



Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Diyala
College of Science



Improvement of Chacha Algorithm in Mobile Communication

A Thesis

**Submitted to the Computer Science Department \College of
Science \University of Diyala
In a Partial Fulfillment of the Requirements for The Degree
of Master of Science in Computer**

**By
Mustafa H. Taha**

**Supervised By
Assistant Prof. Dr. Jamal M. Abbass**

2020 A.D.

1442 A.H.



﴿ قَالَ لَهُ مُوسَى هَلْ أَتَّبِعُكَ عَلَى أَنْ تُعَلِّمَنِ مِمَّا عُلِّمْتَ رُشْدًا ﴾

صدق الله العظيم

سورة الكهف : من الآية 66

ACKNOWLEDGMENTS

*First of all, Praise is to God, Lord of the worlds, for a blessing that helped me in achieving this research until the end of it, I would like to express my thanks and gratitude to my supervisor **Assist Prof. Dr. Jamal Mustafa Abbass** for supervising this research and for the bounty, patience and continued guidance throughout the work.*

My thanks to all academics and administrative staff at the Department of computer science.

ⓂⓊⓈⓉⓐⓕⓐ

Dedication.

*I would like to dedicate this
Work To:*

*The soul of both my father
and my brother, and also
dedicate to The rest of my
family*

MUSTAFA

Supervisor's Certification

We certify that this thesis entitled “*Improvement of Chacha Algorithm in Mobile Communication*” was prepared by "*Mustafa H. Taha*" Under our supervisions at the University of Diyala, Faculty of Science Department of Computer Science, as partial fulfillment of the requirement needed to award the degree of Master of Science in Computer Science.

Signature:

Name: Assistant Prof. Dr. Jamal Mustafa Al-Tuwaijari

Date: 30 / 6 /2020

Approved by the University of Diyala Faculty of Science Department of Computer Science.

Signature:

Name: Assistant Prof. Dr. Taha Mohammed Hassan

Date: 30 /6 /2020

(Head of Computer Science Department)

Linguistic Certification

This is to certify that this thesis entitled "***Improvement of Chacha Algorithm in Mobile Communication***" was prepared by "***Mustafa Hussein Taha***" at the University of Diyala/ Department of Computer Science, is reviewed linguistically. Its language was amended to meet the style of the English language.

Signature:

Name :

Date : / / 2020

Examination Committee Certification

We certify that we have read the thesis entitled “***Improvement of Chacha Algorithm in Mobile Communication***” and an examination committee, examined the student “***Mustafa Hussein Taha***” in the thesis content and that in our opinion, it is adequate as fulfill the requirement for the Degree of Master in Computer Science at the Computer Science Department, University of Diyala.

(Chairman)

Signature:

Name: **Prof. Dr. Ziyad Tariq Mustafa**

Date: / / **2020**

Signature:

Name: **Assist. Prof. Dr. Salah Awad Salman**

(Member)

Date: / / **2020**

Signature:

Name: **Assist. Prof. Dr. Adulbasit Kadhim Shukur**

(Member)

Date: / / **2020**

Signature:

Name: **Asst. Prof. Dr. Jamal Mustafa AL-Tuwaijari**

(Supervisor)

Date: / / **2020**

Approved by the **Dean** of College of Science, University of Diyala

(The Dean)

Signature:

Name: **Prof. Dr. Tahseen H. Mubarak**

Date: / / **2020**

Abstract

Security plays a vital role in digital information and data protection. To provide security reasonably and appropriately, encryption algorithms must be used. One of the most advanced algorithms in data security is Chacha20. It is a lightweight-stream cipher algorithm.

The proposed system was built based on the Chacha20 algorithm. A proposed system consists of a three-stage and NIST test: Chaotic maps stage, chaotic keystream generation stage, and IChacha20 encryption /decryption Operation Stage. The first stage included the creation of an initial matrix with size $[16 \times 16]$ with a random selection of elements from the initial matrix. The second stage included creating input to matrix called Chacha x-matrix With the implementation of the so-called quarter-round that represents the mathematical model of the algorithm. Finally, The third stage included the IChacha20 encryption /decryption operation. National Institute of Standards and Technology (NIST) Package tests prove that the keys (which are generated by the IChacha20 algorithm) are random and unpredictable, so they are robustness against the attacks. IChacha20 keystream is passed most of the NIST tests with high success rates.

The proposed IChacha20 algorithm was applied to multi-media, including text, We used the correlation coefficient metric, the best results have been obtained. Also, the results showed values of metrics ciphering images using the proposed algorithm and they were better than the values obtained using the original algorithm . As well as, the audio signal, the results had provided values of metrics that evaluating the ciphering of audio samples by using IChacha20,

where showed the results best values for the metrics both of correlation coefficient and signal to noise ratio.

The proposed IChacha20 has obtained faster execution time than the original Chacha20, where the execution time of the original Chacha20 algorithm was= 02:07:1sec while the execution time of the proposed algorithm was =01:26:4 sec.

List of Contents

	Chapter One: Introduction	1-9
1.1	Overview	1
1.2	Related Work	3
1.3	Problem Statement	7
1.4	Aim of The Thesis	8
1.5	Contribution	8
1.6	Thesis Outline	8
	Chapter Two: Theoretical Background	10-30
2.1	Introduction	10
2.2	The ChaCha20 Algorithm	10
2.3	Chaotic Maps	13
2.3.1	Chebyshev maps function (1-Dimensional)	14
2.3.2	Tent Maps function (1-Dimensional)	14
2.4	Web Service	15
2.4.1	WSA Functional Ingredients	17
2.4.2	Web Service Security	18
2.4.3	Basic WS Technologies	18
2.4.3.1	Web Services Description Language (WSDL)	19
2.4.3.2	Universal Description, Discovery and Integration (UDDI)	20
2.4.3.3	Simple Objects Access Protocol (SOAP)	20
2.5	The Platform Of Android And A Security Of Its	22
2.5.1	The Architecture of Android System	22
2.5.2	Android Apps Structure	24
2.5.3	Android Security Model	25
2.5.4	Trends In Android Security Research	25
2.6	Randomness Tests	26
2.7	Correlation Coefficient	28
2.8	Mean Square Error (MSE)	28
2.9	Peak Signal to Noise Ratio (PSNR)	29
2.10	Universal Quality Index (UQI)	29
2.11	Normalized Cross-Correlation (NCC)	30
2.12	Signal to Noise Ratio (SNR)	30
	Chapter Three: The Proposed System	31-50
3.1	Introduction	31
3.2	Design Objectives	31
3.3	The Primitive Proposed Model	32

3.4	The Proposed Improvement Cahcha20 Algorithm (ICahcha20)	34
3.4.1	Chaotic Stage	35
3.4.2	Chaotic Chacha Key Stream Generation Stage	41
3.4.3	ICHacha20 Encryption /Decryption Operation Stage	48
	Chapter Four: Results and Analysis	51-66
4.1	Introduction	51
4.2	Initialization	51
4.3	The Implementation Of The Proposed System	51
4.4	Results Of the Proposed System (ICHacha20 Algorithm)	54
4.4.1	Results Of Chaotic Stage	54
4.4.2	Results Of Chaotic Chacha Stream Key Generation Stage	55
4.4.3	Results of Encryption Data Using Proposed IChacha20 Algorithm	57
4.5	Analysis	65
	Chapter Five :Conclusions and Suggestions for Future Work	67-69
5.1	Conclusions	67
5.2	Suggestions For Future Work	68
	References	

List of Figures

2.1	Block diagram of Chacha20	11
2.2	Chacha20 matrix	12
2.3	The Components of The Chacha20 Matrix	12
2.4	Graph of Tent map Function	15
2.5	The three Thoughts Roles and Operations of Web Services	17
2.6	Relations Between Technologies of Web Service	19
2.7	The Format of The Message Elementary Layout	21
2.8	Web Services Communication Stack	21
2.9	Android System Architecture	24
2.10	The Structure of Android Applications	25
2.11	Taxonomy of Literature on Security of Android	26
3.1	Primitive Model of The Proposed System	32
3.2	Exchange Data Between Sender and Recipient using Proposed System	33
3.3	General Block Diagram of The Proposed IChacha20	34
3.4	Initial Matrix [16*16]	37
3.5	Block Diagram of Select Value Step	38
3.6	Selected Value From Initial Matrix	41
3.7	Block Diagram of Create Chacha matrix	42
3.8	256-Bits Key	45
3.9	128-Bits Sigma	46
3.10	96-Bits Nonce	46
3.11	Flowchart of Chacha20 quarter round	48
3.12	Block Diagram of IChacha20 Algorithm	49
4.1	The Main Interface of Receiver Side	52
4.2	The Decryption Interface of The Application	53
4.3	Behaviors of Chaotic Function ; a) Tent Chaotic Function, b) Chebyshev Chaotic Function	54
4.4	Result of Selection Location from Initial Matrix [16*16] based on values of 1d Tent and Chebyshev Chaotic Functions.	55
4.5	Practical Application of Proposed Encryption IChacha20 Algorithm using Text Sample (#1)	59

4.6	Histogram of Original and Cipher Audio Signal; a) Audio Signal #1, b)Audio Signal #2.	64
-----	--	----

List of Tables

4.1	Results of key Setup 256-bit Key,128-bit Sigma, and 96-bit Nonce.	56
4.2	Tests Plaintext Samples	57
4.3	The Tests Image Samples	60
4.4	Histogram of Original Test Images Their and Corresponding Cipher Images using Original Chacha20 &Proposed IChacha20	61
4.5	Results of MSE, PSNR, NCC, and UQI Metrics	62
4.6	Execution Time of All Images by using The Original Algorithm and The Proposed IChacha20 Algorithm	63
4.7	Audio Samples	64
4.8	The statistical Tests of NIST on the keys of the proposed IChacha20 algorithm	66

Abbreviations

1D	One-dimensional
AES	Advanced Encryption Standard
ICHacha20	Improvement Chacha20
IV	Initial Vector
log	Logarithm
MSE	Mean Square Error ¹
n	degree of Chebyshev polynomial
NCC	Normalized Cross-Correlation
NIST	National Institute of Standards and Technology
PNB	Probabilistic Neutral Bits
PSNR	Peak Signal to Noise Ratio
Q.R	Quarter-Round
SNR	Signal to Noise Ratio
SOAP	Simple Object Access Protocol
TLS	Transport Layer Security
UDDI	Universal Description Discovery and Integration
UQI	Universal Quality Index
W.S	Web Service
WSA	Web Services architecture
WSDL	Web Services Description Language
X0	initial value of Chebyshev function
XML	Extensible Markup Language

List of Algorithms

Algorithm (3.1):	Create Initial Matrix using Tent and Chebyshev Function	35
Algorithm (3.2):	Generating Rang [0-1]	39
Algorithm (3.3):	Select Value from Initial Matrix.	40
Algorithm (3.4):	Compute of Chacha20 (256-bit Key, 96-bit Nonce, and 128-bit Block Sigma)	43
Algorithm (3.5):	IChacha20 Quart Round	47
Algorithm (3.6):	The IChacha20 Encryption Algorithm.	50

Chapter One

Introduction

1.1 Overview

Network security is a general term that incorporates a plurality of technologies, devices (hardware), and processes (software). In its simplest term, it is a collection of rules and components engineered to protect the integrity, accessibility as well as the confidentiality of communications systems and information using both software and hardware technology. Each institution, regardless of size, infrastructure or industry, required a degree of security of the network solutions to protect it from the ever-growing digital threat landscape that exists today [1]

The rapid development of modern web technology and information technology causes individuals, businesses, and government departments to join the Internet, which creates more illegitimate users to attack and devastate the network by imitating internet sites, Trojan horses, fake mail, and backdoor viruses at the same time. The aim of the intruding and attacks on the network are computer devices, therefore, as soon as the intruders succeed, they will cause thousands of computers on the network in a crippled state. Besides, some intruders with ulterior motives perceive the government and the military as the goal that poses enormous threats to national and social security [2].

Cryptography is used for information safety of digital reality for today. It means hidden secrets are concerned with encryption. Cryptography is the science of hiding information. More broadly, it is about designing and analyzing protocols that obstruct the enemy[3]. Various aspects of information security, such as data

integrity, confidentiality, authentication, and non-repudiation, are central to modern cryptography [4].

The Cryptography system can be divided into two parts: first, is symmetric-key cryptography and the second is public-key cryptography. symmetric-key cryptography: In a symmetric-key cryptography system dispatcher and recipient share one key which is utilized to encrypt and decrypt a message. It is also named secret-key cryptography. The algorithms used for symmetric key cryptography are named symmetric-key algorithms. There are two kinds of symmetrical algorithms, like block cipher & stream cipher. Stream ciphers encrypt bits of information one at a time but block ciphers encrypt information by fractioned it into blocks [5].

Algorithms in Cryptographic take an important role in providing the data security against malignant attacks. The competence of the cryptographic algorithm does not only rely on its time taken for encryption and decryption, and it also accounts for the number of the stages utilized to obtain the cipher-text from an original-text[1]. Chacha20 algorithm is one of the common encryption algorithms. Chacha20 is a high-speed stream cipher based on the Salsa20 cipher designed by Daniel J. Bernstein. It aims to improve performance by increasing the diffusion in each round [6]. Although it is a powerful and modern algorithm, it is also may be threatened in penetration, therefore in this thesis, chaos functions were used to generate a random key used with the rounds to increase its strength to IChacha20 based on chaotic functions (Chebyshev and Tent maps).

1.2 Related Works

Many researchers and mathematicians have suggested many works about the Improved Chacha20 algorithm. The following are some studies and discussions that can so far associate works to the suggested work in this thesis:

1. **Daniel J. Bernstein in (2008) [7]:** In this article, the author introduced the Chacha family of stream ciphers, a separate version of the Salsa20 family. Chacha implements the same general style principles as Salsa 20 but has modified some of the details. More specifically, there is an improved amount of diffusion each round. He is conceived that the minimal number of secure rounds for Chacha is smaller (and not larger!) than the minimal number of secure rounds for Salsa20. So presents the Chacha family and describes the variation between Salsa 20 and Chacha. The results of the comparison of the Chacha 20 algorithm with the Salsa algorithm showed the following: the same speed on a 64-bit amd64-architecture Core 2 (6f6), the same speed on a 64-bit Sparc-architecture Ultra SPARC IV, 5% faster on a 64-bit amd64-architecture Athlon 64 (15,75,2), 5% faster on a 32-bit x86-architecture Pentium D (f47), 5% faster on a 64-bit ppc64-architecture PowerPC G5 (750), 6% faster on a 32-bit ppc32-architecture PowerPC G4 (7410), 8% faster on a 32-bit x86-architecture Pentium M (695), and 28% faster on a 64-bit amd64-architecture Pentium D (f64).
2. **S.Maitra in (2016) [8]:** In this paper, he revisits the work of Aumasson et.al, whose analyzed the Salsa and Chacha 20 code, to provide a clearer insight of the existing attack (2^{248} complexity for ChaCha7, i.e., 7 rounds) and show certain improvements (complexity around 2^{243}) by exploiting additional Probabilistic Neutral Bits(PNB). More importantly, he described a novel idea that explores the proper choice of the initial vector (IVs) corresponding to the

keys, for which the complexity can be improved further (2^{239}). The choice of IVs corresponding to the keys is the prime observation of this work. he systematically shows how a single difference propagates after one round and how the differences can be reduced with proper choices of IVs. For Salsa too (Salsa20/8, i.e., 8 rounds), he got an improvement in complexity, reducing it to ($2^{245.5}$) from ($2^{247.2}$) reported by Aumasson et.al.

3. **Ganesan et.al** (2016)[9]: In this paper, they 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 the MCC core to generate a collision-resistant function of compression for the encoded hash algorithm.
4. **A. R. Choudhuri & S. Maitra in (2016)** [10]: In this paper, they considered the biases in the forward rounds and estimate an upper bound on the number of rounds till such biases can be observed. For this, they proposed a hybrid model (under certain assumptions), where initially the nonlinear rounds as proposed by the designer are considered, and then they employed their linearized counterpart. The effect of reverting the rounds with the idea of Probabilistic Neutral Bits(PNB)are also considered. Based on the assumptions and analysis, they concluded that (12) rounds of Salsa and ChaCha should be considered

sufficient for 256-bit keys under the current best-known attack models, and they recommended that this model may have potential applications in other ARX based ciphers.

5. **S.Dey and S.Sarkar in (2017)[11]**: They gave a new algorithm to design PNBs "Probabilistic Neutral Bits". They have been using this algorithm to enhance existing attacks to decrease both SalSa and ChaCha rounds. These assaults on SalSa and ChaCha are consecutively around 2.27 and 5.39 times Quickly than the existing works of Maitra and Choudhuri (justifiable in FSE 2017), Where Maitra has enhanced the attack to complication $2^{238.9}$ selecting the (IVs) properly to achieve better outcomes, While Choudhuri et.al proposed to use multi-bit outputs rather than single-bit output, They enhanced the complexity to $2^{237.6}$.
6. **P. A. Babu and J. Thomas in (2018)[12]**: In this paper, they introduced Freestyle, a randomized, and variable round version of the ChaCha cipher. Freestyle demonstrated the concept of hash-based halting conditions, where a decryption attempt with an incorrect key is likely to take a longer time to halt. This makes it resistant to key-guessing attacks i.e. brute-force and dictionary-based attacks. Freestyle used a novel approach for ciphertext randomization by using a random number of rounds for each block of message, where the exact number of rounds is unknown to the receiver in advance. Due to its inherent random behavior, Freestyle provided the possibility of generating up to(2^{256}) different ciphertext for a given key, nonce, and message; thus resisting key and nonce reuse attacks.
7. **A. Miyaji and Y. Matsuoka in (2018) [13]**: They described the existing security analysis, firstly, they described the stream ciphers Chacha and Salsa, then, the existing security analysis because Chacha needs more analysis of

security because it has been suggested more newly so they suggested compared with the AES algorithm. Moreover, Chacha is an enhancement of Salsa from the diffusion and analysis of the diffusion of Chacha and Salsa. It is important to understand the design of security criteria. In this study, the diffusion analysis is reviewed and weak bits and weak columns of Chacha and Salsa are investigated. To the knowledge of the authors, so they considered, this is the first thorough explanation of the diffusion of Salsa and Chacha.

