A Cipher based on Multiple Circular Arrays

Pushpa R. Suri[1], Sukhvinder Singh Deora[2]

**[1]Department of Computer Sc. & Applications,
Kurukshetra University, Kurukshetra,
Haryana, India-136119.**

**[2]Dept. of Computer Applications,
N. C. Institute of Computer Sciences, Israna,
Panipat, India-132107.**

Abstract

Security of information communicated over the internet has been a challenged area. Data is encrypted by using some cryptographic algorithms using symmetric/asymmetric/public key algorithms for its protection. During literature survey of such algorithms, it was observed that there has been use of one or two dimensional array structures in development of cryptographic algorithms. This paper explains the use of alternative structure made of multiple circular arrays in development of a block cipher based on symmetric variable length key encryption for diffusion of information to a larger extent. It uses operations like circular shift operations, merge-swap and XOR to encrypt the data. The key used for encryption algorithm is generated in parts using Random Number Generator (RNG) operated with XOR operations with its previous sub-key bits at both ends. It may also be transferred along with the ciphertext to intended receiver. The cryptographic algorithm generates scrambled data with above mentioned operations for confusion-diffusion of bits to a larger extent of plaintext data using easy to compute operations for efficient security. The algorithm may also work in asymmetric mode using same direction of rotations. The research work presented in the paper can be used for data security over networks.

Keywords: *Arrays, composite data structures, Cryptographic algorithms, data security, encryption-decryption, symmetric key cryptography, Random number generation, Random Number Generators (RNGs).*

1. Introduction

Security of information on computers & networks has been a challenged area. Cryptography is the branch of computer science that deals with hiding information for secure communication of data. It uses encryption-decryption algorithms (Cryptographic algorithms) to convert readable information to some non-readable form (ciphertext). It uses many mathematical operations along with data structures to encrypt and decrypt the information [1,2]. Sender of information shares decryption technique that is required to recover the original information (plaintext). Literature survey on cryptography and security, it was observed that researchers have proposed algorithms using one dimensional or two dimensional arrays only. This paper presents use of circular shift, merge-swap and XOR operations on Multiple Circular Arrays (MCAs) as a symmetric encryption key algorithm [3]. The key used is generated in steps by an agreed upon Random Number Generator at both ends or is communicated along with the ciphertext to its intended recipient as required. The recipient uses the inverse operations in exactly the opposite order to receive back the plaintext from the ciphertext.

2. Cryptographic Algorithms

Cryptographic algorithms are the basic building blocks in cryptography. They are used in various security applications and protocols. A cryptographic algorithm or cipher is a mathematical function used to encrypt information [4]. Generally, there is an alternate function, unique against the encryption function that is used for decryption of the information. If the way that the algorithm works is kept secret, the algorithm is called restricted algorithm. The quality control and standardization for such an algorithm is generally not so good. It is required that the security of algorithm does not depend on the secrecy of its working. In fact the use of operations in the algorithms should be such that the probability of decrypting the information without actually having the decryption key should be almost negligible or very difficult.

The algorithms are mainly divided into three categories on the basis of the fact of using the same or different key for encryption and decryption:

1. Symmetric Algorithms
2. Asymmetric Algorithms
3. Public-Key Algorithms

2.1 Symmetric Algorithms

In these algorithms, the sender and receiver agree on a key before they communicate. Cryptographic algorithms that use same key, K for encryption and decryption of the information are called symmetric algorithm(s) and such technique of communicating securely is called Symmetric Key Cryptography (SKC).
Encryption and Decryption are denoted as Eq. (1) & (2)

$$E_K(M) = C \quad (1)$$
$$D_K(C) = M \quad (2)$$

The strength of a symmetric key cryptography depends on the strength of the algorithm and the length of key used, assuming that the algorithm is perfect, i.e. there is no other way to break the system than the brute-force attack. Symmetric-key systems like Word Auto Key Encryption (WAKE), SEAL, IDEA, RC4, RC5 etc are simpler and faster [1,5,6,7,8,9,10]. Most recently, many symmetric ciphers based on matrix and operations on it were proposed [11].

However, the quest for generating faster symmetric algorithms based on faster hardware properties is also going on [12]. Researchers all over the world are reviving the way symmetric ciphers have been in use in the past [13,14,15]. On the basis of operations performed on single bit or a group of bits, the symmetric ciphers may be divided into stream algorithms or block algorithms respectively.

