A Data Structure Encryption Algorithm based on Circular Queue to Enhance Data Security

Ali N. Albu-Rghaif
Department of Computer Engineering,
College of Engineering, University of
Diyala,Iraq
ali.alburghaif@yahoo.com

Abbood Kirebut Jassim
Department of Computer Science,
College of Science for women,
University of Baghdad, Iraq
zmar5151@yahoo.com

Ali J. Abboud
Department of Computer Engineering,
College of Engineering, University of
Diyala, Iraq
ali.j.abboud@gmail.com

***Abstract*— Data security is an ongoing challenge for developers and hackers. To combat different attacks by hackers, there is a necessity for more reliable security technologies. In this research, a low complexity circular queue data structure security algorithm is developed. Employing multiple complicating variable factors is the strength of this algorithm and makes recovery of original message by attackers more difficult. These tuneable factors are the size of the circular queue, the beginning of the chosen keyword letter and the multiple representations of a number in the Fibonacci format. All letters should be converted into ASCII binary format in order to be used by security algorithm in the logical and shift operations. The results show that our proposed security algorithm has 50% low complexity than compared multiple circular queues algorithm (MCQA). In addition, the Fibonacci format and variable number of challenging factors in this algorithm provide flexibility in changing the security of the algorithm according to the circumstances.**

Keywords— Data Security, Circular Queue, Fibonacci format and complicating factors.

I. Introduction

Data security is one of ultimate crucial topics in the networked community. The importance of this subject belongs to several modern issues including different kinds attacks in cyber networking environment, breaching the privacy of users and increasing usage of internet for electronic transactions [1]. The main tools of data security are cryptography, steganography, watermarking and data integrity algorithms. Firstly, cryptography algorithms are classified into symmetric and asymmetric. The symmetric algorithms use single common key for both sender and receiver while asymmetric algorithms use two keys for each user. Public key cryptography is an example of asymmetric algorithms that involve use public and private keys. Secondly, steganography is another valuable tool to hide information inside a cover carrier to protect information. Thirdly, watermarking is the security apparatus to check the authenticity of the information. Lastly, data integrity algorithms represent the tools that test the integrity of the data. They include hash function, message authentication codes (MAC) and digital signatures [2, 3]. To sum up, information security plays a major role in protection and authenticity of data and applications running in computer networks. It allows people to communicate or transfer data electronically without worries of deception. In addition, ensure the integrity of the message and authenticity of the sender [4]. Furthermore, this paper presents data security algorithm which is based on the circular queue data structure and multiple challenging factors. It can be applied to several applications including communication networks, messaging services, mobile applications. The rest of this paper is organized as follows. Section II summarizes encryption and decryption algorithms. Section III shows the structure of our proposed model of using circular queue to enhance information security. Section IV presents the experimental results and analysis. Section V is used to explain conclusions and future work.

II. Related Work

Circular queue is a data structure can be employed in the information security to make ciphered message more difficult to decipher. For instance, the authors of paper [5] developed an algorithm that uses the shifting and replacing operations of bi-column-bi-row for circular queue to increase security. A random number was used in this algorithm to control the shifting between the row and column, eventually this lead to increase the complexity of plaintext decryption. In the same vein, an elliptic curve algorithm was designed based on matrix scrambling using circular queue [6]. In this research also utilizes shifting process to accomplish the encryption and the decryption of the text. In addition, a multiple circular arrays algorithm was developed to encrypt data using three circular arrays. This algorithm enabled the shifting (elements in the outer or inner array), swapping (elements among the circular arrays) and XORing (for encrypting the text) based on generating random number [7]. In contrast, a double encryption double decryption technique is proposed, which means the transmitter encrypts the text two times that leads the receiver decrypt the cipher text twice using public key [8]. Also, an elliptic curve algorithm is developed to produce a cipher text [9]. Actually, in this work the text firstly formed into ASCII code, and then the prime number and random number are chosen and formed into binary format. Where the “0” representation of the prime number is responsible of shifting the row/column in upward and left respectively. In

978-1-5386-1498-3/ 18/31.00$©2018 IEEE

addition, a multiple access circular queues algorithm is proposed with variable length in [10]. In this work; different numbers of rotations are applied to the circular queues, swapping the elements in the same queue and XORing the elements with generating key number. The authors recommended that these processes would make a secure plaintext over the transmission line. On the other hand, Fibonacci sequence is mostly used for image encryption. A text to image encryption algorithm is designed using Fibonacci sequence [11]. This algorithm firstly converts the plaintext using Fibonacci sequence, and then the Unicode is converted to hexadecimal number and a RGB matrix. Finally, a shuffling operation is made to obtain the image to be sent.