8. **P. McLaren et.al in (2019)[14]:** In this paper, they explained identifies a significant vulnerability within OpenSSH and OpenSSL and which involves the discovery of cryptographic artifact used within the ChaCha20 cipher. This can allow for the cracking of tunneled data using a single targeted memory extraction. With this, law enforcement agencies and/or malicious agents could use the vulnerability to make copies of the encryption keys used for each tunneled connection. The user of a virtual machine would not be alerted to the capturing of the encryption key, as the method runs from an extraction of the running memory. Methods of mitigation include making cryptographic artifacts difficult to discover and limiting memory access.
9. **S. Dey et.al in (2019)[15]:** In this paper, they revisit the existing attacks on Chacha and Salsa ciphers. Firstly, they applied an accurate computation of the attacks complexities of the existing technique instead of the estimation used in previous works. This improves the complexity of something. The differential attacks utilize PNB'S against Salsa and Chacha involve two probability biases: forward probability bias (ϵ_d) and backward probability bias (ϵ_a). In the second part of this paper, a way to growing the backward probability bias is suggested, which helps to decrease the complexity of the attack. Lastly, they concentrate on the principles of the design of Chacha. They suggested a slight medication in the design of this cipher as a countermeasure attack

against differential. They offered that the key recovery attacks suggested against Chacha will not be strong on this improved version.

10.S. Dey and S.Sarkar in (2020)[16]: In this research, they presented a total theoretical confirmation of both the SalSa and ChaCha differentiators observed. The notion of a probabilistically neutral bit often takes on a crucial role in the main recovery attack. Here, too, they theoretically show the reason for a special key bit of Salsa to be neutral. So this is the first attempt to give a theoretical justification for the recovery of differential key attacks versus these ciphers. And also, this work gives a fascinating insight into the notion of a differential assault against SalSa and ChaCha. Such research finds an accurate explanation for the noticed forward biases for both ciphers, and that was the fundamental tool for each distinguishing differentials as well as attacks of key recovery for decades. The researchers' goal has forever been to grow the noticed bias and, extend it to the next round. This abstract description can assist in getting a good vision from the beginning of propagation of the bias, which will help to find a better distinguisher through some effective tool. This study has also established the theoretic basis for the Probabilistic Neutral Bits(PNB). However, this principle can also help to boost the backward bias by applying some suitable techniques.

1.3 Problem Statement:

In the last few years attempts to crack the Chacha 20 rounds were attempted by attackers by directing several types of attacks such as key recovery attacks, differential attacks, collision attacks, and others. The attackers managed to break rounds of the ChaCha 20 algorithm arrived 7 and more rounds. The first problem not to allow attackers to break any round of the Chacha20 algorithm. The second

problem must be generated a key that has enough randomness cannot be broken, to add power the algorithm and the attackers cannot access in easily.

1.4 Aims of The Thesis

The aims of this thesis are :

1. Improvement Chacha20 algorithm based on chaotic maps functions.
2. Increasing the keyspace to expand the probabilities of the key to be difficult to guess, break, or access.
3. transfers secure data (text, images, and audio) in mobile communication.

1.5 Contribution

The main contribution of this thesis is to add a layer of security to the Chacha algorithm based on the Chaotic maps (Chebyshev and Tent map function), thus this new contribution will provide communication channels for the transmission of data such (text and images & audio).

1.6 Thesis Outlines

The remaining chapters are:

Chapter two which is entitled theoretical background: presents An introduction to the importance of security over networks and mobile phone communications, the Chacha 20 algorithm and its details, and its mathematical model with Chaotic map, Web service, android platform, and some metrics for text, image, and audio.

Chapter three which is entitled The Proposed System: presents the main proposed system, design objectives, and covers the steps of the proposed IChacha20 algorithm using a multi-level of a chaotic map with several algorithms that represent these steps.

Chapter four which is entitled The Results: This presents the results and tests of the proposed system.

Chapter five which is entitled Conclusions, and Suggestions for Future Work: presents the conclusions for the proposed systems, and suggestions for future work.

Chapter Two

Theoretical Background

2.1 Introduction

With the coming of the World Wide Web and the development of applications of e-commerce and social networks, the organization's across the world generate a large amount of data every day. Security of network, Information, and communication is the most extreme basic problems in guaranteeing secure transmission of data by the web. Also, the network security issue is now becoming important as society is moving toward the digital information age. As more and more users' connect to the internet it attracts a lot of attacks of cyber. Therefore, security data, information, and networks have become essential issues to provide the highest level of protection for users' privacy over the Internet[17].

This chapter prepares the theoretical background for this work, will introduce the cryptographic algorithm that is used in the proposed system, and make use of in the security of information and communication.

2.2 The Chacha20 Algorithm

The encryption algorithm Chacha20 is a lightweight algorithm that belongs to the Salsa algorithm family but differs in detail, engineered by the researcher D. J. Bernstein. Chacha20 provided the best security than the original Salsa20 cipher, by using somewhat better hash functions. The hash function input data rearranged to allow to implement the algorithm more effectively [7].

Chacha20 It is used in cryptography instead of classic algorithms to transmit data and information over the network and seeing a rapid adoption due to its software-friendly design and natural resistance to timing side-channel attacks.

Chacha20 cipher algorithm generates key-stream of 64-Bytes. Inputs to keystream generation are independent of original text or ciphertext. Chacha20 algorithm is used widely as an alternative to the AES encryption algorithm. The 20 round of stream cipher Chacha20 is systematically quicker and not sensitive to timing attacks as the AES algorithm. Figure (2.1) shows the general diagram of the Chacha 20 algorithm[18].

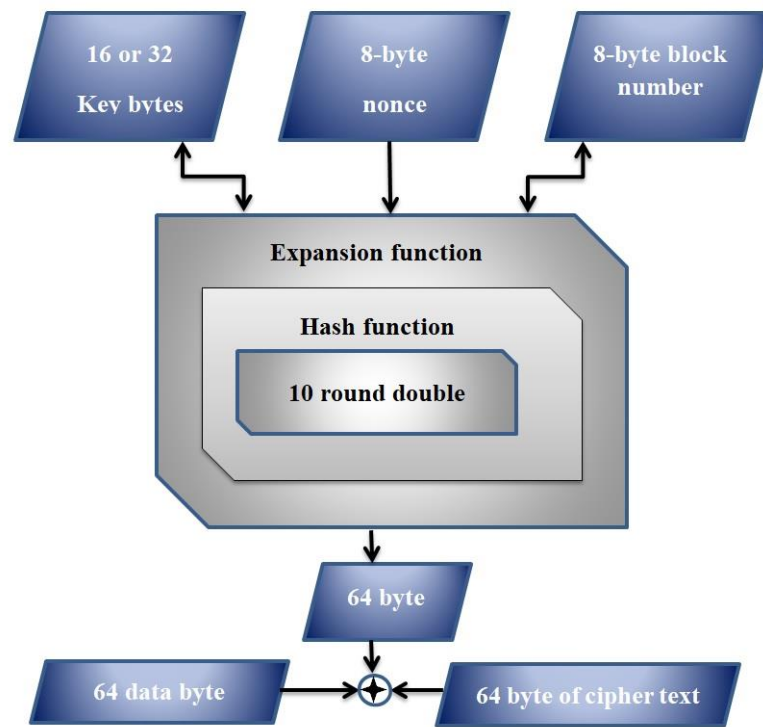


Figure (2.1) : Block diagram of Chacha20 [18]

20 rounds of calculations of mathematical of Chacha utilize XOR, addition & rotation utilize as input four 4-byte constants, a random 32-byte key, a 4-byte counter, and a 12-byte nonce ("Bernstein originally specific counter and the nonce lengths to be eight values"). Chacha20 gather a 256-bit key & a 32-bit nonce (which contain a counter). This leads to creating a keystream which is then XOR-ed with the stream of plaintext. Chacha20 operates on 32-bit words at a time with a key of 256-bits $K = (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$. These blocks are of outputs of 512-bits for the keystream (Z), and which is XOR-ed with a

stream of plaintext. The state of encryption is" stored within 16*32-bit word values and arranged as a 4*4 matrix . Figure (2.2) shows the Chacha 20 matrix[14].

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

Figure (2.2) Chacha20 matrix[19]

primary array of Chacha20 algorithm include 16x32-bit values with constant ("0x61707865", "0x3320646e", "0x79622d32", "0x6b206574") the key(k) =(k₀,k₁, k₂, k₃, k₄, k₅, k₆, k₇), the counter (c₀) & nonce ("n₀, n₁, n₂,n₃"). Figure (2.3) shows the components of the Chacha 20 matrix

$$\begin{bmatrix} 0x61707865 & 0x3320646e & 0x79622d32 & 0x6b206574 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ c_0 & n_0 & n_1 & n_2 \end{bmatrix}$$

Figure (2.3) The components of the Chacha20 matrix[14]

The counter has 32-bits (1 * 32-bits), & nonce has 96-bits (3 * 32-bits).

In Chacha20 algorithm, the nonlinear round function is called The quarter-round. Where Chacha20 is doing by scrambling the hill out of 512-bit blocks, using 80 applications of the quarter-round. The numbered equations (2.1,2.2,2.3 and 2.4) together represent the mathematical model of Chacha 20 algorithm which is called quarter-round (Q.R):

$$a = a + b, d = ((d \oplus a) \lll 16) \dots \dots \dots (2.1)$$

$$c = c + d, b = ((b \oplus c) \lll 12) \dots \dots \dots (2.2)$$

$$a = a + b; d = ((d \oplus a) \lll 8) \dots \dots \dots (2.3)$$

$$c = c + d, b = ((b \oplus c) \lll 7) \dots \dots \dots (2.4)$$

a, b, c and d each one of them represents a single byte of a total of four bytes, which is part of the value of a single location of a matrix Chacha 20.

The nonlinear function application method is not the same in each round. Chacha20 has the function applied along diagonals and columns. The sequence along the column is :

{ "(X0, X4 , X8, X12) , (X1 , X5 , X9; X13), (X2, X6 , X10 ; X14) and (X3, X7; X11 ; X15)" }.

Also along the diagonal, the order is :

{ "(X0, X5, X10; X15), (X1, X6; X11, X12), (X2, X7, X8; X13) and (X3, X4; X9, X14)" } [14] .

2.3 Chaotic Maps

The chaotic sequences have various useful features of application based on security. These features are: - (1) the chaotic is a dynamic system in discrete time to generate complicated sequence which behaves randomly easily and simply. (2) the chaotic signal is not random but it is deterministic, this feature lets us renewal it. (3) The chaotic signal has a high sensitivity of initial condition this leads to any change in the initial condition create another sequence [19]. This feature makes the chaotic sequence very difficult to predict by attackers to renewal it and increase the security level. (4) the chaotic sequence path has random behavior in the specific space, this causes the restoration of this sequence is impossible in its specific space. Chaotic maps are separated into two classes, 1d (one-dimensional) and multidimensional maps[20]:

2.3.1 Chebyshev Maps Function (1-Dimensional)

Chebyshev is one of the most commonly used security mechanisms in authentication methods because it contains a semi-group property. The Chebyshev polynomial is presented in three definitions of as following equation [21]:

Definition 1: The c Chebyshev polynomial in degree n is determined as:

$$T_n(x) = \cos(n \cdot \arccos(x)) \quad \dots (2.5)$$

where (n) is integer digit, $x \in [-1, 1]$

Definition 2: Semi-group properties for Chebyshev maps can be accomplished as:

$$T_r(T_s(x)) = T_s(T_r(x)) \quad \dots (2.6)$$

Definition 3: The equation of Chebyshev in n degree, present :

$$(x, T_n(x)) \quad \dots (2.7)$$

It is infeasible in computation to determine the polynomial order n.

2.3.2 Tent Maps Function (1-Dimensional)

Yoshida et.al [22] studied chaotic behaviors of the Tent map (a piecewise linear, continuous map with a unique maximum) analytically throughout its chaotic region in terms of the invariant density and the power spectrum. As the height of the maximum is lowered, successive band-splitting transitions occur in the chaotic region and accumulate to the transition point into the nonchaotic region. The time-correlation function of nonperiodic orbits and their power spectrum is calculated exactly at the band-splitting points and in the vicinity of these points. The tent map is topologically conjugate, and thus the behaviors of the map are in this sense identical under iteration. Figure (2.4) shows the chaotic tent maps:

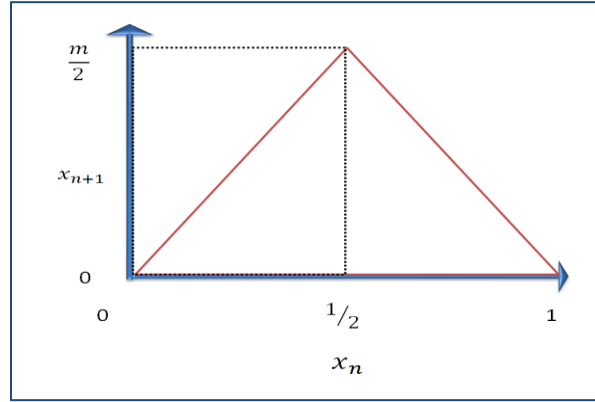


Figure (2.4) Graph of Tent map function[22]

Where (x-axis: Parameter space $\in (0,1)$, y-axis: Chaotic range $\in (0, 1)$)

The chaotic Tent map is described by equation (2.7) :

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

$$F(x_i, \mu) = \begin{cases} FL(x_i, \mu) = \mu x_i, & \text{if } x_i < 0.5 \\ FR(x_i, \mu) = \mu(1 - x_i), & \text{otherwise;} \end{cases} \quad \dots(2.7)$$

Where (x_i) belonged $[0,1]$ for $i \geq 0$. These maps transform a period $[0, 1]$ onto itself and contain only one control coefficient (μ) , Consecutively, where (μ) belonged $[0, 2]$. X_0 is the beginning system value. The collection of real values $(X_0, x_1, \dots, x_n, \dots)$ is named system orbit. By each x_0 , such an orbit is acquired depending on the control coefficient, the figure (2.3) display a range of dynamic behaviors spread from expected to chaotic[22].

2.4 Web Service

A group of operations which are network available over standardized XML messaging that is explained by the Web Service (WS). The description of web service by standard, the notion of formal XML, named its description of service. It includes all the parts essential to interrelate with the service, comprising formats of the message that detail the operations, protocols of the transport, and position[23].

Program-to-program communications are composed of WS. The charge of implementing e-business is reduced by permission of the WS, and also arranges solutions quicker and to exposed new chances. The key to reaching this new horizon is a common program-to-program communications model, built on existing and emerging standards such as HTTP, Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description Discovery and Integration (UDDI) [23].

Among the more important characteristics of WS is integration to established up the environment of the web because it is predicated on a united "Extensible Markup Language (XML)". Will provide this WS syntax with acceptable interface mechanisms to present information for elasticity as well as compatibility. This implies that if the client is continuing to work on a number of the old, sophisticated and new system on various operating systems and in multiple languages of programming, they have to be incorporated [24]. One of the advantages of Web Services is offering a model of combined programming for the services of private Intranet and public Internet growth and usage as a consequence, as a result, the choice of network technology will be transparent to the developer of the service [24]. The architecture of the web services is depending on the exchanges among three parts: provider of the service, registry of the service, and service requestor. The exchanges include the issue, determine, and merge operations. Collected, these parts and operations performed on the artifacts of the web services. Figure (2.5) illustrates these operations, the mechanisms offering them, and their interactions [24].

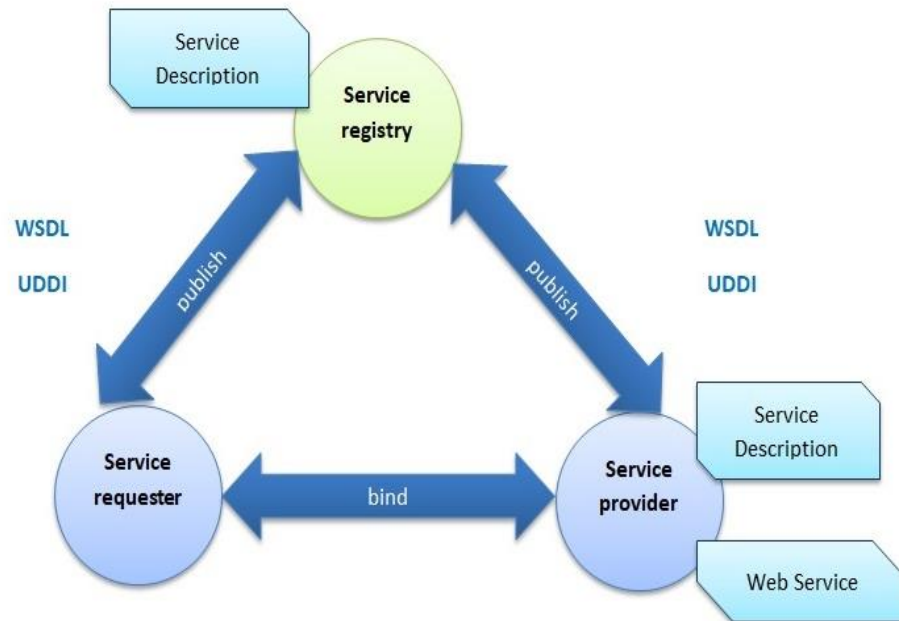


Figure (2.5) The three thoughts roles and operations of web services [25]

2.4.1 WSA Functional Ingredients

The three operations of conceptual WS to complete the roles, WSA (Web Services architecture) must provide three basic functional components of its architecture [26]

1. **Transportation:** used to contact a particular service that includes both protocols as well as format that is part of the transport component. Data features are classified in the message data format. Message transfer control within the semantics of the application is specified in the protocol of the transfer. The transport protocol advertises the transfer of the current message.
2. **Description:** The description of the service throughout the languages of the programming utilized shall be prepared via the ingredients of the characterization. Where even the binding data, the exchange parameters, and the format of the message with the service are available within these ingredients.

3. **Discovery:** The methods for registration of the services or adverts for services or explanations of services are defined within this element.