2.2 Asymmetric Key Algorithms

The algorithm is designed such that there is a separate key for sender and receiver. The key K is used for encryption and K' is used while decryption of the information. Such an encryption is called Asymmetric Key Cryptography (AKC). In this case, there is a combination of keys (K, K') used to encrypt using encryption algorithm (E_K) and decrypt using decryption algorithm ($D_{K'}$). A prior knowledge to be shared in this case may be the rule that can derive K' if K is known or vice-versa or combination keys and key K used while encryption. Encryption and Decryption are denoted as Eq. (3) & (4):

$$E_K(M) = C \quad (3)$$
$$D_{K'}(C) = M \quad (4)$$

2.3 Public Key Algorithms

These are the asymmetric algorithms that are designed in such a way that the key used to encrypt is different than that used for decryption. The encryption key (K) is available to all so that everyone (known as Public Key) can encrypt the information using encryption algorithm (E_K). But only specific person possesses the decryption key (K'), known as Private Key, that is used to decrypt the message using decryption algorithm ($D_{K'}$). It is not easy to predict the decryption key given the knowledge of encryption key and the algorithm. It is not easy to predict the decryption key given the knowledge of encryption key.
Encryption and Decryption are denoted as Eq. (5) & (6):

$$E_K(M) = C \quad (5)$$
$$D_{K'}(C) = M \quad (6)$$

Diffie and Hellman introduced this encryption scheme in 1976 where each person gets a pair of keys, called the public key and the private key. Public key of persons is published while keeping the private key(s) as secret. Public key encryption avoids the problem of securely exchanging the keys because the public key can be distributed in any manner, and the private key is never transmitted. Messages are encrypted using intended recipient's public key which can only be decrypted using his private key. The need for sender and receiver to share secret information (keys) via some secure channel is eliminated. It may be used for authentication, confidentiality, integrity and non-repudiation. RSA encryption algorithm is a public-key cryptosystem [16]. Researchers have been developing many public-key cryptographic algorithms but they are slow due to complex operations carried out based on hard to solve mathematical problems.

The next sections will detail about the proposed cipher that uses circular arrays as mathematical structure and new operation of merge-swap to produce a ciphertext. It also elaborates the structure, operations to be performed, key and its length, detailed encryption-decryption algorithms, the result and analysis of its sample runs, its possible variants. It is explained using a symmetric key and is a block cipher.

3. The Proposed Cipher

This section explains the use of Multiple Circular Arrays (MCAs), a composite structure that each circular array has double the size of its previous circular array. The number of such circular arrays to be used in the structure depends upon the size of data on which the encryption-decryption is to be applied.

3.1 The Structure

The Multiple Circular Arrays (MCAs) structure has been thought of as multiple Rotor Plates numbered from 1 to n from innermost to outside the structure. A example MCA is shown in Fig. 1 that contains three circular arrays and size of innermost array is 4. The number of elements in the innermost circular array is shown with 4 elements with data items filled from 1 to 4. The subsequent circular arrays have size of 8 and 16 having 5 to 12 in second and 13 to 28 in third circular array and so on. There are total of 3 circular arrays having a total of 4(1+2+4)=28 elements (refer to Fig. 2).Persons under communication may decide the size of innermost array and number of such arrays in the MCA structure before actual communication.

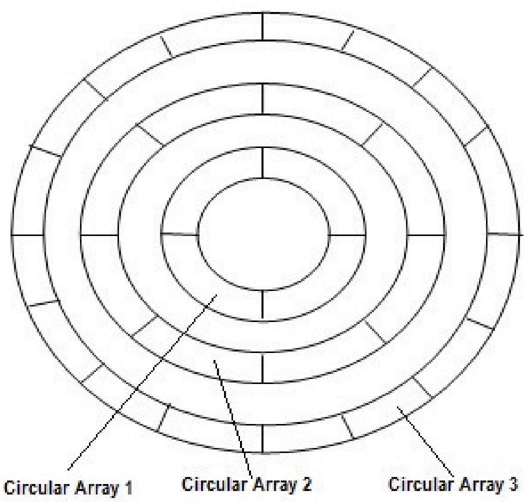

Fig. 1. View of Multiple circular Array

3.2 Operations

The size of the structure will enable some more operations, apart from the traditional operations like XOR and circular right/left shift operations on its elements. The shape of circular arrays is like a rotar plate and hence circular right/left shift operation is shifting the elements of the circular array by some number of positions. E.g. Fig. 3 shows the resultant of the MCAs after the second circular array is rotated right by two positions.

Swapping of variables is operation often used in programming. It is known that swapping can also be viewed as XOR operation. That is, when we XOR any two items, say A and B so that its resultant value is C, refer to Eq. (7). Now, if we XOR C with either of the two initial values, the other will be produced as a resultant value, refer to Eq. (8).

