Improved Salsa20 Stream Cipher Diffusion Based on Random Chaotic Maps

Lamia A. Muhalhal[1], Imad S. Alshawi[*2]
Email: allalemyaa@gmail.com[1], emad.alshawi@uobasrah.edu.iq[2]

[*]Imad S. Alshawi
Department of Computer Science, College of Computer Science and Information Technology,
University of Basrah, Basrah, IRAQ

Keywords: NIST statistical test suite, lightweight stream cipher, salsa20 stream cipher, chaotic map, security

Received: July 8, 2022

To enhance stream ciphers, numerous studies have concentrated on the randomness, unpredictable nature, and complexity of keystream. Numerous stream algorithms have been put forth. Most of them require a significant amount of computational power. Salsa20 is a high-performance stream encryption solution that works on computers with fewer resources and uses a secure method that is faster than AES. They suggest Salsa20 for encryption in common cryptographic applications. Users who value speed over certainty should utilize the Salsa20 family of reduced-round ciphers, such as the (8,12) round cipher. It uses a 256-bit key and a hash algorithm. A successful fusion makes use of both the Salsa20 algorithm's and the random maps' advantages to improve the Salsa20 algorithm's shortcomings by increasing its unpredictability. Particularly now that Salsa20/7 has been hacked and Salsa20/12 is no longer as secure as it previously was. As a result, Salsa20 needs to achieve a high level of diffusion to withstand known attacks. Right now, salsa20 and its shortened versions rank among the fastest ciphers. This study presents a novel lightweight approach to construct a strong keystream that is sufficiently random to avoid being predicted by adversaries, achieve good diffusion, and withstand known assaults. A NIST test found that the performance of the (Salsa20-chaotic maps) approach in terms of data integrity and secrecy is nearly 0.3131 higher than that of the Salsa20.

Povzetek: Predlagan je algoritem generiranja varnih gesel za kriptirni algoritem Salsa20, s čemer se odpravi nedavno odkrite probleme.

1 Introduction

The protection of data from unauthorized access, disclosure, alteration, or destruction while upholding confidentiality, integrity, and availability is known as information security (CIA) [1]–[3]. Cryptography is used to protect data while it is in transit (either electronically or physically) through networks. It is necessary to use current cryptography techniques [1], [3], [4] . The selection of a suitable crypto algorithm will have a dynamic effect on a device's lifetime and performance in terms of battery life, hardware memory, calculation time, and communication bandwidth [4].

Conventional cryptography algorithms are slow, complicated, and energy-intensive when used with resource-constrained systems [5], [6]. The use of simple algorithms is growing in popularity. Symmetric and asymmetric algorithms for lightweight cryptography are separated into two groups. The symmetric encryption method uses the same secret key for both encrypting and decrypting operations. Data is encrypted using a public key and decrypted using a private key in asymmetric key encryption (public-key encryption). Two further symmetric key encryption techniques are block cipher and stream cipher. Trivium, Grain, and Salsa 20/12 are stream ciphers, whereas PRESENT, RECTANGLE, SIMON, and SPECK are block ciphers [2]–[4], [7].

This project is part of the Cryptographic Stream Project (ECRYPT), which was founded in 2005 [4], [7]. This project provides solutions that are both efficient and secure, as well as popular and widely used [4], [7]. Salsa20 was among the winners. A more secure and quicker variant of AES is called Salsa20 (20 rounds). Due to its low hardware requirements and simple structure, Salsa20 is an effective stream cipher for data encryption [4], [7]–[9]. One of the quickest stream ciphers currently accessible is Salsa20, as well as its condensed variants. Salsa20/12 is no longer as secure as it previously was, whereas Salsa20/7 has been broken. To achieve good dissemination and withstand known attacks, Salsa20 must therefore address this issue [10], [11]. A chaotic system or computational intelligence (CI) is therefore the ideal answer. Several techniques for chaotic systems have been devised [12]–[16]. The application of chaos theory, a kind of nonlinear system, in cryptography has recently been made to address issues with existing encryption techniques, which are losing their ability to provide quick and secure encryption for large amounts of data simultaneously [15]. Because of their unique properties and high sensitivity to their beginning conditions, chaotic systems are incredibly unpredictable over the long term. Chaotic systems are extremely sensitive to changes in beginning conditions

and parameter values, which allows for the formation of a wide range of chaotic sequences. The chaotic sequences that are generated [12], [13], [15], [17] are neither periodic nor convergent.

There have been more and more real-world systems in recent years, including wireless sensor networks, smart cities, etc. Information security has faced significant challenges due to the complexity and number of data that are only increasing. To tackle these difficulties, many computational intelligence approaches have been created that can behave intelligently in complicated and dynamic contexts to resolve complex real-world problems that are challenging to solve manually (i.e., through mathematical or traditional modeling). It has been employed to address many information security difficulties, such as selecting the optimum solution, determining normal and abnormal behavior in an intrusion detection system, and data concealing [18]–[21].