2.4.2 Web Service Security

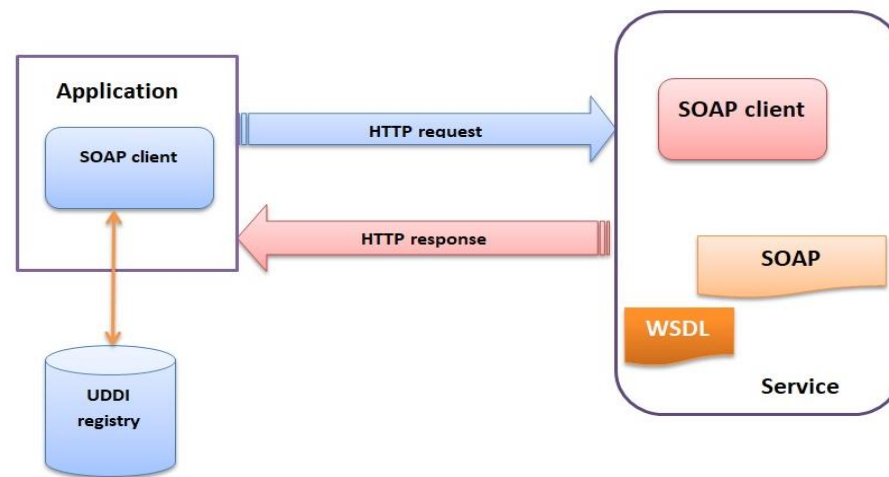
The security layer of the web service is to be delivered in four key primary security needs [24]:

1. **Confidentiality:** The property that information is not made available or disclosed to unauthorized individuals, entities, or processes, and guarantees that the contents of the message are not disclosed to unauthorized individuals.
2. **Authorization:** The characteristic that make sure of the authority, that comprises the permitting of access depending on access rights and promises that the sender will have the authorization for sending a message.
3. **Data Integrity:** The property that data has not been undetectably altered or destroyed in an unauthorized manner or by unauthorized users. Thus make sure that there is no modification on the message unintentionally or purposely in transit.
4. **Proof of Origin:** is evidence identifying the originator of a message or data. It gives emphasizes that the transition of the message was done correctly by the recognized sender and is not a replay of a previously transmitted message. This requirement implies data integrity.

2.4.3 The Basic WS Technologies

There is several technologies have been presented beneath the web service title and additional ones will be presented in the coming years. The paradigm of the web service has grown-up very fast that numerous

technologies of competing are trying to offer the same ability, Figure (2.6) offers a diagram that proves the relationship among these technologies [26].



Figure(2.6) Relationship between technologies of web service[27]

The relationship between these pieces (SOAP, UDDI, and WSDL) can be defined as follows: an application acting in the role of a web services client needs to locate another application or a piece of business logic located somewhere on the network. The client queries a UDDI registry for the service either by name, category, identifier, or specification supported. Once located, the client obtains information about the location of a WSDL document from the UDDI registry. The WSDL document contains information about how to contact the web service and the format of request messages in XML schema. The client creates a SOAP message by the XML schema found in the WSDL and sends a request to the host (where is the service) [26].

2.4.3.1 Web Services Description Language (WSDL)

The technology of XML is WSDL which defines the interface of web services in an identical method. WSDL regulates how the input/output parameters are represented by a web service of an invocation visibly,

the structure of the function, the invocation nature, and the protocol service tie. WSDL permits unlike clients to robotically know how to interrelate to the web-service [26]

2.4.3.2 Universal Description, Discovery and Integration (UDDI)

UDDI presents the web services worldwide registry for detection, announcement, and purpose of integration. Analysts of the business and scientists utilize UDDI to find ready web service by examining titles, identifiers, classes, or the conditions executed through the web services. UDDI present a construction for demonstrating businesses, relationships of business , data set,web services with specification as well as web service admission plugs [26].

2.4.3.3 Simple Objects Access Protocol (SOAP)

SOAP is a protocol used by Web services to construct and understand the messages they exchange. SOAP is at the heart of Web services architecture in that it allows the interacting parties in the architecture to communicate with each other using a standard, well-understood message format. The SOAP specification and development are preserved by the commendation of W3C 27 April 2007. The specification defines a format of XML-based standard message, labeling how the metadata of the message and payload should be packed into a document of XML [28].

The format of the message elementary layout describes in Figure (2.7) SOAP Envelope signals the beginning of the message of the SOAP. every message involves of SOAP sections of the header and body. The body section involved in the payload. The additional details of the processing instruction, like the protocol

of the transaction or policies of the security, go into the header section of the message [28].

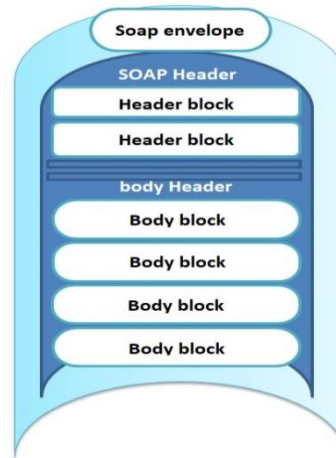


Figure (2.7) The format of the message elementary layout [28]

As shown in the Stack Communication of the Web Service in figure (2.8) of the SOAP messages, any application-level transmission protocols such as SMTP or HTTP is communicated through the Internet. The word "SOAP binding" is meant to refer to the transfer technique through that the SOAP has been communicated. For instance, when the SOAP is linked to the HTTP, the messages of the SOAP are entrenched at the HTTP request segment of a body of the as well as response [28].

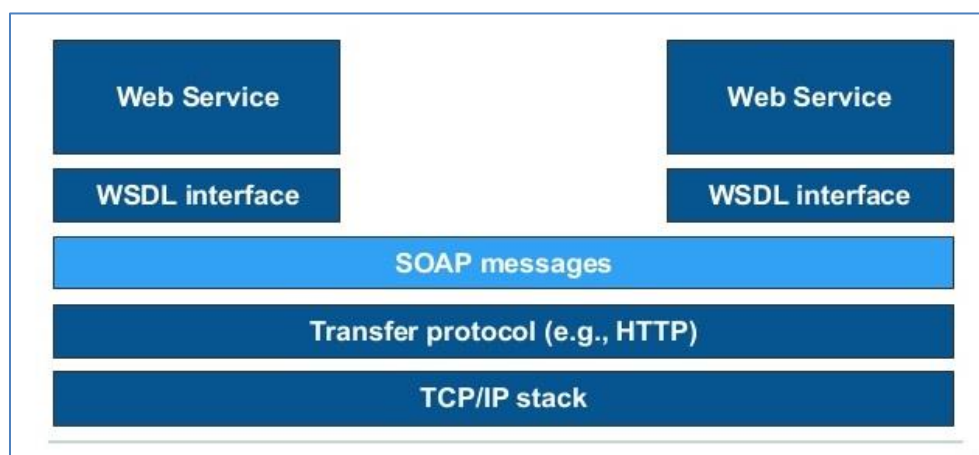


Figure (2.8) Web services communication stack [28]

Web services of SOAP deliver a typical and very active communication method for the web and the significant choice for initiative applications of the web to interconnect with further systems or administrations in an additional elastic method. Extensible communication Technology is easier to use by SOAP via the web. Tracks standards of technology, from architecture to communication security. SOAP security is not only at the level of transport. However, likewise at the application level, for instance, the messages are been transformed only to the requesting application. SOAP is a technology of flexible online communication and has been flexible it improves effectiveness and can guarantee a more effective and serviceable architecture [29].

2.5 The Platform of Android and the Security of Its

The android platform involves some layers and components and characteristics[30]:

2.5.1 The Architecture of the Android System

Android is a Linux-based operation system, originally designed for smartphones. The mobile operating system android is a set of applications of software. Android system is composed of a kernel of Linux, a layer of the middleware, as well as a layer of an application. Google personalized the fundamental internals of Linux that provides powerful isolation among various processes and structure the complete system on the adapted kernel of the Linux. This kernel needs to serve as a layer of abstraction among equipment and other layers of software [30].

The Linux kernel also supplies the usual basic and primary equipment, such as device drivers, management of memory, scheduling of process, and files system. The layer of the middleware of Android is located at the top of the Linux kernel. This layer consists of three major components: the framework of the application,

the environment of the android runtime, and the native software libraries. The framework of the application is typing in the language fo Java as well as is a set of service that determines the environments wherein Android applications are utilized and managed [31].

In the platform of android, system suppliers of content are fundamental datasets, whilst system services supply the necessary high-level functions to monitoring the devices and knowledge on the status of the platform, like area and status of the network. The other middle-ware part of them is a native collection of libraries that supply functionality such as multi-media support and processing of graphics. These forms of libraries are published in the C / C languages [31].

The last part of the layer of a middle-ware of Android was its runtime environment, which requires the java programming libraries as well as the Dalvik Virtual Machine. This layer has been edited by C/C language, excepting sections of root libraries, it is tailored to the specified requirements of limited mobile resources devices [31].

As explained in figure (2.9), the application layer Android is falling to the top of architecture in Android. This involves pre-configured application (i.e. original Android applications) and these part apps created through various (non-official) app designers. Applications are implemented by java, but for execution purposes, the causes might have indigenous code (C / C) which is termed the "Java Native Interface (JNI)". Essentially, Android is a system of multi-user with a unique user ID (UID) with each application. The files in the applications are referred to as UID apps, they are generally cannot access to other apps. Every other application works with its reign process of Linux with a singular UNIX user ID and is separated from other applications because applications can expressly share resources as well as information. With this technique, The android

operating system performs the "least privilege" concept. Anyway, Android application is made of a set of ingredients: interface, Service or background process, broadcast receiver, or mailbox for data transmission and provider content(SQL-like database system) [31].

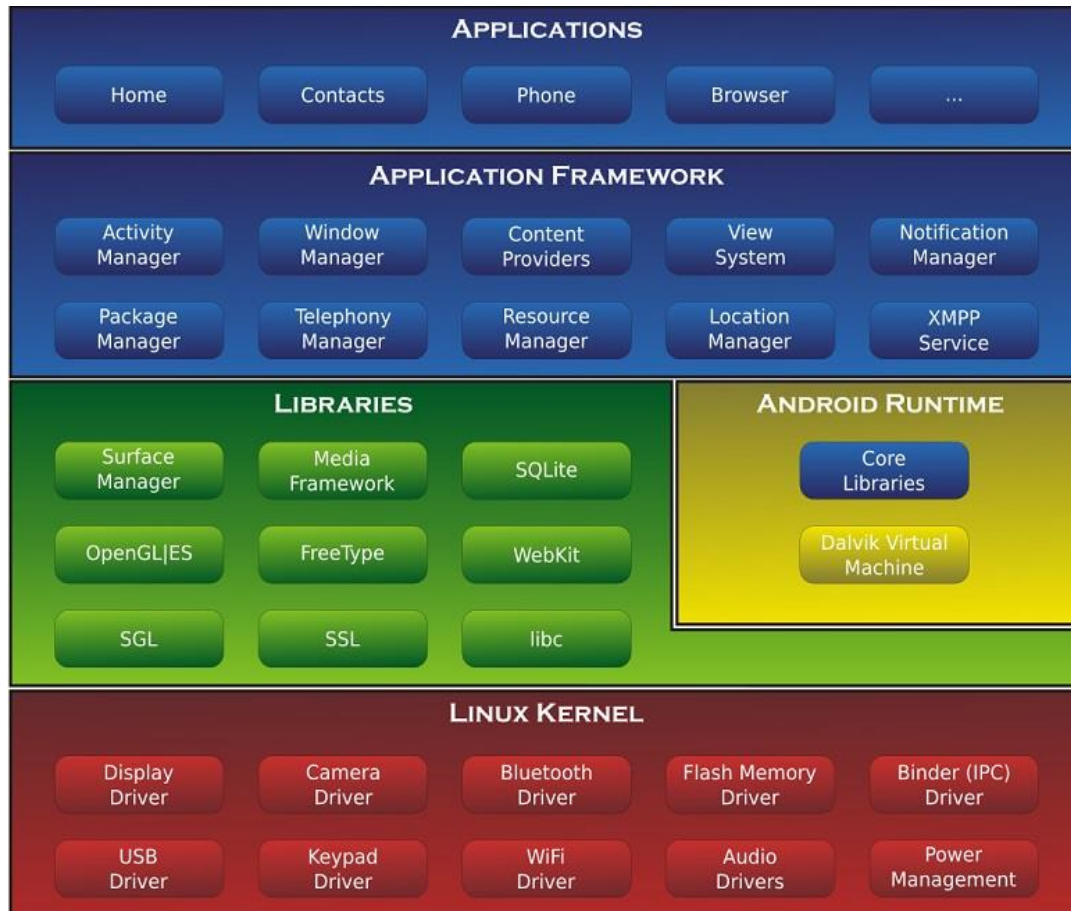


Figure (2.9): Android system architecture[31]

Android Apps Structure

Android is a software package designed for work on mobile platforms, like tablets or smartphones because every Android application usually involves four major ingredients: services, interfaces, content providers, and receivers of the broadcast. communications among applications take place via the layer of middle-ware, shows the various kinds

of inter-process communications (IPC). Figure (2.10) illustrate Android system Architecture [32].

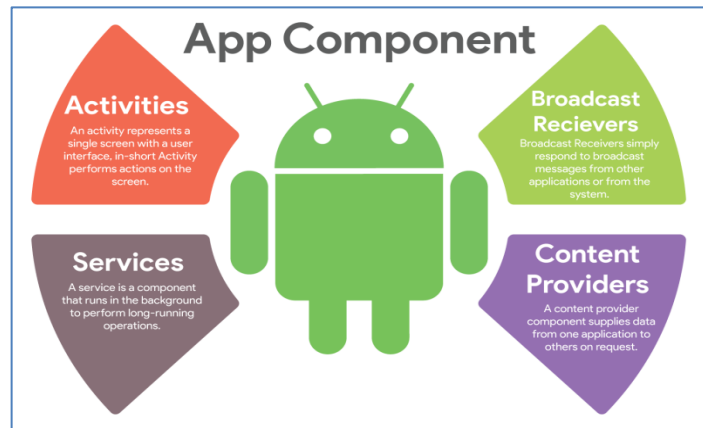


Figure (2.10): The structure of android applications [32]

2.5.2 Android Security Model [33].

Mechanisms of security that the Android platform uses to secure any the environment of the application are mention here. Android implements several security mechanisms, the most prominent of which are:

- Android Permission System.
- Sand-boxing of application
- signing of application
- Secure inter-process communication
- SELinux

2.5.3 Trends In Android Security Research

Recently, several security modifications for Android are being suggested. Figure (2.11) identifies the various pieces of Android security literature as a comprehensive nomenclature of research directions [34]

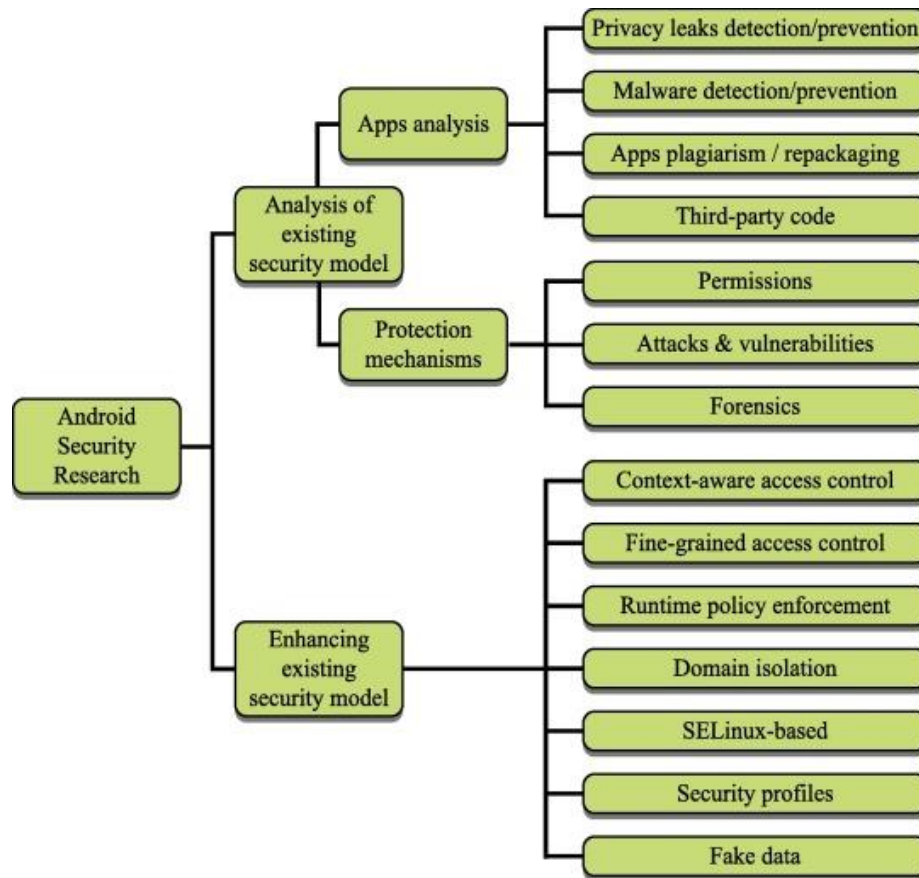


Figure (2.11): Taxonomy of literature on the security of android [34]

2.6 Randomness Tests [35]

A randomness Test is a set of algorithms that find out whether the distribution of data is random. Many tests are used in this thesis to test the randomness of the magic cubes.

- NIST Test Package [35]

To test the randomness of the keys that are generated by the proposed magic cube, the NIST (National Institute of Standards and Technology) package is used. This package is made up of :

- 1- Block Frequency Test
- 2-Cumulative Sums Test
- 3-Spectral DFT Test

- 4-Mono-bit Frequency Test
- 5-Lempel Ziv Compression Test
- 6-Linear Complexity Test
- 7-Longest Run of One's Test
- 8-Non- Overlapping Templates Test
- 9-Overlapping Templates Test
- 10-Binary Matrix Rank Test
- 11-Runs Test
- 12-Serial Test
- 13-Approximate Entropy Test
- 14-Random Excursions Test
- 15-Random Excursions Variant Test
- 16-Maurer's "Universal Statistical" Test

This test is a set of procedures tests, which aims to detect a sequence of binary numbers that do not work randomly.

It is a method, which is the probability that a certain string can be generated. For each test, if the (p-value) is greater than some constant level of confidentiality (α), then it is (passes). If ($P\text{-value} \geq \alpha$), then the string seems to be random, otherwise the string seems to be non-random.

For the tests, a significance level (α) can be chosen. Usually, (α) is chosen in the range [0.001- 0.01].

- When the value of (α) = (0.001), it means that one string in (1000) strings will be rejected by the test. A ($P\text{-value} \geq 0.001$), would mean that the string would be considered to be random with the confidence of (99.9%). A ($P\text{-value} < 0.001$), would mean that the string is non-random with a confidence of 99.9%.