III. Proposed Algorithm

In this section, the proposed algorithm is illustrated thoroughly. It is based mainly on employing a circular queue data structure in encryption and decryption processes. At the beginning, a circular queue that used in our proposed algorithm is shown in Fig 1. The "n" in this figure refers to the size of the circular queue and the plaintext is shown inside the circular queue. The letters in the core of the circle represents the keyword letters to be shifted to the right and left and then XORed with plaintext to obtain ciphertext.

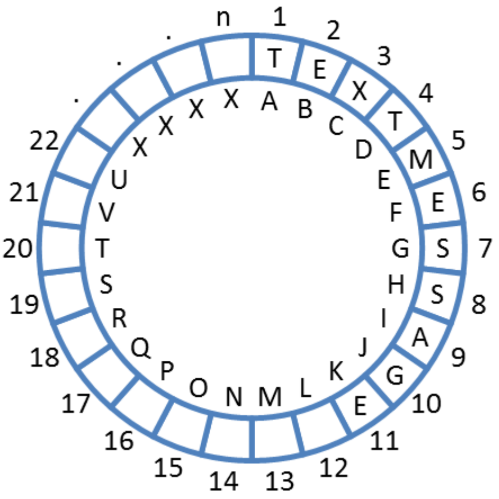

Fig. 1 Circular Queue for Encryption and Decryption Process

Using circular queue in this research provides several factors that make the encryption/decryption process more difficult for eavesdroppers to decrypt the ciphertext. Moreover, these factors are agreed upon by both sender and receiver before the encryption process. These factors can be summarized as follows:

- The size of the circular queue is variable.
- The beginning of the keyword letter is variable.
- The representation number in the Fibonacci format.

A. *Encryption Process*

The encryption process begins by distributing plaintext letters in the circular queue. Then these letters and keyword letters are converted to their equivalent 8 bits ASCII code and XORed with each other. After that, the resultants are represented as decimal numbers. Finally, these numbers are demonstrated into Fibonacci format to be sent as a cipher text, as depicted in Fig. 2.

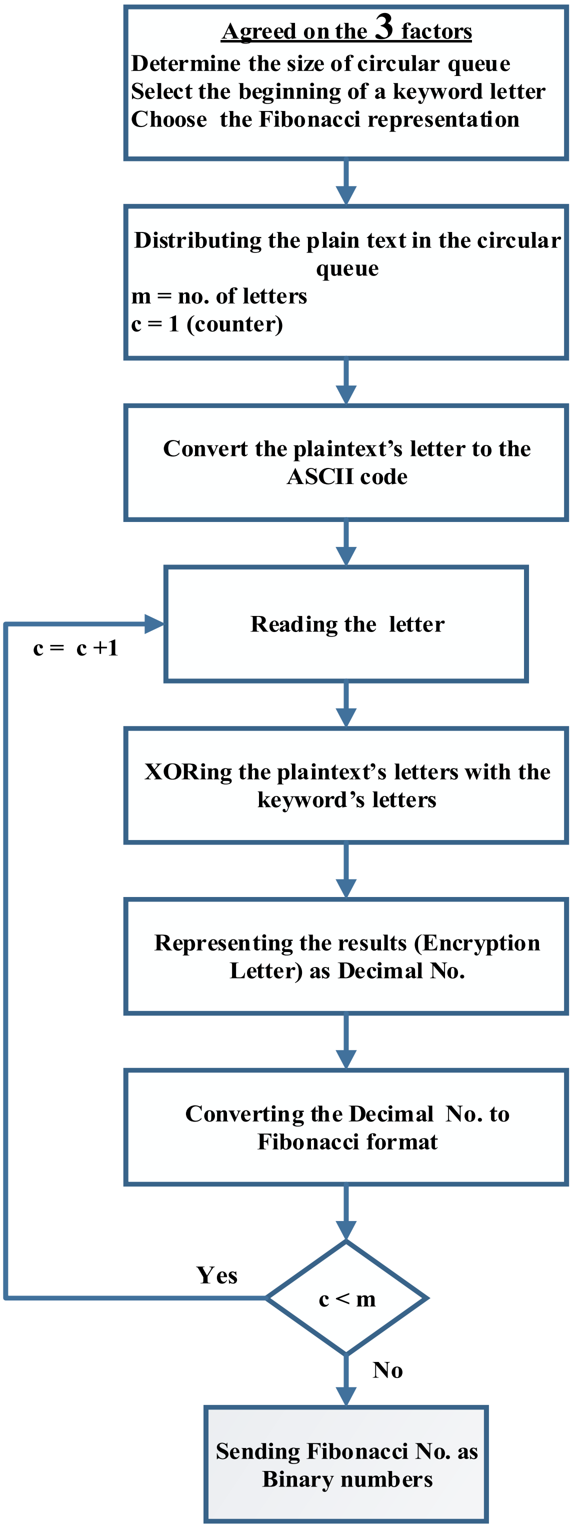
Agreed on the 3 factors
Determine the size of circular queue
Select the beginning of a keyword letter
Choose the Fibonacci representation
Distributing the plain text in the circular queue
m = no. of letters
c = 1 (counter)
Convert the plaintext's letter to the ASCII code
Reading the letter
c = c +1
XORing the plaintext's letters with the keyword's letters
Representing the results (Encryption Letter) as Decimal No.
Converting the Decimal No. to Fibonacci format
Yes
c < m
No
Sending Fibonacci No. as Binary numbers