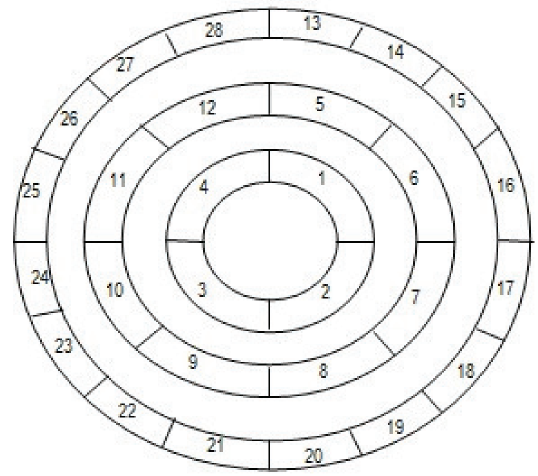

Fig. 2. Data sequence shown

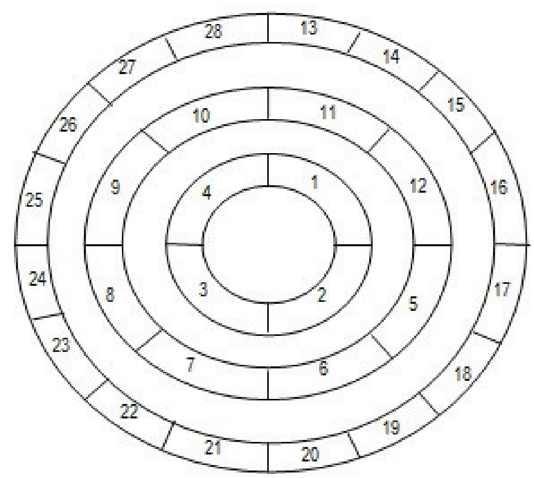

Fig. 3. Multiple Circular Arrays after Circular Rotate operation on 2nd Plate by 2 positions

A XOR B = C and (7)

A XOR C = B or B XOR C = A. (8)

Similarly, swapping can be used as an involution function, i.e. $f^{-1}=f$. Thus, if we have two variables A & B such that A=2 and B=3. Applying swapping operation once will result into A=3 and B=2. If swapping is applied again on the two variables A & B, it will lead to A=2 and B=3. We are proposing a merge-swap operation in the proposed cipher that uses swapping of elements of a particular circular array with the next circular array starting. Swapping may start at a particular index value and can be done with/without gap.

Merge-swap operation of 1st circular array with 2nd circular array of Fig. 2 has been explained without gap in Fig. 4 and with gap option in Fig. 5.

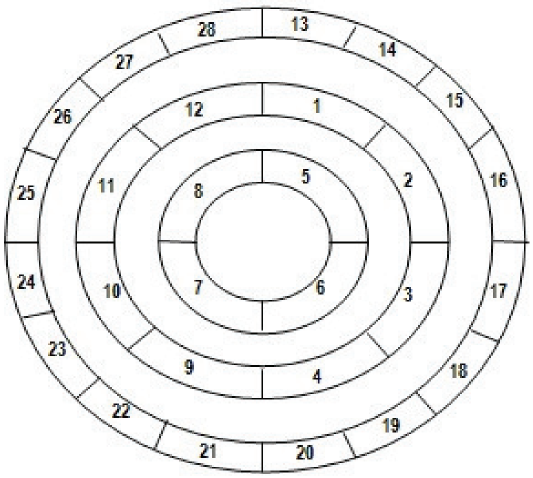

Fig. 4. Sample MCAs after Merge-swap of 1st Circular Array with 2nd Circular Array starting at position 1 in 2nd Circular Array without gap

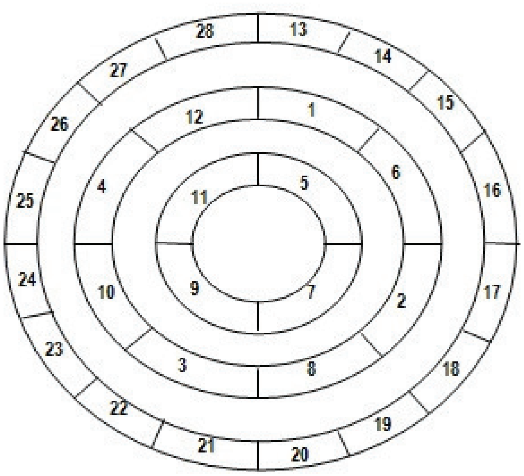

Fig. 5. Sample MCAs after Merge-swap all the elements of 1st Circular Array with 2nd Circular Array starting at position 1 in 2nd Circular Array with gap

3.3 Key, its generations & use

The cipher has been designed in such a manner that one can take a variable sized plaintext by deciding the total number of circular arrays to be used for storing the plaintext initially by padding 0s in the end, if required. Thus, the variable key length is possible for the suggested cipher depending upon the number of circular arrays used in the structure. The general approach to generate the key bits, number of bits used and bit positions to be used for specific operations is discussed in this section.

The proposed key is generated step-wise for each circular array. If we are to encrypt plaintext contained in a multiple circular arrays with innermost array of size = 8 bits, then the first sub-key G_1 of length 8 bits is generated using a Random Number Generator (RNG). It is used as S_1, sub-key for encryption of innermost circular array. For the next circular array, G_2, a set of 8 bit random number is again generated. This is XORed with the previous sub-key of length 8 bits and appended to the resultant 8 bits thereby making it now 16 bits sub-key for 2nd rotar plate. Table 1 provides the the details of generating the sub-keys for each rotar plate.

Table 1: Key Generation w.r.t. Circular Array Number

Circular Array Number	Length of sub-key generated (G_i)	Sub-key used for encryption S_i (i = plate number)
1	G_1=8 bits	S_1=G_1
2	G_2=8 bits	S_2= (S_1 XOR G_2)appendG_2
3	G_3=16 bits	S_3= (S_2 XOR G_3)appendG_3
4	G_4=32 bits	S_4= (S_3 XOR G_4)appendG_4
and so on for next rotor plates/circular arrays …		
i	G_i=(i-1)*8	S_i= (S_{i-1} XOR G_i)append G_i

The cipher uses some number of bits for applying operations discussed in section 3.2. The total number of bits to be used for particular Circular Array Number is provided in the Table 2.

