ISSN: 0067-2904

Improvement of Chacha20 Algorithm based on Tent and Chebyshev Chaotic Maps

Mustafa Hussein Taha* , Jamal Mustafa Al-Tuwaijari
Department of computer science, College of Science, Diyala University, Baqubah, Iraq

Received: 2/6/2020 Accepted: 25/8/2020

Abstract
Chacha 20 is a stream cypher that is used as lightweight on many CPUs that do not have dedicated AES instructions. As stated by Google, that is the reason why they use it on many devices, such as mobile devices, for authentication in TLS protocol. This paper proposes an improvement of chaha20 stream cypher algorithm based on tent and Chebyshev functions (IChacha20). The main objectives of the proposed IChacha20 algorithm are increasing security layer, designing a robust structure of the IChacha20 to be enabled to resist various types of attacks, implementing the proposed algorithm for encryption of colour images, and transiting it in a secure manner. The test results proved that the MSE, PSNR, UQI and NCC metrics of IChacha20 are better than those of the original Chacha20. Also, the proposed method has a faster execution time (01:26:4 sec) compared with the original algorithm (02:07:1 sec).

Keywords: Chacha20, Tent function, Chebyshev function.

تحسين خوارزمية Chacha20 بالاعتماد على الدوال الفوضوية

مصطفى حسين طه*, جمال مصطفى التويجري
قسم الحاسبات, كلية العلوم, جامعة ديالى , بعقوبة , العراق

الخلاصة
خوارزمية تشاتشا ٢٠ هي خوارزمية تشفير من نوع خفيفة الوزن تستخدم للعديد من انوع وحدات المعالجة المركزية التي ليس لها تعليمات (AES) المخصصة لها . لذلك قالت عنها شركة غوغل ان لهذا السبب يتم استخدامها على العديد من انواع الاجهزة المحمولة للمصادقة على بروتوكولTLS. اقترحت هذه الورقة البحثية تحسين هذه الخوارزمية (IChacha20) بالاعتماد على الدوال الفوضوية (tent function, Chebyshev function) تتمثل الاهداف الرئيسية لهذه الخوارزمية المقترحة (IChacha20) في زيادة مستوى الامان وتصميم بنية قوية لتتمكن من مقاومة انواع مختلفة من الهجمات. هنا يتم تنفيذ الخوارزمية المقترحة لتشفير صور ملونة ليتم نقلها بأسلوب أمن .اثبتت التجارب العملية ونتائج الاختبارات ان المقاييس الخاصة بتشفير الصور مثل (MSE, PSNR, UQI and NCC) لـ (IChacha20) افضل من (Chacha20) الاصلية كما أن الطريقة المقترحة لها وقت تنفيذ أسرع مقارنة بالخوارزمية الأصلية, حيث يكون وقت تنفيذ الخوارزمية الأصلية = 02: 07: 1 ثانية بينما يكون وقت تنفيذ الخوارزمية المقترحة = 01: 26: 4 ثانية.

1. Introduction

The development of a system of data communication, such as networks of computers, cellular phones, etc. has been earning momentum in latest years. The advent of such systems of

*Email: mustafa.hussein@sciences.uodiyala.edu.iq

communication leads to major troubles of security on transmission of confidential data such as military data, messages, confidential images, etc. A secure environment or circumference would not be possible without encryption technology. Cryptography, encryption algorithm, and chaotic maps are three prevalent techniques in order to make digital data protected from prohibitive access and illegitimate usage. The algorithms of cryptographic are categorized into cyphers of the stream and cyphers of the block. In stream cyphers, the data are coded bit by bit by utilizing a secure key generator, whilst in block cyphers bits blocks are ciphered [1].

These techniques are used to overcome the difficulties in the original encryption algorithms (AES, DES, and Blowfish, etc.), such as the magnitude of time, the resources of the computations, and the high power for real-time. Various cypher algorithms have been proposed which depended on chaotic maps [2]. The Chacha 20 encryption algorithm is the stream cypher algorithm developed by Bernstein. It is based on the Salsa 20 algorithm; however, it varies in specifics and equips better security than the original Salsa20 cypher, by utilizing somewhat better hash functions. The function of hash input data was rearranged to permit to extra-efficaciously perform the algorithm [3]. In this research paper, an enhancement of the algorithm of Chacha 20 is introduced by using two kinds of functions of the chaotic maps (IChacha20).

2. Related Works

Bernstein (2008) presented a document explaining the Chacha algorithm family of stream cyphers. It has a different form of a family of the Salsa20 algorithm. Chacha20 pursues the same design rules as for the salsa algorithm, but it differs in some of the details. Bernstein designed the Chacha algorithm for an increased amount of diffusion during each round. He realized that the minimal number of secure rounds for the Chacha algorithm is smaller (and not larger!) than the minimal number of secure rounds for Salsa 20. Therefore, if the Chacha algorithm has the same minimum number of secure rounds as the Salsa20 algorithm, then the Chacha algorithm will supply better overall speed than the Salsa20 algorithm for the same level of security [3].

Ganesan and Sobti (2016) analyzed and interpreted the property of the diffusion of Quarter Round (QR) of both the Salsa20 algorithm and the Chacha algorithm, in addition to a proposed alternative design called Modified Chacha Core (MCC). They compared the Quarter round (QR) functions of all these three algorithms using the diffusion matrices that reflect a change in output words with a small change in input words. They generated more than a million diffusion matrices for each algorithm depending on the possible permutations of rotation constants used in the QR. They also proved that, for the Salsa algorithm and Chacha algorithm core, there are a high number of alternative rotation constants that generate more diffusion than the original rotation constants. So, they suggested using MCC core to generate a collision–resistant function of compression for the encoded hash algorithm [4].

