ISSN: 0067-2904

Monitoring System Based on New Chaotic System and IoT

Rawia Abdulla Mohammed [1]*, Maisa'a Abid Ali Khodher [1], Ashwak Alabaichi [2]
[1] *Department of Computer Science, University of Technology, Baghdad, Iraq*
[2]*Department of Biomedical Engineering, University of Kerbala, Iraq*

Received: 16/12/2022 Accepted: 6/5/2023 Published: 30/5/2024

Abstract

The security and integrity of medical data, in particular, have become significant concerns due to the substantial expansion in the healthcare environment due to the internet of things (IoT). To guarantee data security during transmission, this research suggests a chaotic healthcare monitoring system. This system employs a new 5-dimension (5D) chaotic system in conjunction with the secure hash algorithm version three (SHA3-256), salsa 20, and deoxyribonucleic acid (DNA). Lyapunov's proposed 5D chaos system was tested and passed for several initial periods, yielding a super chaos system (3 positive Lyapunov). Salsa 20 and randomly generated chaotic keys were utilized for encryption and decryption operations. The method we created will assess a patient's body temperature, heart rate, and oxygen saturation ($SpO2$)levels in the blood and communicate the data to a healthcare center through the Message Queuing Telemetry Transport (MQTT) protocol. The suggested system ensures that patient data is transmitted via the internet or an intranet in a secure, reliable, and readily accessible manner. In several studies involving security analysis, The encryption time is 1.872, the throughput (in bits per second (bps)) is 273.2642, and the memory usage (in megabytes) is 10.6211. The performance analysis confirmed the proposed cryptosystem's capacity to withstand multiple attacks due to its low computational and communication costs and broad key space.

Keywords: DNA, encryption, salsa 20, MQTT, and hashing function.

نظام مراقبة بالاعتماد على نظام فوضوي خماسي الابعاد جديد وانترنت الاشياء

راويه عبد الله محمد[1*]، ميساء عبد علي خضر[1]، اشواق محمود العبايجي[2]
[1] قسم علوم الحاسبات، الجامعة التكنولوجية، بغداد، العراق
[2] قسم الهندسة الطبية الحيوية، جامعة كربلاء، العراق

الخلاصة

أصبح أمن وسلامة البيانات الطبية ، على وجه الخصوص ، مصدر قلق كبير بسبب التوسع الكبير في بيئة الرعاية الصحية بسبب إنترنت الأشياء (IoT). لضمان أمن البيانات أثناء الإرسال. يصف هذا العمل نظامًا آمنًا لمراقبة الرعاية الصحية يستعمل نظامًا فوضويًا فريدًا خماسي الأبعاد جنبًا إلى جنب مع خوارزمية التجزئة الآمنة الإصدار الثالث (SHA3-256) و Salsa 20 وحمض الديوكسي ريبونوكلييك(DNA) . تم اختبار نظام الفوضى خماسي الابعاد المقترح وتم تمريره لعدة مراحل أولية ، مما أسفر عن نظام فوضوي ممتاز حيث يحتوي على (3 positive Lyapunov) .تم استعمال Salsa 20 لكل من عمليات التشفير وفك التشفير ، جنبًا إلى جنب مع المفاتيح العشوائية التي تم إنشاؤها عشوائيًا. سيقوم النظام الذي أنشأناه بتقييم

*Email: rawiaaljubori@gmail.com

درجة حرارة جسم المريض ومعدل ضربات القلب ومستويات تشبع الأكسجين (SpO2) في الدم وإيصال البيانات إلى مركز الرعاية الصحية من خلال بروتوكول MQTT. يضمن النظام المقترح نقل بيانات المريض عبر الإنترنت أو الإنترانت بطريقة آمنة وموثوقة ويمكن الوصول إليها بسهولة. وجدت العديد من الدراسات ، التي تضمنت التحليل الأمني وأوقات التنفيذ ، أن فك التشفير يستغرق ثانيتين وأن وقت التنفيذ هو ثلاث ثوان عندما تكون ذاكرة الوصول العشوائي (0.23) ميغا بايت. أكد تحليل الأداء قدرة نظام التشفير المقترح على تحمل هجمات متعددة بسبب قلة وقت التنفيذ وكلفة الاتصال وكبر مساحة المفاتيح المتولدة.

1. Introduction

The Internet of Things (IoT) is a network of interconnected physical systems that enables remote data gathering, processing, and analysis of useful information [1] and [2]. Recent developments in IoT technology have been made in computer networks and sensor networks for distributed architecture models responsible for the upkeep and monitoring of intelligent environments [3]. With its improved automation, analytical, and optimization characteristics, the Internet of Things substantially impacts practically every business [4]. Using sensors or other components built into intelligent devices simplifies data gathering, administration, and processing. Using sensors and IoT devices to collect patient health data like body temperature, heart rate, and blood oxygen levels has also caused a significant transformation in the healthcare sector [5]. Defending the security and privacy of this data is essential before developing a healthcare infrastructure design [6]. The lack of updated technology in network equipment resulted in security lapses that caused network attacks, including physical attacks like tampering with nodes to steal personal data [7]. As stated by Oza A.D. et al. [6] in their paper on privacy and integrity attacks, a malicious environment can reveal critical patient data to attackers. Impersonation, eavesdropping, and other attacks are also feasible during data transfer.

Consequently, patients could not have control over their medical information. So, many academics strongly advise encrypting data before transmission to ensure data confidentiality. Different security protocols, like public-key (symmetric encryption) cryptosystems, digital signatures, and authentication algorithms, can provide data protection, authentication, integrity, and confidentiality. Unfortunately, though, the confidentiality and integrity of healthcare data are not generally guaranteed by existing schemes [8]. Therefore, a significant task is investigating cryptographic data security methods in IoT healthcare systems [7]. Recently, crypto methods have been selected and tailored inside HCS under the moniker of "lightweight security cryptography" (LWC) to fit restricted resources, like KATAN, PRESENT, KLEIN, HEIGHT, CLEFIA, Salsa 20, and many other algorithms—lightweight security goals to gain sufficient security levels with optimum resource use [9] and [10]. The abovementioned issues spur efforts to create a more secure healthcare system by identifying the complementary strengths of the three algorithms, i.e., chaos, DNA computing, and Salsa 20.

The remainder of the essay is organized based on the parts below. Related works are discussed in Section 2, and healthcare system techniques are discussed in Section 3. The proposed system architecture is in Section 4. Section 5 provides implementation information particular to the given use case. Finally, the final portion contains the conclusion.

