Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 15 (2020) pp. 034-043
© MIR Labs, www.mirlabs.net/jias/index.html

Circular Queue Based Data Encryption Algorithm Using 512 Random Bits

Kamal Kumar Gola^1, Manika Gupta^2 Gulista Khan^3 and Pankaj Rajput^4

^{1} Faculty of Engineering, TMU, Moradabad, 244001, India.
kkgolaa1503@gmail.com

^{2} College of Computing Sciences and Information Technology, TMU, Moradabad, 244001, India.
manikagupta343@gmail.com

^{3} Faculty of Engineering, TMU, Moradabad, 244001, India.
gulista.khan@gmail.com

^{4} College of Computing Sciences and Information Technology, TMU, Moradabad, 244001, India.
pankaj.rajput.552200@gmail.com

***Abstract*: Modern technology is developing in a very linear way. Security and Privacy issues are of most concern nowadays. With this developing technology privacy and data security has become a very major aspect. To prevent data piracy, many different types of encryption techniques and security algorithms have been developed, which ensures data security. We use authentication so that we can easily identify who is using and accessing our data. We use password system in authentication that contains characters and integers that needs to be encrypted with the help of proper algorithm. In present scenario we cannot say that our data is fully secure. We always need new approaches to prevent our data piracy. Here, this work proposes a low complexity encryption and decryption technique based on random bits and circular queue to encrypt the plain text by making changes in their ASCII values and then shifting them according to random numbers generated. Above all random bits are ever changing bits, which makes it very hard to decrypt the cipher text without knowing the correct random bits generated.**

Keywords: Circular queue, 512 random-bits, ASCII values, encryption and decryption.

I. Introduction

Information, according to its definition is 'The meaningful data derived from raw facts' but to us, information is not merely a meaningful data but it's our personal data, official and private data, our bank account details, and much more. And with the modernization and computerization of physical information into digital data, everything in our computers, mobile phones and tablets has become important and sensitive to us. While sending some message or information to someone, we can't afford to disclose or modify our sensitive data, it is imperative to secure the information in our systems and in the network as well from any kind of intrusion, forging or modifications. So, to secure the information, there is a mechanism that can transform a readable message (plaintext) to an unreadable form of message (ciphertext) and that mechanism is called as 'Encryption'. The ciphertext can be transformed back into the plaintext simply by reversing the Encryption mechanism, which is known as 'Decryption'. The process of encryption of encryption and decryption is achieved by some sort of codes, in [1] the authors have proposed a cyclic codes of length pn over Zp3 that can be used in encryption process.

In modern world, Data security is a major aspect to deal with. Cryptography is a technique to prevent unauthorized access to information through codes so that only person intended for information can process it. The prefix "crypt" means "hidden" and suffix "graphy" means "writing" [2]. To prohibit data-piracy, various cryptography, watermark and stenography algorithms have been adopted. These algorithms are being used in various fields like cryptography firm RSA, network security etc. [3] [4]. The proposed algorithm is based on symmetric key encryption that uses circular queue, 512 random bits, which are generated on the basis of whether the random number is even (1) or odd (0), total count of plain text and the ASCII values of inserted plain text. There are three steps of encryption namely key generation, encryption and then shifting of the encrypted text that will be considered as the final encryption of the inserted given plain text.

In first part of encryption 512 bits are generated that are stored into 3 different arrays of various sizes, which will further work as the keys for encryption and decryption of the text. Secondly, we will enter the plain text and store the characters of plain text in a circular queue and then we will start our encryption based on ASCII values of the plain text and 512 generated random bits [5]. Thirdly, shifting of encrypted text takes place based on random numbers and finally we will receive our fully encrypted cipher text. For decryption the same key and the same process is used in reverse direction to get the plain text from the cipher text [6].

