# Novel Method to Strengthen RC4 Algorithm

**Vishesh Raimugia, Rahul Shah, Kunal Sheth**

*Abstract*— **RC4 is also known as ARC4 or alleged RC4 algorithm. It is widely used in Transport layer security and WEP because of its remarkable simplicity and speed of implementation. The same algorithm is used for encryption and decryption. In our new method to increase the security the data will go through additional steps of adding irrelevant characters which adds to the confusion which complements the diffusion of the state vector in the algorithm. By this implementation we observed that weakness of RC4 can be overcome with the addition of irrelevant characters to the cipher text, making it difficult for a anybody to redesign the algorithm to get the plain text back without the base key and proper cryptanalysis.**

*Index Terms*— **RC4,WPA,WEP,KSA,PRNG**

## I. INTRODUCTION

Encryption is the process of transforming plain text into cipher text in order to conceal its meaning and to prevent it from unauthorized person from retrieving the original data[1]. Cryptography is a tool used to ensure its integrity and authenticity and to keep information confidential [1]. Cryptography algorithms are divided into two classes: Symmetric key (private key) and Asymmetric key (public key)[1]. RSA is the example of asymmetric algorithm and DES, Triple DES, AES are some other examples of symmetric algorithms. There are different kinds of schemes which are also required for digital signatures using both public and private key algorithms [2]. Various Hash algorithms such as MD5 and SHA-1 are also used in different ways to achieve security. The two ways in which the message block is processed are block and stream cipher techniques. Block Cipher technique is used in many of the Modern methods [3].

## II. RC4 AND CRYPTOANALYSIS

RC4 was designed by Ron Rivest for RSA security in 1987 [4]. Official name of RC4 is Rivest Cipher 4. RC4 is a stream cipher, symmetric key algorithm. The same algorithm is used for encryption and decryption. In this paper we will be focusing on stream cipher algorithm, in this method of processing the text is processed either byte by byte or bit-bit or character by character. In cipher stream, the state vector in our case which is also the key stream in many cases is combined with the plain text. In this to get each cipher text part we need to combine one part of the key stream with the plain text one at a time. Pseudo random key stream is typically generated serially from a random seed value which serves as cryptographic keys for decrypting the cipher text stream. RC4 is the simplest method of encrypting data. It is also faster and more suitable for streaming application. RC4 uses stream cipher method. Some of the good points about RC4 algorithm would be that it uses less amount of time and lower amount of resources and it very easy to implement as compared to the other block ciphers. Now the key stream and the data is Ex-ORed, this process is independent of the plain text.

## III. IMPLEMENTATION OF RC4 ALGORITHM

The RC4 algorithm is very simply and quite easy to explain. A variable-length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S [0], S [1],..., S [255]. At all times, the state vector which is used will contain all of the characters from 0 to 255 though not in the same order but will be a permutation of the serial sequence. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion. To achieve more amount of diffusion in the algorithm after every selection of k from the state vector it goes under another permutation operation before a new k is to be selected [7]:

### *A. Initialization of S*

To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is; S[0] = 0, S[1] = 1,..., S[255] = 255. A temporary vector, T, is also created. In order to maintain the size of the key stream to be equal to the state vector which is 256 bytes, the algorithm makes sure that if K is 256 bytes it is directly copied to the vector T, but if it is not then key-len bytes are copied to T and the same K is repeatedly copied into T until it is completely filled out. These preliminary operations can be summarized as follows:

Manuscript received November 5, 2014.

**Vishesh Raimugia**, Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India, 9819125945.
**Rahul Shah**, Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India, 9769568468.
**Kunal Sheth**, Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India, 9769515242.

