## Abstract

In the present work the author has introduced a different symmetric key encryption scheme, named algorithm SKG 2.5, for secured routing through internet using two symmetric keys. This scheme evolves four steps (1) converting each of \(N\) character of the text into binary form using 8-bit ASCII Code (2) swapping integral multiples of leftmost \(N_1\)st bit with corresponding rightmost bits till 4\(N\) bit if remainder of \((8N+1)/N_1\) is zero, otherwise continue the swapping of bits upto 8\(N_1\)th bit (3) shifting \(N_2\) leftmost bits to rightmost in a circular queue or vice versa and (4) Finally converting 8\(N\) bits so obtained into text using 8-bit ASCII Code. This scheme can be applied successfully to encrypt any text/data consisting of 10 or more characters. The results obtained after application of this algorithm are excellent.

## 1 Introduction

The internet [1,2] is now-a-days used to send confidential data from one computer to another computer. The confidential data may be bank statements, bank transaction, military information, confidential data of companies or any other important data etc. But there is always a possibility that any unwanted intruder may intercept our data. So, the security and originality of data [1] has now become very challenging. There are a large number of methods and techniques to achieve security goals, one of these is Cryptography. Cryptography [3,4] is the process used to make a meaningful message to appear meaningless. The cryptographic algorithm can be classified into two categories: (i) Symmetric Key Cryptography where one key is used for both encryption and decryption purpose. (ii) Public Key Cryptography where two different keys are used one for encryption and the other for decryption purpose. Due to massive computation the public key crypto system may not be suitable in security of data in sensor networks [5]. In the present work, algorithm SKG 2.5, the author has used two symmetric keys: (1) swapping integral multiples of leftmost \(N_1\)st bit with corresponding rightmost bits till 4\(N\) bit if remainder of \((8N+1)/N_1\) is zero, otherwise continue the swapping of bits upto 8\(N_1\)th bit and (2) shifting \(N_2\) leftmost bits to rightmost in a circular queue or vice versa.

## 2 Theory

The author has already reported an algorithm SKG 1.2[6], in which positions of characters are manipulated at two stages firstly, by swapping integral multiples of leftmost \(N_1\)st character with...
corresponding rightmost character till N/2 character if remainder of (N+1)/N1 is zero, otherwise continue the swapping of bits upto Nth character and secondly, shifting N2 leftmost characters to rightmost in a circular queue or vice versa. In the present scheme, data is digitized using 8-bit ASCII Code. The bits so obtained are manipulated at two stages firstly, by swapping integral multiples of leftmost N1st bit with corresponding rightmost bits till 4Nth bit if remainder of (8N+1)/N1 is zero, otherwise continue the swapping of bits upto 8Nth bit and secondly, shifting N2 leftmost bits to rightmost in a circular queue or vice versa. This scheme evolves four steps:

1. Firstly, convert each character into binary form using 8-bit ASCII Code, thus we get 8N bits corresponding to text of N characters
2. Secondly, swap integral multiples of leftmost N1st bit with corresponding rightmost bits till 4Nth bit if remainder of (8N+1)/N1 is zero, otherwise continue the swapping of bits upto 8Nth bit
3. Thirdly, shift N2 leftmost bits to rightmost in a circular queue or vice versa and
4. Finally converting 8N bits so obtained into text using 8-bit ASCII Code.

When any text of 10 characters is converted into binary form using 8-bit ASCII Code, we get 80 bits which contains about 50% of 0’s and 1’s each. Therefore, total number of possible combinations is about $80!/(40!)^2 = 1075 \times 10^{20}$. The Super Computer available is Teraflop which is capable of doing $10^{12}$ floating point calculations per second, so a teraflop super computer shall take about 3409 Years to find all possible combinations [7].