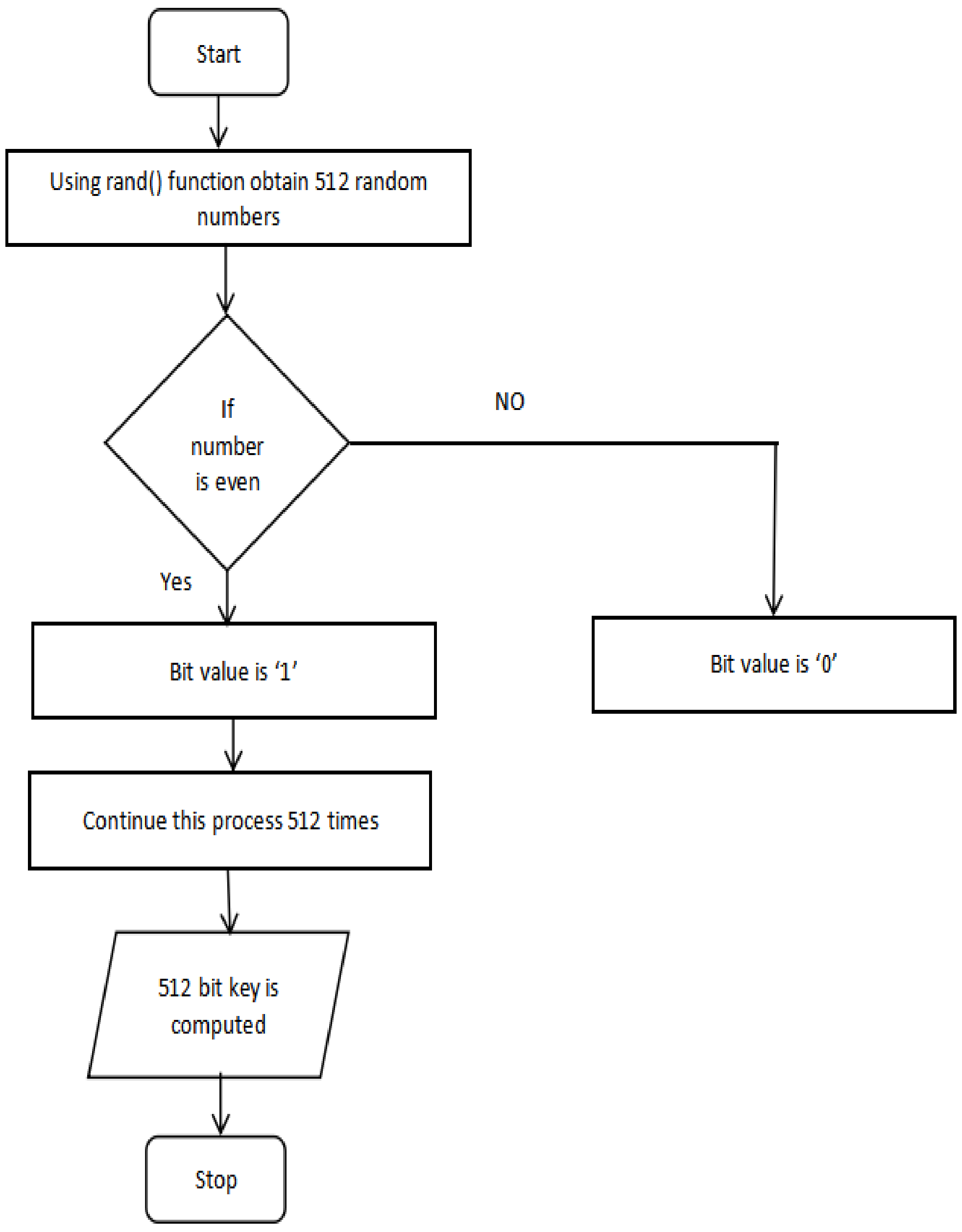
II. Related Work

Using a circular queue gives us the advantage to create a cipher text which is very difficult to decipher. In [7] the authors proposed an algorithm based on shifting and replacing of the circular through bi-column and bi-row to attain more security. Random numbers are used to perform the shifting operation between columns and rows and that leads to increase the complexity of cipher text decryption. In [8] the authors have used the shifting operation to perform the encryption and decryption of the given plain text. In this algorithm data is encrypted using three circular queues which performs shifting, swapping and XORing on the basis of the generated random bits. In [9] an algorithm is proposed with twice the encryption and twice the decryption that means we encrypt the plain text two times and to decrypt the generated encrypted text we have to use the decryption process twice. In [10] the authors firstly converts the given text into their respective ASCII values and then further check whether the value is prime or not and then the prime and random numbers are chosen and form the binary format where ‘0’ shows a prime number responsible for shifting row or column upside or downside. In [11] the author proposed an algorithm that encrypt and decrypt the plaintext based upon the binary conversion. In this algorithm binary numbers are converted from ‘0’ to ‘1’ and ‘1’ to ‘0’.
Authors in [12] examined on symmetric encryption in which the content to be encrypted is first transformed into encapsulation cipher which cannot be recognized by cipher algorithm. Authors in [13] proposed an algorithm in which plain text is first conveyed in circular queue, transformed into 8bit ASCII values, XORed with keyword letter and then converted into respective decimal numbers which is further translated into Fibonacci format to be sent as binary numbers. In [14], authors proposed an algorithm that uses 128/256 bits encryption algorithm that will execute two layer encryption to enhance security and effectiveness.