In this paper, we propose a stream cipher encryption scheme based on these concepts. This method must meet all NIST SP 800-22 requirements and be able to encrypt and decode data rapidly and simply. In this way, not only confidentiality is important, but also performance (speed, memory reduction, etc.) Our approach is to accelerate computation while maintaining the highest level of security. In the Salsa20 method, the spread of a 64-byte stream key is augmented using the chaotic maps (Henon, Rabinovich Fabrikant equation, Lorenzo, and Chua circuit) and also extends the initial matrix from 4x4 t0 4x8 [22].

The following is how the rest of the paper is organized: The literature review will be discussed in the next section. The chaotic map and the history of Salsa20 are covered in Section 3. Section 4 will go over the preferred technique. The fifth section will consist of an evaluation of the proposed method as well as a discussion of the findings. Finally, in Section 6, the conclusions will be provided.

2 Related work

Many different articles were used to develop Salsa20. Table 1 summarizes the related works with their methodology, performance, and results. According to chaos theory [23]. To improve the speed of the Salsa20-based cipher, an upgrade based on the logistic map is presented. They've been able to achieve a faster spread than the basic Salsa20. The majority of examples revealed that a modern two-iteration strategy is faster than a traditional four-iteration procedure while maintaining the same diffusion grade. In comparison to Differential Equations, the approach performed well.

Salsa20's array (4, 4) with 512 bits is changed to Salsa20's array (3, 3) with 576 bits in [24], i.e. Each location in the Salsa (20) array is 64-bit words, and each iteration involves modifying their locations by applying nine operations, resulting in a more diffuse result than the simple Salsa (20). To break down salsa, a variety of papers were used (20). Ahmad.G.et.al. in [25] generated pseudorandom numbers and employed them with a variety of encryption methods, including Salsa20. The findings of NIST measures were used to assess different encryption algorithms that used pseudorandom numbers, and the results showed that utilizing random numbers with the Salsa20 stream cipher algorithm obtained faster implementation and more diffusion than the original Salsa20 algorithm.

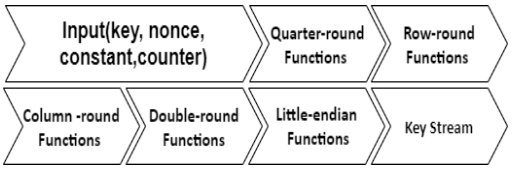

Figure 1 :Salsa20 hash function operating [8],[28].

Eman L. Mohaisen. et.al [26] presented a review of a variety of recent research that deals with stream ciphers based on Chaotic functions, as well as testing the randomness of these chaotic functions. The results of the testing revealed that stream ciphers based on chaotic functions are more secure and robust.

Maitra et al. [27] improved Salsa20's valid initial state after just one round and increased Salsa20's non-randomness after five rounds. It uses the Probabilistic Neutral Bit (PNB) to lessen the complexity of typical attacks by selecting appropriate parameters.

3 Background

3.1 Salsa20 algorithm

Salsa20 is a highly reliable stream cipher algorithm that encrypts quickly with a key size of 128 or 256 bits [8],[28], [29]. It is submitted to eSTREAM, the Encrypt Stream Cipher Project. The hash function is used in Salsa20, which takes 64-byte inputs and outputs 64 bytes. This hash function is implemented as a stream cipher in counter mode [8], [28],[29].

Hash functions include the quarter round (QR), row round (RR), column round (CR), and double round functions (DR) as illustrated in algorithm 1 pseudo-code and Figure 1. It accepts as input a 256-bit key (k0, k1... k7), a 64-bit counter (t0, t1), a 64-bit nonce (v0, v1), and 128-bit constants (c0, c1... c3). Salsa20 operates on 32-bit words and maps inputs to a 4 x 4 matrix. As in the following Equations (1), (2), (3), (4), and (5) [8],[28],[29].

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \sigma_0 & k_0 & k_1 & k_2 \\ k_3 & \sigma_1 & v_0 & v_1 \\ t_0 & t_1 & \sigma_2 & k_4 \\ k_5 & k_6 & k_7 & \sigma_3 \end{bmatrix} \quad (1)$$

The QR (a, b, c, d) transformation updates the matrix X four 32-bit words as below [8]. Where the symbol ≪ represents the rotation to the left, + is arithmetic addition, and (⊕) represents a bitwise XOR.

$$\text{QR} = \begin{cases} b = b \oplus [(a + d) \lll 7] \\ c = c \oplus [(b + a) \lll 9] \\ d = d \oplus [(c + b) \lll 13] \\ a = a \oplus [(d + c) \lll 18] \end{cases} \quad (2)$$

$$\mathrm{RR} = \begin{matrix}\mathrm{QR}(x_0, x_4, x_8, x_{12}) \\ \mathrm{QR}(x_5, x_9, x_{13}, x_1) \\ \mathrm{QR}(x_{10}, x_{14}, x_2, x_6) \\ \mathrm{QR}(x_{15}, x_3, x_7, x_{11})\end{matrix}, \mathrm{CR} = \begin{matrix}\mathrm{QR}(x_0, x_1, x_2, x_3) \\ \mathrm{QR}(x_5, x_6, x_7, x_4) \\ \mathrm{QR}(x_{10}, x_{11}, x_8, x_9) \\ \mathrm{QR}(x_{15}, x_{12}, x_{13}, x_{14})\end{matrix} \quad (3)$$

$$DR = (X) = RR\big(CR(X)\big) \quad (4)$$

$$\text{Keystream} = X + DR^r(X) \quad (5)$$

There are various types of rounds; for example, Salsa20/12 and Salsa20/8 are considered to be the fastest of the other stream cipher algorithms. Salsa20, on the other hand, is faster than the AES cipher algorithm and is therefore recommended for typical cryptosystems where speed is more important than confidence [4], [7], [26], [30].