2.7 Correlation Coefficient

The correlation coefficient factor is used to measure the relationship between two variables: The plaintext and its encryption. This factor demonstrates to what extent the proposed encryption algorithm strongly resists statistical attacks. Therefore, ciphertext must be completely different from the plaintext. The correlation coefficient is measured by the following equation.(2. 8, 2. 9, and 2. 10) [36].

$$Corr\ Coef(x, y) = \frac{\sum_{i=1}^n (x_i - \mu(x))(y_i - \mu(y))}{\sigma(x)\sigma(y)} \quad \dots (2.8)$$

where $\mu(x)$ and $\mu(y)$ are the respective means of x and y:

$$\mu(x) = \frac{1}{n} \sum_{i=1}^n x_i, \text{ and } \mu(y) = \frac{1}{n} \sum_{i=1}^n y_i \quad \dots (2.9)$$

x and y are variables of the plaintext and ciphertext.

and the terms in the denominators (It is called the standard deviations of x and y) are :

$$\sigma(x) = \sqrt{\sum_{i=1}^N (x_i - \mu(x))^2}, \text{ and } \sigma(y) = \sqrt{\sum_{i=1}^N (y_i - \mu(y))^2}, \dots (2.10)$$

If the correlation coefficient equals one, that means the plaintext and its encryption is identical. If the correlation coefficient equals zero, that means the ciphertext is completely different from the plaintext (i.e. good encryption). So, the success of the encryption process means smaller values of the correlation coefficient.

2.8 Mean Square Error (MSE)

MSE is used to specified hiding quality. It appears the cumulative squared error between the ciphered and the original one. If it is low, the process will be accepted else it will be rejected. It can be calculated by using the next equation (2.11) [37]

$$\mathbf{MSE} = \frac{\sum_{i=1}^M \sum_{j=1}^N (a(i,j) - b(i,j))^2}{M * N} \dots\dots\dots(2.11)$$

Where $M \times N$ is the image size and the parameters $a(i, j)$ and $b(i, j)$ refer to the pixels located at the i th row and the j th column of the original image and encrypted image, respectively. The smaller the MSE value, the better the encryption security.

2.9 Peak Signal to Noise Ratio (PSNR)

PSNR is the peak signal power about noise power. It has been calculated for the quality of the image. For the image is of good quality, the PSNR should be large. It can be measure as following equations (2.12)(2.13) [37]

$$\text{PSNR} = 10 \log_{10} (I_{\max}^2 / \text{MSE}) \text{ dB} \dots\dots\dots(2.12)$$

where: I_{\max} is the maximum intensity of the image.

The previous equation can also be expressed as follows:

$$\text{PSNR} = 10 \log_{10} (255^2 / \text{MSE}) \text{ dB} \dots\dots\dots(2.13).$$

2.10 Universal Quality Index (UQI)

The UQI is relied upon to valuation the deformation of two images. This quality index model any distortion as a combination of three various factors: Loss of correlation, contrast distortion, and luminance distortion [38]. Let assume x represents input image, and y represents output distorted image, therefore the loss of correlation is valued by the correlation coefficient which measures the linear correlation degree between x and y . The contrast and luminance are estimated by mean and standard deviation respectively. Mathematically, the UQI is expressed by equation (2.14) [39]:

$$\mathbf{UQI} = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\sigma_x^2 + \sigma_y^2)[\bar{x}^2 + \bar{y}^2]} \dots\dots\dots(2.14)$$

UQI has a dynamic range $[-1, 1]$. The optimal value 1 is achieved if and only if $x=y$ [36]

2.11 Normalized Cross-Correlation (NCC)

The NCC is a metric of similarity of two-wavelength as a function of the lost time applied to one of the wavelengths, the value of NCC must be lower that represents the low image quality, the NCC represents in equation(2.15) [40].

$$NCC = \frac{\sum_{m=1}^M \sum_{n=1}^N x(m,n).y(m,n)}{\sum_{m=1}^M \sum_{n=1}^N (x(m,n))^2} \quad \dots(2.15)$$

Where x is the original image, y is the encryption image, and the size of images equal to m×n.

If the NCC value equal to one, two images are completely similar, and if NCC equals zero, it declares that two images have no similarities[40].

2.12 Signal to Noise Ratio (SNR)

To evaluate the performance of the audio encryption, the detectability rate of encrypted data is measured by utilizing signal-to-noise ratio SNR, often expressed in decibels(dB). A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise as in the equation (2.16) [41]:

$$SNR_{MS} = \sum_{X=0}^{N-1} (g(x))^2 / \sum_{X=0}^{N-1} (g(x) - h(x))^2 \quad \dots(2.16)$$

The root means a square value of Signal-to-Noise Ratio (SNR_{rms}) is then given by equation (2.17).

$$SNR_{rms} = (SNR_{ms})^{0.5} \quad \dots(2.17)$$

Chapter Three

The Proposed System

3.1 Introduction

Multimedia data like images, text, video, audio, etc. are widely used and shared over the internet. These data are very sensitive, therefore security of transmission channels issues should be taken into consideration because hackers can unauthorized access to these sensitive data and misused it.

This chapter provides a general model illustrating the proposed system, which consists of the steps of the Chacha 20 algorithm after adding several levels of chaotic functions to produce the improved Chacha 20 algorithm. In this chapter, section (3.2) presents the design aims of the proposed system. Section (3.3) illustrates the primitive model of the proposed system. The design specifics and techniques of the proposed system are described in (3.4).

3.2 Design Objectives

The design aims of the proposed system are:

1. The proposed system transfers secure data (text, images, and audio) using the improvement of the Chacha20 algorithms based on a chaotic map.
2. Increase diffusion of Chacha20 stream key using a multi-level of chaotic function to increase the robustness and resistance of the proposed algorithm.
3. Build a faster encryption algorithm using a multi-layer of chaotic function to make the proposed IChacha20 more suitable for modern applications.

3.3 The Primitive Proposed Model

The initial model of the proposed system consisting of three aspects. sender, receiver, and the webserver. These sides collaborate as shown in figure (3.1).

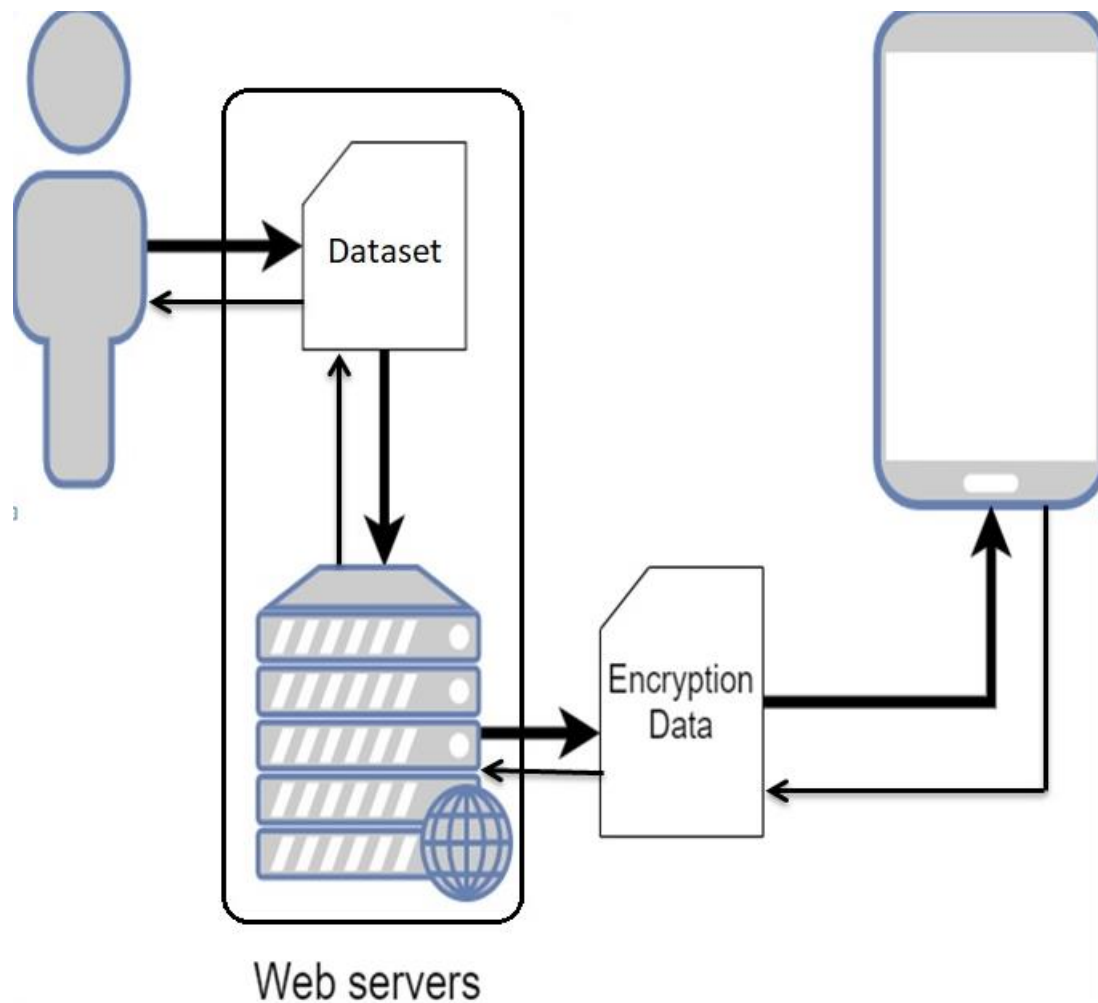


Figure (3.1) Primitive model of the proposed system

As clarifies in figure (3.1), both sender and recipient sides can be access to web servers through login into application in their mobile, and then exchange data (texts, image, and audio) between them through secure communication channels.

In the webserver side, the proposed system using proposed improvement chaotic Chacha20 (IChacha20) to encryption users data (text, image, and audio) and stored in the database, when the recipient login in their mobile application to request from web servers to receive the data from the sender, the web servers get data requested from the database and using proposed IChacha20 to decryption data and display it on the recipient mobile as shown in figure (3.2).

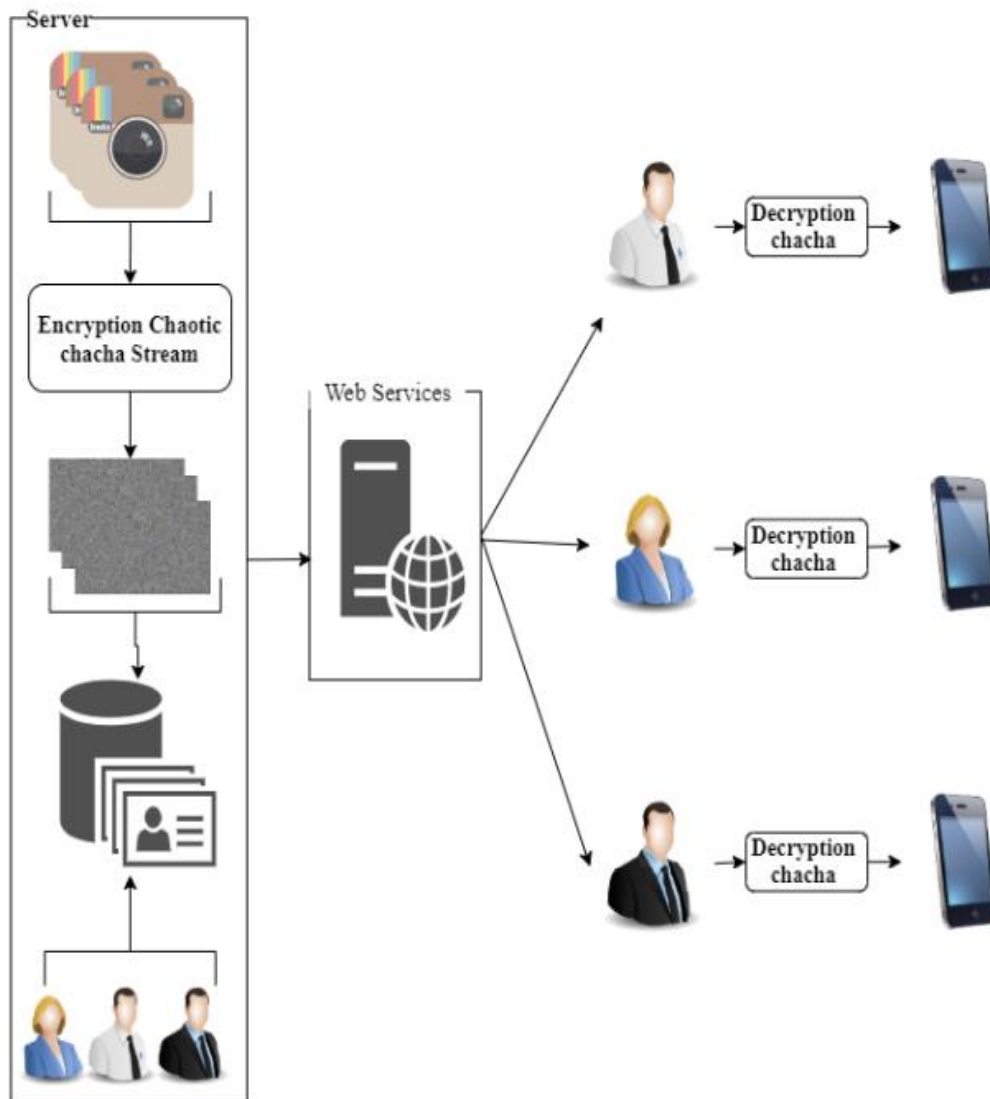


Figure (3.2) Exchange data between the sender and recipient using proposed system.

3.4 The Proposed Improvement Cahcha20 Algorithm (ICahcha20)

Figure (3.3) shows a general block diagram of the proposed IChacha20 algorithm using a multi-layer of a chaotic map (1d Tent map and Chebyshev).

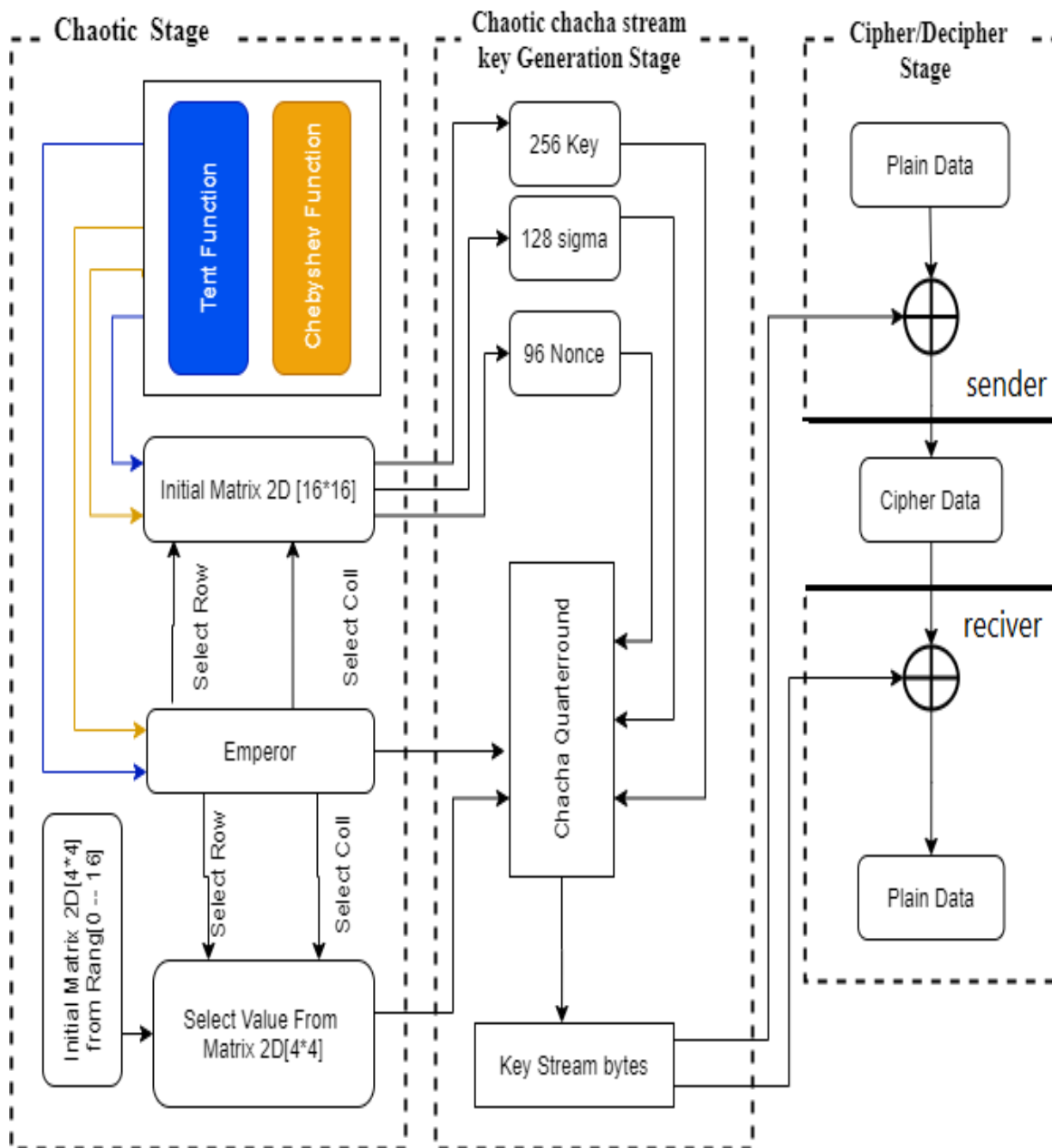


Figure (3.3) General block diagram of the proposed IChacha20

As shown in figure (3.3), the first stage is a chaotic layer, in this stage using (1D Tent and Chebyshev) chaotic function to generate random unsigned integer number, the second stage is the second chaotic layer to generate Chacha20 keystream of 64 –bytes, the final stage is encryption/decryption stage using the same key, all these stages collaborate to make improvement Chacha20 (IChacha20) algorithm able to encoding/decode data (text/image/audio) securely as illustrated in subsections (3.4.1,3.4.2, and 3.4.3)

3.4.1 Chaotic Stage

The first stage in the proposed method is generating random unsigned 32-bits integer numbers based on two types of chaotic maps (Tent and Chebyshev), this step aims to add a security layer to the proposed IChacha20 algorithm, the components of this stage are two sub-steps explained in the subsection (i and ii).

i. Create Initial Matrix

Algorithm (3.1) shows details to create an initial matrix with size [16*16] and the values of this matrix are random unsigned (Uint) 32-bit integer using 1DTent with setting initial parameters (R, X0) and Chebyshev chaotic functions with setting initial parameters (K, X0).