Table 2: Number of bits to be used during Encryption/Decryption w.r.t. Circular Array Number

Circular Array Number	Total No. of bits					
	Newly Generated	Used for				
		Merge-swap		Rotate operation	Rotate and Merge-swap	XOR
		Gap	Start Position			
1	8	1	4	3	8	8
2	8	1	5	4	10	16
3	16	1	6	5	12	32

4	32	1	7	6	14	64
and so on for next rotor plates/circular arrays…						
i	2^{i+1}	1	3+i	3+(i-1)	7+2i-1	2^{i+2}

A 8 bit sub-key is used for applying operations on innermost circular array bits in some order; say circular rotate, Merge-swap and XOR key bits with the circular array bits. The bit numbers that were used for applying the three operations in the test runs of the proposed cipher is given in Table 3.

Table 3: Key bit positions used for various operations during Encryption/Decryption w.r.t. Circular Array Number

Circular Array Number	Length of sub-key used	Rotate bits	Merge-swap bits	
			With Gap bit	Position of bit where to start swap in next Plate
1	8	5 to 7	0	1 to 4
2	16	12 to 15	6	7 to 11
3	32	27 to 31	20	21 to 26
4	64	58 to 63	50	52 to 57
and so on for next rotar plates…				
i	2^{i+2}	$(2^{i+2}-(i+2))$ to $(2^{i+2}-1)$	$(2^{i+2}-2(i+3))$	$(2^{i+2}-2(i+2)-1)$ to $(2^{i+2}-(i+2)-1)$

Sub-keys may be repeatedly used for the same circular array i, after circular right shift operation on the sub-key S_i by 3 positions. The number of repetitions used for applying operations on circular array bits will increase the overall security of the algorithm. While decryption, the sub-key S_i is rotated circular left shift operation by the same number of bit positions.

3.2 Encryption Algorithm

The proposed cipher has been designed using Multiple Circular Arrays (MCAs) with number of such arrays equal to n. The plaintext may be inserted into the MCAs starting from innermost to outermost circular array taken in order of the index values. The operations carried out on the circular array "i" under consideration may be repeated for some odd number of times as per the requirements of level of data security to be achieved. The algorithm uses number of bits and bit positions of key as discussed in table 1, 2 and 3 of section 3.3. Variable i denotes the number, CA_i denotes i-th Circular Array, Count variable is used to calculate the number of iterations performed on CA_i, ⊕ denotes append/concatenate operation, << and >> denotes the circular left shift and circular right shift operations respectively in the Algorithms given below:

Encryption Algorithm

1. Set Count :=0.
2. Repeat Steps 3 to 10 For i=1 to n, do the following for circular array CAi:
3. Generate the sub-key G_i for i-th circular array.
4. If (i <> 1) then
   Update sub-key $S_i = (S_{i-1} \text{ XOR } G_i) \oplus G_i$
   Else
   Set $S_i = G_i$
   [End of If condition, step no. 4]
5. Obtain the circular right bit values as discussed in table 1, 2 & 3.
6. Perform circular right shift operation on CA_i as per the value of bit positions discussed in Section 3.3
7. If (i <> n) then
   Obtain the merge-swap bit values
   Perform merge-swap operation between CA_i and CA_{i+1}
   [End of If condition, step no. 7]
8. Perform CA_i XOR S_i.
9. Count:=Count + 1.
10. If (Count < 3) then
    $S_i = (S_i << 3)$.
    Else
    Set Count := 0
    [End of If condition, step no. 10]
    [End of For loop, step no. 2]
11. End

At the end of the encryption algorithm, the CA_is will contain the ciphertext that can be collected in the order that they were inserted into the CA_is. The ciphertext so obtained along with the key S_i used in last circular array iteration needs to be communicated to its recipient.

The recipient may start the decryption using the decryption algorithm that uses the key S_i for i-th circular array CA_is. The detailed decryption algorithm is as follows:

Decryption Algorithm

1. Set Count :=0. Set i=n.
2. Repeat Steps 3 to 10 While i >= 1, do the following for circular array CAi:
3. If (Count = 0) then
   Use the sub-key S_i for i-th circular array.
   Else
   Set $S_i = S_i >> 3$.
   [End of If condition, step no. 3]

4. Perform CA_i XOR S_i.
5. If (i <> n) then
   Obtain the merge-swap bit values
   Perform merge-swap operation between CA_i and CA_{i+1}
   [End of If condition, step no. 7]
6. Obtain the circular right bit values as discussed in table 1, 2 & 3.
7. Perform circular left shift operation on CA_i as per the value of bit positions discussed in Section 3.3
8. Set Count = Count + 1.
9. If (Count = 2 and i > 1) then
   Set G_i = right half key bits of S_i and
   Set S_{i-1} = left half key bits of S_i.
   Obtain sub-key S_{i-1}= (S_{i-1} XOR G_i)
   Set Count = 0.
   [End of If condition, step no. 8]
10. Set i := i – 1.
   [End of While loop, step no. 2]
11. End

