.738	0.724	0.708
Binary Matrix Rank	0.866	0.845	0.840	0.821
DFT	0.923	0.909	0.900	0.890
Non-Overlapping Template Matching	0.674	0.665	0.654	0.633
Overlapping Template Matching	0.874	0.854	0.843	0.821
Maurer's Universal	0.698	0.672	0.660	0.645
Linear Complexity	0.827	0.809	0.801	0.789
Serial	0.955	0.940	0.935	0.913
Approximate Entropy	0.776	0.769	0.760	0.749
Cumulative Sums	0.850	0.844	0.837	0.811
Random Excursion	0.898	0.886	0.879	0.860
Random Excursion Variant	0.989	0.960	0.952	0.944

Table 3 shows that overall, the results indicate that the proposed encryption algorithm performs well on most of the tests, with most values above 0.8. The "Random Excursion Variant" test had the highest passing rate across all round counts, with values above 0.94. However, the "Non-Overlapping Template Matching" and "Maurer's Universal" tests had lower passing rates, indicating that the proposed algorithm may have some weaknesses in these areas. Additionally, as the number of rounds decreases, the passing rates generally decrease as well, suggesting that increasing the number of rounds may improve the algorithm's security. Overall, while the proposed algorithm shows promising results, further testing and analysis may be needed to fully assess its strengths and weaknesses. Top of FormBottom of FormWhen testing, standard tests are used to test the proposed system encryption algorithm when encrypting unique characteristics such as time tests and NIST tests. Table 4 and Table 5 demonstrate the proposed system encryption algorithm timeline compared with the GOST algorithm and also check the encryption of files of various sizes. The tests were obtained as follows.

Table 4. Benchmarking performance of the proposed encryption algorithm (32, 28, 24, and 20 rounds) and average time (in msec) using the random input data

Operation	Proposed 32 rounds	Proposed 28 rounds	Proposed 24 rounds	Proposed 20 rounds
Encryption (1KB)	0.183	0.169	0.159	0.150
Decryption (1KB)	0.186	0.172	0.163	0.151
Encryption (10KB)	1.943	1.779	1.723	1.510
Decryption (10KB)	1.949	1.788	1.733	1.511
Encryption (100KB)	23.231	21.465	20.675	17.132
Decryption (100KB)	24.380	22.775	21.650	17.145
Encryption (1MB)	210.123	193.657	183.077	171.476
Decryption (1MB)	212.110	195.331	185.201	173.561
Encryption (10MB)	2210.867	2180.810	2171.355	2140.708
Decryption (10MB)	2214.487	2185.734	2174.111	2143.654

Table 5. Different statistical measures comparison

	Traditional GOST	**Proposed system**
Hamming distance	0.2692	0.2055
Entropy	3.3669	3.9974
MAE	48.776	42.332
UACI	35.739	36.667

4. **Conclusions.** The GOST cipher algorithm is an efficient encryption technique due to its low-complexity operations and high-security properties. GOST's mathematical technique is vital for encrypting and decrypting data detection, and the working procedures take only a few seconds. Consequently, the proposed algorithms were created to increase the difficulty of the cipher text while keeping a low execution time based on many methodologies, as well as the confusion and diffusion generated by the proposed system using the Avalanche effect and the NIST suite tests, the proposed cryptographic algorithms are studied and evaluated. Experiments reveal that the suggested algorithms beat the existing GOST method in terms of unpredictability and execution time. In addition, many tests were carried out to determine the resilience of the proposed algorithms to differential attacks. To be considered efficient in cryptosystems, the measured execution time of the proposed algorithms demonstrated that they exhibit a substantial amount of unpredictability. The future direction for GOST and SALSA20 encryption methods may include new chaotic system-based algorithms that are offered as a new trend that can be applied to various hybrid cryptosystems to achieve the trade-off between complexity and speed as a secure and hard-to-crack cipher algorithm. Hybridization with other encryption techniques such as quantum encryption or homomorphic encryption may provide even stronger security and privacy guarantees for sensitive data. Increasing their speed and efficiency, as the demand for faster and more efficient encryption continues to grow, there may be efforts to optimize the algorithms and make them more efficient in terms of processing speed and resource usage.

REFERENCES

[1] Z. Mohammad, Cryptanalysis of an efficient protocol for authenticated key agreement, *ICIC Express Letters*, vol.13, no.4, pp.293-301, DOI: 10.24507/icicel.13.04.293, 2019.

[2] N. Zitouni, M. Sedrati and A. Behaz, Comparing lightweight algorithms to secure constrained objects in Internet of Things, in *New Realities, Mobile Systems and Applications. IMCL 2021. Lecture Notes in Networks and Systems*, M. E. Auer and T. Tsiatsos (eds.), Cham, Springer, 2022.

[3] M. Y. Rhee, *Internet Security: Cryptographic Principles, Algorithms and Protocols*, John Wiley & Sons, 2003.

[4] Y. Zhang, W. Wang and H. Zhang, Neural cryptography based on quaternion-valued neural network, *International Journal of Innovative Computing, Information and Control*, vol.18, no.6, pp.1871-1883, DOI: 10.24507/ijicic.18.06.1871, 2022.