Fig. 2 Encryption Process

Fig. 3 shows the distribution of the plaintext in the circular queue. The encryption process in this data structure starts with XORing the plaintext letters with the keyword letters that started with letter” E”.

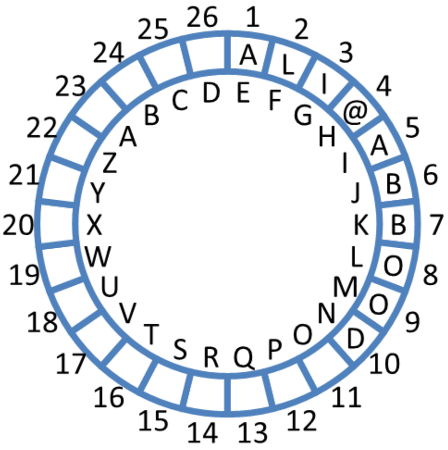

Fig. 3 Circular Queue Example

Given example below illustrates the encryption process in a number of steps. Let take the first name of the authors as a plaintext “ALI@ABBOOD”. As shown in Fig. 2 earlier, the sender and receiver should agree on three factors. The values of these factors in this example as follows:

factor1: the size of circular queue is **26** as shown in Fig. 3.

factor2: the keyword letter started with letter “**E**” as shown in Fig.3.

factor3: the representation of the Fibonacci number is the 1st representation as shown in Table I. Also, this table shows the first ten element of Fibonacci number in two representations to demonstrate that a number can have several representations.

1st representation of number 5 is: 1 0 0 0 0
2nd representation of number 5 is: 1 1 0 0

TABLE I. FIBONACCI SEQUENCE

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
1	1	2	3	5	8	13	21	34	55

The steps of an example are:

Step 1: convert the plaintext and keyword letters into the ASCII code as shown below in Table II for plaintext letters.

TABLE II. REPRESENTATION THE PLAINTEXT IN ASCII CODE

Letters	ASCII code							
A	0	1	0	0	0	0	0	1
L	0	1	0	0	1	1	0	0
I	0	1	0	0	1	0	0	1
@	0	1	0	0	0	0	0	0
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
B	0	1	0	0	0	0	1	0
O	0	1	0	0	1	1	1	1
O	0	1	0	0	1	1	1	1
D	0	1	0	0	0	1	0	0

Step 2: XORing the plaintext letters with keyword letters.

Plain letter	**A**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
keyword letter	***E***	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**
Encrypted letter		**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
	L	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
	F	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
	I	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**
	G	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**
	@	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
	H	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
	A	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
	I	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
	B	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**
	J	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
	B	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**
	K	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**
	O	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
	L	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
Encrypted letter		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
	O	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
	M	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**
Encrypted letter		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
	D	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
	N	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**
Encrypted letter		**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**

Step 3: Represent the encrypted outputs as decimal numbers.

Encrypted letter								**Decimal Number**
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**4**
0	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**10**
0	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**14**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**8**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**8**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**8**
0	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**9**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**3**
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**2**
0	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**10**

Step 4: This step is concerned with converting the decimal numbers into Fibonacci format to be sent as binary numbers.