2. Related work

Researchers and medical industry executives are paying close attention to the development of monitoring systems for healthcare. Several more effective research initiatives are now being done in this field [11] and [12]. Real-time patient health status monitoring is a problem that the IoT solves with the aid of sensor data and connectivity in an efficient and usable

manner. Moreover, it has been demonstrated that the IoT may offer various upgraded and better services when used with intelligent technology. By recording significant vital signs, information on general health and potentially harmful conditions may be gathered [13]. An overview of current studies on IoT-based healthcare monitoring systems is given in this section. *J. Rokan Naif et al.* [14] presented a safe method for securing sensing data moving between embedded subsystems coupled by networks and IoT sensors/devices. This method is based on the modified, lightweight Advanced Encryption Standard (AES) and a newly suggested 4D chaotic system.

M. M. Dhanvijay et al. [15] introduced IoT-based healthcare in wireless body sensor networks, where the challenges and issues of wireless body area networks are discussed. *M. M. Islam et al.* [16] presented an e-healthcare system. This system consists of three components: a sensor module, a data-processing module, and a web user interface. *V. Tamilsevi et al.* [17] introduced IoT-based e-healthcare using an Arduino Uno processing device. The system can employ a temperature sensor, a spo2 sensor, a heartbeat sensor, and an eye blink sensor to predict the patient's state. *H. T. Yew et al.* [18] suggest a real-time remote patient monitoring system based on the IoT that can ensure the accuracy of the real-time electrocardiogram (ECG). Nevertheless, this system has significant levels of jitter delay and noise signal. *H. A. El Zouka* [19] introduced IoT-based e-healthcare using fuzzy logic. But this work lacks experimental measurements, statistical analysis, examination, and security testing and introduces IoT-based e-healthcare using fuzzy logic. But this work lacks practical measures, statistical exploration, analysis, and security testing. *M. M. Khan* [20] presents a COVID-19 monitoring system. A mobile application can get information from the system on a patient's body temperature, heart rate, and blood oxygen saturation ($SpO2$) levels through Bluetooth. Nevertheless, this system's drawback is that it is slow. *H. Y. Mohamed* [21] presented a system that can continuously track vital signs, update data online, inform clinicians if anything is out of the ordinary, and even predict whether a patient has a diagnosis. However, because the system uses deep-belief neural networks, training them requires massive amounts of data.

Since most of the healthcare systems mentioned above do not rely on data encryption, they cannot all be guaranteed to be reliable and honest. People may, as a result, lose control over their medical data. Therefore, the innovative technology is suggested to employ Salsa 20 for its higher security, DNA for security owing to its uniqueness, a new 5D chaotic map because of its complexity, and SHA3-256 for integrity.

3. HealthCare System Techniques

To encrypt digital data, several attributes are considered the main requirements to achieve this goal: security, integrity, and confidentiality. The integrity requirement is fulfilled by using SHA3-256. Security and confidentiality requirements are performed using the Lorenz chaotic map, Salsa 20, and DNA. Chaos is a mathematical theory investigating dynamical systems' behavior [22]. The system's initial conditions are critical because they contain secret keys that only authorized individuals should understand.

Furthermore, because of the extreme sensitivity of the system, changing minor numbers will result in profoundly bizarre attractors, rendering hackers unable to decrypt the original content [23]. Deoxyribonucleic acid (DNA) is a type of biological macromolecule comprised of nucleotides. These nucleotides are in four categories of bases: (A) is adenine, (C) is cytosine, (G) is guanine, and (T) is thymine. The complementary nucleotides A and T (or C and G) generate hydrogen bonds that hold the two DNA strands together to form a double-helix shape. Watson and Crick, two physicists, found this structure. This phenomenon is referred to as "Watson-Crick complementarity" [24]. The Salsa20 algorithm stream cipher was created by Daniel J. Bernstein and submitted to the eStream (Encrypt Stream Cipher

Project) in 2005 [25]. In reality, the core functionalities of Salsa20 can be described by combining expansion operations with hash functions. The expansion carefully combines an 8-byte nonce and an 8-byte block counter to create 512 bits by cryptographically connecting 16 bytes of data to 32 bytes of a secret key (a 64-byte block). Salsa20 is still regarded as suitable for our IoT healthcare system since the best-recorded attack against its security was made using a 256-bit Brute Force search, which takes longer than harming medical security [26]. Table 1 lists the functions of DNA.

Table 1: a- Coding, b- Add, c-Subtract operation in DNA

Co-de	Bits	Add	A	T	C	G	Sub	A	T	C	G
A	00	A	A	T	C	G	A	A	T	G	C
T	01	T	T	A	G	C	T	C	A	T	G
C	10	C	C	G	A	T	C	G	C	A	T
G	11	G	G	C	T	A	G	T	G	C	A

4. The Proposed System

The proposed system's architecture combines three phases, as shown in Figure 1. The blue arrows indicate the direction of the data flow between the necessary system components and the human actors, as well as through the internal system architecture. The privacy of the processed data is essential because of the three crucial steps that enable the data transmission pipeline's functionality. The first phase included the client-side (patient node), the second phase included sending cipher text or secreted data over the MQTT protocol, and the third phase included the server-side (healthcare node). The main objectives of the proposed system are to achieve confidentiality, integrity, and availability of patient data during transmission over the internet or intranet to the healthcare center. In the proposed system, the wearable gadget has one microprocessor ($Raspberry\ Pi4$) and two sensors ($MAX30102$, $MLX90614$).

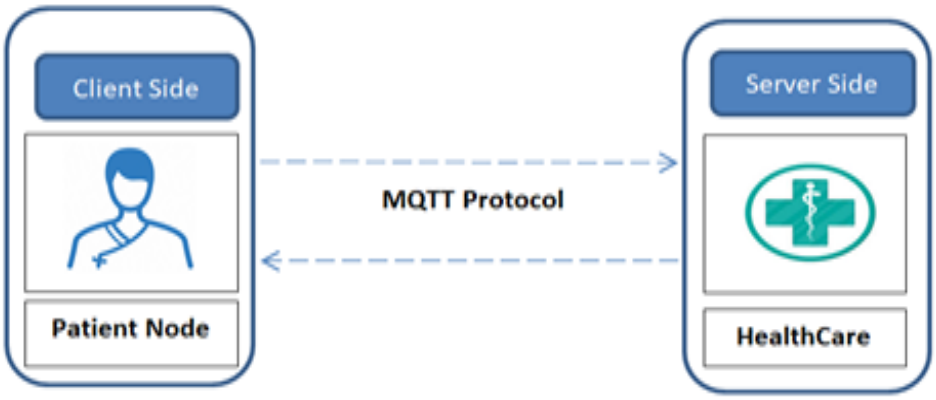

Figure 1: General Structure

The following sub-section will describe these phases in detail:

4.1 Client-Side (Patient Side)

In the first phase of the proposed healthcare system, a pulse oximeter and heart-rate sensor ($MAX30102$) were used to measure the heart rate (O2,HR) and SpO2. In addition, the infrared thermometer sensor (MLX90614) also provided body temperature (TEMP). Then, these reading sensor data are padded when the length of (O2<3, HR <3, and TEMP<2) is left padded for each sensor data. After that, we concatenate reading sensor data (O2, HR, and TEMP). Next, to provide confidentiality, sensor data has been encrypted by a new five-

dimension randomness chaotic system to generate 32 keys, the DNA algorithm, Add Round, and Salsa20 algorithm to generate a 512-bit block of a key stream (16 words).
Additionally, we apply SHA3-256 to sensor data to provide integrity. Finally, hashed and encrypted data are concatenated to prepare to transmit them over the internet or intranet to the healthcare center. The proposed system's wearable sensors include ($MAX30102$ and $MLX90614$). The $MAX30102$ takes precise SpO2 and heart rate measurements and sends those values to the Raspberry Pi4 through the I2C communication line. MLX90614 (non-contact infrared thermometer sensor) can be easily connected to a Raspberry Pi4 for measuring the temperature of a patient's body from a distance. Before sending data to concatenation processing, it must ensure the length of string (O2) equals 3, the length of string (HR) equals 3, and the length of string (TEMP) equals 2. Therefore, the padding on the three variables is performed if the length of sensor data is not equal to the present lengths. After completing the padding process for sensor data, the concatenation data is saved into a state. The string (state) must be eight characters long to be ready to start the encryption and hashing procedure. Patient data must be encrypted to ensure its confidentiality during transmission over the internet or intranet. Algorithm 1 and Figure 2 present our proposed system on the patient side. Also, algorithm 1 shows our proposed system on the inpatient side:
Algorithm 1: Proposal Encryption Algorithm on Patient Side
Input: State, Initial Values (xs, ys, zs, ks, ps) to 32 Keys Generation, and Initial Values for Salsa20
Output: E-State.
Step 1: Convert State from String to Number.
Step 2: Generate 32 keys using 5D Chaotic System.
Step3: Apply Salsa20
Step4: Apply to Add Round on (State, 32 Chaotic Keys, and Final Salsa20)
Step 5: Apply Pre-processing (Padding) if the length of Final State! = 64 bits
Step6: Encoding Final State with DNA Code into (DNA State)
Step7: Select Final Salsa 20
Step8: Convert Final Salsa20 to Binary Form
Step 9: Apply Pre-processing (Padding) if the length! = 32 bits
Step10: Split every two-bit
Step11: Encoding Final Salsa 20 with DNA Code into (DNA Salsa20)
Step12: Applying DNA- Addition Operation on (DNA State, and DNA Salsa20) into (DNA Add State)
Step 13: Convert DNA Add State from String to Number into (E-State).
Step14: Return integer (E-State)
End.

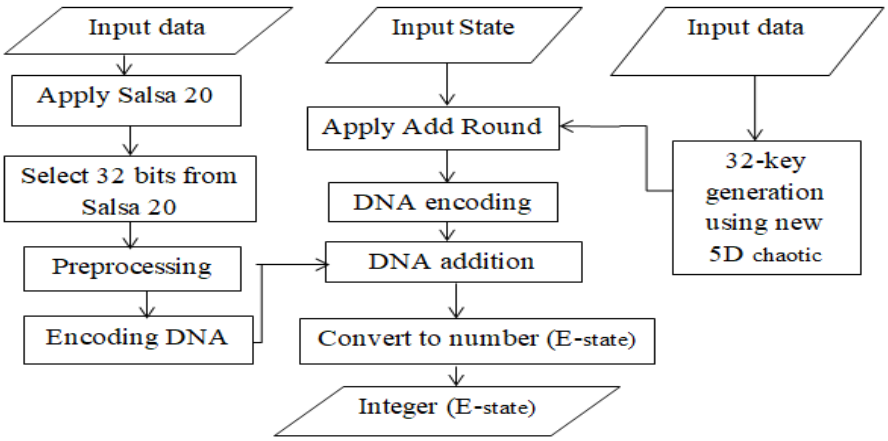

Figure 2: Proposal Encryption Algorithm on the patient side

The proposed encryption algorithms explain in detail:

4.1.1 Salsa 20

Salsa20 is used in our proposed system to create 512-bit blocks of keystream (Matrix 16 words). Salsa20's default in our suggestion is as follows: Twenty rounds are involved; the length of the keys is 32; the nonce is eight bytes long; the counter is eight bytes long; and the default value of the keys, nonce, and block counters is zero. First, the initial values of Salsa20 are as follows: For keys, we used default values (all zeroes). Nonce, the initial values are [3, 1, 4, 1, 5, 9, 2, and 6]. Block Counters: the initial values are [7, 0, 0, 0, 0, 0, 0, 0]. Constants: the initial values are [61707865, 3320646e, 79622d32, 6b206574]. Then, for keys, we select the first four numbers from the initial values of keys to k [0], the second four numbers from the initial keys to k [1], the third four numbers from the initial keys to k [2], and so on, until we reach the final four numbers from the initial keys to k [7]. For the nonce, we select the first four numbers from the initial values of the nonce to n [0] and the second four numbers from the initial nonce to n [1]. For block counters, we choose the first four numbers from the block counters' starting values up to b [0] and the following four numbers from b [1]. For constants, c [0] = 61707865, c [1] = 3320646e, c [2] = 79622d32, and c [3] = 6b206574. Then, we apply the little-endian function only on keys (k [0], k [1],..., k [7]), nonce (n [0], and n [1]), and block counters (b [0], b [1]). The block of keystream (16 words), which generates the starting matrix (S), 512 bits, is organized as follows:

$$S=\begin{pmatrix} \mathbf{c_0} & \mathbf{k_0} & \mathbf{k_1} & \mathbf{k_2} \\ \mathbf{k_3} & \mathbf{c_1} & \mathbf{v_0} & \mathbf{v_1} \\ \mathbf{i_0} & \mathbf{i_1} & \mathbf{c_3} & \mathbf{k_4} \\ \mathbf{k_5} & \mathbf{k_6} & \mathbf{k_7} & \mathbf{c_4} \end{pmatrix}$$

Next, for each round ($\boldsymbol{r-value}$), S is selected as input (only for the first iteration). Then, apply quarter-round equations in salsa 20. After that, the column-round equation is used. Finally, we transpose every row as a column to generate (S1), a new 512-bit block of keystream. Note that S1 is the next iteration's input until all twenty rounds are completed. Finally, we apply the adding operation between S1 (after 20 rounds) and S for every word; the result is the final salsa.