[5] A. G. Konheim, *Computer Security and Cryptography*, John Wiley & Sons, 2006.

[6] Murinto, A. Harjoko, S. Hartati and P. Danoedoro, Modified particle swarm optimization with chaos-based particle initialization and logarithmic decreasing inertia weight, *ICIC Express Letters*, vol.16, no.2, pp.195-203, DOI: 10.24507/icicel.16.02.195, 2022.

[7] S. C. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin and M. Szydlo, Security analysis of a cryptographically-enabled RFID device, *Proc. of the 14th Conference on USENIX Security Symposium (SSYM'05)*, 2005.

[8] J. Cho, J. Yoo and J.-I. Lim, Analysis of job stress's impact on job performance and turnover of cybersecurity professionals, *ICIC Express Letters*, vol.14, no.4, pp.409-415, DOI: 10.24507/icicel.14.04.409, 2020.

[9] A. W. Limanto and S. M. Isa, Securing apple watch ECG data transmission using enhanced randomized rail fencing chiper, *ICIC Express Letters*, vol.15, no.12, pp.1253-1259, DOI: 10.24507/icicel.15.12.1253, 2021.

[10] L. S. Andrés, A method for identification of force coefficients in flexible rotor-bearing systems, *Texas A&M Univ.*, no.5, pp.1-31, 2003.

[11] P.-L. Carmen and L.-R. Ricardo, Notions of chaotic cryptography: Sketch of a chaos based cryptosystem, *Applied Cryptography and Network Security*, pp.267-294, DOI: 10.5772/36419, 2012.

[12] M. W. Habiby and D. Lestari, Cryptography system for information security using chaos Arnold's cat map function, *Proc. of ICRIEMS*, pp.61-66, 2017.

[13] H. Ansaf, H. Najm, J. M. Atiyah and O. A. Hassen, Improved approach for identification of real and fake smile using chaos theory and principal component analysis, *Xinan Jiaotong Daxue Xuebao/Journal of Southwest Jiaotong University*, vol.54, no.5, DOI: 10.35741/issn.0258-2724.54.5.20, 2019.

[14] M. Mahdi and N. Hassan, A suggested super salsa stream cipher, *Iraqi J. Comput. Informatics*, vol.44, no.2, pp.5-10, DOI: 10.25195/ijci.v44i2.51, 2018.

[15] M. S. Mahdi, *Proposed Secure Internet of Everything (IoE) in Health Care*, Ph.D. Thesis, University of Technology, 2018.

[16] H. Najm, H. K. Hoomod and R. Hassan, A proposed hybrid cryptography algorithm based on GOST and Salsa (20), *Period. Eng. Nat. Sci.*, vol.8, no.3, pp.1829-1835, DOI: 10.21533/pen.v8i3.1619, 2020.

[17] T. Isobe, A single-key attack on the full GOST block cipher, *J. Cryptol.*, vol.26, no.1, pp.172-189, DOI: 10.1007/s00145-012-9118-5, 2013.

[18] I. Dinur, O. Dunkelman and A. Shamir, Improved attacks on full GOST, in *Fast Software Encryption. FSE 2012. Lecture Notes in Computer Science*, A. Canteaut (ed.), Berlin, Heidelberg, Springer, 2012.

[19] H. Ansaf, S. Hussain, H. Najm and O. Hassen, Face smile detection and cavernous biometric prediction using Perceptual User Interfaces (PUIs), *Proc. of the 1st International Conference on Computing and Emerging Sciences*, pp.62-68, DOI: 10.5220/0010412200620068, 2021.

[20] M. Khalaf, H. Najm, A. A. Daleh, A. H. Munef and G. Mojib, Schema matching using word-level clustering for integrating universities' courses, *Proc. of the 2nd Al-Noor International Conference for Science and Technology (NICST2020)*, pp.1-6, DOI: 10.1109/NICST50904.2020.9280318, 2020.

[21] H. Najm, H. K. Hoomod and R. Hassan, Intelligent Internet of Everything (IOE) data collection for health care monitor system, *International Journal of Advanced Science and Technology*, vol.29, pp.2341-2350, 2020.

[22] H. Najm, H. K. Hoomod and R. Hassan, A new WoT cryptography algorithm based on GOST and novel 5D chaotic system, *Int. J. Interact. Mob. Technol.*, vol.15, no.2, pp.184-199, DOI: 10.3991/ijim.v15i02.19961, 2021.

[23] S. Vaidyanathan, C. Volos and V. T. Pham, Hyperchaos, adaptive control and synchronization of a novel 5-D hyperchaotic system with three positive Lyapunov exponents and its SPICE implementation, *Arch. Control Sci.*, vol.24, no.4, pp.409-446, DOI: 10.2478/acsc-2014-0023, 2014.

[24] N. T. Nguyen, T. Bui, G. Gagnon, P. Giard and G. Kaddoum, Designing a pseudorandom bit generator with a novel five-dimensional-hyperchaotic system, *IEEE Trans. Ind. Electron.*, vol.69, no.6, pp.6101-6110, DOI: 10.1109/TIE.2021.3088330, 2022.