Algorithm (3.1): Create an initial matrix using Tent and Chebyshev function.

Input: R ,X ₀ //Tent initial Parameter K, X ₀ //Chebyshev initial Parameter No. iteration // Number of iteration
Output: Initial matrix2D [16,16] Initial_Matrix1D[4*4) Tent data ,Chebyshev data //float Un-Tent data, Un-Chebyshev data //Unsigned
Begin: Step 1: // process of getting Unsigned data from Tent For each(i=0; i< number of iterations; i++) Tent _{value} = value from apply equation (2.7)

```

    Rtent value = separation float (Tentvalue) based on "."
    Value1= RTent value
    Value2= Reverse (RTent value)
    Value1= Convert to Unsigned (value1)
    Value2= Convert to Unsigned (value2)
    Add to Tent data (Value1)
    Add to Tent data (Value2)
    X0 = Tentvalue
End for
Step 2:// process of getting Unsigned data from Chebyshev
For each( i=0; i< number of iterations; i++)
    Chebyshevvalue = apply equation (2.5)
    Rtent value = separation float (Chebyshevvalue) based on "."
    Value1= R Chebyshev value
    Value2= Reverse (Chebyshev value value)
    Value1= Convert to Unsigned (value1)
    Value2= Convert to Unsigned (value2)
    Add to Chebyshev data (Value1)
    Add to Chebyshev data (Value2)
    X0 = Chebyshevvalue
End for
Step 3: // initial Initial_Matrix2D selection
k = 0;
Foreach(row=0,col=0;row<16,col<16;row++,col++)
    Initial_Matrix2D[column , Row ] = Chebyshev data(k)
    Increment k by one
End for

Step 4// initial Initial_Matrix1D selection index using Round
k = 0;
Foreach(row=0,col=0;row<4,col<4;row++,col++)
    Initial_Matrix1D[column , Row ] = k
    Increment k by one
End for
End algorithm

```

As shown in step1 and 2 of an algorithm (3.1) both Tent and Chebyshev function follow the same procedure, as an example where each output of a

chaotic function are 100-unsigned 32-bits random number and they are opposed, so the final outputs of all chaotic maps are 400-unsigned random number each number of them consists from 32-bits. To achieve this, for example, Tent the function utilized setting parameters as inputs to its function, these parameters are $X_0 = 0.6556$ and $R = 0.954545$ and number of round [0-100] if the Tent value = 0.65388127808236, spilled this value and take digits after dots '.' and save in variable Tent value after dots =(65388127808236), find opposed of (65388127808236) and save in variable RTent value= 63280872188356, and finally convert both Tent value after dots and RTent value to unsigned 32-bit integer number and stored in data set with size [No.iteration*4].

In step 3 and 4 in the algorithm (3.1) shown how to get values from data set that stored 400 –unsigned chaotic numbers in random form and used these values to fill the initial matrix with size [16*16]. Figure (3.4) shows the initial matrix.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	598688909	3119016388	538125114	3597925130	538125114	3597925130	2304878418	4147852162	3061761334	3886128936	3806572452	920694348	4251077341	1500069026	1826686959	3674387265
2	2530821460	1129874523	4148231517	3368818746	4148231517	3368818746	1727776607	3065737591	1479979671	1164897711	68644362	2747741959	1110768457	3204899667	2570865542	1373998925
3	1117071528	3613818695	1687591872	4212110689	1687591872	4212110689	2888800135	1088420863	2275176538	990046302	4265099417	2770670250	3417476521	859235384	2975039552	1921880971
4	1902613347	786490252	2319218671	29970167	2319218671	29970167	1981038710	2122392046	1245748997	3797248895	2495948781	3154239249	948087204	3604570246	492652339	3321343942
5	3151246003	386070460	1319036796	3177303682	1319036796	3177303682	2525212812	4071046039	2203255601	15485660177	323001163	1652665195	2171586649	1835679174	4101862389	3908480255
6	1639345557	2814127236	264722208	3984910152	264722208	3984910152	2742365164	2598327915	436363403	3284391142	1756965943	912396601	1620395849	2968265150	2694786554	2114540187
7	711133391	3213041782	3342069818	4293456029	3342069818	4293456029	41516239084	107643285	416947349	2062627919	2706988738	18691880	40038152	3507923239	3346321983	808287764
8	2385862029	2475185565	2856169041	1312987008	2856169041	1312987008	4056698285	3921541332	3039043794	2229814742	1394906384	3784276334	3638044866	233967185	1741169908	452468129
9	1531737879	3788341348	3336388261	4237128227	3336388261	4237128227	4216636622	333963758	1560934592	3820155188	1485536548	1488284470	1092744547	1153730374	4176692299	1727464104
10	1736674199	3887190984	1328419763	3494869154	1328419763	3494869154	166378165	760274759	1641029255	23783217167	778892386	2842926385	832291951	887674549	45455471	1197128563
11	2663217975	117979456	1350311399	2174462741	1350311399	2174462741	1566091480	3479718331	4152569479	1982867142	3703486439	3217079655	2592623037	1589336153	83570441	302606068
12	1364052233	3094206821	4259080988	1306717033	4259080988	1306717033	1567648218	1422707826	3173366036	1167039954	3089786833	3782066128	3438921718	4075483478	4057448323	1852011241
13	598688909	132108542	2268903089	3896598278	2268903089	3896598278	3736345977	708803041	3198751251	2413080761	77184261	3926623230	1324317607	887170017	3228671951	1730213444
14	2530821460	4166430645	3179956906	3839815527	3179956906	3839815527	150126368	3225700237	3607986248	3586265279	699331693	2831775446	1166402488	2947717275	175027228	1476660763
15	1117071528	4154140198	672838576	4175401497	672838576	4175401497	1875344438	1843441639	1108904761	2066275202	2827969232	212137255	3250899408	1525616747	1675448843	111037353
16	1902613347	19278853	3807921691	1244856689	3807921691	1244856689	417500233	1140305659	1499602085	1751591231	1781238113	159733871	3503566184	217904251	148182913	3429831151

Figure (3.4) Initial matrix [16*16].

ii. Select Values from Initial Matrix Step

The second step in the chaotic map stage of the proposed IChacha20 algorithm is selected element randomly from the initial matrix. Figure (3.5) shows a block diagram of the chosen element from the initial matrix.

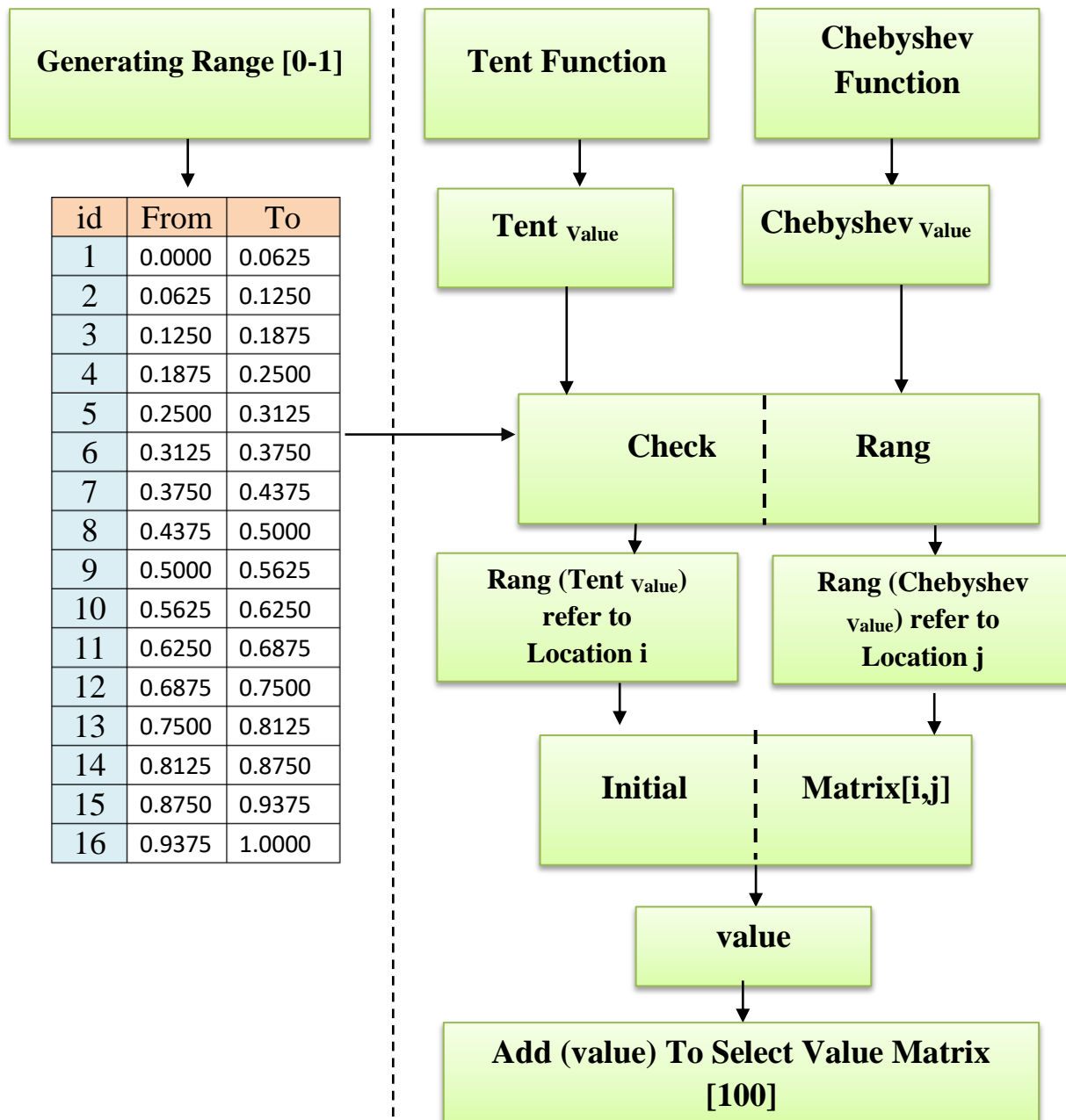


Figure (3.5) Block diagram of the select value step.

As illustrated in figure (3.5) this step consists of two sub-step: firstly, generate 16 intervals between [0-1] called Range as shown in the algorithm (3.2) and the second step is to check the values that generate by the Tent and Chebyshev function in the previous step (Tent value and Chebyshev value) with Range to return index (id) of that Range to choose location [i,j] of initial matrix and return value of that location and save in array called select value [100] as shown in an algorithm (3.3).

Algorithm (3.2) : Generating range [0-1]

Input: From // initial value of Rang To // End value of Rang Xstep // Decrement value for each step
Output: period table (id ,From,To)
Begin: Step 1://generation parameters Rang xstrat = 1 : xStep = 1/ 16 xend = xstrat - xStep Step 3: // full Rang from value Add period table (1, xstrat, xend) xstrat =xend for each(i = 2 ; i< 16; i++) xend = xstrat - xStep Add period table (i, xstrat, xend) xstrat = xend end for step4 :end

Algorithm (3.3): Select value from the initial matrix.

Input: Tent data (Tent _{value}) , chebyshev data (chebyshev _{value}) Initial Matrix2d [16*16] ,Rang
Output: full-value
Begin Step 1: for each element in Ten data (Tent _{value}) and chebyshev data (chebyshev _{value}) Goto step 2 with value Tent _{value} loci = check (element Tent) with Range Goto step 2 with value Chebyshev _{value} locj = check (element Chebyshev)with Range full-value = get value from initial Matrix[loci, locj] Increment k by one End for Step2://return check in Range table using enter xvalue For each element in Range do If (Element. Xstrat > xvalue and Element. Xend <= xvalue) then return Element.location End for Step 3 :Return (full-value) End algorithm

Figure (3.6) clarifies the output of algorithm (3.3), where the red location represents the element in the Initial matrix that is chosen by Tent and Chebyshev's chaotic function.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3151249603	3788341348	3335358261	942724221	3802896469	333963758	4227240563	952620725	1590934592	3592864109	3516839106	1488284470	3419106534	1153730374	4178692299	1727464204
2834962651	1129874523	4148231517	3358815746	1727276507	3055737591	1364030328	1974920407	1479979571	1164897711	68544362	2747741959	1110768457	3204099667	2570505542	1373995825
2663217975	4154140198	3894576425	4175401497	1875344438	4211238330	3483399614	796961467	1109304761	177270929	1101334131	2149984232	3250899408	1825616747	1497253685	1110037353
2536082450	2814127236	2217259996	3984910152	2742355164	3082251777	2860475204	1579092846	436363403	3284391142	2878213366	912396601	1620395849	1700926910	396206548	2114540187
2535119603	785690252	2319218571	29970167	1981038710	2122392046	2739877055	438562406	1345749597	3737243895	2495948781	3154329349	948037204	3604576246	492552339	622388116
2385862029	3094206821	4259080988	2819604623	1567648218	3580167104	3553482014	2655014460	3173396036	1167039864	2533074973	3782066128	3438921718	204329059	1132599472	846270705
1902613347	2475185565	2856169041	1312987008	4056698285	3922541332	1469929012	3829765683	2756420002	2229814742	1394906384	953005839	4008502369	1516085030	715704937	452468129
1735674199	3755151355	3179956906	1966720477	150126368	3225700237	204309067	3267181433	3458658785	3586265279	699331693	3061893403	1166402488	4148817697	1293364268	1476660763
1639345557	304401443	1328419763	3494869154	166378165	760274759	2108105397	4176291425	3796843874	3960196783	1862346865	3765388870	832291951	887674549	45455471	11971385
1545693932	3119016388	538125114	3597925130	2304878418	4147852162	907764496	3149748329	3061761334	3886128936	3800572452	920684348	4251077341	1580069026	1825686959	36743572
1532737879	1321008542	752624057	3896598278	2649371972	219998386	3445260937	180611206	3406560773	2413080761	77184361	3926523230	1324317607	1580485702	3228671951	17302134
1364052233	193788853	479583430	3877429321	579591771	1140305650	1839704113	3261078434	2948501137	1065273437	4180633865	1683558438	3103936600	3829349781	148182913	34298311
1117071528	3213041782	3242069818	4042790300	4151629084	1076431285	446869595	1379901486	416947349	2062627919	3476816122	18691880	40038152	3507923239	3346321983	80828776
1031376673	3613818695	1333911864	4212110689	4156223829	1098420863	983602489	2459939750	2275176538	990046302	3550025037	516637849	3417470521	859325384	511220279	14420264
711133391	117379456	3062584478	2174462741	1565091480	3355966177	3804919623	3457269805	4152569478	1982857142	3703486439	3217079855	2592052037	1589336153	835270441	30260606
598688909	386070460	1319036796	3177303682	2525212812	4071046039	1013741445	1121984227	220325561	1548556177	323001163	1652565195	3720517263	1835679174	4101862099	39094902

Figure (3.6) Selected value from initial matrix.

3.4.2 Chaotic Chacha Key Stream Generation Stage

The second layer of chaotic producing Chacha stream key stage, the input of this stage is a select value matrix that contained 100 values with the type of an unsigned random integer of 32 bits and the output is keystream 64-bytes.

To generate Chacha20 keystream 64 bytes using the chaotic function, the proposed algorithm follow three main sub-steps as illustrated in section (i, ii, and iii)

i. Create Matrix or (X-matrix)

To create Chacha x-matrix, the proposed algorithm computes a 256-bit key, 128-bit sigma, and 96 –bits nonce and fill x-matrix as shown in figure (3.7).

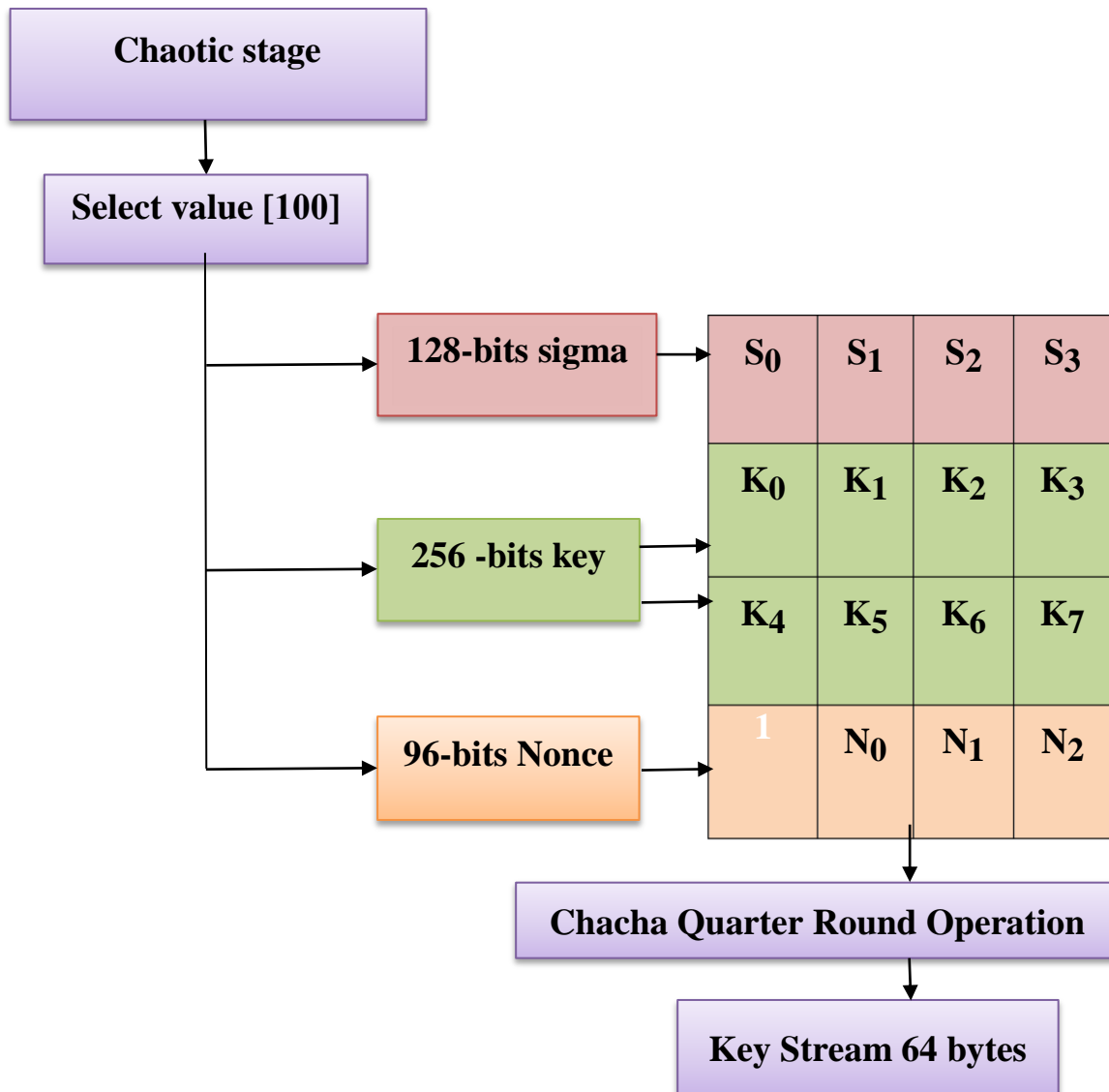


Figure (3.7) Block diagram of creating the IChacha matrix.