Choudhuri and Maitra (2016) proposed a mixture system, under confirmed hypotheses, where nonlinear rounds are initially considered as suggested by the designer, then their linear counterpart is used. The effect of reversing rounds was also being considered with the Probabilistic Neutral Bits(PNB) idea. Drawing on assumptions and analysis, they concluded that 12 rounds of Chacha and Salsa algorithms must be considered sufficient for keys of 256 bit under the current better-known models of attack. They advised that this model can have the prospected applications in other ARX based cyphers **[5]**.

Miyaji and Matsuoka (2018) described the existing security analysis. Firstly, they described the stream cyphers Chacha and Salsa, then the existing security analysis, because Chacha needs more analysis of security since it has been suggested more recently. Hence, they suggested a comparison with the AES algorithm. Moreover, Chacha is an enhancement of Salsa from the perspective of diffusion . Salsa algorithm was significant to understand their design of criteria of security. In that research paper, the researchers revisited the diffusion analyzis and investigated weak columns and bits of Salsa and Chacha. They confirmed that their work was the first of its kind [6].

Dey *et al*. (2019) initially revisited the existing attacks on Chacha and Salsa cyphers. Firstly, they applied an accurate computation of the attack complexities of the existing technique instead of the estimation used in previous works, which improved the complexity time to some extent. The differential attacks that utilize Probabilistic Neutral Bits(PNB) against Salsa and Chacha include two biases of probability, namely the forward bias of probability ($\in$d) and backward bias of probability ($\in$a). In the second part of that paper, an approach to grow the backward bias of probability was suggested, which aids to decrease the attack complexity. Lastly, they concentrated on the principles of the design of Chacha. They proposed a slight or simple modification in the designing of this cypher as a countermeasure against differential attacks. They showed that the key recovery attacks suggested against this modified version are not effective for Chacha **[7]**.

3. Chacha20 Algorithm

The Chacha20 algorithm produces the keystream of 64-Bytes. The input to the keystream matrix is separate from the plain text or coded text [8]. This allows equivalent cypher text generation with a consequent performance enhancement. Chacha algorithm includes twenty rounds of computations of mathematical They are all used X-OR operation, the addition operation and bit rotation operation then take inputs such as a 4-byte constant, a random 32–byte key, a 12–byte nonce , and a 4-byte counter (In original, Bernstein determine the values of each of counter and the lengths of the nonce to be eight values).The 4-Byte constants are: (σ0, σ1, σ2, σ3) = ("0x61707865"," 0x3320646e"," 0x79622d32" , "0x6b206574"), or in ASCII they are "apxe", "3 dn", "yb-2", "k et". In Chacha20 algorithm, these strings are successive. The counter typically begins with 0 or 1, with increment to every 64Byte plaintext block [9]. Chacha20 gathers a 256–bit key and a 32–bit nonce that contain a counter. This leads to creating a keystream, which is a combination of using X–OR with the plaintext. Chacha20 algorithm operates on 32bit words at a time with a key of 256–bits where K= k0, k1, k2, k3, k4, k5, k6, k7. These blocks are the output of 512–bits for the keystream Z, Then the X-OR operation is executed between the keystream and the original text. The encryption state is stored within a 16x32bit word's value and arranged as a 4x4 matrix [9], as follows:

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} \qquad \dots(1)$$

Chacha20 algorithm then locates a four-quarter round function, as shown in the algorithm (1) [9].

Algorithm (1): Quarter round

```
Result: QR (a, b, c, d)
a=a + b ; d = d ⊕ a ; d=(d) << 16;
c=c + d ; b = b ⊕ c ; b=(b) << 12;
a=a + b ; d = d ⊕  a ;   d=(d) << 8;
c=c + d ; b = b ⊕  c ;   b= b) << 7;
```

The structure of the primary array is defined as shown below:

$$x = \begin{pmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \sigma_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_0 & k_0 & k_2 & k_3 \\ n_0 & n_1 & c_0 & c_1 \end{pmatrix} \text{ (For a 256-bit key)} \qquad \dots(2)$$

$$or = \begin{pmatrix} T_0 & T_1 & T_2 & T_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ v_0 & v_1 & i_0 & ci_1 \end{pmatrix} \text{ (for a 128-bit key)} \qquad \dots(3)$$

The quarter-round functions are exercised to the column (x0, x4, x8, x12), (x5, x9, x13, x1), (x10, x14, x2, x6) & (x15, x3, x7, x11) in odd rounds, and diagonal (x0, x5, x10, x15), (x1, x6, x11, x12), (x2, x7, x8, x13) &(x3, x4, x9, x14) in even rounds.

Algorithm (2) characterizes the full procedures of Chaha20 algorithm [9].

Algorithm (2): Chacha20 algorithm

```
Required: Key K, Block Counter C, and Nonce ( N)
Ensure: Keystream (Z)
X ←primary array (K, C, N)
Y ←X;
for i = 1 to 10 do
// Column Round
(x0,x4;x8,x12) ← QuarterRound (x0,x4,x8,x12)
```

```
(x5,x9,x13,x1) ← QuarterRound (x5,x9,x13,x1)
(x10,x14,x2,x6) ← QuarterRound (x10,x14,x2,x6)
(x15,x3,x7,x11) ← QuarterRound (x15, x3, x7, x11)
// Diagonal Round
(x0, x5;x10, x15) ← QuarterRound (x0,x5;x10,x15)
(x1, x6, x11, x12) ← QuarterRound (x1,x6,x11,x12)
(x2;x7,x8,x13) ← QuarterRound (x2,x7,x8,x13)
(x3, x4, x9, x14) ← QuarterRound (x3, x4, x9, x14)
end
Z ←X + y
Return Z
```

4. Chaotic Maps

The chaotic series has various useful attributes of application depending on security; it is a dynamic method in a discrete time to result in a complex sequence. The noun is not random; however, it is deterministic. This characteristic allows us to renew it with a very high sensitivity in the initial condition. This leads to another initial arrangement which makes another sequence. Also, the chaotic sequence path has an entropy behaviour in a particular space, which causes the recovery of this sequence to be impossible in its special space. The chaotic maps are separated into two portions, namely one-dimensional and multi-dimensional maps [10].