3.2 Chaotic maps

a) Three-dimension chaotic Lorenz map

The Lorenz is a chaotic dynamical map in three dimensions. Edward Lorenz invented the coupled differential equation in 1963. When the Lorenz system is plotted, it produces a Butterfly-like attractor. A simple formula can be used to explain the system Equation (6) [31].

$$\begin{aligned} x' &= a(y - x) \\ y' &= (\sigma - z)x - y \\ z' &= xy - bz \end{aligned} \quad (6)$$

b) Henon map

The Henon map looks to be one of the well-studied instances of chaotic discrete-time dynamical systems. In 1978, the Henon chaotic map [32] was discovered for the first time. Equation7 shows a two-dimensional map with

Table 1: Summarization table on the related works.

Ref	Methodology	Performance/Results
[23]	• Chaotic sequences • Logistic map	• In order to overcome the problems of logistics maps. Some changes were made in this study. According to the simulation results, the parameter ranges to transform the logistic map distribution into a uniform distribution, which is very suitable for elastic quantification, can be greatly expanded. • The energy spectrum of the modified logistic map has been shown to be smooth and identical to the logistic map. This indicates that the pseudo-chaotic quality, which is crucial for the chaotic sequence generators.
[24]	• Sponge function • Salsa20 • Double - A	• The design decisions made when creating the Double-A cryptographic hash algorithm by the sponge. First, the relative benefits of using a stream mode cipher over a block style cipher are discussed. • Secondly, a description of how a sponge works, how it is made, and what its primary parts are follows. • Finally , the selections of the states width, rounds, and operations in the pseudorandom function are then thoroughly explained to demonstrate how and why they are employed in the permutation of Double - A, following a brief recap of the Salsa20 stream cipher and its structure.
[25]	• Pseudo-Random Sequences • NIST tests • Block Ciphers • Stream ciphers	• Use The algorithm was deleted from the list after examining a variety of pseudo-random number generators, as well as block and stream ciphers, to determine what flaws each one has, in order to choose the optimal one using the model. • An According to the studies that have been done, the algorithms AES256, MT19937, and Salsa, which serve as block cryptographers, pseudorandom number generators, and stream cryptographers, respectively, have successfully completed all of the required steps and can be awarded the privacy certificate in accordance with the model.
[26]	• Chaotic maps • keystream • Randomness of chaotic	• The majority of more recent investigations relied on chaotic map with cipher systems, which had garnered the attention of most researchers in an effort to increase their security and robustness. • This paper provides an overview of contemporary chaotic map-based stream ciphers. Additionally, it displays how chaotic maps' randomness is evaluated. According to the survey, Chen map is the randomness approach.
[27]	• Salsa20/12 • Probabilistic Neutral Bit (PNB) • ARX Cipher	• Describes how the Salsa20 state after one run can be used to create a valid initial state. After five rounds, this helps examine the Salsa20's lack of randomness. • It reconsiders the concept of probability neutral (PNB) and how the complexity of existing attacks can be reduced by carefully choosing some parameters in order to improve the results obtained today. For cryptanalysis against Salsa20, 8 rounds.

Table 2 Salsa20 Column Round(CR) , Row Round(RR)

Column Round(CR)	Row Round(RR)
(x0, x4, x8, x12)	(x0, x1, x2, x3)
(x5, x9, x13, x1)	(x5, x6, x7, x4)
(x10, x14, x2, x6)	(x10, x11, x8, x9)
(x15, x3, x7, x11)	(x15, x12, x13, x14)
(x16, x20, x24, x28)	(x16, x17, x18, x19)
(x21, x25, x29, x17)	(x21, x22, x23, x20)
(x26, x30, x18, x22)	(x26, x27, x24, x25)
(x31, x19, x23, x27)	(x31, x28, x29, x39)

Algorithm 1: Salsa20 Keystream generator [8],[28].

Input: key (k), block counter (c) and nonce (n)
Output: keystream
1: X ← IntMatrix(k,c,n)
2: for i=0 to r=9 do
▷ Column Round (CR)
3: (x0, x4, x8, x12) ← QR(x0, x4, x8, x12)
4: (x5, x9, x13, x1) ← QR((x5, x9, x13, x1)
5: (x10, x14, x2, x6) ← QR(x10, x14, x2, x6)
6: (x15, x3, x7, x11) ← QR(x15, x3, x7, x11)
▷ Row Round (RR)
7: (x0, x1, x2, x3) ← QR((x0, x1, x2, x3)
8: (x5, x6, x7, x4) ← QR(((x5, x6, x7, x4)
9: (x10, x11, x8, x9) ← QR(x10, x11, x8, x9)
10: (x15, x12, x13, x14) ← QR(x15, x12, x13, x14)
13: DR ← RR(CR(X))
14: Keystream ← X + DR^r(X)
15: end for

quadratic non-linearity that describes and represents it [32], [31].