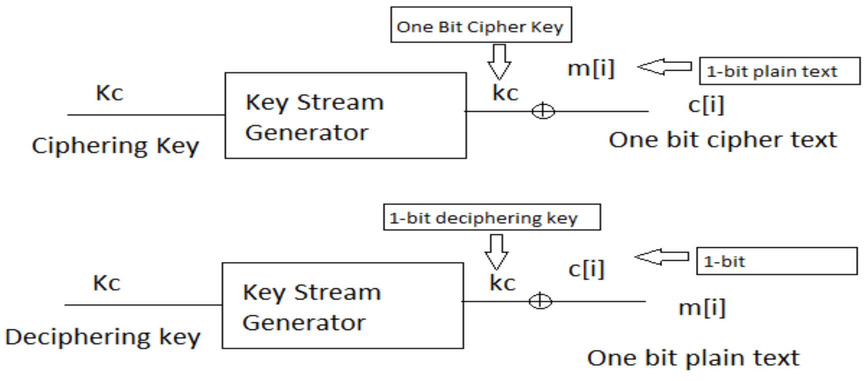

Fig. 1 Basic Implementation of algorithm

```
/* Initialization, */

for i = 0 to 255 do
        S[i] = i;
T[i] = K [i mod keylen];
```

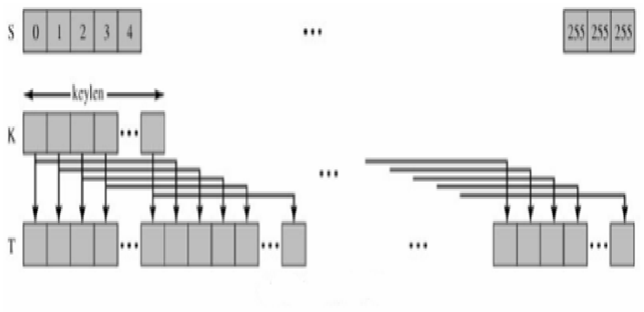

Fig. 2 Initialization Step

Next we will use T to produce the initial permutation which is another way of ordering of S. This involves starting with S [0] and going through to S [255], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]

```
/* Initial Permutation of S */

int j=0;
for(int i=0;i<256;i++)
{
   j=(j+S[i]+T[i])%256;
   int temp=S[i];
   S[i]=S[j];
   S[j]=temp;
}
```

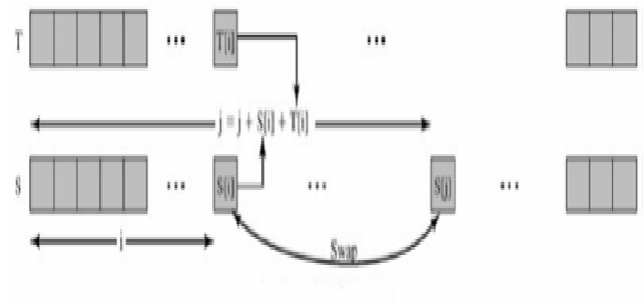

Fig. 3 Initial Permutation of S

Because the only operation on S is a swap, the only effect is a permutation. We still have not changed the original requirement as the sequence is just a swapping operation and the vector contains a permutation of 0 to 255.

*B. Stream Generation*

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S. After S [255] is reached, the process continues, starting over again at S [0],

```
/* Stream Generation */

int i=0,z=0;
j=0;
for(int l=0;l<mln;l++)
{
   i=(l+1)%256;
   j=(j+S[i])%256;
   int temp=S[i];
   S[i]=S[j];
   S[j]=temp;
   z=S[(S[i]+S[j])%256];

   //Ex-ORing
   cipher_text+=(char)(z^msgi[l]);
}
```

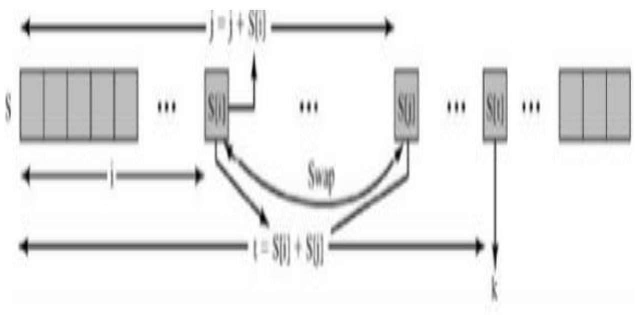

Fig 4. Stream Generation

To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of cipher text.

## IV. WEAKNESS IN RC4

All RC4 is mainly used to secure internet traffic, E-Commerce transactions and information over the network. It is also adopted by WEP and WPA to secure wireless network. Many applications like WEP (Wired Equivalent Privacy), WPA (Wi-Fi Protected Access), SSL (Secured Socket Layer) uses RC4 with improved features such as additional initialization vector to form RC4 traffic key, increase key length and size of initialization vector. RC4 has much weakness on key size. It is advised to keep the key length to be as high as possible because shorter keys are very much susceptible to attack from hackers.