4.1 Chebyshev 1D Chaotic Map

Definition **1**: The Chebyshev polynomial by degree n is specified as shown below.

$$T_n(x) = \cos(n * arc(\cos(x))) \qquad \dots (4)$$

where **n** is an integer value, $\mathbf{x} \in [1, -1]$

Definition 2: Semi-group properties for Chebyshev can be achieved as shown below.

$$Trs(x)=Tr(Ts(x)) = Ts(Tr(x)) \qquad \dots (5)$$

Definition 3: The Chebyshev polynomial in (n) degree presents: x,Tx(x). It is unfeasible in computation to define the polynomial order (n).

4.2 The Chaotic Tent Map Function

Yoshida *et al*. [11] studied the chaotic behaviours of the tent maps' functions (a linear of piecewise, continued map with a unique maximum) analytically, which is a messy region in terms of the fixed density and the power spectrum. With lowering the max limit, the sequential band-splitting transitions happen in a messy region and accumulate to the transition point into the non-messy region. The time of the correlation function of nonperiodic orbits and their spectrum of power are computed exactly at the band splitting points and in the neighborhood of these points. The tent map function is topologically associated, and thus, the behaviour of the maps is identical in this sense, under iteration.

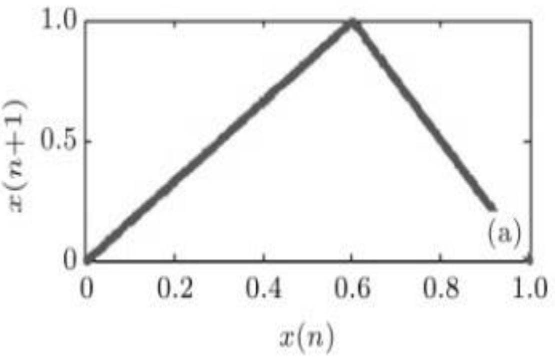


Figure 1-Graph of Tent Map Function [12].

The chaotic tent map [13] is given by

$$x_{i+1} = f(x_i, \mu)$$

$$F(x_i, \mu) = \begin{cases} FL(x_i, \mu) = \mu x_i, & \text{if } x_i < 0.5 \\ FR(x_i, \mu) = \mu(1 - x_i), & \text{O.w} \end{cases} \quad \dots (6)$$

where $x_i \in [0, 1]$, for $i \geq 0$.

This map converts a period [0, 1] onto itself and includes only one control parameter μ, where μ belonged [0, 2]. While x0 is the initial value of the method. The real set of values (x0, x1, dn) provides the name of the orbit of the method.

5. The Proposed Method

The proposed method is represented by an improvement of the Chacha20 algorithm that uses two chaotic maps as pseudo number generator to be used in internal operations, as shown in Figure-2 which clarifies the general block diagrams. The proposed algorithm consists of three main stages; the chaotic stage, generating Chacha20 key stage, and the final stage is the encode/decode of the colour image. Each of these stages is clarified in details in the following subsections.

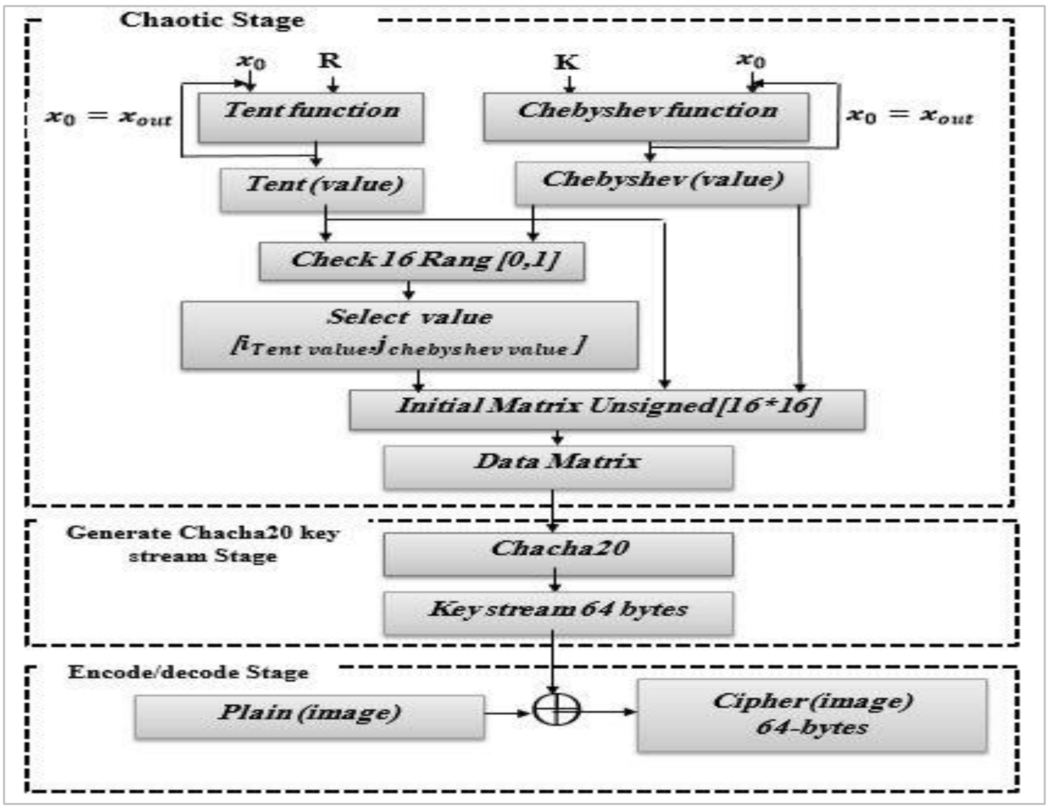


Figure 2-General block diagrams of the proposed (IChacha20).