$$\begin{aligned} x_{i+1} &= 1 - ax^2 + y_n \\ y_{i+1} &= bx_n \end{aligned} \quad (7)$$

c) Chua circuit map

The Chua circuit has a 3D continuous-time characteristic and cited as a fundamental example of chaos, exhibits chaotic behavior. That uses non-linear equations. Both the Chua circuit and other chaos-based analogy circuits can be used in a variety of applications. The Chua circuit's equations are as follows [33].

$$\begin{aligned} x' &= \alpha(y - x - g(x)) \\ y' &= x - y + z \\ z' &= -\beta y \end{aligned} \quad (8)$$

d) Rabinovich Fabrikant equation map

The Rabinovich–Fabrikant equations are a set of three linked ordinary differential equations that exhibit chaotic behavior given definite constraint values. They were first named in 1979 [14] by Mikhail Rabinovich and Anatoly Fabrikant. The chaotic system Rabinovich-Fabrikant, which is connected to the Lorenz chaotic system, is defined as follows [14].

$$\begin{aligned} x' &= y(z - 1 + x^2) + \gamma x \\ y' &= z(3z + 1 = x^2) + \gamma x \\ z' &= -2z(\alpha + xy) \end{aligned} \quad (9)$$

4 Salsa20 and chaotic maps

The proposed approach builds a stream cipher using chaotic maps and the Salsa20 algorithm. The plaintext and keystream are XORed to encrypt streams of data. The proposed method also uses the same keystream XOR with the plaintext to produce the ciphertext. Every stream cipher has the benefit of being easy to use. On the other side, the keystream generation process completely determines how strong these ciphers are. To build a strong enough unexpected keystream that attackers won't expect, the goal of this research is to enhance the Salsa20 algorithm. The cipher system's complete recommended block diagram is shown in Figure 2. The following issues will be covered.

We modify the Salsa20 algorithm to make the Salsa20 algorithm more diffusion and randomness using (the Lorenzo map (L), Henon map (H), Rabinovich Fabrikant equation map (RF), and Chua circuit map (CC)) Because they have good qualities that are suited for keystream generation, that are utilized to modify the Salsa20 algorithm. So using this chaos to generate the input Salsa20 algorithm (key, nonce, counter, constants), is used to generate a keystream. Salsa20 algorithm accepts a key 256-bit (k_{L1}, k_{H1}, k_{RF1}, k_{CC1}, k_{L2},k_{H2},k_{RF2},k_{CC2}, k_{L6}, k_{H5}, k_{RF5}, k_{CC5}, k_{L8},k_{H8},k_{RF8},k_{CC7}),a 64-bit counter (t_{L3}, t_{RF3},t_{L7},t_{RF6}]), 64-bit nonce (v_{CC3}, v_{CC3},v_{CC6}, v_{CC7}), and 128-bit constants (σ_{L4}, σ_{H4}, σ_{RF4}, σ_{CC4},σ_{L5},σ_{H6},σ_{RF7},σ_{CC8}]). (L, H, RF, and CC) is the chaotic map, and (1, 2, 3, 4) is the number of random number strings that result from the chaos. Extends Equation (1)'s initial 4x4 matrix (X) to a 4x8 matrix (X) [22], modify Equation (2) quarter-round QR (a, b, c, d) shown in Figure 3, modify column round(CR) and row around(RR) shown in Table 2 . Where K key generate by chaos and (L, H, RF, CC) is a chaotic map, and (1, 2, 3, 4) is the number of random number strings that result from the chaos. Equations (4), and (5) are still the same.

$$X = \begin{bmatrix} \sigma_{L4} & k_{L1} & k_{H1} & k_{RF1} & \sigma_{L5} & k_{L6} & k_{H5} & k_{RF5} \\ k_{CC1} & \sigma_{H4} & v_{CC3} & v_{H3} & k_{CC5} & \sigma_{H6} & v_{CC6} & v_{H7} \\ t_{L3} & t_{RF3} & \sigma_{RF4} & k_{L2} & t_{L7} & t_{RF6} & \sigma_{RF7} & k_{L8} \\ k_{H2} & k_{RF2} & k_{CC2} & \sigma_{CC4} & k_{H8} & k_{RF8} & k_{CC7} & \sigma_{CC8} \end{bmatrix} \quad (10)$$