4. Some Test Results

The proposed algorithm was implemented in C language. Multiple Circular Arrays of were initialized with half 0s and half 1s as per the size of the arrays. The operations as discussed in Section 3 were applied for certain number of rounds and the resultant bits were collected by reading the arrays from innermost to outermost. Randomly selected subsequence of bits were chosen and tested for the amount of randomness produced after the shuffling of plaintext bits as per the NIST specification tests [17]. The tests were applied to know how much shuffling of bits has been done by the operations. The performance of the above discussed algorithm in some selected tests was recorded and is explained in the following subsections:

4.1 Monobit Test

This test is used to determine the number of 0s and 1s in a randomly selected bit sequence of ciphertext is same or not. If the resultant sequence is a random sequence, then any arbitrarily selected bits must have equal proportion of 0s and 1s. The objective of the test is to assess closeness of the fraction of ones to ½. That is, thenumber of zeros and ones in a sequence should be almost equal. It uses parameters n - the length of the bit string, $\varepsilon = \varepsilon_1, \varepsilon_2, \varepsilon_3,\ldots, \varepsilon_n$ the sequence of bits. It then calculates S_{obs} – the absolute value of the sum of X_i (where $X_i = 2\varepsilon - 1 = \pm 1$) given by Eq. (9).

$$S_{obs} = \frac{|S_n|}{\sqrt{n}} \quad (9)$$

The p-value is then computed as given by Eq. (10),

$$p - value = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right) \quad (10)$$

where *erfc* is the complementary error function.

The tests were applied on sample data of 7 and 8 circular arrays. The p-value calculated after 5, 6 or 7 rounds were evaluated. The details of the min., max., average p-value and number of tests failed and passed is provided(see Table 4 & 5). Tests were applied on structures with some number of circular arrays after applying certain number of rounds/iterations (see Table 4 & 5).

Table 4: Test Results of Monobit Test on 7 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 7	6, 7	7, 7
	p-value	p-value	p-value
1	0.657969	0.282297	0.087705
2	0.230139	1	0.109599
3	1	0.423711	0.689157
4	0.841481	0.548506	0.689157
5	0.027807	0.423711	0.689157
6	0.230139	0.027807	0.317311
7	0.548506	0.689157	0.027807
8	0.230139	0.109599	0.423711
9	0.689157	0.689157	0.423711
10	0.841481	0.423711	0.841481
11	0.841481	0.548506	0.423711
12	0.949571	0.282297	0.113846
13	0.423711	0.548506	0.161513
14	0.423711	0.841481	1
15	1	0.423711	0.689157
16	0.009322	0.109599	0.423711
17	0.548506	0.230139	0.317311
18	0.230139	0.689157	0.009322
19	0.230139	0.109599	0.317311
20	0.423711	0.841481	0.423711
21	1	0.841481	0.689157
22	0.548506	0.841481	0.841481
Min	0.009322	0.027807	0.000689
Max	1	1	1
Average	0.542073	0.496595	0.446975
Result:	Count	Count	Count
FAIL	1	0	1
PASS	21	22	21

The results of the test indicate that more than 95% of times the resultant ciphertext contained equal proportion of 0s and 1s again. It indicates that operations have shuffled, XORed and merge-swapped the bits, still the basic nature of bits have not been lost. Thus there is a good shuffling of the elements of the plaintext.

Table 5: Test Results of Monobit Test on 8 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 8	6, 8	7, 8
	p-value	p-value	p-value
1	0.375921	0.949571	0.282297
2	0.548506	0.423711	0.689157
3	0.689157	1	0.230139
4	0.841481	0.689157	0.548506
5	0.423711	0.689157	0.841481
6	0.841481	0.689157	0.689157
7	0.109599	0.230139	0.548506
8	0.071861	1	0.548506
9	0.689157	0.161513	0.423711
10	0.0455	0.689157	0.0027
11	0.423711	1	0.109599
12	0.569214	0.612882	0.447884
13	0.548506	0.548506	0.689157
14	0.071861	0.689157	0.317311
15	0.841481	0.841481	0.109599
16	0.841481	0.230139	0.841481
17	0.548506	0.071861	0.548506
18	0.841481	0.689157	0.317311
19	0.548506	0.016395	0.230139
20	0.548506	0.841481	0.841481
21	0.689157	0.423711	0.423711
22	0.841481	0.109599	0.317311
Min	0.0455	0.016395	0.000689
Max	0.841481	1	1
Average	0.543194	0.572542	0.452748
Result:	Count	Count	Count
FAIL	0	0	1
PASS	22	22	21

4.2 Frequency within a Block Test

The focus of this test is on the proportion of 1s within M-bit blocks of the data. It evaluates that the number of 1s in a M-bit data block is approximately M/2. This test uses M-the length of each block, n – the length of the bit string and sequence of bits $\varepsilon = \varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots, \varepsilon_n$. Firstly, non-overlapping blocks N are formed given by Eq. (11):

$$N = \left\lfloor \frac{n}{M} \right\rfloor \tag{11}$$

Now, use Eq. (12) to determine the proportion π_i of ones in each M-bit block.

$$\pi_i = \frac{\sum_{j=1}^{M} \varepsilon_{(i-1)}M + j}{M} \tag{12}$$

and compute the χ^2 statistic as per Eq. (13) to further obtain the p-value using Eq. (14).