5.1 Chaotic Maps

This stage demonstrates an additional layer of security to the original Chacha 20 algorithm. The objective is to generate a random unsigned number based on the two chaotic functions of Tent maps and Chebyshev. This stage consists of two sub-steps:

i) **Creating the Initial Matrix**

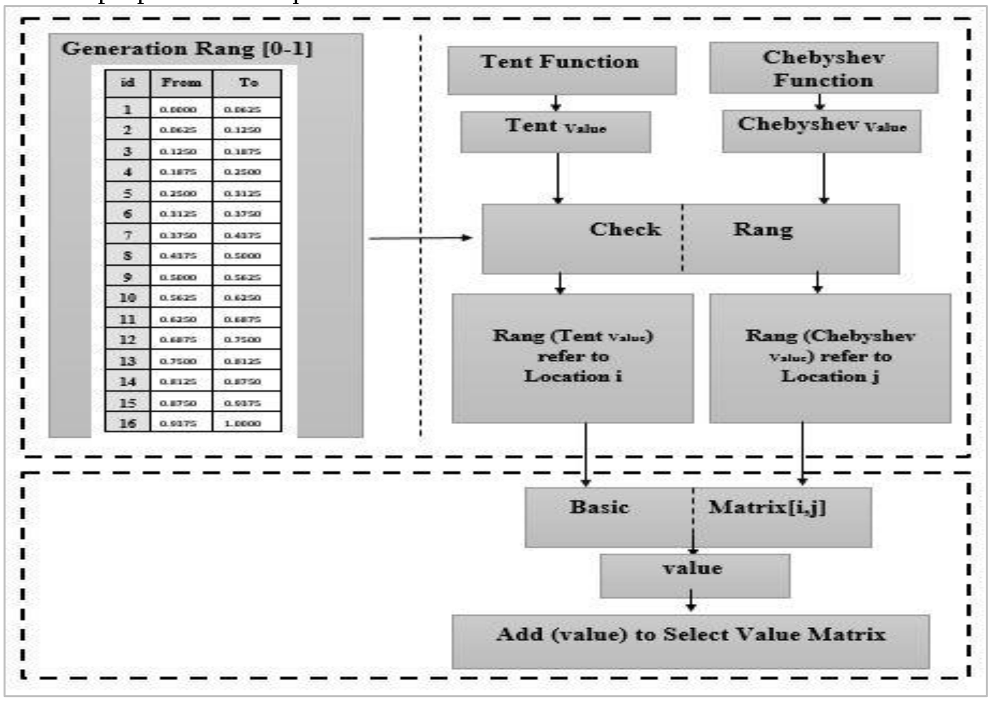
This step aims to create an initial matrix with the size of 16*16 and the value of this matrix is 32bit integer randomly unsigned using the chaotic functions of 1d Tent maps and Chebyshev. Both Tent maps and Chebyshev follow the same procedure to generate the randomly unsigned 32-bit integer number. These numbers are used to full the initial matrix. So as to achieve this, for example, the Tent maps utilize parameters as inputs to its function. These parameters include X0 (Primitive value) =

0.6556, R= 0.954545, and the number of rounds which is 0-100. If the value of Tent map = 0.65388127808236, then:

1. Separate this value and take the digits after the decimal point '.' .
2. Save this value in an array xdigit[0] =(65388127808236).
3. Find the opposite of 65388127808236 and save it in an array xdigit[1] = 6328087218836.
4. Finally, convert the xdigit value [ ] to an unsigned 32bit integer number and store it in an array data [No.Gen×4].

ii) Selecting Values from Initial Matrix

The objectives of this step are to increase complexity and diffusion that results in more security in the proposed method. In this step, the proposed technique is utilized to select, in a random manner, one element each time from the initial matrix that was created in the previous step. Figure-3 illustrates the specifics of the proposed technique.


Generation Rang [0-1]
Tent Function
Chebyshev Function
Tent Value
Chebyshev Value
Check
Rang
Rang (Tent Value) refer to Location i
Rang (Chebyshev Value) refer to Location j
Basic
Matrix[i,j]
value
Add (value) to Select Value Matrix

id	From	To
1	0.0000	0.0625
2	0.0625	0.1250
3	0.1250	0.1875
4	0.1875	0.2500
5	0.2500	0.3125
6	0.3125	0.3750
7	0.3750	0.4375
8	0.4375	0.5000
9	0.5000	0.5625
10	0.5625	0.6250
11	0.6250	0.6875
12	0.6875	0.7500
13	0.7500	0.8125
14	0.8125	0.8750
15	0.8750	0.9375
16	0.9375	1.0000

Figure 3- General block diagrams of the proposed technique to select values from the matrix.

As shown in Figure-3, the proposed technique is divided into three steps to enable the random selection from the initial matrix:

1- Generating a period table which consists of 16 rows, since the size of the initial matrix is [16*16]. The period table is called “range”, starting from 0 and ending at 1.

2- Generating a random number using equation (4) for Chebyshev function and equation (6) for the Tent maps function.

3- Dropping values of tent maps and Chebyshev functions into the range table to check for any periods with values that belong to the return index for that period. For example, if a chaotic value for Tent or Chebyshev = 0.65, then when dropping this value on the range table it appears in a period of 11,0.6250,0.6875, then in a return index of this period which is = 11.

4- Determining the location [Row $_i$, Column $_j$] of the element from the initial matrix [16*16], where Row $_i$ = is the index of the period that crossponds to the Tent value, whereas Column $_j$= is the index of period that crossponds to the Chebyshev value.

5- Adding the value of the element that was selected in the previous step into the selected matrix.

5.2 Generating Chacha20 Key Stream Stage

Figure-4 shows the stage of keystream generating for the proposed algorithm.