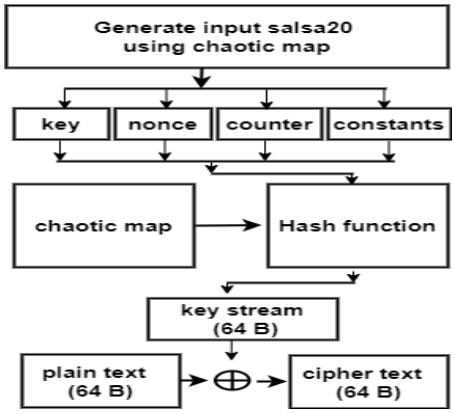

Figure 2 :Salsa20-chaotic system

$$\mathrm{QR} = \begin{cases} b = b \oplus \left[\left(K_{\mathrm{L5}} \oplus (a+d)\right) \lll 7\right] \\ c = c \oplus \left[(K_{\mathrm{RF5}} \oplus (b+a)) \lll 9\right] \\ d = d \oplus \left[\left(K_{\mathrm{CC5}} \oplus (c+b)\right) \lll 13\right] \\ a = a \oplus \left[\left(K_{\mathrm{H5}} \oplus (d+c)\right) \lll 18\right] \end{cases} \quad (11)$$

5 Results and discussion

In this paper, we propose lightweight stream encryption based on the Salsa20 algorithm and chaotic maps to achieve a high level of randomness and propagation to resist known attacks. A good fusion combines the qualities of both the Salsa20 algorithm and the random maps to become more random to improve the weaknesses of the Salsa20 algorithm. Especially after the Salsa20/7 algorithm was broken and Salsa20/12 is no longer as safe as it was before. As a result, Salsa20 achieves a high level of diffusion to resist known attacks. Salsa20 and its reduced variations are among the fastest ciphers currently available.

The proposed method was built and implemented in a Python 3.9.7 environment, on a machine with an Intel(R) Xeon(R) CPU E3-1545M v5 running at 2.90 GHz and 8 GB of RAM running Windows 10, Intel(R) Xeon(R) CPU E3-1545M v5 running at 2.90GHz.

Algorithm 2: Salsa20 with chaotic maps.

Input : key (k), block counter (b) , nonce (n), constants (c)
Output : keystream
▷ Chaotic maps
1: $RF_i(\ i\ ,1 \text{ to } 8)$= Rabinovich-Fabrikant map Generation(1 to 8)
2: $L_i(\ i\ ,1 \text{ to } 8)$= Lorenz map Generation(1 to 8)
3: $H_i(\ i\ ,1 \text{ to } 8)$= Henon map Generation(1 to 8)
4: $CC_i(\ i\ ,1 \text{ to } 8)$= Chua-circuit map (1 to 8)
▷ Initial matrix
5:k=[$k_{\mathrm{L1}}, k_{\mathrm{H1}}, k_{\mathrm{RF1}}, k_{\mathrm{CC1}}, k_{\mathrm{L2}}, k_{\mathrm{H2}}, k_{\mathrm{RF2}}, k_{\mathrm{CC2}}$, $k_{\mathrm{L6}}, k_{\mathrm{H5}}, k_{\mathrm{RF5}}, k_{\mathrm{CC5}}, k_{\mathrm{L8}}, k_{\mathrm{H8}}, k_{\mathrm{RF8}}, k_{\mathrm{CC7}}$]
6: c=[$\sigma_{\mathrm{L4}}, \sigma_{\mathrm{H4}}, \sigma_{\mathrm{RF4}}, \sigma_{\mathrm{CC4}}, \sigma_{\mathrm{L5}}, \sigma_{\mathrm{H6}}, \sigma_{\mathrm{RF7}}, \sigma_{\mathrm{CC8}}$]
7: b=[$t_{\mathrm{L3}}, t_{\mathrm{RF3}}, t_{\mathrm{L7}}, t_{\mathrm{RF6}}$]
8: n=[$v_{\mathrm{CC3}}, v_{\mathrm{CC3}}, v_{\mathrm{CC6}}, v_{\mathrm{CC7}}$]
9: X ← IntMatrix(k,c,b,n)
▷Quarter Round (QR)
10: x1 ← XOR (x1,Left Shift (XOR(K_{L5}, Add(x0 + x3)), 7)
11: x2 ← XOR (x2,Left Shift (XOR(K_{RF5}, Add(x1 + x0)), 9))
12: x3 ← XOR (x3,Left Shift (XOR(K_{CC5}, Add(x2 + x1)), 13))
13: x4 ← XOR (x4,Left Shift (XOR(K_{H5}, Add(x3 + x2)), 18))
14: for i=0 to r=9 do
▷ Column Round (CR)
15: (x0, x4, x8, x12) ← QR(x0, x4, x8, x12)
16: (x5, x9, x13, x1) ← QR((x5, x9, x13, x1)
17: (x10, x14, x2, x6) ← QR(x10, x14, x2, x6)
18: (x15, x3, x7, x11) ← QR(x15, x3, x7, x11)
19: (x16, x20, x24, x28) ← QR(x16, x20, x24, x28)
20: (x21, x25, x29, x17) ← QR(x21, x25, x29, x17)
21: (x26, x30, x18, x22) ← QR (x26, x30, x18, x22)
22: (x31, x19, x23, x27) ← QR(x31, x19, x23, x27)
▷ Row Round (RR)
23: (x0, x1, x2, x3) ← QR(x0, x1, x2, x3)
24: (x5, x6, x7, x4) ← QR(x5, x6, x7, x4)
25: (x10, x11, x8, x9) ← QR(x10, x11, x8, x9)
26: (x15, x12, x13, x14) ← QR(x15, x12, x13, x14)
27: (x16, x17, x18, x19) ← QR(x16, x17, x18, x19)
28: (x21, x22, x23, x20) ← QR(x21, x22, x23, x20)
29: (x26, x27, x24, x25) ← QR(x26, x27, x24, x25)
30: (x31, x28, x29, x30) ← QR(x31, x28, x29, x30)
31: DR ← RR(CR(X))
32: Keystream ← X + DR^r(X)
33: end for