$$\chi^2(\text{obs}) = 4M \sum_{i=1}^{N} \left(\pi - \tfrac{1}{2}\right)^2 \tag{13}$$

$$\text{p-value} = igamc\left(N/2, \chi^2(obs)/2\right) \tag{14}$$

Table 6: Test Results of Frequency within a Block Test on 7 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 7	6, 7	7, 7
	p-value	p-value	p-value
1	0.999438	0.999887	0.99982
2	0.991468	0.999993	0.999988
3	0.964295	0.998474	0.998821
4	0.911413	0.999951	0.999856
5	0.834308	0.998821	0.99982
6	0.739918	0.999777	0.997823
7	0.637119	0.999107	0.999934
8	0.534146	0.999982	0.999777
9	0.437274	0.999438	0.999988
10	0.350485	0.99923	0.999887
11	0.999524	0.998474	0.99472
12	0.999107	0.99934	0.998058
13	0.998971	0.999438	0.999951
14	0.998275	0.998058	0.998821
15	0.999668	0.998971	0.999438
16	0.999601	0.999668	0.998655
17	0.999107	0.997568	0.999668
18	0.999668	0.999107	0.999974
19	0.999934	0.99934	0.998275
20	0.995969	0.999726	0.999524
Min	0.350485	0.997568	0.99472
Max	0.999934	0.999993	0.999988
Average	0.851823	0.999178	0.998978
Result:	Count	Count	Count
FAIL	0	0	0
PASS	20	20	20

Table 7: Test Results of Frequency within a Block Test on 8 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 8	6, 8	7, 8
	p-value	p-value	p-value
1	0.999726	0.998821	0.999913
2	0.99934	0.999668	0.999524
3	0.999934	0.999913	0.996677
4	0.999913	0.999777	0.997823
5	0.999438	0.999988	0.995578
6	0.999668	0.999777	0.99923
7	0.998655	0.996996	0.999934
8	0.998474	0.999601	0.99934
9	0.996335	0.99934	0.997823
10	0.998058	0.985339	0.999668
11	0.999934	0.999856	0.999668
12	0.998655	0.999887	0.998971
13	0.999601	0.999726	0.999668
14	0.997568	0.999524	0.999726
15	0.999777	0.985339	0.999438
16	0.999668	0.999964	0.999934
17	0.999974	0.999524	0.999601
18	0.99982	0.999601	0.999913
19	0.998474	0.99472	0.999934
20	0.999993	0.998655	0.999951
Min	0.996335	0.985339	0.995578
Max	0.999993	0.999988	0.999951
Average	0.99906	0.997334	0.998993
Result:	Count	Count	Count
FAIL	0	0	0
PASS	20	20	20

The parameters of the frequency within a block test are evaluated using (14). The resulted values show that all tests were passed by the proposed cipher for 7 and 8 Circular Array block (see Table 6 & 7). It confirms that the operations applied on the structure are scrambling the data bits amongst themselves.

4.3 Runs Test

The focus of this test is the total number of runs of 1s in the sequence. A run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits that are bounded before and after with a bit of the opposite value. The main purpose of the runs test is to determine whether the number of runs of one and zeros of various lengths is as expected for a random sequence. Moreover, it checks the oscillations of zeros and ones are more or low in number. For a n - length of bit string, let ε - denotes the sequence of bits $\varepsilon = \varepsilon_1\varepsilon_2\varepsilon_3 \ldots \varepsilon_n$. The parameter V_n(obs) – denotes the total number of runs (zero runs + one runs) across all n bits evaluated given by Eq. (16) and

$$\tau = \frac{2}{\sqrt{n}} \qquad (15)$$

$$V_n(\text{obs}) = \sum_{k=1}^{n-1} r(k) + 1 \qquad (16)$$

where r(k)=0 if $\varepsilon_k=\varepsilon_{k+1}$ r(k)=1 otherwise.

The p-values is then calculated given by Eq. (17)

$$p - value = erfc\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n\pi}(1-\pi)}\right) \qquad (17)$$

Table 8: Test Results of Runs Test on 7 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 7	6, 7	7, 7
	p-value	p-value	p-value
1	0.79047	0.041017	0.996745
2	0.080606	0.421211	0.548506
3	0.161513	0.284236	0.537243
4	0.894201	0.500798	0.284236
5	0.443224	0.505677	0.919542
6	0.385117	0.308855	0.894201
7	0.505677	0.379678	0.919542
8	0.948642	0.545683	0.640508
9	0.689157	0.844549	0.308855
10	0.100609	0.425847	0.83829
11	0.795064	0.161513	0.576645
12	0.071861	0.203323	0.828718
13	0.23151	0.812771	0.676922
14	0.4518	0.948642	0.828718
15	0.643606	0.739835	0.266521
16	0.571368	0.223244	0.739835
17	0.643606	0.278479	0.948642
18	0.853782	0.308855	0.948642
19	0.545683	0.346178	0.421211
20	0.315185	0.52328	0.503957
Min	0.071861	0.041017	0.266521
Max	0.948642	0.948642	0.996745
Average	0.506134	0.440183	0.681374
Result:	Count	Count	Count
FAIL	0	0	0
PASS	20	20	20