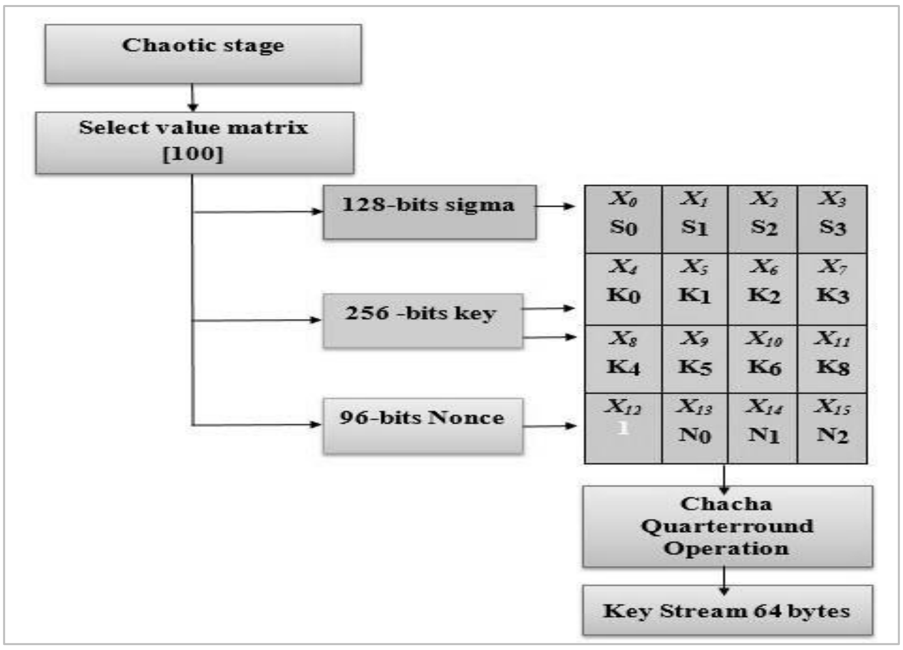


Figure 4-Block diagram of Chacha20 64-bytes key stream generating.

This step is objective to compute values of (key| nonce| sigma); the IChacha20 cipher operates on 32-bit words. It takes as input a 256-bit key K = (k_0, k_1, k_2, k_3, k_4, k_5,k_6 & k_7) and a 96-bit nonce N =(N_0, N_1, N_2 & N_3) , a 128-bit block sigma S=(S_0,S_1,S_2 and S_3), and a 32-bits counter which is equal to 1. The IChacha20 operates on a 4×4 matrix of 32-bit words called X matrix. To compute the 256-bit key, 96-bit nonce, and 128-bit block sigma, algorithms (1) and (2) are applied.

5.3 Encoding /Decoding Color Image Stage

Figure-5 shows the block diagram of IChacha20 Encryption /Decryption operation for color image.

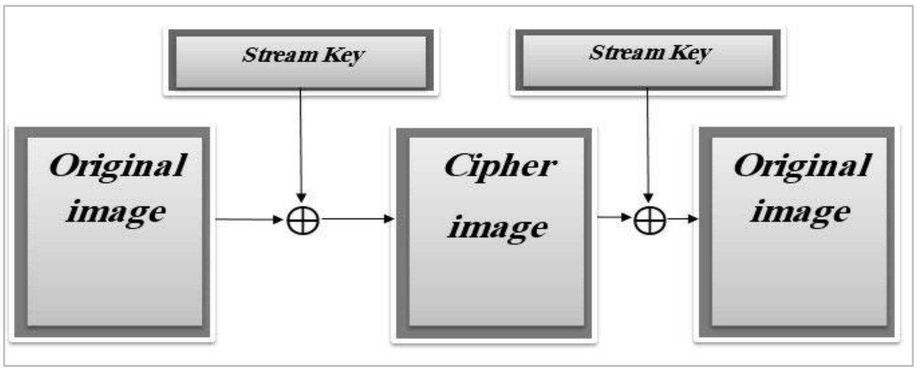


Figure 5-Block diagram of the encryption /decryption for color image using IChacha20 algorithm.

The IChacha20 successively calls the Chacha20 block function, with the same key , nonce, and sigma, with successively increasing block counter parameters to form a keystream. The IChacha20 algorithm then performs an XOR operation between the keystream and the entered data that are represented by the color images. Alternatively, each keystream block can be XORed with a plaintext (color image) block before proceeding to create the next block, saving some memory. The input of this encryption /decryption algorithm includes a 256-bits key, a 32-bit initial counter, a 96-bits nonce, 128-bits sigma, and finally, the original image with random size. The output is an encrypted image of the same size as that of the original image.

6. Experimental Results

The proposed IChacha20 algorithm is coded in C# and the tests are conducted on a PC with Intel(R) Core(TM) i7-5500U, CPU@ 2.40GHz, a memory of 16.0 GB RAM, and a 64-bit system kind. The implementation of key stream generation of the proposed algorithm is shown in Figures- 6.a, 6.b, and 6.c,where the setting parameters of the chaotic Tent map for all experiments are $R =$

0.954545, $x_0 = 0.6556$, number of iteration $= 100$, whereas the setting parameters of the Chebyshev chaotic map for all experiment are k=5, $x_0 = 0.2$ and number of iteration $= 100$.