Decimal number	Fibonacci format							
	21	13	8	5	3	2	1	1
4	0	0	0	0	1	0	0	1
10	0	0	1	0	0	1	0	0
14	0	1	0	0	0	0	0	1
8	0	0	1	0	0	0	0	0
8	0	0	1	0	0	0	0	0
8	0	0	1	0	0	0	0	0
9	0	0	1	0	0	0	0	1
3	0	0	0	0	1	0	0	0
2	0	0	0	0	0	1	0	0
10	0	0	1	0	0	1	0	0

Step 5: The final step is to send the Fibonacci numbers as binary numbers that are represented in decimal as shown below.

Fibonacci numbers (AS) binary numbers								Decimal Number
0	0	0	0	1	0	0	1	9
0	0	1	0	0	1	0	0	36
0	1	0	0	0	0	0	1	65
0	0	1	0	0	0	0	0	32
0	0	1	0	0	0	0	0	32
0	0	1	0	0	0	0	0	32
0	0	1	0	0	0	0	1	33
0	0	0	0	1	0	0	0	8
0	0	0	0	0	1	0	0	4
0	0	1	0	0	1	0	0	36

B. *Decryption Process*

The decryption process is the reverse operation of encryption to recover the original message. After receiving the ciphertext in Fibonacci format, it converted back to the decimal number as shown in Fig. 4. Then, these numbers are XORed with keyword letter according to their locations. Eventually, the original message is restored. The key point in this process is to perform the conversion process of the Fibonacci number carefully. To demonstrate the decryption process, we have used encryption letters of preceding example. The steps of an example are:

Step 1: convert the received encrypted message (Fibonacci number) to decimal number.

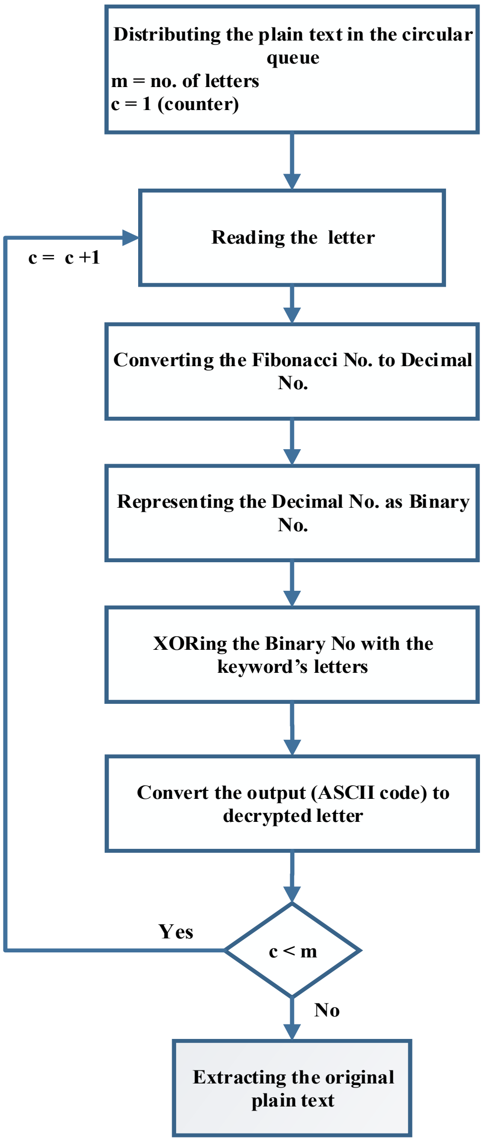
Distributing the plain text in the circular queue
m = no. of letters
c = 1 (counter)
Reading the letter
c = c +1
Converting the Fibonacci No. to Decimal No.
Representing the Decimal No. as Binary No.
XORing the Binary No with the keyword's letters
Convert the output (ASCII code) to decrypted letter
Yes
c < m
No
Extracting the original plain text

Fig. 4 Decryption Process

Fibonacci numbers								Decimal Number
21	13	8	5	3	2	1	1	
0	0	0	0	1	0	0	1	4
0	0	1	0	0	1	0	0	10
0	1	0	0	0	0	0	1	14
0	0	1	0	0	0	0	0	8
0	0	1	0	0	0	0	0	8
0	0	1	0	0	0	0	0	8
0	0	1	0	0	0	0	1	9
0	0	0	0	1	0	0	0	3
0	0	0	0	0	1	0	0	2
0	0	1	0	0	1	0	0	10

Step 2: convert the decimal number to binary format.