In this step, the proposed method computer values of (key| nonce| sigma) :the Chacha20 cipher operates on 32-bit words, takes as input a 256-bit key $K = ("k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7")$ as well as 96-bit nonce $N = ("N_0, N_1, N_2, N_3")$, a 128-bit block sigma $S = ("S_0, S_1, S_2, S_3")$, and a 32-bits counter which is equal (1). The proposed IChacha20 operate on a 4×4 matrix of 32- bits words

called (X matrix). Algorithm (3.4) clarifies details of computes (256-bit key, 96-bit nonce, and 128-bit block sigma).

Algorithm (3.4): Compute of Chacha20 (256-bit key, 96-bit nonce, and 128-bit block sigma)

Input: full-value
Output: 256-bit key, 96-bit nonce, and 128-bit block sigma
Begin Step 1 // create key K=0 For each(i=0; i<8;i++) byte[] output = new byte[4] xvalue= full-value[i] // Unsigned value goto step 4// Unsigned To Byte where input = xvalue key[k] = output[0] key[k+1] = output[1] key[k+2] = output[2] key[k+3] = output[3] k +=4 End for Step 2 :://Create nonce K=0 For each(i=0; i<3;i++) byte[] output = new byte[4] xvalue= full-value[i] // Unsigned value goto step 4// Unsigned To Byte where input = xvalue nonce [k] = output[0] nonce [k+1] = output[1] nonce [k+2] = output[2] nonce [k+3] = output[3] k +=4 End for Step 3 :://Create sigma K=0 For each(i=0; i<4;i++) byte[] output = new byte[4] xvalue= full-value[i] // Unsigned value goto step 4// Unsigned To Byte where input = xvalue

```

        sigma [k] = output[0]
        sigma [k+1] = output[1]
        sigma [k+2] = output[2]
        sigma [k+3] = output[3]
        k +=4
    End for
Step 4:// Convert Unsigned ToBytes
    outputOffset=0;
    output[outputOffset] = (byte) xvalue
    output[outputOffset + 1] = (byte)( xvalue >> 8)
    output[outputOffset + 2] = (byte)( xvalue >> 16)
    output[outputOffset + 3] = (byte)( xvalue >> 24)
    return output
step 5: end
End algorithm

```

In step1 of the algorithm (3.4), the proposed system has 100 unsigned number save in an array called select value [100], so, to generate 256 –bits key (8- words) the proposed technique used first 8 number from select value array and following in order several steps, for example :

- 1- Select value [1] = **2706988738** , select value [2] = **778892386** , select value[3] = **1752591231** , select value[4] = **3039043794** , select value [5] = **1306717033** , select value[6] = **217904261**, select value[7] = **3246222359**, and select value[8] = **264722208**
- 2- Take first value (**2706988738**) and convert it to bytes as following
 Input=27066988738
 Output [0] =(byte) input
 Output [1] = (byte)(input >> 8)
 Output [2] = (byte)(input >> 16)
 Output[3] =(byte) (input >> 24)

Where operation ">> n" denotes an n-bit right rotation, so the results of this step are:

Output[0]	Output[1]	Output[2]	Output[3]
194	94	89	161

3- Create key [k] , where $k = 0, \dots, 255$ as

$$\text{Key}[k] = \text{Output}[0]$$

$$\text{key}[k+1] = \text{Output}[1]$$

$$\text{key}[k+2] = \text{Output}[2]$$

$$\text{key}[k+3] = \text{Output}[3]$$

4- Repeated step (2 and 3) for each value of select value [2-8] .Further, the results of the key expansion of a 256-bits are $K = (" k_0, k_1, k_2; k_3, k_4, k_5; k_6, k_7")$ as illustrated in figure (3.7).

In step 2 and 3 of an algorithm (3.4), follow the same procedure of step1 and the results are 96-bit nonce $N = (N_0, N_1, N_2)$, and a 128-bit block sigma $S = (S_0, S_1, S_2, S_3)$ are shown in figure(3.8, 3.9 and 3.10).

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

Figure (3.8): 256-bits key.

S_0				S_1			
39	245	165	12	244	36	200	103
S_2				S_3			
213	218	164	222	136	123	212	208

Figure (3.9): 128-bits sigma.

N_0				N_1				N_2			
228	61	197	22	225	239	34	212	97	245	32	42

Figure (3.10): 96-bits nonce.

ii. The Chaotic Layer

In this step, is used values of Select matrix to using the value of Tent function to select a row from the Chacha matrix (x matrix) as well as using the value of Chebyshev function to select the column from the Chacha matrix (x matrix), and the conjunction of row and column in this matrix represents the value of X_h [0,...15]. Finally, using the selected values of X_h [0,...,15] as inputs to Chacha quart round operation.

iii. The Quarter Round of The Chacha20:

The main function of the ChaCha 20 algorithm is the quarter round. It runs on four 32—bit unsigned integers, did refer to as a, b, c, and d. The operation is carried out using equations (2.1,2.2,2.3, and 2.4). Chacha20 runs 20 rounds, alternating between "column rounds" and "diagonal rounds". Each round, consisting of four quarter rounds, is run as shown in the algorithm. (3.5).

Algorithm (3.5): IChacha20 Quart Round

Input: Key K, Block Counter C, and Nonce N

x0,

Matrix2D[16,16) ,Matrix1D[4*4)

period table (id ,From,To)

Output: Z // keystream

Step1: $X \leftarrow \text{InitialMatrix}(K, C, N)$

$Y \leftarrow X;$

For each($i=0; i < 9; i++$)

Got to step2// input x0 and return array 1d call loc[]

/* Column Round */

$(x_0, x_4, x_8, x_{12}) \leftarrow \text{QuarterRound}(\text{loc}[0], \text{loc}[4], \text{loc}[8], \text{loc}[12])$

$(x_5, x_9, x_{13}, x_1) \leftarrow \text{QuarterRound}(\text{loc}[5], \text{loc}[9], \text{loc}[13], \text{loc}[1])$

$(x_{10}, x_{14}, x_2, x_6) \leftarrow \text{QuarterRound}(\text{loc}[10], \text{loc}[14], \text{loc}[2], \text{loc}[6])$

$(x_{15}, x_3, x_7, x_{11}) \leftarrow \text{QuarterRound}(\text{loc}[15], \text{loc}[3], \text{loc}[7], \text{loc}[11])$

/* Diagonal Round */

$(x_0, x_5; x_{10}, x_{15}) \leftarrow \text{QuarterRound}(\text{loc}[0], \text{loc}[x_5], \text{loc}[10], \text{loc}[15])$

$(x_1, x_6, x_{11}, x_{12}) \leftarrow \text{QuarterRound}(\text{loc}[1], \text{loc}[6], \text{loc}[11], \text{loc}[12])$

$(x_2; x_7; x_8, x_{13}) \leftarrow \text{QuarterRound}(\text{loc}[2], \text{loc}[7], \text{loc}[8], \text{loc}[13])$

$(x_3, x_4, x_9, x_{14}) \leftarrow \text{QuarterRound}(\text{loc}[3], \text{loc}[4], \text{loc}[9], \text{loc}[14])$

end

$Z \leftarrow X + y$

Goto step 4

Step2: Quarter Round new location

loc = uint[16]

K = 0

For each($i=0, j=0; i < 4, j < 4; i++, j++$)

Got to step3// get location (x0) return location

loc [k] = loc t

k++

End for

return array 1d call loc;

step3: return get location in rang table using enter xvalue

For each element in Rang do

If (Element. Xstrat > xvalue and

Element. Xend <= xvalue) then

return Element.location

End for

Step 4: Return Z

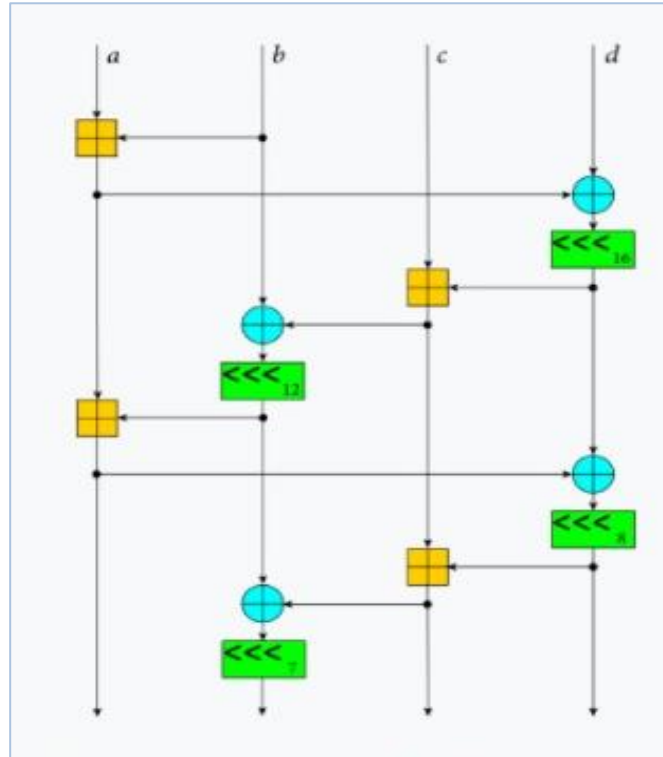


Figure (3.11) Flowchart of Chacha20 quarter round

3.4.3 IChacha20 Encryption /Decryption Operation Stage

Figure (3.12) shows the block diagram of IChacha20 Encryption /Decryption operation for data (text, image, or audio).

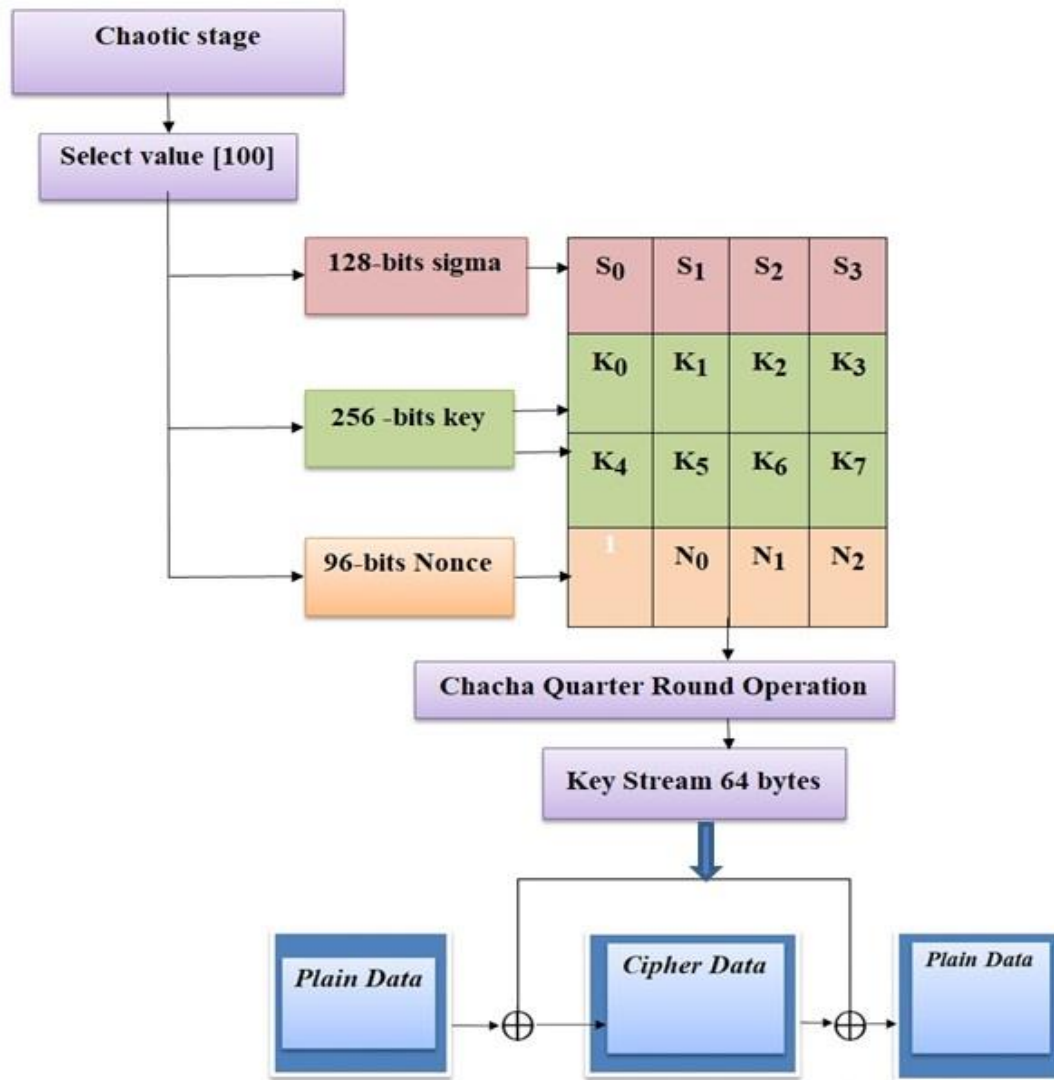


Figure (3.12): Block diagram of IChacha20 algorithm

The IChacha20 sequentially needs to call the Chacha20 block function, with the same key, nonce, sigma, and consecutively continuing to increase the block counter variables. Concatenating the keystream blocks from the successive blocks forms a keystream. The IChaCha20 algorithm then performs an XOR of this keystream with the plain data. Alternatively, each keystream block can be XORed with a plain data block before proceeding to create the next block, saving some memory. Algorithm (3.6) the encryption data using IChacha20 algorithm,

where the input of this algorithm are: a 256-bits key, a 32-bit initial counter, a 96 – bits nonce, 128-bits sigma, and finally an arbitrary-length plaintext or an arbitrary size of the image and the outputs are an encrypted text of the same length or cipher image with the same size.

Algorithm (3.6): The IChacha20 encryption algorithm.

Input: key, sigma, nonce Plain text// text, image, audio
Output: cipher message //ciphertext or image
Begin For each(j=0 ; j<len(plaintext)/64)-1;j++) keystream=Chacha20-block(key,sigma,nonce) block = plaintext[(j*64)..(j*64+63)] encrypted-messages += block \oplus key-stream if ((len(plaintext) % 64) <> 0) then j = len(plaintext)/64 key_stream = Chacha20-block(key,sigma, nonce) block = plaintext[(j*64)... len(plaintext) -1] encrypted_mes += (block^key-stream) [0..len(plaintext)%64] endif end return encrypted -message End algorithm

The decipher operation of the IChacha20 algorithm is to follow the same steps of the encryption IChacha20 algorithm but in opposed order.

Chapter Four

Results and Analysis

4.1 Introduction

This chapter is dedicated to present the results and tests that evaluate the performance of the proposed method. Since the proposed system consists of three stages (Chaotic Stage, Chaotic Chacha keystream Generation Stage, and IChacha20 encryption /decryption Operation Stage) therefore different measures are used to evaluate these stages. An initialization is given in section (4.2). The implementation of the proposed system is illustrated in section (4.3). While section (4.4) clarifies the results of the proposed system.

4.2 Initialization

The proposed system is implemented by using Microsoft visual studio (2019) by the C# programming language, Android Studio 3.4.1, SQL Server 2012, with a laptop computer (core i7 - seven generations) and (Windows 10) as an operating system.

4.3 The Implementation of the Proposed System

The implementation of the proposed system starts when the sender attends to send data to the receiver via the server, the server gives to that user (user names and password) with cipher data. On the receiver side, starting with the main interface of the application that requires (username and password) to log in, as shown in figure (4.1). After the receiver entering its username and password then show the decryption interface of the application as shown in figure (4.2).



Figure (4.1) The main interface of the receiver side



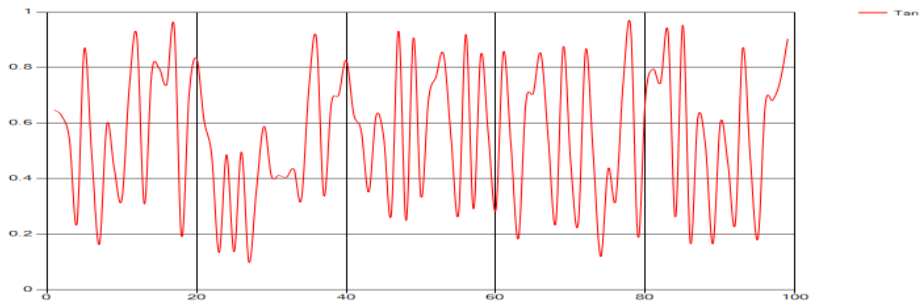
Figure (4.2) The decryption interface of the application

4.4 Results of The Proposed System (IChacha20 Algorithm)

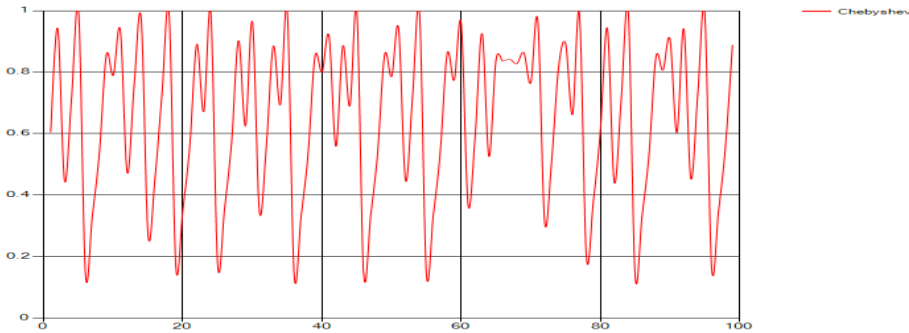
In this section, present in order the results of each step of the proposed IChacha20 algorithm respectively in subsection (4.4.1,4.4.2, and 4.4.3).