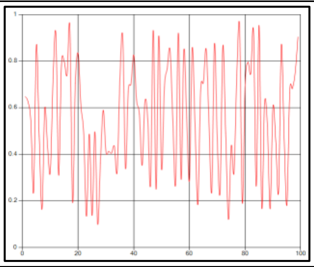
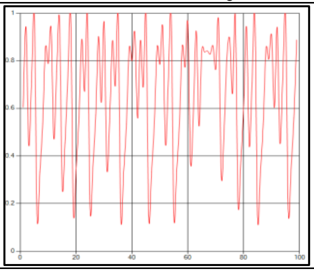
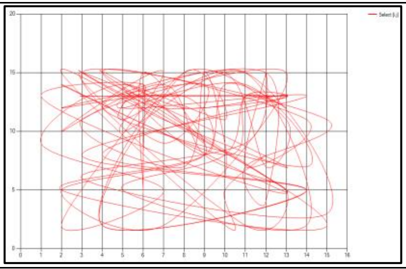
		
a) Tent behaivours with its setting parametes	b) Chebyshev behaivours with its setting parametes	c) Location (i,j) of elements selected from the select matrix

Figure 6- illustrates the key setup to compute the 256-bit key |96-bit-nonce|128-bits sigma.

K_0				**K_1**			
194	94	89	161	98	244	108	46
K_2				**K_3**			
127	107	118	104	210	32	36	181
K_4				**K_5**			
105	235	226	77	133	244	252	12
K_6				**K_7**			
23	108	125	193	32	87	1199	15

a) 256 –bits

N_0				**N_1**			
228	61	197	22	255	239	34	212
N_3							
97		245		32		42	

b) 96 –bits Nonce

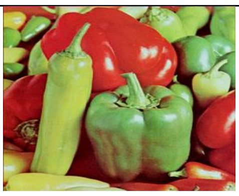
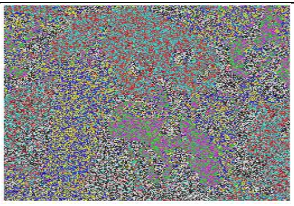
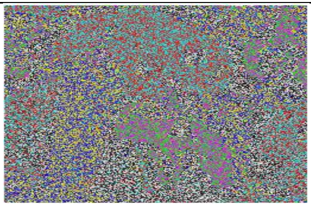
S0				**S1**			
39	245	164	12	244	36	200	103
S2				**S3**			
213	218	164	222	136	123	212	206

c) 128-bits Sigma

Figure-**6** Implementation of the key setup

The performance of the proposed IChacha20 and Original Chacha20 algorithms are evaluated using two different images, Pepper and Baboon, with a size of 256*256, in a RGB colour space based on XOR – operation. Both encryption algorithms work with a 256-bits key, 32-bit initial counter, 96 –bits nonce, and 128-bits sigma, as shown in Table-1. Table-2 shows the histogram of original and encryption images that are illustrated in Table-1.

Table (1) Image Encryption using IChacha20 and Original Chacha20

Image name	**Original image**	**Cipher image using original Chacha20**	**Cipher image using IChacha20**
Pepper			

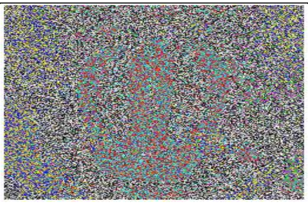
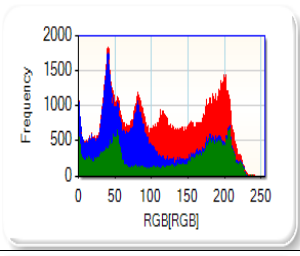
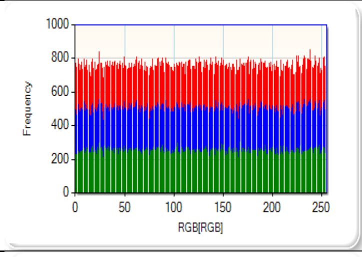
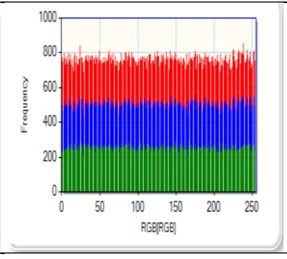
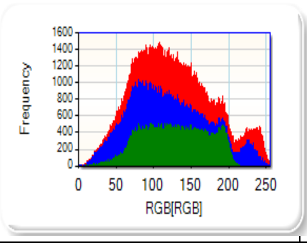
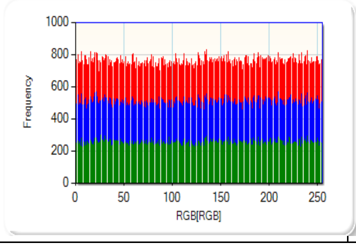
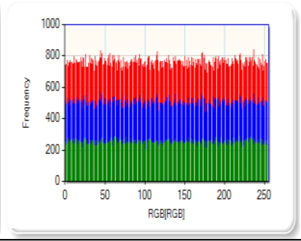
Baboon			

Table 2-Histogram of Image Encryption Using IChacha20 and Original Chacha20

Image name	**Histogram of Original image**	**Histogram of Cipher image using original Chacha20**	**Histogram of Cipher image using IChacha20**
Pepper			
Baboon			

As depicted in Table-2, it is clear that the visual histogram of cypher images for both the proposed IChacha20 and original chacha20 algorithms is distributed uniformly, and does not provide any information, so the attacker cannot know anything about the information. The resulting analysis is dependent on Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), Universal Quality Index (UQI), and Normalized Cross-Correlation (NCC). The MSE is a quantitative measure that clarifies the distinction between plain image and cypher image, as in Equation (7) [6, 14].

$$\mathrm{MSE} = \frac{1}{\mathrm{m}\times\mathrm{n}}\sum_{\mathrm{i}=0}^{\mathrm{m}-1}\sum_{\mathrm{j}=0}^{\mathrm{n}-1}[\mathrm{P(i,j)} - \mathrm{C(i,j)}]^2 \quad \ldots. \; (7)$$

where P (i, j) is the input image pixel value, C (I, j) is the value of cypher image pixel, N and M are the dimensions of the images.