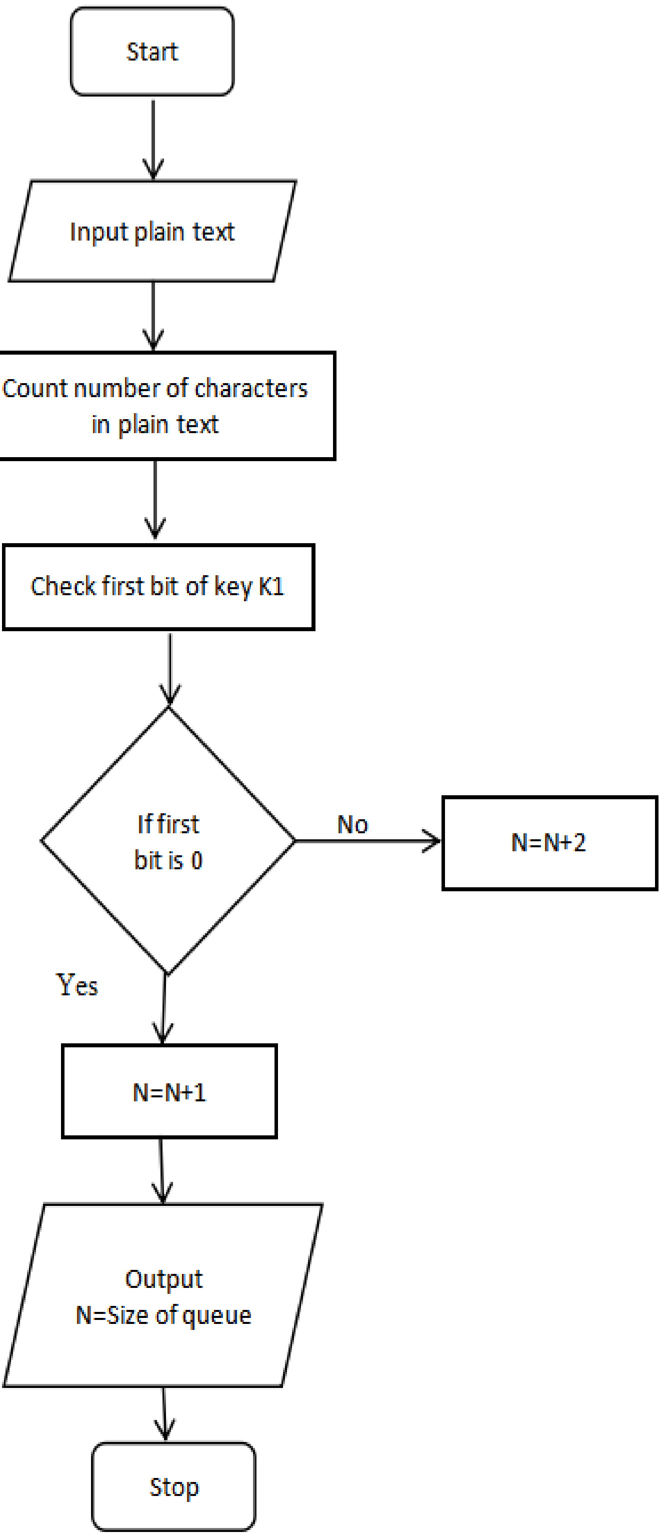
III. Proposed Algorithm with Flowchart

Step 1: Generate 512 random numbers using rand ( ) function.
Step 2: If (random number == even), then bit value =1, else, bit value = 0
Step 3: A series of 512 bits is computed.
Step 4: Divide the keys into three parts K1 with 256 bits, K2 with 128 bits and K3 with 128 bits. After splitting the keys into three parts we check whether the first bit of K1 is either 1 or 0. If it is 0 then we add 1 in the length of the number count of the plain text and, if it is 1 then we add 2 in the length of the number count of the plain text. The number count obtained will become the size of the circular queue.
Step 5: Obtain the plain text and count the number of characters (N) in it including the blank-spaces.
Step 6: Check first bit of key K1.
Step 7: If key_K1 [0] =0, N=N+1, else, N=N+2.
Step 8: Size of circular queue = N.
Step 9: Insert plain text into the queue.
Next we check the bit value from the key K1 to determine the direction to count the position of the letters of the plain text in the circular queue. If the bit value of key K1 is ‘1’ we will count the position of letters in clockwise direction else if it is ‘0’ count the position of letters in anticlockwise direction.
Step 10: Check bit value from key K1 to determine the direction to count the position of letters in the circular queue.
Step 11: Obtain the ASCII value (AV) of the letters in the queue.
Step 12: For i=0 to 255 If (key_K1 [i] ==1), then count the position of a letter in clockwise direction within the queue, else, count the position in the anticlockwise direction.
Step 13: Compute new ASCII value (NAV) =AV + C for every letter and substitute every letter with the letter obtained by value NAV.
Step 14: Repeat step 10, 11, 12 & 13 until the whole text is converted. Before getting the position count of the letters of the plain text in the queue, we first get the ASCII value of the letters. When we get the ASCII values and position count of the letters we add both of them (AV + position count) and obtained a new ASCII value for every letter. We then get the new letter assigned to that ASCII value. Replace that old letter with the new one that