4.4.1 Results of Chaotic Stage

Figure (4.3) shows the outputs of chaotic functions, where the setting parameters of chaotic Tent map are $R = 0.96, x_0 = 0.66, \text{number of iteration} = 100$, and the setting parameters of the Chebyshev chaotic map are $k=5, x_0 = 0.21, \text{and number of iteration} = 100$.



(a)



(b)

Figure (4.3) Behaviors of chaotic function; a) Tent chaotic function,
b) Chebyshev chaotic function.

Where (x-axis: Parameter space $\in (0,1)$, y-axis: Chaotic range $\in (0, 1)$)

Figure (4.4) illustrated results of selection location from the initial matrix in a random manner based on values of 1d Tent and 1d Chebyshev chaotic functions to selection number of row i and column j .

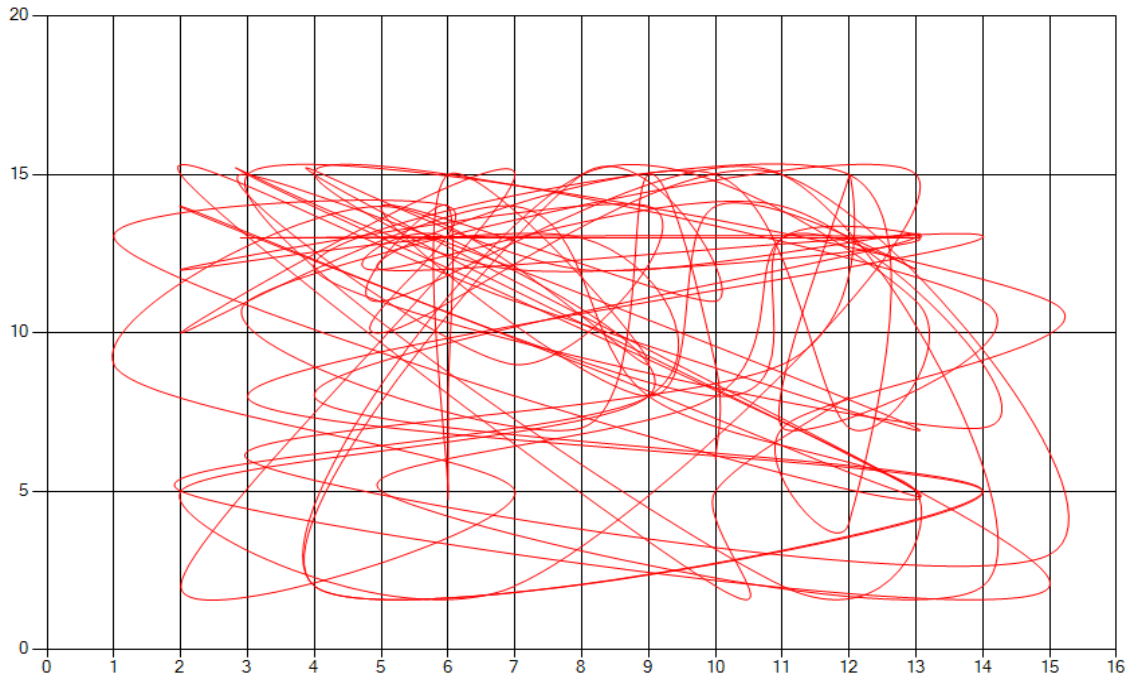


Figure (4.4) Result of selection location from the initial matrix [16*16] based on values of 1d Tent and 1d Chebyshev chaotic functions.

Where row represented by Tent, column represented by Chebyshev map.

4.4.2 Results of Chaotic Chacha Keystream Generation Stage

Table (4.1) shows the results of the stream key setup which includes a 256-bit key, 96-bit nonce, 128 –bits sigma.

Table (4.1) Results of the key setup 256-bit key,128-bit sigma, and 96-bit nonce.

256-bit key						128-bit sigma			96-bit Nonce		
item no.	Chaotic[x]	Byte	item no.	Chaotic[x]	Byte	item no.	Chaotic[x]	Byte	item no.	Chaotic[x]	Byte
1	2706988738	194	5	1306717033	105	1	212137255	39	1	382025188	228
		94			235			245			61
		89			226			164			197
		161			77			12			22
2	778892386	98	6	217904261	133	2	1741169908	244	2	3559059455	255
		244			244			36			239
		108			252			200			34
		46			12			103			212
3	1752591231	127	7	3246222359	23	3	3735345877	213	3	706803041	97
		107			108			218			245
		118			125			164			32
		104			193			222			42
4	3039043794	210	8	264722208	32	4	3503586184	136			
		32			87			123			
		36			199			212			
		181			15			208			

4.4.3 Results of Encryption Data using Proposed IChacha20 Algorithm

The encryption process using the proposed IChacha20 algorithm Applying to three types of data (text, image, and audio), so this section is shown the results of encryption each type of data respectively in subsection (i, ii, and iii).

i. Results of Encryption Text using IChacha20 Algorithm

Table (4.2) shown text samples for using an encryption test of the proposed IChacha 20 algorithm. The testing methodology consisted of randomly generated data split in predefined sizes (8, 16, 32, 64, 128, 256 MBs) and sent to the encryption modules. Figure (4.5) clarifies the practical application of the proposed encryption IChacha20 algorithm using a text sample (#1)

Table (4.2) Tests plaintext samples

No.Text	Text	Text size
#1	"Ministry of Higher Education and Scientific Research/Diyala University/College of Science/ Department of Computers ."	114 Char
#2	"Convince what God has given you may be the richest people, but you don't know"	77 Char

Hash	Char #1	Char #2	Char #3	Char #4	Char #5	Char #6	Char #7	Char #8	Char #9	Char #10	Char #11	Char #12	Char #13	Char #14	Char #15	Char #16
0000	M	i	n	i	s	t	r	y		o	f		H	i	g	h
0016	e	r		E	d	u	c	a	t	i	o	n		a	n	d
0032		S	c	i	e	n	t	i	f	i	c		R	e	s	e
0048	a	r	c	h	/	D	i	y	a	l	a		U	n	i	v
0064	e	r	s	i	t	y	/	C	o	l	l	e	g	e		o
0080	f		S	c	i	e	n	c	e	/	D	e	p	a	r	t
0096	m	e	n	t		o	f		C	o	m	p	u	t	e	r
0112	s	.														

a) Split plain text #1

Hash	Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8	Byte #9	Byte #10	Byte #11	Byte #12	Byte #13	Byte #14	Byte #15	Byte #16
0000	77	105	110	105	115	116	114	121	32	111	102	32	72	105	103	104
0016	101	114	32	69	100	117	99	97	116	105	111	110	32	97	110	100
0032	13	10	83	99	105	101	110	116	105	102	105	99	32	82	101	115
0048	101	97	114	99	104	47	68	105	121	97	108	97	32	32	14	85
0064	110	105	118	101	114	115	105	116	121	47	67	111	108	108	101	103
0080	101	13	10	111	102	32	83	99	105	101	110	99	101	47	67	111
0096	109	112	117	116	101	114	115	32	68	101	112	97	114	116	109	101
0112	110	116	13	10												

b) ASCII code of each character of the plaintext#1

1374659166	3334001404	3459224807	1503453264
3288287840	2769032972	2097303670	809070472
2344945394	2802150961	833558683	3134718571
1	2328993245	3208036621	2440375337
2204560438	955107783	1169771929	1394803218
4203182440	3039001457	839477794	696523354
1189532304	238946272	1492119273	1192979903
1145592442	95381558	3553174574	1081436883

c) keystream blocks

Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8	Byte #9	Byte #10	Byte #11	Byte #12	Byte #13	Byte #14	Byte #15	Byte #16
192	164	145	88	86	119	112	158	33	77	65	111	165	252	189	53
210	214	198	241	62	119	73	154	160	205	137	212	240	254	6	235
197	113	15	74	222	209	152	74	197	172	13	50	85	4	163	215
172	120	26	247	219	215	37	104	211	104	140	61	57	84	159	139
91	103	66	242	202	118	102	31	147	205	111	155	68	144	130	48
201	222	143	87	178	231	236	44	181	197	114	52	109	180	169	32
247	81	122	194	213	0	220	82	23	108	125	193	5	32	215	217
203	63	178	108												

d) 64-bytes of stream key using chaotic map

Hash	Char #1	Char #2	Char #3	Char #4	Char #5	Char #6	Char #7	Char #8	Char #9	Char #10	Char #11	Char #12	Char #13	Char #14	Char #15	Char #16
0000	141	205	255	49	37	3	2	231	1	34	141	205	255	49	37	3
0016	183	164	230	180	90	2	42	251	212	164	183	164	230	180	90	2
0032	200	123	92	41	183	180	246	62	172	202	200	123	92	41	183	180
0048	201	25	104	148	179	248	97	1	170	9	201	25	104	148	179	248
0064	53	14	52	151	184	5	15	107	234	226	53	14	52	151	184	5
0080	172	211	133	56	212	199	191	79	220	160	172	211	133	56	212	199
0096	154	33	15	182	176	114	175	114	83	9	154	33	15	182	176	114
0112	165	75	191	102												

e) ASCII code for cipher text#1

Hash	Char 1	Char 2	Char 3	Char 4	Char 5	Char 6	Char 7	Char 8	Char 9	Char 10	Char 11	Char 12	Char 13	Char 14	Char 15	Char 16
0000	•	İ	ÿ	1	%			ç		"	'	O	í	•	Ú]
0016	•	α	æ	´	Z		*	û	Ô	α	æ	°	Đ	Ÿ	h	•
0032	È	{	\)	•	´	ö	>	¬	Ê	d	Q	u	V	Æ	α
0048	É		h	”	3	ø	a		a		à	\		t	‘	Þ
0064	5		4	—				k	ê	â	,	ô	(ü	ç	W
0080	¬	Ó	...	8	Ô	Ç	¿	O	Ü			W		>	ê	O
0096	š	!		¶	°	r	-	r	S				w	T	°	¼
0112	¥	K	¿	f												

f) Ciphertext #1





Figure (4.5) Practical application of the proposed encryption IChacha20 algorithm using a text sample (#1).

The text samples presented in Table (4.2) using the proposed IChacha20 algorithm evaluated based on the metric of the correlation coefficient, as indicated in equation no.(2.8). Where the value of the correlation coefficient appeared for the first text and using the IChacha 20 algorithm (0.5952), While the value of the correlation coefficient is for the second text encoded using the IChacha 20 algorithm (0.6162).

ii. Results of Encryption Image using IChacha20 Algorithm.

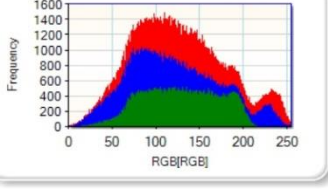
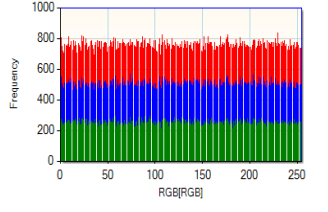
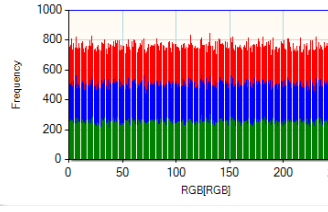
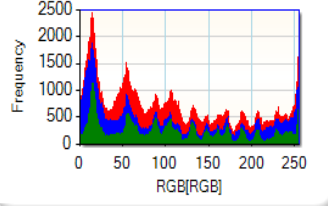
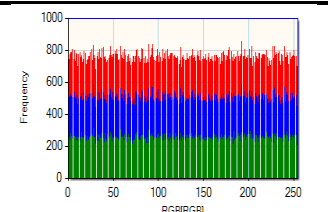
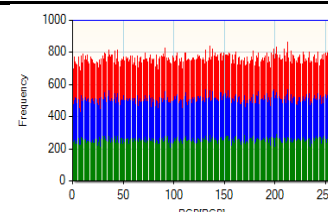
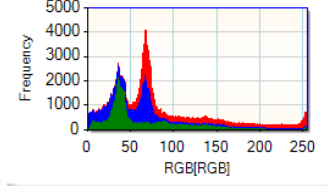
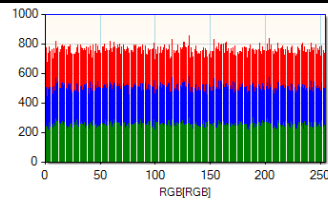
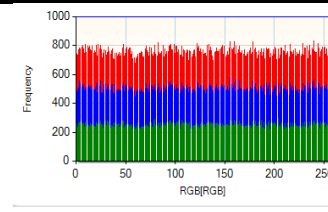
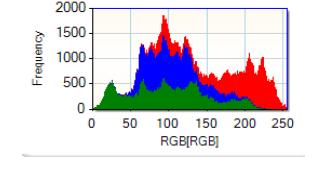
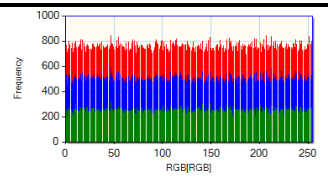
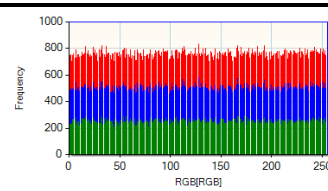
The good cipher algorithm must be sensitive to the secret key and the plain data. If the proposed algorithm is sensitive to the secret key, when the proposed algorithm is using another key with only a small difference for image encryption, the results should be very different. Table (4.3) shows four-color images samples in RGB color space which are using in the proposed IChacha 20 algorithm.

Table (4.3) The Tests Image Samples

No. Image	Images	Image Size	Image Dimensions
#1		28.6KB	256x256
#2		48.0KB	256x256
#3		20.0KB	256x256
#4		192KB	256x256

Table(4.4) show the original test image with its histogram using both cipher algorithm that original Chacha20 and proposed IChacha20. . where illustrated results of the table the distributed pixels of original test images and their corresponding cipher images, the results provide the histogram images that ciphering by using proposed IChacha20 more homogeneous and uniform, making it difficult for unauthorized people to obtain any data from the encoded images.

Table (4.4) Histogram of original test images and their corresponding cipher images using original Chacha20 & proposed IChacha20

No. Image	Histogram original image	Histogram Cipher image Using Chacha20	Histogram Cipher image Using IChacha20
#1			
#2			
#3			
#4			

The results of image quality metrics are applied between original test images and their corresponding cipher images using both original Chacha20 and IChacha20 algorithms, the metrics are: Mean Square Error (MSE) as in equation no. (2.11) and the best value of it must be lower, Peak Signal-to-Noise Ratio (PSNR) as in equations (2.12)(2.13) respectively and the best value of it

must be higher. Universal Quality Index Image (UQI) must be closed to 1 as in equation no. (2.14) and Normalized Correlation Coefficient(NCC) must be closed to 1 as in equation no. (2.15). The results of these metrics showed in a table (4.5), the IChacha20 has the best results more than the original Chacha20. The best value of MSE =9516.2558403015, the best value of PSNR= 8.3461425153Db, the best value of NCC =0.4374341199, and finally, the best value of UQI =0.5105258736.

Table (4.5) Results of MSE, PSNR, NCC, and UQI metrics

Metrics	No. Image	Image Size	Original Chacha20	Proposed IChacha20
MSE	#1	28.6KB	6297.3720207214	6301.7086143494
	#2	48.0KB	9511.8949851990	9516.2558403015
	#3	20.0KB	8497.7762374878	8502.9133377075
	#4	192KB	6794.4051132202	6815.6725168228
PSNR	#1	28.6KB	10.1392101062	10.1362204297
	#2	48.0KB	8.3481331400	8.3461425153
	#3	20.0KB	8.8377506976	8.8351260813
	#4	192KB	9.8092892296	9.7957164583
NCC	#1	28.6KB	0.9270183361	0.9282682998
	#2	48.0KB	0.4382184016	0.4374341199
	#3	20.0KB	0.5255600819	0.5252820990
	#4	192KB	0.4751681715	0.4731776050
UQI	#1	28.6KB	0.4952951992	0.4937781146
	#2	48.0KB	0.4333429762	0.4323990544
	#3	20.0KB	0.5107446324	0.5105258736
	#4	192KB	0.4714120255	0.4694661374

Table (4.6) includes execution time (per second) between encrypted images by using the original algorithm and the proposed IChacha20 algorithm, where the proposed IChacha20 algorithm was faster than the original algorithm in execution time over all test images.