The PSNR mathematical representation is as in Equation (8) below [6, 14].

$$PSNR = 10\log_{10}(MAX^2|MSE) \quad (8)$$

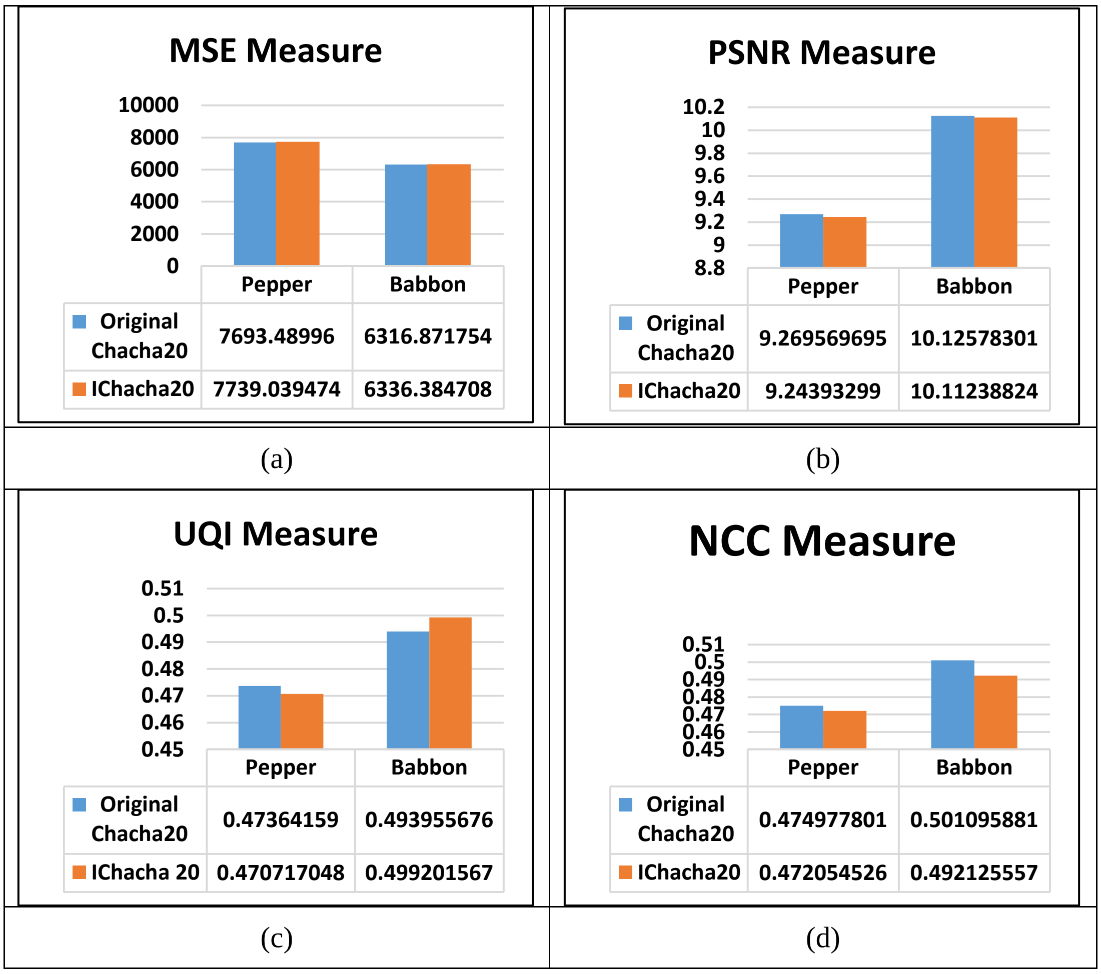
The UQI is relied upon to estimate the deformation of two the images, as shown in equation (9) below [8, 15].

$$UQI=\frac{4\sigma_{xy}\overline{x}\overline{y}}{(\sigma^2x+\sigma^{2y})[\overline{x}^2+\overline{y}^2]} \quad \ldots \; (9)$$

NCC is a metrics of similarity of two wavelengths as a function of the lost time applied to one of wavelengths. The value of NCC must be lower, that represents the low image quality. The NCC is calculated as shown in equation (10) below [8, 15].

$$\mathbf{NCC}=\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}x(m,n).y(m,n)}{\sum_{m=1}^{M}\sum_{n=1}^{N}(x(m,n))^2} \quad \ldots \; (10)$$

The performance comparisons between the proposed IChacha20 and the original Chacha20 encryption algorithms are illustrated in Figure-7, based on MSE, PSNR, UQI, and NCC metrics. Figure-8 shows a comparison between them based on execution time in seconds.



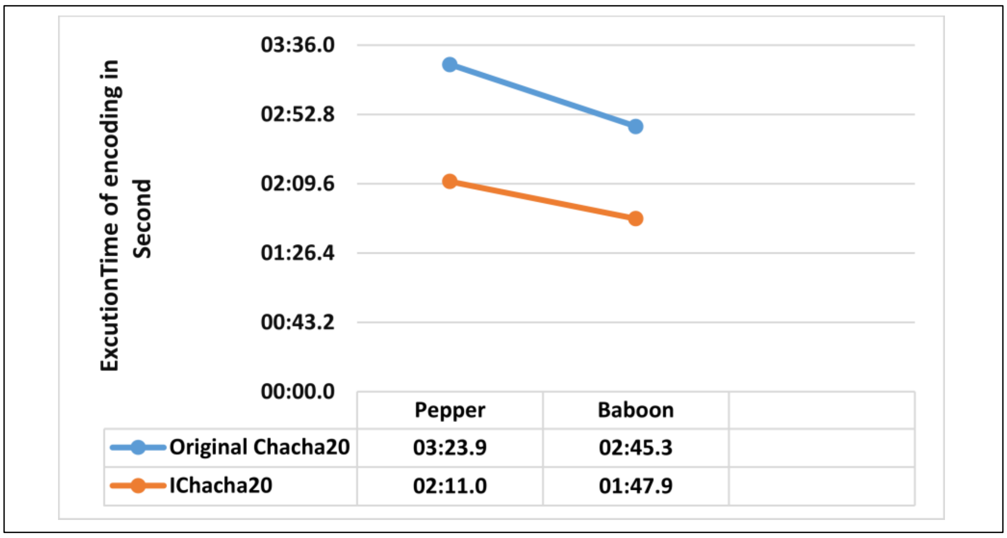
MSE	Pepper	Babbon
🟦 Original Chacha20	7693.48996	6316.871754
🟧 IChacha20	7739.039474	6336.384708