Table 3 NIST Statistical Test Suite

NIST Tests	Salsa20	Proposed Algorithm
Frequency	0.3748	0.5472
Block Frequency	0.3659	0.7667
Cumulative Sums	0.2409	0.5892
Runs	0.4168	0.8867
Longest Run	0.1381	0.7607
Rank	0.0356	0.3626
FFT	0.24904	0.5309
Non Overlapping	0.9993	0.9999
Overlapping Template	0.1515	0.9517
Universal	0.3319	0.6951
Approximate Entropy	0.9999	0.9999
Random Excursions	0.4798	0.5178
Random Excursions Variant	0.4236	0.6046
Serial	0.3595	0.8231
Linear Complexity	0.3833	0.6101

Several statistical tests are also available to evaluate the randomness features of cryptographic algorithms. Statistical analysis is evaluated using NIST SP 800-22. Based on the significance value, the NIST tests determine whether the sequence ratio is random. When the P-value is less than 0.01, the sequence is considered random or vice versa and the non-random sequence is called [34],[35]. The proposed method and the Salsa20 cipher algorithm are subjected to each of the 15 NIST tests [34],[35]. Test results will also be discussed below.

- **Frequency Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.1724 more than the Salsa20 algorithm, according to NIST tests.
- **Frequency Block Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.4008 more than the

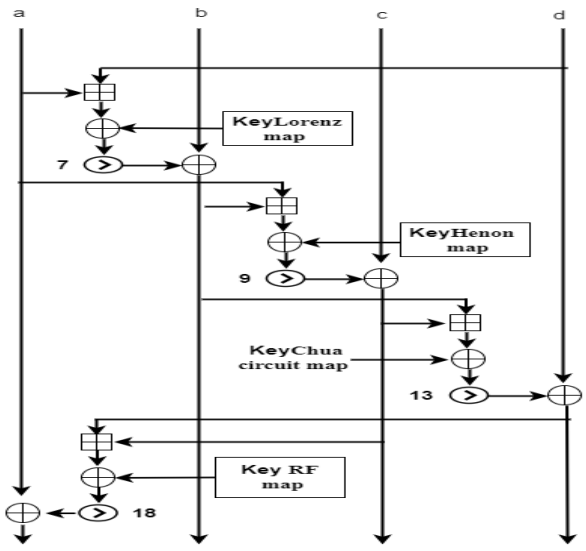

Figure 3 :QR for Salsa20-chaotic

Salsa20 algorithm, according to NIST tests.

- **Runs Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.4699 more than the Salsa20 algorithm, according to NIST tests.
- **Longest Run Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.6226 more than the Salsa20 algorithm, according to NIST tests.
- **Binary Matrix Rank Test: T**he proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.327 more than the Salsa20 algorithm, according to NIST tests.
- **Discrete Fourier Transform Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.28186 more than the Salsa20 algorithm, according to NIST tests.
- **Non-overlapping Template Matching Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.0006 more than the Salsa20 algorithm, according to NIST tests.
- **Overlapping Template Matching Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.8002 more than the Salsa20 algorithm, according to NIST.
- **Universal Test: T**he proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.3632 more than the Salsa20 algorithm, according to NIST tests.
- **Linear Complexity Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.2268 more than the Salsa20 algorithm, according to NIST tests.
- **Serial Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.4636 more than the Salsa20 algorithm, according to NIST tests.
- **Approximate Entropy Test:** The proposed method is generally equal to the Salsa20, shown in Table 3.
- **Cumulative Sums (Cusum) Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.3483 more than the Salsa20 algorithm, according to NIST tests.
- **Random Excursions Variant Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.181 more than the Salsa20 algorithm, according to NIST tests.
- **Random Excursions Test:** The proposed method is generally superior to the Salsa20, as shown in Table 3. , which increases nearly 0.038 more than the Salsa20 algorithm, according to NIST tests.

6 Conclusion

Salsa20/7, on the other hand, was broken, and Salsa20/12 is no longer as safe as it once was. As a result, Salsa20 requires greater randomness and diffusion. To improve Salsa20 is described in this paper as a cipher that uses four types of chaotic maps (3D Lorenzo map, 2D Henon map, 3D Rabinovich Fabrikant equation map, and 3D Chua circuit map) to obtain a decent diffusion level and speeds up the cipher. The proposed method improves security by using more randomization and a hash function, the hash function, and key generation, and it could be used with other techniques like a lightweight block cipher. Key generation raises the complexity of keys and adds greater flexibility. Salsa20 has been updated so that it can be used as a lightweight stream cipher on devices with limited resources. The performance of random ciphers was evaluated using 15 NIST statistical tests, which were developed to evaluate pseudo-random numbers in cryptographic applications and successfully bypassed the randomness of the proposed method. According to a NIST test, the performance achieved by the proposed method is increased by nearly 0.3131 more than that achieved from the Salsa20 in terms of data integrity and confidentiality.