Table (4.6) Execution time of all images by using the original algorithm and the proposed IChacha20 algorithm

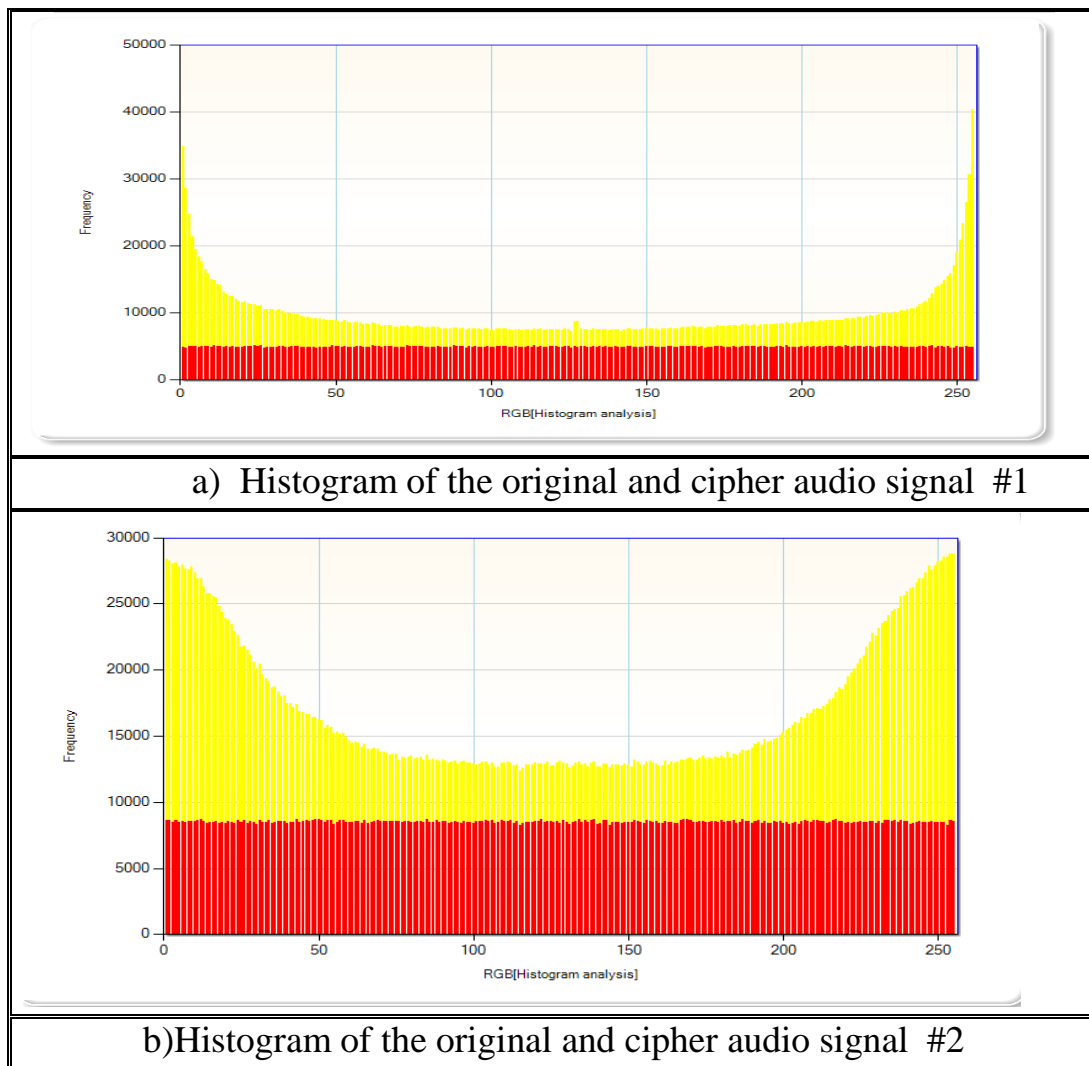
Item no.	Original algorithm	IChacha20 algorithm
Img1	03:34.1 sec	02:53.6sec
Img2	03:16.7 sec	03:10.0 sec
Img3	02:07.1 sec	01:26.4 sec
Img4	02:12.1 sec	01:27.6 sec

iii. Results of Encryption Audio using IChacha20 Algorithm.

Table (4.7) show two audio samples using in test performance of the proposed IChacha20 algorithm based on Correlation analysis as in equations no. (2.8) (2.9) (2.10) and Signal to noise ratio(SNR) as in equation no. (2.16) (2.17) metrics. Figure(4.6) illustrated the histogram of original audio and its corresponding cipher audio using proposed IChacha20 algorithms, where the yellow pixels represent original audio and red pixels represent cipher audio.

Table (4.7) Audio samples

No. Audio	Audio time	Size	Bitrate	Format
#1	00.00.11 Sec	2.08 MB	1536kbps	wave
#2	00.00.07 Sec	1.21 MB	1411kbps	wave

**Figure (4.6)** Histogram of original and cipher audio signal;

a) audio signal #1, b) audio signal #2.

Encrypted data is represented as numbers in a histogram in the encryption operations. If the distributions of these numbers are somewhat close, then the performance of the encryption process is good as shown in figure (4.6). the histogram of the ciphered audio signal using the proposed IChacha20 algorithm is significantly different from the original audio signal, and it is completely uniform. This means that the proposed IChacha20 obtains a very good quality of encryption which explains the high-security level.

The value of the correlation coefficient appeared for the first audio sample by using the IChacha 20 algorithm (0.000521), while the value of the correlation coefficient is for the second audio sample encoded using the IChacha 20 algorithm (0.001071). As for the value of the SNR metric for the first audio sample by using the IChacha 20 algorithm (1.7969), While the value of the correlation coefficient is for the second audio sample encoded using the IChacha 20 algorithm (1.957).

4.5 Analysis

Testing the randomness of keys sequence is an essential part of security evaluation for a cryptographic system. To test the randomness of the keys that are generated by the proposed IChacha20 algorithm, the NIST (National Institute of Standards and Technology) package is used. The results for (11) statistical tests of the NIST are shown in table (4.8).

Table (4.8) The statistical Tests of NIST on the keys of the proposed IChacha20 algorithm

Test Name	Key 128bit	Key 255bit	Key 512bit	Key 1024bit	Key 2,048bit
Block-frequency	0.215925	0.859684	0.376759	0.157299	1
Cumulative-sums	0.627432	0.744238	0.627432	0.627432	0.265697
FFT	0.95949	0.95949	0.115349	0.387868	0.95949
Frequency	0.49092	0.49092	0.234115	0.415592	0.661128
Lempel-Z	0.676064	0.674771	0.67823	0.676933	0.676933
Linear-complexity	1	1	1	1	1
Longest-run	1	1	1	1	1
Nonperiodic- templates	0.999987	0.999987	0.999987	0.999987	0.044965
Overlapping- templates	1	1	1	1	1
Runs	0.552229	0.035714	0.546147	0.600654	0.088358
Serial	0.84229	0.922714	0.157711	0.498531	0.498961

Chapter Five

Conclusions and Suggestions for Future Works

5.1 Conclusions

The conclusions can be drawn from the results and tests of this work as follows:

- 1- The proposed system has been built using Chacha20 based on chaotic functions (Tent maps and Chebyshev maps).
- 2- The Chacha20 input has been containing four constants, a key, a nonce, and a counter, which had entered manually, while the proposed IChacha20 had inputs randomly generated in one stage.
- 3- IChacha20 generated a high random keystream because of the output of chaotic functions.
- 4- The proposed algorithm can generate very sensitive keys to change in the initial values with higher diffusion.
- 5- The keys were tested using the NIST metric, this metric showed that the resulting key is more random, which gives the proposed algorithm more security in encrypting data.
- 6- The proposed IChacha20 encryption algorithm having generated a secret key that completely different from the original algorithm based on the correlation coefficient metric.
- 7- The correlation coefficient metric has gotten the best value using IChacha20 = 0.617 with text size =77 char.
- 8- The results had provided a histogram to all images that ciphered using proposed IChacha20 more homogeneous and uniform from images that ciphered using original Chacha20 as shown in table (4.6).

- 9- The proposed IChacha20 had obtained the best values of metrics for all testing image samples. The best value of MSE =9516.256, the best value of PSNR= 8.346, the best value of NCC =0.437, and finally, the best value of UQI =0.511, while the original chacha20 had obtained values of metrics for all testing image samples using the same metrics. The best value of MSE =9511.894, the best value of PSNR= 8.348, the best value of NCC =0.438, and finally, the best value of UQI =0.511. This indicated that the performance of the proposed algorithm is better than the original algorithm.
- 10-The proposed IChacha20 is faster than the original Chacha20, where the best value of execution time using the proposed IChacha20 had been obtained in the cipher process was(01:26:4sec) while the best value of execution time using original Chacha20 had been obtained in the cipher process was(02:07:1sec).
- 11-The audio samples that have been ciphered using the proposed IChacha20 obtained a completely different histogram from that of the audio samples that have been ciphered using the original Chacha20, They are completely uniform.
- 12-The proposed IChacha20 has obtained the best value for all ciphered audio samples. Where it was the best value of correlation analysis=0.000521 and the best value of SNR = 1.7969.

5.2 Suggestions For Future Work

In this work, several subjects have been specified that will provide significant support in the field of using Chacha20:

1. Developing a new different method for information and communication security using lightweight Chacha20 with another chaotic map functions such as: (Symplectic, Tangent, Tinkerbell, and Interval exchange map).

2. Applying the proposed system using Chacha20 based on DNA sequence and multi-chaotic maps.
3. Employing the proposed system in blockchain (encryption of database).
4. Using an attack, such as a keys recovery attack, to the proposed IChacha20 method for measuring the strength of a cipher.

References

- [1] D. I George Amalarethinam, J. Sai Geetha, and K. Mani, "Add-on Security Level for Public Key Cryptosystem using Magic Rectangle with Column/Row Shifting," in *IJCA*, 2014, vol. 96, no. 14, pp. 38–43.
- [2] X. Yue, W. Chen, and Y. Wang, "The research of firewall technology in computer network security," in *2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA)*, 2009, vol. 2, pp. 421–424.
- [3] S. Tayal, N. Gupta, P. Gupta, D. Goyal, and M. Goyal, "A Review paper on Network Security and Cryptography," *Adv. Comput. Sci. Technol.*, vol. 10, no. 5, pp. 763–770, 2017.
- [4] J. F. Dooley, *History of cryptography and cryptanalysis: Codes, Ciphers, and their algorithms*. Springer, 2018.
- [5] R. Girija and H. Singh, "A new substitution-permutation network cipher using Walsh Hadamard Transform," in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, 2017, pp. 168–172.
- [6] D. A. F. Saraiva, V. R. Q. Leithardt, D. de Paula, A. Sales Mendes, G. V. González, and P. Crocker, "Prisec: Comparison of symmetric key algorithms for iot devices," *Sensors*, vol. 19, no. 19, p. 4312, 2019.
- [7] D. J. Bernstein, "ChaCha, a variant of Salsa20," in *Workshop Record of SASC*, 2008, vol. 8, pp. 3–5.
- [8] S. Maitra, "Chosen IV cryptanalysis on reduced round ChaCha and Salsa," *Discret. Appl. Math.*, vol. 208, pp. 88–97, 2016.
- [9] R. Sobti and G. Ganesan, "Analysis of Quarter Rounds of Salsa and ChaCha Core and Proposal of an Alternative Design to Maximize Diffusion (Only Abstract)," 2016.
- [10] A. R. Choudhuri and S. Maitra, "Differential Cryptanalysis of Salsa and ChaCha-An Evaluation with a Hybrid Model," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 377, 2016.
- [11] S. Dey and S. Sarkar, "Improved analysis for reduced round Salsa and Chacha," *Discret. Appl. Math.*, vol. 227, pp. 58–69, 2017.
- [12] P. A. Babu and J. J. Thomas, "Freestyle, a randomized version of ChaCha for resisting offline brute-force and dictionary attacks," *arXiv Prepr. arXiv1802.03201*, 2018.
- [13] Y. Matsuoka and A. Miyaji, "Revisited Diffusion Analysis of Salsa and ChaCha," in *2018 International Symposium on Information Theory and Its Applications (ISITA)*, 2018, pp. 452–456.

- [14] P. McLaren, W. J. Buchanan, G. Russell, and Z. Tan, "Deriving ChaCha20 key streams from targeted memory analysis," *J. Inf. Secur. Appl.*, vol. 48, p. 102372, 2019.
- [15] S. Dey, T. Roy, and S. Sarkar, "Revisiting design principles of Salsa and ChaCha.," *Adv. Math. Commun.*, vol. 13, no. 4, 2019.
- [16] S. Dey and S. Sarkar, "Proving the biases of Salsa and ChaCha in differential attack," *Des. Codes Cryptogr.*, pp. 1–30, 2020.
- [17] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS One*, vol. 11, no. 6, p. e0155781, 2016.
- [18] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson, "ChaCha20-Poly1305 cipher suites for transport layer security (TLS)," *RFC 7905*, no. 10, 2016.
- [19] J. M. Al-Tuwaijari, "Image Encryption Based on Fractal Geometry and Chaotic Map," *Diyala J. Pure Sci.*, vol. 14, no. 1-Part 1, pp. 166–182, 2018.
- [20] H. A. Ismael, J. M. Abbas, S. A. Mostafa, and A. H. Fadel, "An enhanced fireworks algorithm to generate prime key for multiple users in fingerprinting domain," *Bull. Electr. Eng. Informatics*, vol. 10, no. 1, pp. 337–343, 2020.
- [21] J. M. Al-Tuwaijari, "Multi-Cipher Technique based on RNA and Chebyshev Map," *Iraqi J. Inf. Technol.*, vol. 7, no. 1, pp. 114–125, 2015.
- [22] M. A. Khan and V. Jeoti, "On the enlargement of robust region of chaotic tent map for the use in key based substitution-box (S-Box)," *J. Comput. Sci.*, vol. 11, no. 3, p. 517, 2015.
- [23] S. Dustdar and W. Schreiner, "A survey on web services composition," *Int. J. web grid Serv.*, vol. 1, no. 1, pp. 1–30, 2005.
- [24] H. Kreger, "Web services conceptual architecture (WSCA 1.0)," *IBM Softw. Gr.*, vol. 5, no. 1, pp. 6–7, 2001.
- [25] S. A. Hamid, R. A. Abdalrahman, I. A. Lafta, and I. Al Barazanchi, "Web Services Architecture Model to Support Distributed Systems," *J. Southwest Jiaotong Univ.*, vol. 54, no. 6, 2019.
- [26] D. A. Chappell and T. Jewell, *Java web services*. Tecniche Nuove, 2002.
- [27] A. Nassiry and A. Kardan, "Grid learning; computer grid joins to e-learning," *World Acad. Sci. Eng. Technol.*, vol. 49, pp. 280–284, 2009.
- [28] H. Paik, A. L. Lemos, M. C. Barukh, B. Benatallah, and A. Natarajan, *Web Service Implementation and Composition Techniques*, vol. 256. Springer, 2017.
- [29] A. W. Mohamed and A. M. Zeki, "Web services SOAP optimization techniques," in *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 2017, pp. 1–5.

- [30] B. Rashidi and C. J. Fung, "A Survey of Android Security Threats and Defenses.," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 6, no. 3, pp. 3–35, 2015.
- [31] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastri, "Practical and lightweight domain isolation on android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 51–62.
- [32] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 235–245.
- [33] M. Hussain *et al.*, "Conceptual framework for the security of mobile health applications on android platform," *Telemat. Informatics*, vol. 35, no. 5, pp. 1335–1354, 2018.
- [34] W. Enck, "Defending users against smartphone apps: Techniques and future directions," in *International Conference on Information Systems Security*, 2011, pp. 49–70.
- [35] E. B. Smid, S. Leigh, M. Levenson, M. Vangel, A. DavidBanks, and S. JamesDray, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *Her Res. Interes. Incl. Comput. Secur. Secur. Oper. Syst. Access Control. Distrib. Syst. Intrusion Detect. Syst.*
- [36] K. M. Sundaram, T. Santhanam, M. Saroja, and C. P. Sumathi, "A performance analysis of modified mid-square and mid-product techniques to minimize the redundancy for retrieval of database records," *J. Comput. Sci.*, vol. 6, no. 4, p. 386, 2010.
- [37] A. B. Dhole and N. J. Janwe, "An Implementation of Algorithms in Visual Cryptography in Images," *Int. J. Sci. Res. Publ.*, vol. 3, no. 3, pp. 1–5, 2013.
- [38] M. E. Hodeish and V. T. Humbe, "An optimized halftone visual cryptography scheme using error diffusion," *Multimed. Tools Appl.*, vol. 77, no. 19, pp. 24937–24953, 2018.
- [39] M. J. Barani, M. Y. Valandar, and P. Ayubi, "A new digital image tamper detection algorithm based on integer wavelet transform and secured by encrypted authentication sequence with 3D quantum map," *Optik (Stuttg.)*, vol. 187, pp. 205–222, 2019.
- [40] N. S. Mohammed, Z. T. M. Al-Ta, and S. S. Mohammed, "An Enhancement of LSB Audio Steganography Using Magic Cube," *Diyala J. Pure Sci.*, vol. 15, no. 03, pp. 88–102, 2019.
- [41] Y. Wang, K.-W. Wong, X. Liao, and G. Chen, "A new chaos-based fast image encryption algorithm," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 514–522, 2011.

الخلاصة

النظام المقترح تم بناءه باستخدام تشاتشا ٢٠ (Chacha20) بالاعتماد على الخرائط الفوضوية (Tent maps and Chebyshev functions). يتكون النظام المقترح من ثلاث مراحل واختبار NIST: مرحلة الخرائط الفوضوية , مرحلة توليد المفاتيح الفوضوية ومرحلة عملية التشفير / فك التشفير IChacha20.

المرحلة الاولى تضمنت إنشاء مصفوفة أولية بحجم $[16 * 16]$ مع عناصر عشوائية منتقاة من المصفوفة الأولية, تضمنت المرحلة الثانية إنشاء مدخلات لمصفوفة تسمى Chacha x-matrix مع تنفيذ ما يسمى ربع الجولة الذي يمثل النموذج الرياضي للخوارزمية وأخيرًا , المرحلة الثالثة تضمنت عملية تشفير / فك تشفير IChacha20. أثبتت اختبارات حزمة المعهد الوطني للمعايير والتكنولوجيا (NIST) أن المفاتيح التي تم إنشاؤها بواسطة خوارزمية تشاتشا ٢٠ المحسنة (IChacha20) عشوائية وغير متوقعة , لذا فهي قوية ضد الهجمات , حيث اجتازت مفاتيح تشاتشا ٢٠ المحسنة (IChacha20) معظم اختبارات (NIST) و بمعدلات نجاح عالية.

تم تطبيق خوارزمية تشاتشا 20 المحسنة (IChacha20) على وسائط متعددة , بما في ذلك النص , واستخدمنا مقياس معامل الارتباط وتم الحصول على افضل النتائج. كما أظهرت النتائج قيم مقياس تشفير الصور باستخدام الخوارزمية المقترحة وكانت أفضل من القيم التي تم الحصول عليها باستخدام الخوارزمية الأصلية. بالإضافة إلى الإشارة الصوتية , فقد قدمت النتائج قيم المقياس الخاصة بتشفير عينات الصوت باستخدام تشاتشا 20 المحسنة (IChacha20) , حيث ظهرت أفضل النتائج لمقياس كل من معامل الارتباط ونسبة الإشارة إلى الضوضاء.

حصلت تشاتشا ٢٠ المحسنة (IChacha20) على وقت تنفيذ اسرع من تشاتشا ٢٠ (Chacha20) الأصلية , حيث كان وقت تنفيذ تشاتشا ٢٠ (Chacha20) الأصلية = (02:07.1) ثانية, بينما كان وقت تنفيذ تشاتشا ٢٠ المحسنة (IChacha20) = (01:26.4) ثانية.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة ديالى
كلية العلوم



تحسين خوارزمية تشاتشا (Chacha) في الاتصالات المتنقلة

رسالة

مقدمة الى قسم علوم الحاسوب /كلية العلوم /جامعة ديالى وهي جزء

من متطلبات نيل درجة الماجستير في علوم الحاسوب

من قبل

مصطفى حسين طه

بإشراف

الأستاذ المساعد الدكتور جمال مصطفى عباس