4.1.2 Add Round

As shown in Figure 3, the inputs in our suggested system include starting values (i=0, and j=0), 32 chaotic keys (5-dimensional), the final Salsa20 (16 words), and state (concatenate data sensors). We translate a state from binary to an integer. For 32 rounds, if the round (i) number is even, First, we apply XOR between keys (zs [i] and ks [i]) into (CK). Then, we use the XOR operation between (State and CK) into (State). After that, we apply the XOR operation between keys (xs[i] and ys [i]) into (CK). Next, we use the XOR operation between (State and CK) into (State). Then, we apply XOR between key ps[i] and Final Salsa20 [j] into (CK) and$j = j + 1$. Finally, we use the XOR operation between (State and CK) into State and $i = i + 1$. If the number of the round (i) is odd. First, we apply XOR between keys (zs [i] and ks[i]) into (CK). Then, we use the XOR operation between (State and CK) into (State). After that, we apply the XOR operation between keys (xs[i]and ys[i]) into (CK). Next, we use XOR between (State and CK) into (State). Finally, if the value of (i) is less than or equal to thirty-one, then $i = i + 1$; otherwise, final state equals state. In our proposed system, we need to generate keys for the encryption process. Therefore, we use the 5-Dimension Randomness Chaotic System to generate 32 keys. First, we create two empty lists (x, y, z, k, and p) and the second (x1, y1, z1, k1, and p1). Then input the initial value (Seed) for variables

xs, ys, zs, ks, and ps in location zero as xs [0] = 0.01, ys [0] = 0.02, zs [0] =0.03, ks [0] =0,01, and ps [0] =0.02. These initial values can change at any time (any change on the patient side must be the exact change on the healthcare side) to create new keys. Any simple change can create new and different keys.

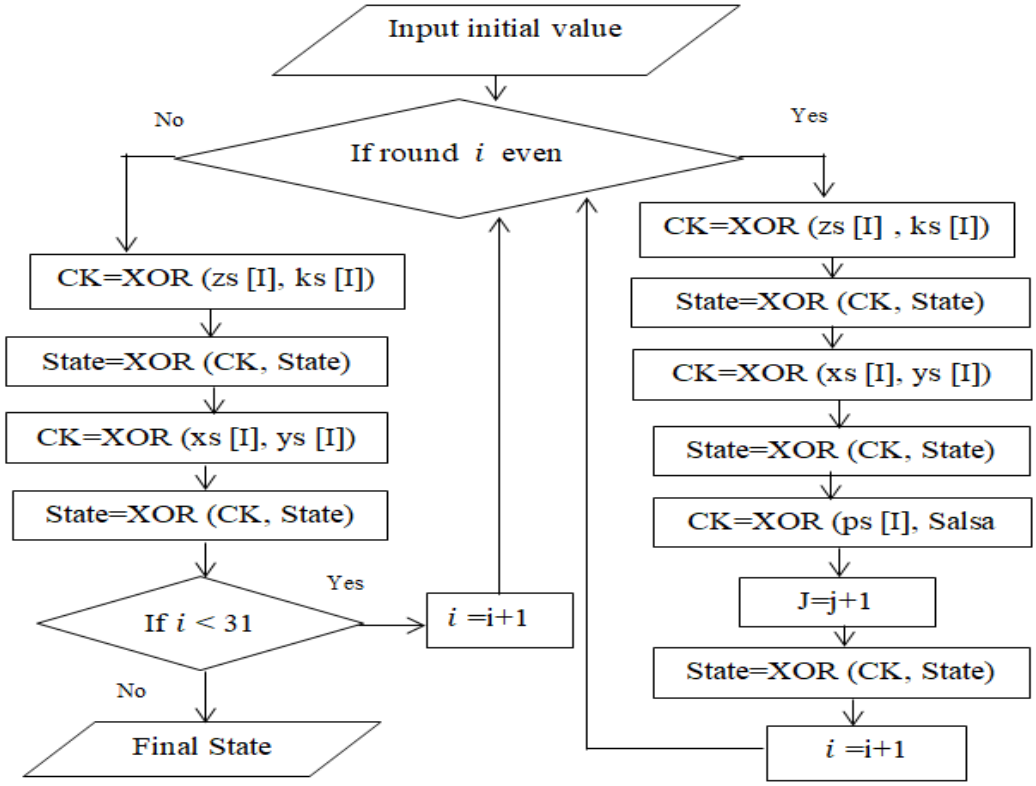

Figure 3: Add Round Process

4.1.3 DNA Encryption

Our proposed system's starting settings for DNA encryption are Final State and Final Salsa20. For Final Salsa20, first select a random key and then convert the selected key to binary form. After that, we apply padding to ensure the length of the string is exactly 32 bits; if not, we add zeros on the left of the string. Next, every two-bit is split and encoded with the *DNA* codes (A, C, G, and T) using Table 1. a. After that, concatenate these *DNA* characters into a string. Finally, this concatenation results in a *DNA* string length of 16 characters (*DNA* Salsa20). Also, the same thing applies for final state encodings with *DNA* Code: apply padding to ensure the length of the string is 64 bits; if not, we add zeros on the left of the string. Then, concatenate these *DNA* characters into a string. Finally, this concatenation results in a *DNA* string length of 32 characters named "*DNA* State." Finally, the *DNA*-Addition operation is applied to (*DNA* Salsa20) and (*DNA* State) using Table 1. b. Because the length of the *DNA* Salsa20 string is less than the length of the *DNA* State string, we repeat the *DNA* Salsa20 string to ensure the length of the two strings is equal. Next, concatenate these *DNA* characters to the string, and the result from the addition operation is a *DNA* string length of 32 characters named (*DNA* Add State).

Algorithm 2: DNA-Encryption Process
Input: Final Salsa20, Final State
Output: DNA String (DNA Add State)
Begin
Step1: Convert Final Salsa20 to Binary Form

Step2: Apply Pre-processing (Padding) if the length of the Final Salsa20 <> 32 bits
Step3: Split every two-bit
Step4: Encoding Every Two bits with DNA Code into (DNA Salsa20) using (Table 1. a)
Step5: Concatenate DNA Salsa20
Step6: Convert Final State to Binary Form
Step7: Apply Pre-processing (Padding) if the length of the Final State <> 64 bits
Step8: Apply Steps 3 to 5 on State into (DNA State)
Step9: Apply DNA-Addition Operation on (DNA Salsa20) and (DNA State) into (DNA Add State) using (Table 1. b)
Step10: Concatenate DNA Add State
Step11: Return DNA Add State
End.

