Encrpytion using Circular Queues

January 15, 2024

Sujal Sabavat ,
Shivam Maske

Instructor:
Dr. Anil Shukla

Teaching Assistant:
Vinay

Summary:
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 project, 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 tune-able 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 percent low complexity than compared multiple circular queues algorithm. 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.

1. 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.

2. 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 Figure 1.(a). The "n" in this

figure refers to the size of the circular queue and the plain text 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 plain text to obtain cipher text. Using circular queue in this research provides several factors that make the encryption/decryption process more difficult for eavesdroppers to decrypt the cipher text. 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 plain text 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 Figure 2.(a).

Given example below illustrates the encryption process in a number of steps. Let take the first name of the authors as a plain text "ALI@ABBOOD". As shown in Figure. 2. (a) 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 Figure 1.(b).
factor2: the keyword letter started with letter **"E"** as shown in Figure.1.(b).
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 The steps of an example are:

Step 1: convert the plain text and keyword letters into the ASCII code as shown below in Table II for plain text letters.
Step 2: XORing the plaintext letters with keyword letters.
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 cipher text in Fibonacci format, it converted back to the decimal number as shown in Figure.2.(b). 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.
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.

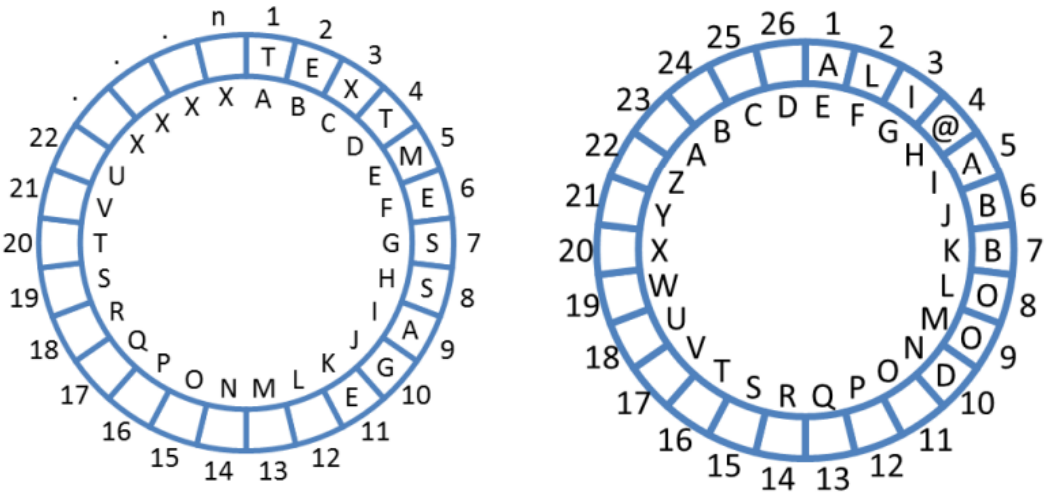
Binary No.	4	0	0	0	0	0	1	0	0
keyword letter	*E*	0	1	0	0	0	1	0	1
Decryption letter		0	1	0	0	0	0	0	1
	10	0	0	0	0	1	0	1	0
	F	0	1	0	0	0	1	1	0
Decryption letter		0	1	0	0	1	1	0	0
	14	0	0	0	0	1	1	1	0
	G	0	1	0	0	0	1	1	1
Decryption letter		0	1	0	0	1	0	0	1
	8	0	0	0	0	1	0	0	0
	H	0	1	0	0	1	0	0	0
Decryption letter		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 plain text letters to original message as illustrated in Table III.

It is worthwhile to mention the comparison of complexity of our algorithm with other algorithm such as multiple access circular queues. As shown in the Table IV.

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	$

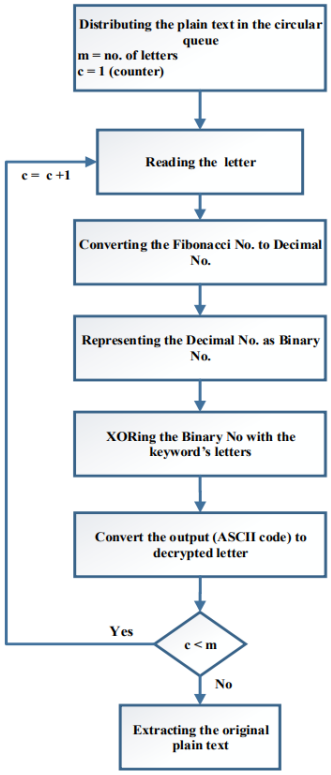
2.1. Figures



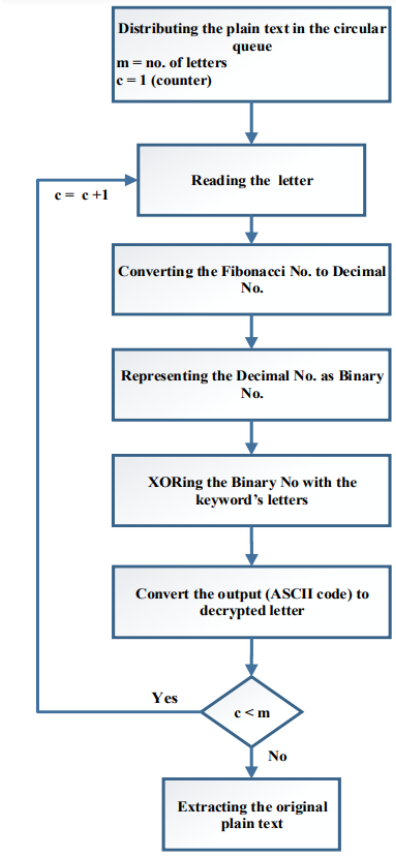
(a) Circular Queue for Encryption and Decryption.

(b) Example of Circular Queue.

Figure 1: Circular Queue Representation



(a) Encryption process.



(b) Decryption Process.

Figure 2: Flowchart Representation of the Algorithm.

2.2. Tables

TABLE I. FIBONACCI SEQUENCE

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
1	1	2	3	5	8	13	21	34	55

TABLE II. PLAIN TEXT TO ASCII CODE

Letter	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

TABLE III. 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

TABLE IV. BREAKDOWN COMPLEXITY

Our Algorithm	Complexity
Converting the plaintext to 8 bite	O log(n)
XORing with keyword letters	O(n^2)
Converting the output to decimal number	O log(n)
Converting the decimal number to Fibonacci	O log(n)

3. Conclusions

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.

4. Bibliography and Reference

[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 "Recommendation for Key Management-Part 1: General (Revision 3)", Computer Security Division (Information Technology Laboratory), 2016.

5. Acknowledgements

We would like to thank Mr. Anil Shukla, our Professor-in-charge and Mr. Vinay, mentoring guide, for the support and guidance in completing our project. Thanking Ali N. Albu-Rghaif, Abbood Kirebut Jassim, Ali J. Abboud for their work on the related topic.