Decimal number	Binary format							
4	0	0	0	0	0	1	0	0
10	0	0	0	0	1	0	1	0
14	0	0	0	0	1	1	1	0
8	0	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1
3	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0
10	0	0	0	0	1	0	1	0

Step 3: XORing the binary numbers with the keyword letters.

Binary No.	**4**	0	0	0	0	0	1	0	0
keyword letter	***E***	0	1	0	0	0	1	0	1
Decryption letter		0	1	0	0	0	0	0	1
	10	0	0	0	0	1	0	1	0
	F	0	1	0	0	0	1	1	0
Decryption letter		0	1	0	0	1	1	0	0
	14	0	0	0	0	1	1	1	0
	G	0	1	0	0	0	1	1	1
Decryption letter		0	1	0	0	1	0	0	1
	8	0	0	0	0	1	0	0	0
	H	0	1	0	0	1	0	0	0
Decryption letter		0	1	0	0	0	0	0	0
	8	0	0	0	0	1	0	0	0
	I	0	1	0	0	1	0	0	1
Decryption letter		0	1	0	0	0	0	0	1
	8	0	0	0	0	1	0	0	0
	J	0	1	0	0	1	0	1	0
Decryption letter		0	1	0	0	0	0	1	0
	9	0	0	0	0	1	0	0	1
	K	0	1	0	0	1	0	1	1
Decryption letter		0	1	0	0	0	0	1	0
	3	0	0	0	0	0	0	1	1
	L	0	1	0	0	1	1	0	0
Decryption letter		0	1	0	0	1	1	1	1
	2	0	0	0	0	0	0	1	0
	M	0	1	0	0	1	1	0	1
Decryption letter		0	1	0	0	1	1	1	1
	10	0	0	0	0	1	0	1	0
	N	0	1	0	0	1	1	1	0
Decryption letter		0	1	0	0	0	1	0	0

Step 4: convert the XORing output which represent the ASCII code of the plaintext letters to original message as illustrated in Table III.

TABLE III. REPRESENTATION OF THE ASCII CODE OUTPUT INTO LETTERS

XORing Output (ASCII code)								Letter
0	1	0	0	0	0	0	1	A
0	1	0	0	1	1	0	0	L
0	1	0	0	1	0	0	1	I
0	1	0	0	0	0	0	0	@
0	1	0	0	0	0	0	1	A
0	1	0	0	0	0	1	0	B
0	1	0	0	0	0	1	0	B
0	1	0	0	1	1	1	1	O
0	1	0	0	1	1	1	1	O
0	1	0	0	0	1	0	0	D

IV. SECURITY ANALYSIS

In order to analysis the security performance of our encryption/decryption algorithm, let assume eavesdroppers have the ciphertext as illustrated in the example. They try to decrypt the message by converting these binary digits to the ASCII code. Hence, the cracked result as follows:

Original Letters	Output Message								ASCII Code
A	0	0	0	0	1	0	0	1	HT
L	0	0	1	0	0	1	0	0	$
I	0	1	0	0	0	0	0	1	A
@	0	0	1	0	0	0	0	0	Space
A	0	0	1	0	0	0	0	0	Space
B	0	0	1	0	0	0	0	0	Space
B	0	0	1	0	0	0	0	1	!
O	0	0	0	0	1	0	0	0	BS
O	0	0	0	0	0	1	0	0	EOT
D	0	0	1	0	0	1	0	0	$

The recovered message by eavesdroppers is totally different from original message and such result proves difficulty of some kinds of attacks such as statistical and linear. Such achievement of difficulty is attributed to the challenging factors of variable queue size, variable key size and different Fibonacci format representations. Consequently, our proposed algorithm is satisfying required confidentially by employing variable size factors. Moreover, Fibonacci format adds more complexity to the decryption process. Furthermore, Table IV demonstrates another example of our encryption algorithm, where the symbol "**…**" is an extended of the ASCII code and has no description.

TABLE IV. OUR ALGORITHM'S EXAMPLE

Plain-text	**Beginning of Keyword Letter**	**Cipher-text**
HELLOCIPHER	B	$ DC1 Space ! ! HT SOH ... EOT $ i
CRYPTOGRAPHY	D	DC4 ... ¤ ... ... DC1 @ ... @ ... DC1 ...