Table 9: Test Results of Runs Test on 8 Circular Arrays

S No	NoOfRounds, NoOfCAs		
	5, 8	6, 8	7, 8
	p-value	p-value	p-value
1	0.29846	0.503957	0.308855
2	0.853782	0.841481	0.338168
3	0.545683	0.413753	0.812771
4	0.79047	0.676922	0.551016
5	0.108573	0.413753	0.537243
6	0.13511	0.338168	0.661694
7	0.937395	0.230139	0.661694
8	0.432303	0.154473	0.948642
9	0.531971	0.853782	0.322658
10	0.140635	0.0455	0.379678
11	0.479523	0.384538	0.987214
12	0.166685	0.551016	0.52328
13	0.160152	0.443224	0.197399
14	0.423711	0.028706	0.23151
15	0.853782	0.338168	0.869265
16	1	0.761867	0.812771
17	0.432303	0.212076	0.835168
18	0.52328	0.171022	0.223244
19	0.00511	0.167799	0.714876
20	0.100609	0.588131	0.09399
Min	0.00511	0.028706	0.09399
Max	1	0.853782	0.987214
Average	0.445977	0.405924	0.550557
Result:	Count	Count	Count
FAIL	1	0	0
PASS	19	20	20

The test is applicable when all the frequency tests are cleared by some sequence of bits. Since the proposed cipher has cleared all the tests in Section 4.2, we may apply this test. This test was applied on 7 and 8 circular array structures with different number of rounds. The evaluated p-values show that most of the tests were passed (see Table 8 & 9).

4.4 Random Excursion Test

This test focuses on the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is derived from partial sums after the (0,1) sequence is transferred to the appropriate (-1, +1) sequence. A cycle of a random walk consists of a sequence of steps of unit length taken at random that begin at and return to the origin. The objective of the test is to find out if the number of visits to a particular state within a cycle deviates from an random sequence. This test is a series of eight tests, one test and conclusion for each of the states: -4, -3, -2, -1 and 1, 2, 3, and 4.

Let n denote the length of bit string, ε - denotes the sequence of bits $\varepsilon = \varepsilon_1 \varepsilon_2 \varepsilon_3 \ldots \varepsilon_n$. The 0s and 1s of the input sequence (ε) are changed to values -1 and +1 using $X_i=2\varepsilon_i-1$. Then partial sums S_i are computed given by Eq. (18)

$$
\begin{aligned}
S_1 &= X_1 \\
S_2 &= X_1 + X_2 \\
S_3 &= X_1 + X_2 + X_3 \\
&\vdots \\
S_n &= X_1 + X_2 + X_3 \ldots X_n
\end{aligned}
\qquad (18)
$$

A new sequence S' is formed by attaching zeros before and after the set S. $S'=0, S_1, S_2, S_3, \ldots S_n$. Total number of zero crossings in S' are calculated represented as J (the number of cycles in S'). Using the calculated values, p-value is computed and checked. The results (see Table 10 & 11) show that most of the tests have passed.

Table 10: Test Results of Random Excursion Test on 7 Circular Arrays

Test Result	NoOfRounds, NoOfCAs		
	5 7	6 7	7 7
PASS/FAIL	Count	Count	Count
FAIL	1	0	0
PASS	359	360	360
Min	0.000407	0.157299	0.157299
Max	1	1	1
Average	0.680464	0.644273	0.665769

Table 11: Test Results of Random Excursion Test on 8 Circular Arrays

Test Result	NoOfRounds, NoOfCAs		
	5 8	6 8	7 8
PASS/FAIL	Count	Count	Count
FAIL	0	0	0
PASS	360	360	360
Min	0.220672	0.157299	0.220672
Max	1	1	1
Average	0.658914	0.665038	0.652101

5. Analysis

Our proposed symmetric key algorithm based on multiple circular arrays use the operations of circular rotate, merge-swap and XOR three times on each of the circular array starting from the innermost array. Circular rotate operation shifts the plaintext data bits of CA_i by some integer value thereby ensuring change in the actual value. Merge-swap operation shuffles the plaintext data bits of some circular array, CA_i, with some subset of next circular array, CA_{i+1}. Lastly, the XOR operation with the sub-key S_i will change some of the data bits according to bit value of S_i. Repeating the process for some number of times ensures that scrambling of plaintext data bits to ciphertext. It is proposed that the repetition of the process must be done odd number of times so that the original plaintext is not produced as the ciphertext due to even iteration working as an inverse to the previous odd iteration. Sub-key is rotated circularly by 3 bits (or for any odd number) before applying further iterations on CA_i, thus reducing the probability that second iteration on CA_i becomes exactly inverse of the first iteration is very low.

6. Possible Variants and Future Scope

The proposed cipher is a variable length block cipher has a property to select as many multiple arrays as per the size of data (appending some dummy information bits/bytes) and may use variable length of key size (depending upon size of MCAs). It may be used with some secret S-box used for substituting step on data values to further increase the complexity of the encryption/decryption. This step will convert the cipher into a complex Feistel cipher. The cipher may be used in composite ciphers in which more than one type of cryptographic algorithms may be applied in sequence to produce harder algorithms. The operations suggested in the paper may be used in authentication and verifiability of data in data files using some modifications also.