we have obtained with the help of new ASCII value. Now, we will check the bit values from the key K2 to determine the number of shifts for the queue. After that we will check the bit values from key K3 to determine the direction of shifts (whether clockwise or anti clockwise). If the first bit key K3 is 0 then we shift the letters in clockwise direction else in anti-clockwise direction. Lastly, we extract the obtained cipher text from the circular queue, which is required encrypted text.
Step 15: Check values of key K2 to determine number of shifts (count) for the queue.
Step 16: Check values of key K3 to determine direction of shifts (clockwise/anti clockwise). For i=0 to count, If (key_K3 [i] ==0), then shift the letters in clockwise direction, else, shift the letters in anti -clockwise direction.
Step 17: Shift the letters in the circular queue using the shift offset and shift direction obtained from step 15 & 16.
Step 18: Extract the cipher text from the circular queue.
For decryption, we insert the cipher text in the circular queue. Now we check the bit values from key K3 to determine the direction of shifts(clockwise / anti-clockwise) and if the first bit of K3 is 0 we shift the letters in anti-clockwise direction else in clockwise direction.
Step 19: Insert cipher text in circular queue.
Step 20: Go to step 15.
Step 21: Check bit values from key K3 to determine direction of shifts (clockwise/anti clockwise).
For i=0 to count, If (key_K3 [i] ==0), then shift the letters in anti-clockwise direction, else, shift the letters in clockwise direction.
Step 22: Shift the letters in the circular queue using the shift offset and shift direction obtained in 20 and 21. Repeating the steps of insertion and encryption compute the new ASCII values of the cipher text for every letter and substitute every letter with the letter obtained by the new ASCII value, and then we finally extract the plain text from the circular queue.
Step 23: Go to step 11 & 12.
Step 24: Compute new ASCII value (NAV) = AV - C for every letter and substitute every letter with the letter obtained by the value NAV.

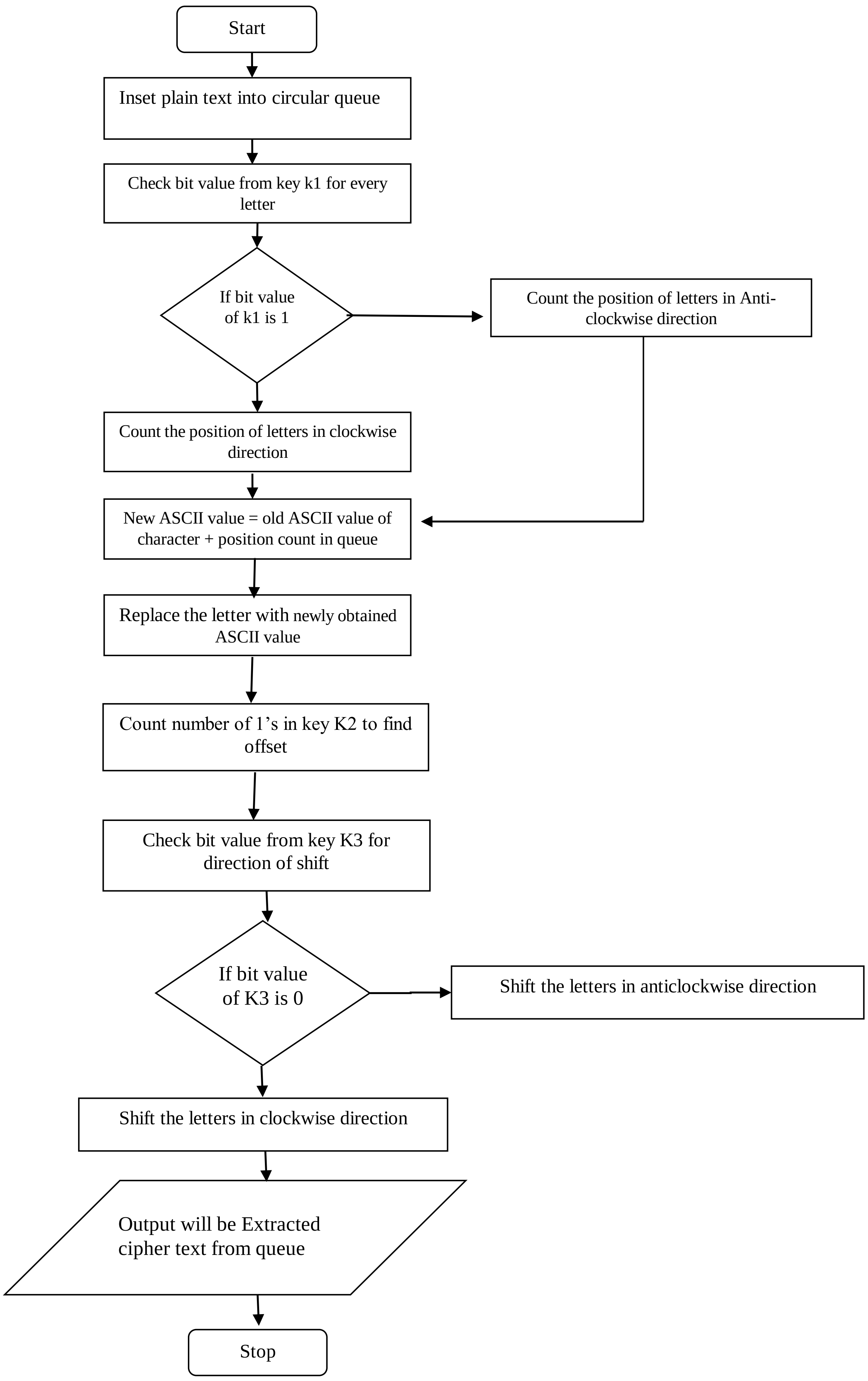
Step 25: Extract the plain text from the circular queue.