References

[1] H. Wu and H. Wu, “Research on Computer Network Information Security Problems and Prevention Based on Wireless Sensor Network,” in 2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2021, pp. 1015–1018,https://doi.org/10.1109/IPEC51340.2021.9421303.

[2] K. Gupta, D. Gupta, S. K. Prasad, and P. Johri, “A Review on Cryptography based Data Security Techniques for the Cloud Computing,” in 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 1039–1044, https://doi.org/10.1109/ICACITE51222.2021.9404568.

[3] M. A. Latif, M. Bin Ahmad, and M. K. Khan, “A Review on Key Management and Lightweight Cryptography for IoT,” in 2020 Global Conference on Wireless and Optical Technologies (GCWOT), 2020, pp. 1–7, doi: 10.1109/GCWOT49901.2020.9391613.

[4] S. S. Dhanda, B. Singh, and P. Jindal, “Lightweight cryptography: a solution to secure IoT,” Wirel. Pers. Commun., vol. 112, no. 3, pp. 1947–1980, 2020, https://doi.org/10.1007/s11277-020-07134-3.

[5] R. Anusha, M. J. Dileep Kumar, V. S. Shetty, and N. Prajwal Hegde, “Symmetric Key Algorithm in Computer security: A Review,” in 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020, pp. 765–769, https://doi.org/10.1109/ICECA49313.2020.9297547.

[6] H. H. Al-badrei and I. S. Alshawi, “Improvement of RC4 Security Algorithm,” Adv. Mech., vol. 9, no. 3, pp. 1467–1476, 2021.

[7] L. Jiao, Y. Hao, and D. Feng, “Stream cipher designs: a review,” Sci. China Inf. Sci., vol. 63, no. 3, pp. 1–25, 2020, https://doi.org/10.1007/s11432-018-9929-x.

[8] D. J. Bernstein, “The salsa20 family of stream ciphers,” in Lecture Notes in Computer Science

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 4986 LNCS, M. Robshaw and O. Billet, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 84–97,https://doi.org/10.1007/978-3-540-68351-3_8.

[9] Z. M. J. Kubba and H. K. Hoomod, "A hybrid modified lightweight algorithm combined of two cryptography algorithms PRESENT and Salsa20 using chaotic system," in 2019 First International Conference of Computer and Applied Sciences (CAS), 2019, pp. 199–203, https://doi.org/10.1109/CAS47993.2019.9075488.

[10] T. Ishiguro, S. Kiyomoto, and Y. Miyake, "Latin dances revisited: new analytic results of Salsa20 and ChaCha," in International Conference on Information and Communications Security, 2011, pp. 255–266, https://doi.org/10.1007/978-3-642-25243-3_21.

[11] S. Maitra, "Chosen IV cryptanalysis on reduced round ChaCha and Salsa," Discret. Appl. Math., vol. 208, pp. 88–97, 2016, https://doi.org/10.1016/j.dam.2016.02.020.

[12] A. Gaeini, A. Mirghadri, G. Jandaghi, and B. Keshavarzi, "Comparing some pseudo-random number generators and cryptography algorithms using a general evaluation pattern," IJ Inf. Technol. Comput. Sci., vol. 9, pp. 25–31, 2016, https://doi.org/10.5815/ijitcs.2016.09.04.

[13] L. O. Tresor and M. Sumbwanyambe, "A selective image encryption scheme based on 2d DWT, Henon map and 4d Qi hyper-chaos," IEEE Access, vol. 7, pp. 103463–103472, 2019, https://doi.org/10.1109/ACCESS.2019.2929244.

[14] A. Alghafis, N. Munir, and M. Khan, "An encryption scheme based on chaotic Rabinovich-Fabrikant system and S8 confusion component," Multimed. Tools Appl., vol. 80, no. 5, pp. 7967–7985, 2021, https://doi.org/10.1007/s11042-020-10142-x.

[15] M. Hamdi, J. Miri, and B. Moalla, "Hybrid encryption algorithm (HEA) based on chaotic system," Soft Comput., vol. 25, no. 3, pp. 1847–1858, 2021, https://doi.org/10.1007/s00500-020-05258-z.

[16] N. Mohananthini, M. Y. Mohamed Parvees, and J. Abdul Samath, "Lightweight Image Encryption: A Chaotic ARX Block Cipher," J. Circuits, Syst. Comput., vol. 30, no. 02, p. 2150026, 2021, https://doi.org/10.1142/S0218126621500262.

[17] R. Anandkumar and R. Kalpana, "Analyzing of Chaos based Encryption with Lorenz and Henon Map," in 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on, 2018, pp. 204–208, https://doi.org/10.1109/I-SMAC.2018.8653652.

[18] A. H. Jabbar and I. S. Alshawi, "Spider monkey optimization routing protocol for wireless sensor networks.," Int. J. Electr. \& Comput. Eng., vol. 11, no. 3, 2021, https://doi.org/10.11591/ijece.v11i3.pp2432-2442.