4.1.4 Hashing Sensors Data

We implemented SHA3-based cryptography to ensure integrity in our suggested system when delivering patient data over the network (256). SHA-3 is a unidirectional function that may create input data into digital print lengths $(224, 256, 384, and\ 512)$. The r and c for our proposed SHA3-256 are as follows: $n = 256, r = 1088, and\ c = 512$, where r is the bit rate $(the\ value\ of\ block\ size\ used\ to\ partition\ the\ input\ message)$. C denotes capacity (a measure of the achievable complexity of the sponge construction). Using the reading data sensor, hashing is performed (state). The state is initially padded with the first "1" and then zeroes to achieve a length of 1088 bits (r). The state variable's operation, b = r + c bit, is initialized to all zeroes when making a sponge and is updated each time. The two phases of sponge construction are the absorption phase and the compression phase. The absorbing phase is carried out as follows: the input block to be processed is padded with all zeroes for each iteration (in our proposed system, just one iteration), increasing its length from r (1088 bits) to b (1600 bits), and then a bitwise XOR of the extended message block and s is formed to create a b bit input to iteration function f. The output of f is the value of s, which is the input for the following stage. We need 24 rounds in f; five internal steps are in each round. The first block Z_0 in the squeezing phase is the first 256 bits of s. (in our proposed system block Z_0 is hash data). Then, for each iteration, the values of s are updated by repeatedly running f; the first 256 bits are used as a block Z_i and are concatenated with previously created blocks. The result of the hashing process under the system we've described is H-State.

4.1.5 Key Generation using 5D Chaotic System

In order to analyze chaotic qualities like $randomness, dynamics, and\ sensitivity$ to the initials and equation parameters and produce a set of numerical output sequences, the suggested new 5D chaotic system of differential dynamic, chaotic equations is being put into reality. The following are the recently suggested 5D chaotic system equations:

$$xs[\mathrm{i}+1] = (\mathrm{xs}[\mathrm{i}] + \big(s \times (ys[i] + ks[i])\big))\%1 \quad (1)$$

$$\mathrm{ys}[\mathrm{i}+1] = \Big(\mathrm{ys}[\mathrm{i}] + \big((\mathrm{xs}[\mathrm{i}] * (\mathrm{r} - \mathrm{zs}[\mathrm{i}] + \mathrm{ps}[\mathrm{i}])\big)\Big)\%1 \quad (2)$$

$$\mathrm{zs}[\mathrm{i}+1] = \mathrm{zs}[\mathrm{i}] + ((\mathrm{ps}[\mathrm{i}] \times \mathrm{ys}[\mathrm{i}] - b \times ks[i]))\%1 \quad (3)$$

$$\mathrm{ks}[\mathrm{i}+1] = (1 - (\mathrm{L} \times \mathrm{ks}[\mathrm{i}] \times \mathrm{xs}[\mathrm{i}]) + \mathrm{ps}[\mathrm{i}])\%1 \quad (4)$$

$$\mathrm{ps}[\mathrm{i}+1] = (\mathrm{B} \times \mathrm{ks}[\mathrm{i}])\%1 \quad (5)$$

where xs, ys, zs, ks, and ps are the beginning conditions for the chaos map, while s, r, b, L, B, and dt are the chaotic parameters. Lyapunov exponents were calculated for various initials and parameters after the suggested new 5D chaotic system was implemented and tested. For example, with parameters (s =0.01, r =2.8, s =0.95, b =2.667, L=1.4, B=0.3, and dt=0.01), the

initial values for xs = 0.01, ys = 0.02, zs = 0.03, ks = 0.01, ps = 0.02, the proposed new 5D chaotic system displays super-chaotic Lyapunov values with positive values. All suggested systems employ the created chaotic keys (K_1, K_2,..., K_5), which will be kept in the file to make it easier to use and refer to them in subsequent processes. The proposed 5D Lorenz chaotic map's chaotic attractors for each plane are shown in Figure 4.

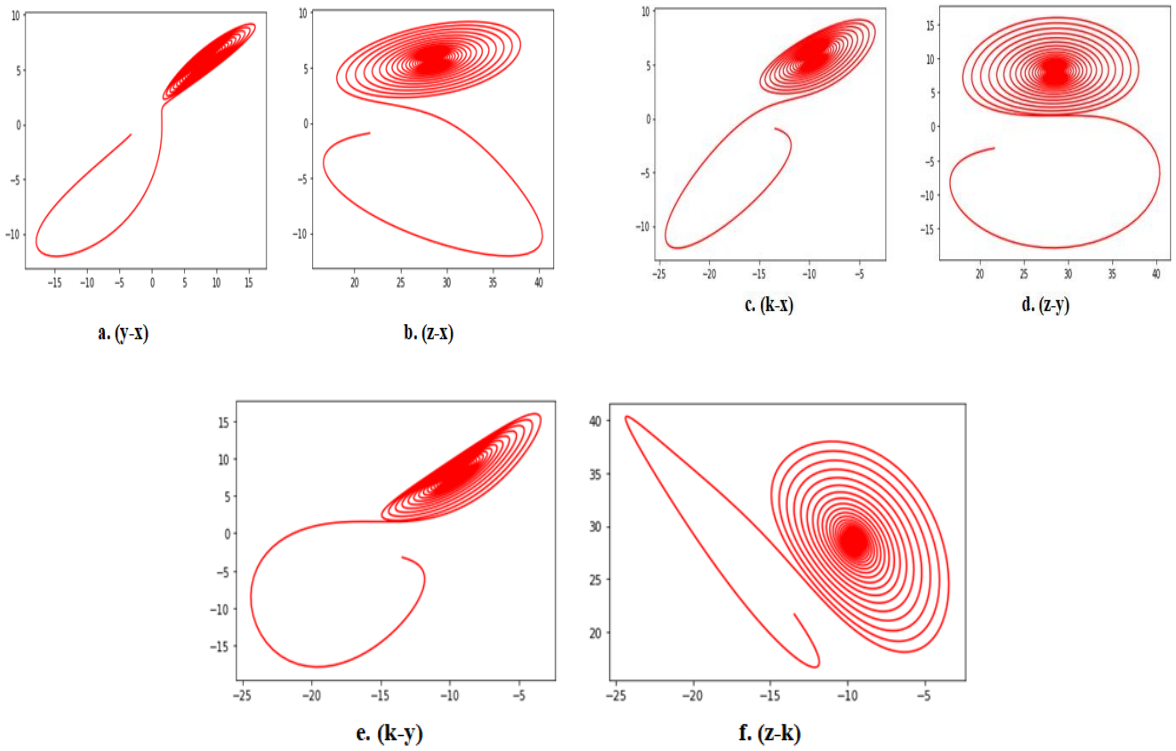

Figure 4: Chaotic attractors of each plane of 5D Lorenz chaotic map

4.2 Sending Data over the MQTT Protocol

Now the hashed and encrypted data are ready to be sent over the internet or intranet, so we used MQTT protocols to send the data. First, we named the patient node ("node") and the healthcare node ("server"). The configuration of the MQTT protocol is as follows: IP number, unencrypted MQTT port (1883), QoS (0), and keep alive (60).