### 3 Encryption Algorithm

// Read the text input and check length of Input, if less than 10, give error message
Step 1: Start
Step 2: Read input text N
Step 3: If (N.length() < 10)
    Print error message that program is not applicable;
//Convert the text of N characters to binary form using 8-bit ASCII Code
Step 4: initialize character array str[] // copying each character of text N to character array str[]
Step 5: for(i=0; i<N.length(); i++)
    {
        If(i==N.length() – 1)
            Str[i] = N.substring(i);
        else
            Str[i] = N.substring(i, i+1);
    }
Step 6: initialize byte array bytes[] // copy each text character of character array converted to byte char
Step 7: for(i=0; i<str.length; i++)
    {
        bytes[i] = (byte) Str[i];
    }
Step 8: for each byte in array, convert each byte to binary bits and create string of those binary bits
// Interchange the leftmost integral multiple of N1 bits with corresponding rightmost integral multiple of N1 bits
Step 9: initialize l = length of binary string //this will be used at number of places in program
Step 10: initialize integer j (to store length to be traversed [loop through])
Step 11: Read value of N1
Step 12: integer r = remainder of (binary+1)modulus N1
Step 13: if (r==0)
        j = l/2;
    else
        j = l;
Step 14: for(i=1;N1*i<=j;i++)
        {
            ch = charAt(N1*i-1);
            Replace/ set charAt((N1*i)-1)th position with charAt((l+1)-(N1*i))th position;
            Replace/ set charAt((l+1)-(N1*i))th position with char stored in variable ‘ch’;
        }
// Shift leftmost N2 bits to rightmost of the string of bits and vice versa)
Step 15: Read value of N2 and Shifting
Step 16: initialize array a1[l]// of length of total bits stored in step 8
Step 17: If shifting rightmost bits to leftmost, then go to step 19
Step 18: If shifting leftmost bits to rightmost
        for each bit starting from N2+1 to end of total bits
            copy each bit to array a1
        for each bit starting from 1st bit upto N2
            copy(append) each bit to a1
Step 19: for each bit starting from (totalbits-N2+1) to end of total bits
        copy each bit to array a1
        for each bit starting from 1” bit to (totalbits-N2) bit
            copy (append) each bit to a1
Step 20: print and return string st to output file
// Convert 8N bits so obtained into the text of N characters using 8-bit ASCII Code
Step 21: String s2 = ‘’;
Step 22: String s=binary; // to store the binary string
Step 23: char nextChar; // declare the variable to store next significant character in string
Step 24: for(int i = 0; i<s.length(); i += 8) //this is a little tricky, as we want [0, 7], [9, 16], etc
        {
            nextChar = Integer value of s.substring(i, i+8);
            s2 = s2 + nextChar;
        }
Step 25: return s2 to output file // return the final string
The decryption algorithm is just the reverse of the encryption algorithm

4 Implementation Of Algorithm SKG 2.5, Results And Discussion
The author has implemented the said algorithm SKG 2.5 on Java platform for different values of N1 = 3 to 8N/3 and N2 =1 to 8N-1. e.g., for input text :
Located in Kurukshetra, the land of Bhagwadgita, Kurukshetra University is a premier institute of higher learning in India. Its foundation stone was laid on January 11, 1957 by Bharatratna Dr. Rajender Prasad, the first President of the Indian Republic. The output is given Table 1 :
<table>
<thead>
<tr>
<th>S.No.</th>
<th>Comparison of Encrypted Output Text of Algorithms SKG 1.2 and SKG 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=3, N2=2 and Left Shift</td>
</tr>
<tr>
<td>1.2.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=3, N2=2 and Left Shift</td>
</tr>
<tr>
<td>1.3.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=3, N2=2 and Right Shift</td>
</tr>
<tr>
<td>1.4.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=3, N2=2 and Right Shift</td>
</tr>
<tr>
<td>1.5.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=5, N2=2 and Left Shift</td>
</tr>
<tr>
<td>1.6.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=5, N2=2 and Left Shift</td>
</tr>
<tr>
<td>1.7.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=5, N2=2 and Right Shift</td>
</tr>
<tr>
<td>1.8.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=5, N2=2 and Right Shift</td>
</tr>
<tr>
<td>1.9.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=5, N2=5 and Left Shift</td>
</tr>
<tr>
<td>1.10.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=5, N2=5 and Left Shift</td>
</tr>
<tr>
<td>1.11.</td>
<td>Encrypted Output Text of Algorithm SKG 1.2 for N1=5, N2=5 and Right Shift</td>
</tr>
<tr>
<td>1.12.</td>
<td>Encrypted Output Text of Algorithm SKG 2.5 for N1=5, N2=5 and Right Shift</td>
</tr>
</tbody>
</table>
From Table 1, it is clear that if we change even a single variable (N1, N2 or direction of shifting the characters either Left or Right) then output of the Algorithm SKG 2.5 is entirely different. The algorithm SKG 2.5 is successful for encrypting any text/string consisting of 10 or more characters. Minimum time required to decrypt any text/string consisting of 10 or more characters is about 3409 Years to find all possible combinations, which is sufficiently large to decrypt any text.

5 Conclusion

The proposed scheme named as algorithm SKG 2.5 was tested in Java platform for different values of N1 (= 1 to 8N-1) and N2(= 3 to N/3). In all cases the result came as per the literature and work seems to be satisfactory based on security metrics. It has been estimated that to crack the code we will require more time than the data will reside on the medium to travel. So, it can be said that the proposed scheme will produce an efficient secured algorithm for data transfer in both wired and wireless networks.

References