However, there is exceptional difficult case if we have the symbol ">" to be encrypted. When this symbol is XORed with keyword letter "A", the length of encrypted letter will be sixteen bits instead of eight bits. This issue is solved by adding eight bits (one byte) at the end of most significant bit as illustrated below.

>	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**0**
A	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
output	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

The decimal representation of the XORing output is the number 127. Then the Fibonacci format of resultant decimal number is shown below, and these two bytes represent "ENQ" and "I" respectively.

0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 1

It is worthwhile to mention the comparison of complexity of our algorithm with other algorithm such as multiple access circular queues "MACQ" [10]. As shown in the Table V the breakdown complexity of our algorithm.

TABLE V. BREAKDOWN COMPLEXITY

Our Algorithm	**Complexity**	**MACQ Algorithm**	**Complexity**
Converting the plaintext to 8 bite	*O* log(n)	Converting the plaintext to 8 bite	*O* log(n)
		XORing with generating key	$O(n^2)$
XORing with keyword letters	$O(n^2)$	XORing the key with innermost queue	$O(n^2)$
		XORing the key with second innermost queue	$O(n^2)$
Converting the output to decimal number	*O* log(n)	XORing the circular queues with innermost queue	$O(n^2)$
		Converting to Hexadecimal	*O* log(n)
Converting the decimal number to Fibonacci	*O* log(n)	Applying right shift	*O* (1)
		Swap operation	*O n*log(n)

Where "n" refers to number of bits, whereby the complexity of our algorithm is much less than the MACQ algorithm by around 50%. Such result achieved using less number of operations in our algorithm.

V. CONCLUSIONS AND FUTURE WORK

In this research a new data structure based security algorithm is proposed. The significance of this algorithm is using circular queue and Fibonacci sequence to be sent as a cipher text. The decryption process is challenging due to the usage of variable factors. In addition, our algorithm offers flexible size tunable mechanism. The most important issue is the agreement between sender and receiver to change circular queue size, keyword letter/symbol and the representation of the Fibonacci number before establishment of connection. Furthermore, our algorithm is faster than MACQ algorithm due to low complexity of the encryption/decryption processes. We have to mention that proposed algorithm is used for text messages. However, we will plan to encrypt other types of data such images, voice and video using this algorithm.

REFERENCES

[1] Kahate, Atul., "Cryptography and Network Security", Tata McGraw-Hill Education, 2013.

[2] Ali J. Abboud, "Multifactor Authentication For Software Protection", Diyala Journal of Engineering Sciences, Vol. 08, No. 04, Special Issue, 2015.

[3] Ali J. Abboud ,"Protecting Documents Using Visual Cryptography", International Journal of Engineering Research and General Science ,2015.

[4] E. Barker, W. Barker, W. Burr, W. Polk and M. Smid, "Recommendation for Key Management-Part 1: General (Revision 3)", Computer Security Division (Information Technology Laboratory), 2016.

[5] Wu, Suli, Yang Zhang, and Xu Jing. "A Novel Encryption Algorithm based on Shifting and Exchanging Rule of bi-column bi-row Circular Queue", IEEE International Conference on Computer Science and Software Engineering, Vol. 3. , 2008.

[6] Amounas, Fatima., "An Elliptic Curve Cryptography based on Matrix Scrambling Method", IEEE International Conference on Network Security and Systems (JNS2), 2012.

[7] S. S. D. Pushpa R. Suri, "A Cipher based on Multiple Circular Arrays", International Journal of Computer Science Issues (IJCSI), Vol. 10, No. 5, pp. 165-175, 2013.

[8] Merkle, Ralph C., and Martin E. Hellman. "On the Security of Multiple Encryption", Communications of the ACM, Vol. 24, No.7, 465-467, 1981.

[9] Hankerson, Darrel, Alfred J. Menezes, and Scott Vanstone," Guide to Elliptic Curve Cryptography", Springer Science and Business Media, 2015.

[10] S. Phull and S. Som, "Symmetric Cryptography using Multiple Access Circular Queues (MACQ)", Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, 2016.

[11] P. Agarwal, N. Agarwal and R. Saxena, "Data Encryption Through Fibonacci Sequence and Unicode Characters", MIT International Journal of Computer Science and Information Technology, Vol. 5, No. 2, pp. 79-82, August 2015.