4.3 Healthcare Side

In this phase of the proposed system, we used a PC as the server for the healthcare unit. First, the healthcare server receives the cipher text from the patient side. Then, we perform a separation process to separate the hashed data (H-State) and encrypted data (E-State). After that, we decrypt (E-State) by generating 32 random chaotic system keys (five dimensions), Salsa20 to generate a 512-bit block of the keystream (16 words), the DNA algorithm, and adding a round into (D-State). Then, we separate D-State into O2, *HR*, and TEMP. Also, we hash decrypt data using SHA3-256 into HD-State. Finally, we check if the HD-State equals the H-State; if yes, then print authorized and store O2, *HR*, and TEMP in the dataset; otherwise, print unauthorized and discard the data. Algorithm 3 and Figure 5 present the server side of the proposed system.

Algorithm 3: Server-Side
Input: Ciphertext
Output: O2, *HR*, and TEMP
Begin
Step1: Received Ciphertext from the Patient's Side
Step 2: Separate Ciphertext into (H-State) and (E-State).
Step3: Keys Generation using 5D Chaotic Systems
Step4: Decrypt E-State using (32 Keys Generation, Salsa20, DNA, and Add Round) into (D-State)
All processes can explain in algorithm four on the server side.
Step5: Separate D-State into (O2, HR, and TEMP)
Step6: Hashing D-State into (HD-State)
Step7: If H-State== HD -State
Step8: Print Authorized and Save (O2, HR, and TEMP)
Step9: Else
Step10: Print Unauthorized and Discard (O2, HR, and TEMP)
End

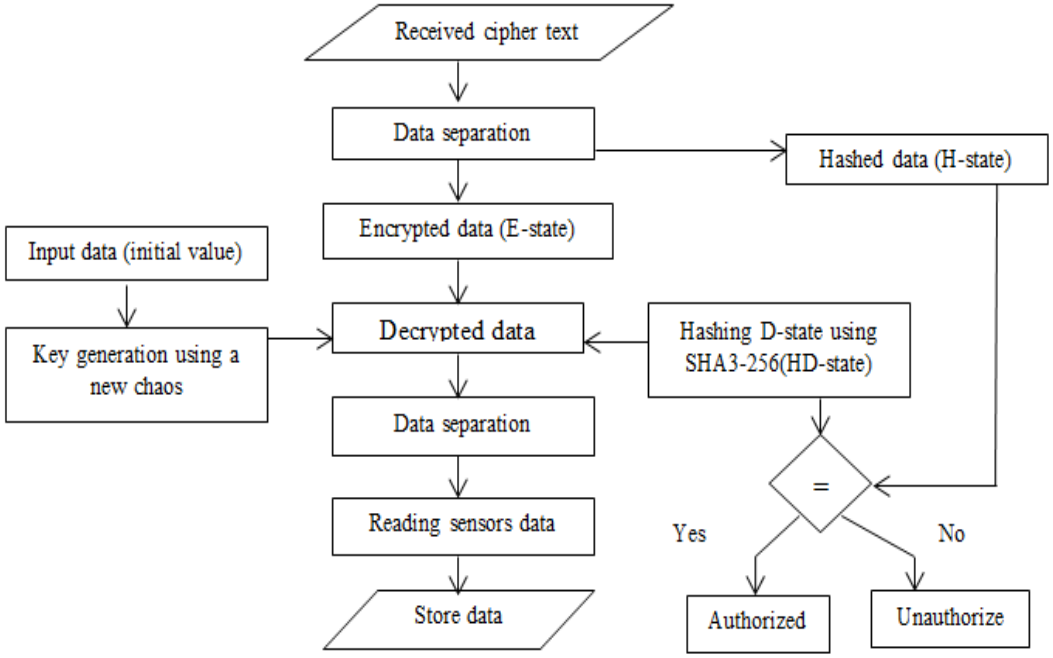

Figure 5: Server-Side

The server side has three processes:

4.3.1 Receiving cipher data

In the proposed system, after transmitting cipher text over the internet or intranet using the MQTT protocol, the healthcare side receives the data, and now it is ready to begin decryption and hashing the data and make sure the data is secure from threats and prepared to be stored in the database.

4.3.2 Cipher data separation

On the healthcare side, the first operation separates ciphertext into hacked data (H-State) and encrypted data (E-State). So we select the first 62 characters from the ciphertext string as H-State and the remaining string as decrypted data E-State.

4.3.3 Decryption proposed

On the healthcare side, after separation and receiving decrypted data (E-State), now it must perform the decryption process. In the proposed system, to decrypt data sensors, we used the Five-Dimensions Randomness Chaotic System to generate 32 keys, the Salsa20 algorithm to generate a 512-bit block of the keystream, the DNA algorithm, and $Add\ Round$ to decrypt data to avoid threats. Each previous process does the inverse; Figure 6 and algorithm (4) present the decryption process of the proposed system.