Flowchart 1: Key Generation Process



Flowchart 2: Insertion of Plaintext in Circular Queue


Start
Inset plain text into circular queue
Check bit value from key k1 for every letter
If bit value of k1 is 1
Count the position of letters in Anti-clockwise direction
Count the position of letters in clockwise direction
New ASCII value = old ASCII value of character + position count in queue
Replace the letter with newly obtained ASCII value
Count number of 1's in key K2 to find offset
Check bit value from key K3 for direction of shift
If bit value of K3 is 0
Shift the letters in anticlockwise direction
Shift the letters in clockwise direction
Output will be Extracted cipher text from queue
Stop

Flowchart 3: Encryption and shifting

IV. Implementation

Firstly we get the text from the user. In this process we'll be using the text "HELLO TMU FOECS". Count the number of characters in the given text. In this example the count of the given characters including spaces is 15. Generate K [512] random bits using rand () function and divided them into three parts namely K1 [258], K2 [128] and K3 [128]. For further execution check whether the first bit of K1 is '0' or '1'. If first bit of key K1 is 0, add 1 to the number count of the characters and if it is 0, add 2 to the number count of the characters. In this example we have 15 characters in the given string and let the first bit of K1 be '0' so add 1 in the length of the string and create a queue of length 16 as shown in figure 1.

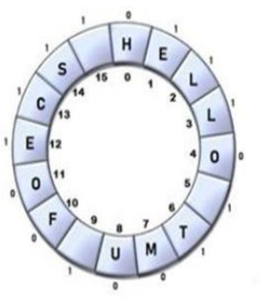


Figure 1
Letters in circular queue before transformation

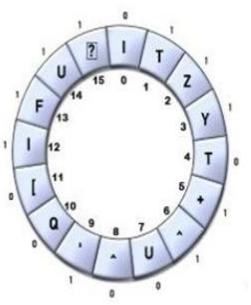


Figure 2
Letters in circular queue after substitution

Now store the string in the queue and get their ASCII values. After getting the ASCII values of every character start the encryption of the plain text. Now using 256 bits of key k1, we check the bit values to count the position of every character in the queue on the basis of K1's bit. If the bit value of K1 is '1' we count the position of the character in clockwise direction and if it is '0' we count the position in anticlockwise direction. For example the first bit in the given example is '0' so we count the position of 'H' in anticlockwise direction which is 1, then we will add 1 to the ASCII value of 'H' which is 72, the new ASCII value attained will be 73 which is the ASCII value of 'I', so the letter 'H' will be substituted by 'I'. Similarly, for the given message 'HELLO TMU FOECS' new encrypted message will be I T Z Y T + ^ U ^ ' Q [I F U ® Refer to table 1.