PSNR	Pepper	Babbon
🟦 Original Chacha20	9.269569695	10.12578301
🟧 IChacha20	9.24393299	10.11238824

UQI	Pepper	Babbon
🟦 Original Chacha20	0.47364159	0.493955676
🟧 IChacha 20	0.470717048	0.499201567

NCC	Pepper	Babbon
🟦 Original Chacha20	0.474977801	0.501095881
🟧 IChacha20	0.472054526	0.492125557

Figure 7-Evaluation of the Performance of the Proposed IChacha20 and the Original Chacha20 based on values of measures: a) MSE; b) PSNR; c) UQI; d) NCC



	Pepper	Baboon	
● Original Chacha20	03:23.9	02:45.3	
● IChacha20	02:11.0	01:47.9	

Figure 8-Execution Time of Encoding per Seconds

In Figure-7, the results indicate that the MSE value between the cypher and the input images is large. Also, the results of PSNR, UQI, and NCC show that the values between the original and the encrypted images using the proposed method are smaller than those of the original method. Figure-8 illustrates that the proposed algorithm has a faster execution time of encoding; 00:02:10.994 Sec) for the pepper image and 00:01:47.860 Sec for the baboon image.

7. Conclusions

Chacha20 is a 64-bytes stream cypher based on the 20-round cypher. It is designed to enhance the diffusion per round, conjecturally increasing resistance to cryptanalysis, while preserving, and often enhancing, time per round. In this work, an efficient method for the improvement of Chacha20 encryption algorithm for RGB image is proposed. The 64-bytes stream key of the suggested algorithm is generated by using the Tent maps function and Chebyshev-chaotic maps in addition to the proposed combination between the two chaotic maps. Colour image was divided into blocks with a size of 64-bytes based on XOR operation, applied with a 64-bytes stream key to produce the cypher image. Experimental results reveal that the proposed IChacha20 is more robust against attacks that intend to obtain any information from the encrypted image, being also faster compared with original Chacha20. Furthermore, the proposed algorithm provides a large key space and has a very high sensitivity to any simple change in the secret key.

Acknowledgement

We wish to express our sincere thanks to the University of Diyala as well as the College of Science- Department of computer science for providing a pleasurable and stimulating environment to carry out the research and for their timely support.

References

1. George, D.I., Geetha, J.S. and Mani, K., **2014**. Add-on Security Level for Public Key Cryptosystem using Magic Rectangle with Column/Row Shifting. In *IJCA,* **96**(14): 38-43.

2. Stallings, W., **2017**. "*Cryptography and network security: principle and practice*". (p. 743). Upper Saddle River, Nj: Pearson.

3. Bernstein, D. J., **2008**. Chacha, a variant of Salsa20. In Workshop Recorded of SASC,. **8**: 3-5.

4. Sobti, R., & Ganesan, G., **2016.** Analysis of quarter rounds of Salsa and Chacha core and proposal of an alternative design to maximize diffusion. *Indian Journal of Science and Technology*, **9**(3): 1 -10.

5. Choudhuri, A. R., & Maitra, S.,**2016**. Differential Cryptanalysis of Salsa and Chacha - An Evaluation with a Hybrid Model. IACR Cryptology ePrint Archive, 2016, 377.

6. Matsuoka, Y., & Miyajei, A., **2018**. Revisited Diffusion Analysis of Salsa and Chacha. In 2018 International Symposium on Information Theory and Its Applications (ISITA) (pp. 452-456). IEEE

7. Dey, S., Roy, T., & Sarkar, S.**,2019**. Revisiting design principle of Salsa and Chacha. *Advancers in Mathematics' of Communication*, **13**(4).

8. Robshaw, M., & Billet, O. (Eds.). **2008**. New stream cipher designs: the eSTREAM finalists (Vol. 4986). Springer.

9. Peter McLaren, William J Buchanan , Gordon Russell , Zhiyuan Tan. **2019**.Deriving Chacha20 Key Streams From Targeted Memory Analysis, *Journal of Information Security and Applications,* **48**: 102372 http://DOI: 10.1016/j.jisa.2019.102372 .

10. A Al-Tuwaijari, J. M. **2015**. Multi-Cipher Technique based on RNA and Chebyschev Map. *Iraqi Journal of Information Technology*, **7**(1):114-125.

11. Yoshida, T., Shige Matsu, H., Mori, H., **1983**. Analytic study of chaos of the tent map: band structures, power spectra, and critical behaviors. *J. Stat. Phys*. **31**(2): 279–308 .

12. Lv-Chen, C., Yu-Ling, L., Sen-Hui, Q., & Jun-Xiu, L., **2015**. A perturbation method to the tent map based on Lyapunov exponent and its application. *Chinese Physics* B, **24**(10): 100501.

13. D. Chappell and Jewell. **2002**. "Java Web Services", First Edition book of O'Reilly, ISBN: 0-596-00269-6, March.

14. Al-Tuwaijari, J. M. **2018**. Image Encryption Based on Fractal Geometry and Chaotic Map. *Diyala Journal For Pure Science*, **14**(1-Part 1): 166-182.

15. Kumar, Vijay and Dr: Badal Neelendra . **2017**. An approach of visual cryptography for grayscale and color Images using error –diffusion halftoniing technic, *International Journal of Computers Sciences and I information Technology AND Security* (IJCSITS), ISSN:2249-9555, **7**(1).