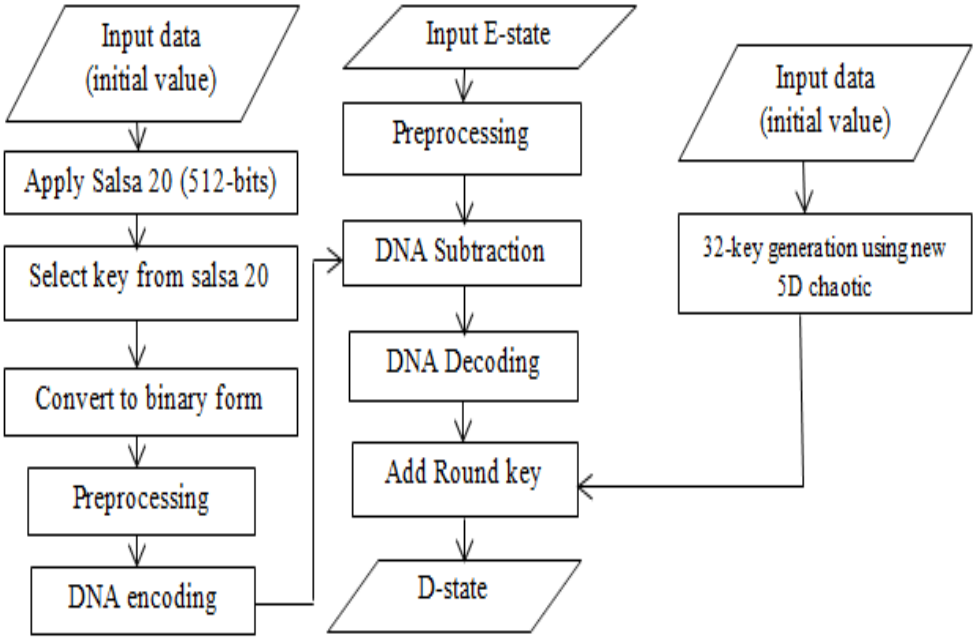

Figure 6: Decryption Process

Algorithm 4: Decryption Process
Input: Decrypted Date Sensors (E-State), initial values (x, y, z, k, p) to Generate Keys, and initial values for Salsa20
Output: D-State
Begin
Step 1: Convert E-State from Number to String into (DNA Add State).
Step 2: Generate 32 keys using Five-Dimensions Chaotic System.
Step 3: Apply Salsa20 to generate 512 bits block of keystream into (Final Salsa20).
Step4: Select Final Salsa20 (one word)
Step5: Convert Final Salsa20 to Binary Form
Step 6: Apply Pre-processing (Padding) if the length of the Final Salsa20 <> 32 bits.
Step7: Encoding Every Two bits with DNA Code into (DNA Salsa20) using (Table 1. a)
Step8: Applying DNA Subtraction Operation on (DNA Add State) and (DNA Salsa20) into (DNA Sub State) (using Table 1. c)
Step10: Decode DNA State using (Table 1. a).
Step10: Convert Decode DNA State to Decimal
Step11: Apply to Add Round
Step12: Return D-State
End.

5. Test and Experimental

In this section, we discuss how key space, key sensitivity, NIST, computing, and time, which are crucial assessment criteria for a text encryption system resisting an all-out attack, are met by our scheme in terms of the security parameters indicated above.

5.1 Security Space

The suggested strategy uses iteration times N0 as the secret key, with a value of 32. The five starting values of the 5-D chaotic system are indicated as ($xs[0]$ = 0.01, $ys[0]$ = 0.02, $zs[0]$ =0.03, $ks[0]$ =0,01, and $ps[0]$ =0.02). $Key - x_0(-40, 40)$, $-y_0$ $(40, 40)$, $Key - z_0$ $1, 81)$, $Key - k_0$ $(-250, 250)$, and $Key - p_0$ have the following ranges (1000, 2500). We may determine the key space by using the formula key space = 80×1015 ×80×1015 ×80×1015 ×500×1015 ×1500 ≈ 2^{237} > 2^{128} if all of the starting numbers have a precision of 10^{15}. The suggested approach may therefore be guaranteed to withstand a thorough attack because of the essential space volume.

5.2 Security Key Sensitivity

The degree of difference between two cipher texts, derived by encrypting the identical source text with two secret keys that have a slight modification, can be used to gauge the sensitivity of a key. In our suggested approach, there are five secret keys (keys x_0, y_0, z_0, k_0, and p_0). We randomly choose one set of keys from the secret key space to evaluate the sensitivity of the secret keys. Then, the plaintext is encrypted using the selected keys to produce the matching cipher text C1. Maintaining the same values for the other four keys (keys y_0, z_0, w_0, and N_0) will allow us to assess the sensitivity of key x_0. To get a fresh set of secret keys, swap out key x0 with key x_0 + 10-15. After that, use the new keys to encrypt the identical starting text to create text C2.

5.3 NIST Roundness Test

The NIST test suite includes 15 statistical metrics. The NIST metrics verify the randomness of binary series, such as cipher text or crypto PSNG. A P-value is calculated for each of these tests. The results for each NIST test were displayed in a table, revealing that every test had a success rate of > 0.01 when it came to execution [27] and [28]. For example, Table 2 shows the NIST statistical test results for the proposed algorithm for encryption cipher text. All of these tests yielded essentially identical results.

Table 2: NIST Test Results

NIST statistical tests Name	Length of input size in bits and other parameters	Proposed Algorithm
Frequency	100 bits	Pass
Block Frequency's	1000 bits, M=128	Pass
Discrete Fourier Transform	6000 bits	Pass
A**pproximate** Entropy	1000 bits, M=11	Pass
Cumulative sums	100 bits	Pass
Serial	100 bits, M=4	Pass
Runs	10000 bits	Pass
Longest Runs of ones in a block	6272 bits, M = 128	Pass
Overlapping template of all one's	1000 bits, M=9	Pass
Non- overlapping templates	1000 bits	Pass
Linear complexity	10000 bits, M=500	Pass
Binary matrix ranks	38,912 bits, M=32, Q=32	Pass
Maurer's universal statistical	387,840 bits, L=5, Q=320	Pass
Random Excursion	1000000 bits	Pass
Random Excursion Variant	1000000 bits	Pass

5.4 Encrypted Time Analysis

The running speed of the cryptosystem is a key indicator in addition to the security performance. The execution time is measured in seconds. Therefore, our technique performs better in terms of encryption speed. Additionally, completing encryption and decryption is related to computational complexity. The time-consuming aspects of our suggested strategy mainly involve the creation of the key streams in the Salsa20 chaotic system: DNA encoding, decoding, and hashing 256. Time complexity studies show that the DNA encoding and decoding processes have greater time complexity. The encryption time is 1.872, the throughput (in bits per second (bps)) is 273.2642, and the memory usage (in megabytes) is 10.6211.

6. Conclusion

We all know that the Internet of Things is one of the most coveted solutions for health monitoring right now. The most crucial aspect is that any doctor may remotely check on any patient's health since it ensures that the data is safe while being transmitted over the internet or intranet to the healthcare facility. The MQTT protocol is used by this device to send data to the healthcare facility about a patient's body temperature, heart rate, and blood oxygen saturation ($SpO2$) levels. The key principle of this study is to assure the confidentiality, availability, and integrity of the data by decreasing the use of memory and processing time. Experimental results demonstrate that the proposed system guarantees increased security while still providing integrity through SHA3-256, where The encryption time is 1.872, and the throughput (in bits per second (bps)) is 273.2642. On the other hand, the memory usage (in megabytes) is 10.6211. Furthermore, good results were obtained from evaluating the randomness of the cipher's output using three statistical test suites: NIST, key-sensitive, and computing. Therefore, the proposed system is a perfect fit for healthcare and telemedicine applications, which often send contextual data in small packets, avoiding the needless use of memory and processing power.