Letters	ASCII VALUE	Bit from K2	Counted number	New ASCII value	New Letter
H	72	0	1	73	I
E	69	1	15	84	T
L	76	1	14	90	Z
L	76	1	13	89	Y
O	79	0	5	84	T
Space	32	1	11	43	+
T	84	1	10	94	^
M	77	0	8	85	U
U	85	0	9	94	^
space	32	1	7	39	'
F	70	0	11	81	Q
O	79	0	12	91	[
E	69	1	4	73	I
C	67	1	3	70	F
S	83	1	2	85	U
unknown	*unknown*	1	1	*unknown*	*unknown*

Table 1. Substitution Encryption Table

New ASCII value = ASCII value of character + position count in Queue

After getting position of every character in the queue we find the new ASCII value by adding ASCII value with the position count and replace the previous letter with the new letter.

After replacing all values with the same process we get a new circular queue which is shown in figure 2. Now 128 bits of key K2 will be used for the second process of the encryption which is determining the number of shifts in letters of the generated cipher text.

For shifting of the cipher text inside the queue we will count how many number of 1's arises in the Key K2. In this example that we've used to explain the process there are 8 1's inside the K2 key so we shift the cipher text 8 times. The binary values in 128 bits of key k3 will be used to decide the direction of shift in the circular queue. If the bit value of k3 is '1' then we shift the text in anticlockwise direction and if it is '0' then we shift it in clockwise direction. Let the first bit of k3 be '0', then we will shift the letters 8 position in clockwise direction.

Similarly, if second bit of key k3 is '1', then letters in the queue will be shifted to 8 positions in anticlockwise direction. Repeating same process of shifting for k3 [2] and k3 [4], we get queue given in figure. 3 and for k3 [3] and k3 [5], we get the same queue as shown in figure. 4. After this process of encryption, a new queue with fully encrypted cipher text is obtained. We will then extract the cipher I F U ® I T Z Y T + ^ U ^ ' Q[

For k3 [0] =0

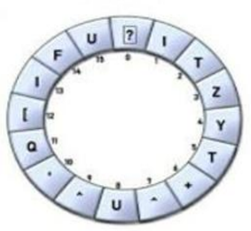


Figure 3
Clockwise Direction

For k3 [6] =0

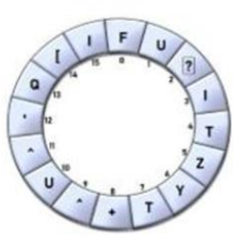


Figure 5
Anticlockwise Direction

For k3 [1] =0

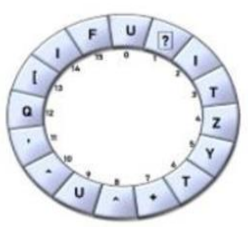


Figure 4
Clockwise Direction

For k3 [7] =0

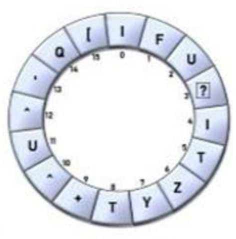


Figure 6
Clockwise Direction

For decryption of the cipher text the same process in reverse format must be executed to get the plain text again. 128 bits from the key K2 and 128 bits from K3 will be used to put the characters to their original positions. In this process firstly we count how many 1's are there in the Key K2 and count their occurrence. Then we check the first bit of the Key K3 to determine the direction of the shifting of the queue. If the bit is '1' then the letters in the queue will move in clockwise direction and if it is '0' then it will move in anticlockwise direction.
For further decryption we use the Key K1 to get the original plain text. We count the position of the character according to the bits of Key K1. If bit value is '1', we count the position in anticlockwise direction and if it is '0', then count the position of letters in clockwise direction. Now, subtract the position count of the character with the ASCII value of the character and replace the letters with the letters attained by new ASCII value. In this way, we apply the same process to every character we'll get our original plain text. Refer to table2.

New letter	**New ASCII value**	**Bit from K2**	**Counted number**	**Original ASCII VALUE**	**Original letters**
I	73	0	1	72	H
T	84	1	15	69	E
Z	90	1	14	76	L
Y	89	1	13	76	L
T	84	0	5	79	O
+	43	1	10	32	Space
^	94	1	10	84	T
U	85	0	8	77	M
^	94	0	9	85	U
‘	39	1	7	32	Space

Q	81	0	11	70	F
[	91	0	12	79	O
I	73	1	4	69	E
F	70	1	3	67	C
U	85	1	2	83	S
unknown	*unknown*	1	1	*unknown*	*unknown*

Table I1. Substitution Decryption Table

Generated 512 bits:

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

```
1    0    0    0    0    0    0    0    1    1
 0 1

256 bits of k1 are:
0    1    1    1    0    1    1    0    0    1
0    0    1    1    1    1    1    0    1    0
1    1    0    1    0    0    1    0    0    1
0    1    0    0    0    1    1    1    1    0
1    1    1    1    1    1    0    1    1    1
0    0    0    0    1    1    0    1    0    1
1    0    0    0    0    0    0    0    1    0
0    0    1    0    0    0    1    1    0    0
1    0    1    0    1    1    1    1    1    1
1    1    0    1    1    1    0    0    1    0
0    0    0    0    1    0    1    0    0    0
1    0    0    0    1    1    1    0    0    0
0    0    0    1    0    1    0    0    0    0
1    0    0    0    0    1    1    1    0    1
1    0    1    0    0    1    0    1    1    0
0    1    0    1    0    0    0    1    1    0
0    1    0    1    0    0    1    0    1    0
0    0    0    1    0    1    0    1    0    1
1    1    1    1    1    1    0    1    0    1
0    1    0    1    1    0    1    1    0    0
1    0    0    0    1    0    1    0    1    0
0    1    0    0    0    1    1    0    1    1
0    0    0    0    0    0    0    0    0    1
0    0    0    1    0    0    1    1    0    0
1    0    0    1    1    1    1    1    1    1
1    0    1    0    0    0

128 bits of k2 are:
0    1    0    0    1    0    0    1    1    0
0    1    1    1    1    0    1    1    1    0
1    1    0    0    0    0    0    1    0    1
0    0    0    1    1    1    1    1    1    0
1    1    0    0    1    1    1    1    0    1
1    1    0    1    0    1    1    1    0    1
0    1    0    1    0    0    1    1    0    1
0    0    0    0    0    0    0    0    0    1
0    1    0    1    0    1    1    1    1    0
1    1    0    0    0    0    0    1    1    1
1    1    0    0    0    1    1    1    1    1
1    1    0    1    1    1    1    1    0    1
0    1    1    0    0    0    0    0

128 bits of k3 are:
0    0    1    0    1    0    0    0    0    0
0    0    0    0    0    0    1    0    0    1
1    0    0    0    0    0    0    1    0    1
1    0    0    0    1    1    0    0    1    1
1    1    1    0    0    0    1    1    1    0
0    1    0    0    0    1    0    1    1    1
1    0    0    0    0    1    1    1    1    1
0    0    0    1    0    0    0    0    1    1
0    0    1    0    1    0    0    1    1    0
1    1    0    1    1    1    1    0    0    1
0    0    1    0    0    0    0    1    0    0
0    1    1    0    1    0    1    0    0    0
0    0    0    0    1    1    0    1
Enter any text HELLO TMU FOECS
No. of letters= 15
Since first bit of k1 is 0: N = 16
Elements in queue are: H E L L O   T M U   F O E C S
1. Encryption
2. Decryption
3. Exit
Enter choice: 1
k1 0    A 1
k1 1    C 15
k1 1    C 14
k1 1    C 13
k1 0    A 5
k1 1    C 11
k1 1    C 10
k1 0    A 8
k1 0    A 9
k1 1    C 7
k1 0    A 11
k1 0    A 12
k1 1    C 4
k1 1    C 3
k1 1    C 2
k1 1    C 1
I T Z Y T + ^ U ^ ' Q [ I F U
No. of 1 in k2 = 8
k3 0    C
k3 0    C
k3 1    A
k3 0    C
k3 1    A
k3 0    C
k3 0    C
k3 0    C
Encrypted text is I F U    I T Z Y T + ^ U ^ ' Q [
1. Encryption
2. Decryption
3. Exit
Enter choice: 2
No. of 1 in k2 = 8
Shifting in opp. direction: I T Z Y T + ^ U ^ ' Q [ I F U
Decrypted text is H E L L O   T M U   F O E C S
1. Encryption
2. Decryption
3. Exit
Enter choice: 3
```

V. Results and Discussions

On comparing our algorithm with the previous one's we get to know that our proposed algorithm takes less time to execute i.e. the compilation time of our proposed algorithm is much less than previously proposed or defined algorithms. In this comparison we have used RSA, RS4, Blowfish, AES and our proposed algorithms. Data is collected by implementing all algorithms in Dev C++ language using all standard specifications given in algorithm. All the algorithms are implemented on three different hardware systems to compare their encryption/decryption speed using different memory sizes (32, 64, 128, 256 and 512) kb. According to the result proposed algorithm is more efficient than previous ones. To compare algorithm's security we've used DES and AES algorithms because they both uses 64 bit and 128 bit key system in their algorithm for encryption but in our proposed algorithm we are using a 512 bit key which makes it more complex for anyone to decrypt it once it is encrypted.