7. Conclusions

The security of the cipher is dependent upon the symmetric key, number of iterations used for each circular array and rotation policies. Thus the attacker has to predict all three parameters well so as to decipher the ciphertext. The chance of making a brute force attack is very complex as there is a possibility of arriving at many number of possible plaintext data bits in that case. Error in prediction of a single bit value of the sub-key may affect many values in the ciphertext. This error will be further propagated to more plaintext bits due to merge-swap operation with their immediate successor circular arrays, if applicable. To sum up, the proposed symmetric block cipher has a good security with variable key length option. The key length to be used and number of iterations may also be increased/decreased as per agreed upon requirements of the entities involved in communication. Thus total number of possible attempts to be checked for n rounds will be 2^{n+2}. The complexity of decrypting the ciphertext to original plaintext increases with the increase in value of n. Moreover, error in prediction of one bit value will lead to multiple errors propagated to next rounds of iterations, a selective bit attacks may also not prove good. In nutshell, the proposed cipher is a flexible symmetric key cipher with variable key length option.

References

[1] Bruce Schneier, Applied Cryptography: John Wiley & Sons (Asia) Pte Ltd, ISBN 9971-51-348-X, 2006.
[2] William Stallings, Cryptography and Network Security, Principles and Practices, Fourth Edition: Pearson Education, 2006.
[3] Seymour Lipschutz, Data Structures: Tata McGraw Hill Publishing Company Ltd., 2005.
[4] Jonathan Katz and Yehuda Lindell, Introduction to Modern Cryptography, CRC Press, New York, pp. 3-4, 2007.
[5] D.J. Wheeler, "A Build Data Encryption Algorithm", Fast Software Encryption, Cambridge Scurity Workshop Proceedings, Springer-Verlag, 1994, pp. 127-134.
[6] P. Rogaway and D. Coppersmith, "A Software-Oriented Encryption Algorithm", Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 56-63.
[7] X. Lai, "Detailed Description and a Software Implementation of the IPES Cipher", unpublished manuscript, 8 Nov. 1991.
[8] X. Lai, "On the Design and Security of Block Ciphers", ETH Series in Information Processing, Vol. 1, Konstanz: Hartung-Gorre Verlag, ETH No. 9752, Doctor of Technical Sciences, Xidian University, Xian, China, 1992.
[9] R.L. Rivest, "The RC5 Encryption Algorithm", Dr Dobb's Journal, Vol. 20, No. 1, Jan 1995, pp 146-148.
[10] R.L. Rivest, "The RC5 Encryption Algorithm", K. U. Leuven Workshop on Cryptographic Algorithms, Springer-Verlag, 1995.
[11] Neeraj Khanna et. al, New Symmetric key Cryptographic algorithm using combined bit manipulation and MSA encryption algorithm: NJJSAA symmetric key algorithm, International Conference on Communication Systems and Network Technologies, 2011, pp. 125 - 130.
[12] D. Gollmann and W.G. Chambers,: Clock-controlled shift registers: a review, IEEE Journal on Communications, Vol. 7 , Issue : 4, May 1989, pp. 525 - 533.
[13] P. R. Suri, Sukhvinder Singh Deora: A Cipher based on 3D Array Block Rotation, IJCSNS International Journal of Computer Science and Network Security, Vol.10 No. 2, February 2010, pp. 186-191.

[14] Mohammed M. Alani: Testing Randomness in Ciphertext of Block-Ciphers Using DieHard Tests, IJCSNS International Journal of Computer Science and Network Security, Vol. 10 No. 4, April 2010, pp. 53-57.
[15] S. Dhall, S.K. Pal: Design of a New Block Cipher Based on Conditional Encryption, Seventh International Conference on Information Technology, pages 714, Las Vegas, NV, 12-14 April 2010, pp. 714 - 718.
[16] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21, No. 2, Feb 1979, pp. 120-126.
[17] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Specifications. SP800-22.

Dr. (Mrs) Pushpa R. Suri is Professor in the Department of Computer Science and Applications at Kurukshetra University, Haryana, India. She has supervised a number of PhD students in the area of Network Security and Cryptograhic techniques. She has also published a number of research papers in National and International Journals and Conference Proceedings. Her areas of interest are Data Structures, Network Security & Cryptography.

Sukhvinder Singh Deora is a Research Scholar at Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India. He is M.C.A., M.Phil. (Computer Science) and is working as Assistant Professor in N.C. Institute of Computer Sciences, Israna, Panipat, India. To his credit are many prominent papers in the area of data security and issues related to it published in eminent International/National Journals & Conferences. He is a Reviewer to many International Journals of repute and worked as Editor for Indian Society of Information Theory and Applications (ISITA), India. He is having an industry experience of 1.5 years in the areas of Testing, Java, Database design and security issues. His interest areas include Network Security, Theoretical Computer Sciences, Data Structures, S/W Testing and Database Designing. He is also a permanent member of ISITA and many other professional bodies.