References

[1] A. T. M. Mohammed Shakir, "Lightweight Authentication Model for IoT Environments Based on Enhanced Elliptic Curve Digital Signature and Shamir Secret Share," *INASS, International Journal of Intelligent Engineering and Systems*, vol.15, no.5, pp. 81-90,2022.

[2] A. Y. Hussein and A. T. Sadiq, "Meerkat Clan-Based Feature Selection in Random Forest Algorithm for IoT Intrusion Detection," *Iraqi Journal of Computers, Communications, Control & Systems Engineering*, vol. 22, no. 3, pp. 15–24, 2022.

[3] M. A. Khodher, A. M. Alabaichi and A. S. Abbas, "Dual method cryptography image by two force secure and steganography secret message in IoT," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 6, pp. 2928-2938, 2020.

[4] I. M. Hasan and R. F. Ghani, "Blockchain for Authorized Access of Health Insurance IoT System," *Iraqi Journal of Computers, Communications, Control & Systems Engineering*, vol. 21, no. 3, pp. 76–88, 2021.

[5] R. W. L. Coutinho and A. Boukerche, "Modeling and Analysis of a Shared Edge Caching System for Connected Cars and Industrial IoT-Based Applications," *IEEE Trans. Ind. Informatics*, vol. 16, no. 3, pp. 2003–2012, 2020.

[6] A. Karati, S. K. Hafizul Islam, G. P. Biswas, M. Z. A. Bhuiyan, P. Vijayakumar, and M. Karuppiah, "Provably secure identity-based signcryption scheme for crowdsourced industrial internet of things environments," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2904–2914, Aug. 2018.

[7] D. G. Korzun, "Internet of Things Meets Mobile Health Systems in Smart Spaces: An Overview," *In book: Studies in Big Data, Berlin/Heidelberg, Germany, Springerlink*, pp 111–129, 2017.

[8] H. Deng, Z. Qin, L. Sha, and H. Yin, "A Flexible Privacy-Preserving Data Sharing Scheme in Cloud-Assisted IoT," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11601–11611, 2020.

[9] C. G. Thorat and V. S. Inamdar, "Implementation of new hybrid lightweight cryptosystem," *Appl. Comput. Informatics*, vol. 16, no. 1/2, pp. 195–206, 2020.

[10] S. A. Jassim, A. K. Farhan, "Designing a New Lightweight AES Algorithm to Improve the Security of the IoT Environment," *Iraqi Journal of Computers, Communications, Control & Systems Engineering*, vol. 22, no. 2, pp. 96–108, 2022.

[11] Y. N. Teja, L. Vooha, A. R. Priya, and N. V. K. Ramesh, "IOT based smart health care," *Int. J. Eng. Technol.*, vol. 7, no. 11, pp. 470–472, 2018.

[12] C. Li, X. Hu, and L. Zhang, "The IoT-based heart disease monitoring system for pervasive healthcare service," *Procedia Comput. Sci.*, vol. 112, pp. 2328–2334, 2017.

[13] A. S. El Batouty , H. H. Farag , A. A. Mokhtar , A. El-Badawy and M. H. Aly, "Improvement of Radio Frequency Identification Security Using New Hybrid Advanced Encryption Standard Substitution Box by Chaotic Maps," *Electronics*, vol. 9, no. 1168, 2021.

[14] J. Rokan Naif, G. H. Abdul-Majeed, and A. K. Farhan, "Internet of Things Security using New Chaotic System and Lightweight AES," *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 11, no. 2, pp. 45–52, 2019.

[15] M. M. Dhanvijay and S. C. Patil, "Internet of Things: a survey of enabling technologies in healthcare and its applications," *Comput. Networks*, vol. 153, pp. 113–131, Apr. 2019.

[16] M. M. Islam, A. Rahaman, and M. R. Islam, "Development of Smart Healthcare Monitoring System in IoT Environment," *SN Comput. Sci.*, vol. 1, no. 3, pp. 1–11, 2020.

[17] V. Tamilselvi, S. Sribalaji, P. Vigneshwaran, P. Vinu, and J. Geetharamani, "IoT Based Health Monitoring System," *2020 6th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2020*, pp. 386–389, 2020.

[18] H. T. Yew, M. F. Ng, S. Z. Ping, S. K. Chung, A. Chekima, and J. A. Dargham, "IoT Based Real-Time Remote Patient Monitoring System," *Proc. - 2020 16th IEEE Int. Colloq. Signal Process. its Appl. CSPA 2020, no. March 2021*, pp. 176–179, 2020.

[19] H. A. El Zouka and M. M. Hosni, "Secure IoT communications for smart healthcare monitoring system," *Internet of Things*, vol. 13, p. 100036, Mar. 2021.

[20] M. M. Khan, S. Mehnaz, A. Shaha, M. Nayem, and S. Bourouis, "IoT-Based Smart Health Monitoring System for COVID-19 Patients," *Comput. Math. Methods Med.*, vol. 2021, 2021.

[21] H. Y. Mohamed, R. Shahin, O. R. Hamed and M. H. Abdellattif, "Analyzing the Patient Behavior for Improving the Medical Treatment Using Smart Healthcare and IoT-Based Deep Belief Network," *Journal of Healthcare Engineering*, vol. 2022, no. 63, pp. 8, 2022.

[22] J. P. Eckmann and D. Ruelle, "Ergodic theory of chaos and strange attractors," *Reviews of Modern Physics*, vol. 57, no. 3, pp. 273–312, 2008.

[23] F. Masood, M. Driss, W. Boulila, J. Ahmad, S. Ullah et al., "A Lightweight Chaos-Based Medical Image Encryption Scheme Using Random Shuffling and XOR Operations," *Wirel. Pers. Commun.* vol. 2021, no. 127, pp. 1-28, 2021.

[24] A. Alabaichi, "True color image encryption based on dna sequence, 3D chaotic map, and key-dependent DNA S-box of AES," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 2. pp. 304–321, 2018.

[25] P.A. Nikolov, "Analysis and Design of a Stream Cipher," *M.S. thesis, Univ. of Alicante, San Vicente del Raspeig*, España, 2019.

[26] G. Meiser, T. Eisenbarth, K. Lemke-Rust, and C. Paar, "Efficient implementation of eSTREAM ciphers on 8-bit AVR microcontrollers," *in Proceedings of the International Symposium on Industrial Embedded Systems*, Le Grande Motte, France , 2009.

[27] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *NIST Special Publication*, vol. 22, no. 4,

pp. 1-1-G-1, 2010.

[28] M. S. Fadhil , A. K. Farhan, M. N. Fadhil, "A lightweight AES Algorithm Implementation for Secure IoT Environment, " *Iraqi Journal of Science*, vol. 62, no. 8, pp. 2759-2770, 2021.