VI. Conclusions

The goal of cryptography is to ensure the preservation of information. This algorithm provides a method to encrypt the plain text and accordingly decrypt it. This algorithm is implemented using arbitrary numbers, circular queue, and a technique for encryption and decryption of data. Here, we have worked upon 32-bit release Dev C++ software. We have generated 512 random numbers using rand () function and further divided them into three parts. In this we have inserted the plain text into the circular queue and applied the substitution encryption method and encrypted the plain text into cipher text by incrementing their ASCII values. Further applying the shifting method we have converted plain text, which is harder to be decrypt-ed. This algorithm is being proposed to make the passwords or the plain text more secure, so that cyber-crime or cyber piracy could not take place on any of the important and valuable data. In future, cryptography will become more vital. On comparing we found that this proposed algorithm uses less number of lines of code and has less complexity as compared to other algorithms and our proposed algorithm also provides less security issues. Henceforth, we conclude that this proposed algorithm can be used to save our data or to protect us from all cyber piracy as it has less complexity and is more secure for all types of passwords.

References

[1] Cyclic codes of length pn over Zp3, Kuwait Journal of Science, Kuwait University, pp 15-24, 2016.

[2] Gola K.K., Khan G., Joshi A., Rathore R. (2020) KKG-512: A New Approach for Kryptos Key Generation of Size 512 Bits Using Plaintext. In: Singh Tomar G., Chaudhari N., Barbosa J., Aghwariya M. (eds) International Conference on Intelligent Computing and Smart Communication 2019. Algorithms for Intelligent Systems. Springer, Singapore.

[3] Kamal Kr. Gola, Bhumika Gupta and Zubair Iqbal. Article: Modified RSA Digital Signature Scheme for Data Confidentiality. International Journal of Computer Applications 106(13):13-16, November2014.

[4] Zubair Iqbal, Kamal Kr. Gola, Bhumika Gupta and Manisha Kandpal. Article: Dual Level Security for Key Exchange using Modified RSA Public Key Encryption in Playfair Technique. International Journal of Computer Applications 111(13):5-9, February2015.

[5] Gola K.K., Sharma V., Rathore R. “SKT: A New Approach for Secure Key Transmission Using MGPISXFS”. Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing, Vol 433. 2016, Springer, New Delhi.

[6] Gulista Khan, Bhumika Gupta, Gola K.K. MDS3C: Modified Digital Signature Scheme for Secure Communication. International Conference on Intelligent Communication, Control and Devices. Advances in Intelligent Systems and Computing, Vol 479, 2017.

[7] 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.

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

[9] 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.

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

[11] Kanika Sharma and Nischay Bahl, Taxonomy of CryptographyTechniques for Network Security, International Journal of Engineering and Computer Science, ISSN: 2319-7242, Volume 5 Issue 8 August 2016, Page No.17787-17793.

[12] N. Varol, F. Aydoğan and A. Varol, "Cyber Attacks Targetting Android Cellphones," in The 5th International Symposium on Digital Forensics and Security (ISDFS 2017), Tirgu Mures, 2017.

[13] K. Harini, N. Pravallika, K. Sashi Rekha,“Enhancement of Data Security using Circular Queue Based Encryption Algorithm,” in International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-12, October 2019.

[14] Gayatri Kapil1 , Alka Agrawal1 , Abdulaziz Attaallah2 , Abdullah Algarni2 , Rajeev Kumar1 and Raees Ahmad Khan1,”Attribute based honey encryption algorithm for securing big data: Hadoop distributed file system perspective,” in peerj journal, 2020.

Author Biographies

Kamal Kumar Gola is working as Assistant Professor in Faculty of Engineering, Teerthanker Mahaveer University, Moradabad, India. He received his B.Tech. Degree from Moradabad Institute of Technology in Computer Science and Engineering and M.Tech. Degree from Uttarakhand Technical University in Computer Science and Engineering. He boasts of more than ten years of University teaching experience and around six months industrial experience. He has more than thirty international publications to his credit in reputed journals. Apart from this, he has participated in various International Conferences and National Conferences and has presented the papers as well.

Manika Gupta was born in Rampur city on 14, October 1999, India. She is currently pursuing Bachelor of Science (Honours) Computer Science from Teerthanker Mahaveer University, Moradabad (U.P.), India and major field of study include designing and coding in C++.

Gulista Khan is B.Tech from Kurukshetra University and M.Tech from MMEC Mullana. Ph. D. in the area of UWSN from Teerthanker Mahaveer University, Moradabad. She has more than 12 years of experience. Published more than 15 papers in International Journals of high impact factor and 18 research papers in Conferences. Her research interest includes the Wireless Sensor Network. Currently working as Assistant Professor in faculty of Engineering, Teerthanker Mahaveer University, Moradabad, India.

Pankaj Rajput was born in Dhampur city on 25, august 1998, India. He is currently pursuing Bachelor of Science (Honours) Computer Science from Teerthanker Mahaveer University, Moradabad (U.P.), India and major field of study includes coding and designing.