[19] I. S. Alshawi, A.-K. Y. Abdulla, and A. A. Alhijaj, "Fuzzy dstar-lite routing method for energy-efficient heterogeneous wireless sensor networks," Indones. J. Electr. Eng. Comput. Sci., vol. 19, no. 2, pp. 906–916, 2020, https://doi.org/10.11591/ijeecs.v19.i2.pp906-916.

[20] G. Singh and S. Garg, "Fuzzy Elliptic Curve Cryptography based Cipher Text Policy Attribute based Encryption for Cloud Security," in 2020 International Conference on Intelligent Engineering and Management (ICIEM), 2020, pp. 327–330, https://doi.org/10.1109/ICIEM48762.2020.9159961.

[21] A. Abdaoui, A. Erbad, A. Al-Ali, A. Mohamed, and M. Guizani, "Fuzzy Elliptic Curve Cryptography for Authentication in Internet of Things," IEEE Internet Things J., p. 1, 2021, https://doi.org/10.1109/JIOT.2021.3121350.

[22] M. Mahdi and N. Hassan, "A suggested super salsa stream cipher," Iraqi J. Comput. Informatics, vol. 44, no. 2, pp. 5–10, 2018, https://doi.org/10.25195/ijci.v44i2.51.

[23] J. Zhang, Y. Zhu, H. Zhu, and J. Cheng, "Some improvements to logistic map for chaotic signal generator," 2017 3rd IEEE Int. Conf. Comput. Commun. ICCC 2017, vol. 2018-Janua, no. 1, pp. 1090–1093, 2018, https://doi.org/10.1109/CompComm.2017.8322711.

[24] A. Issa, M. A. Al-Ahmad, and A. Al-Saleh, "Double-A-A Salsa20 Like: The Design," Proc. - 2015 4th Int. Conf. Adv. Comput. Sci. Appl. Technol. ACSAT 2015, pp. 18–23, 2016, https://doi.org/10.1109/ACSAT.2015.25.

[25] A. Gaeini, A. Mirghadri, G. Jandaghi, and B. Keshavarzi, "Comparing Some Pseudo-Random Number Generators and Cryptography Algorithms Using a General Evaluation Pattern," Int. J. Inf. Technol. Comput. Sci., vol. 8, no. 9, pp. 25–31, 2016, https://doi.org/10.5815/ijitcs.2016.09.04.

[26] E. L. Mohaisen and R. S. Mohammed, "Stream Cipher Based on Chaotic Maps," 1st Int. Sci. Conf. Comput. Appl. Sci. CAS 2019, pp. 256–261, 2019, https://doi.org/10.1109/CAS47993.2019.9075490.

[27] S. Maitra et al., "Salsa20 Cryptanalysis : New Moves and Revisiting Old Styles," Int. Work. Coding Cryptogr., p. 11, 2015, https://eprint.iacr.org/2015/217.

[28] Z. M. Jawad Kubba and H. K. Hoomod, "A Hybrid Modified Lightweight Algorithm Combined of Two Cryptography Algorithms PRESENT and Salsa20 Using Chaotic System," in 2019 First International Conference of Computer and Applied Sciences (CAS), 2019, pp. 199–203, https://doi.org/10.1109/CAS47993.2019.9075488.

[29] E. L. Mohaisen and R. S. Mohammed, "Improving Salsa20 Stream Cipher Using Random Chaotic Maps," in 2020 3rd International Conference on Engineering Technology and its Applications (IICETA), 2020, pp. 1–6, https://doi.org/10.1109/IICETA50496.2020.9318902.

[30] S. Maitra, "Chosen IV cryptanalysis on reduced round ChaCha and Salsa," Discret. Appl. Math., vol. 208, pp. 88–97, 2016,

https://doi.org/10.1016/j.dam.2016.02.020.

[31] R. Anandkumar and R. Kalpana, "Analyzing of chaos based encryption with Lorenz and Henon map," Proc. Int. Conf. I-SMAC (IoT Soc. Mobile, Anal. Cloud), I-SMAC 2018, pp. 204–208, 2019, https://doi.org/10.1109/I-SMAC.2018.8653652.

[32] L. O. Tresor and M. Sumbwanyambe, "A selective image encryption scheme based on 2D DWT, henon map and 4D Qi hyper-chaos," IEEE Access, vol. 7, pp. 103463–103472, 2019, https://doi.org/10.1109/ACCESS.2019.2929244.

[33] F. Caldarola, P. Pantano, and E. Bilotta, "Computation of supertrack functions for Chua's oscillator and for Chua's circuit with memristor," Commun. Nonlinear Sci. Numer. Simul., vol. 94, p. 105568, 2021, https://doi.org/10.1016/j.cnsns.2020.105568.

[34] L. E. Bassham III et al., "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications." National Institute of Standards \& Technology, 2010, https://dl.acm.org/doi/pdf/10.5555/2206233.

[35] E. A. Luengo and L. J. G. Villalba, "Recommendations on Statistical Randomness Test Batteries for Cryptographic Purposes," ACM Comput. Surv., vol. 54, no. 4, May 2021, doi: https://doi.org/10.1145/3447773.