## V. MODIFICATION OF RC4

In order to strengthen the RC4 algorithm, the algorithm was modified so that the plain text is put through the same steps with one change in the final cipher text. In the initial plain text irrelevant characters are added after each vowel in order to add confusion to the algorithm. Each vowel encountered in the cipher text is followed by a random irrelevant character with no dependency on the key or the plain text. Following are the steps of the algorithm after the changes

Initially, irrelevant characters are added to the plaintext.

1. Take the Base Key provided by the user and divide it into N equal parts.
2. If length of Key, k, is not perfectly divisible by N, then pad the key with zeros to make it perfectly divisible by N.
3. Create N equal sub keys from K.
4. Then ciphering the plain text using PRNG, this is not the final Cipher text.

5. Add irrelevant random character after each vowel in the Cipher text.

6. This is the final Cipher text.

7. While Deciphering, follow the same steps as Ciphering for generating the key stream.

8. Before EX-ORing remove the irrelevant character after the vowels.

The size of the message increases by the number of vowels in the message and this amount is not fixed and will be really difficult for a hacker to figure out the real cipher text. Even if the hacker is able to find the key, the real message will not be recovered easily.

## VI. SOFTWARE IMPLEMENTATION

All The code is implemented in C++ to demonstrate the changes in the state vector and the original message with respect to our algorithmic changes. Step by step result of the state vector and the plain text as well as the intermediate cipher text is given below:

```
string encrypt(string msg)
 {
   string cipher_text="";
   int ln=msg.length();
   srand(ln*time(NULL));
   for(int i=0;i<ln;i++)
   {
     cipher_text+=msg[i];

 if(msg[i]=='a'||msg[i]=='e'||msg[i]=='i'||msg[i]=='o'||msg[i]
=='u'||msg[i]=='A'||msg[i]=='E'||msg[i]=='I'||msg[i]==
'O'||msg[i]== 'U')
   {
     cipher_text+=(char)(rand()%256);
                //%256 means we are restricting the
                random character in the range of 256 which
                is added to the //cipher text.
   }
   }
   return cipher_text;
 }
```

This method is applied twice in our approach; initially it is applied to the plain text as it is to add confusion into the message and the next when the plain text is Ex-ORed with the state vector which adds diffusion as well as confusion into the algorithm and thereby making it more robust.

## VII. RESULTS

This algorithm was implemented in C++ and was tested to calculate the running time of the methods for encryption and decryption for various file sizes. We used file input output to supply key stream and the message to be decoded.

Table 1. Result of above algorithm

| File Size(1Kb) | Encryption Time | Decryption Time |
|---|---|---|
| 1 | 0.00110s | 0.00010s |
| 10 | 0.00230s | 0.00100s |
| 20 | 0.00360s | 0.00300s |
| 50 | 0.00820s | 0.00700s |
| 100 | 0.01700s | 0.01300s |

## VIII. CONCLUSION

We can observe that as file size increases encryption, decryption time also increases however due to more number of characters present the hackers may not get the exact trace of the data and it will be more time consuming as well as tedious for hackers to get the original message. The basic idea of increasing the confusion and diffusion is achieved using this method and also the simplicity of the algorithm also is an advantage.

## IX. FUTURE WORK

There will be increase in the running time of this algorithm, the encryption and decryption a block of data will increase the security at the cost of increase in its run time. This algorithm is still widely used in WEP and other transport layer security protocols so there is still need for more optimization to improve its security and make it more robust.

## ACKNOWLEDGEMENT

Special thanks to Mrs. Lakshmi Kurup for guiding us through the implementation of the algorithm and enriching the quality of the research.

## REFERENCES

[1] A.Mousa, " Data Encryption Performance Based on Blowfish," 47th International symposium ELMAR 2005 focused on multimedia systems and applications,pp.131-134, Zadar, Croatia 08-10, June 2005

[2] A.Mousa, "Evaluation of RC4 Algorithm for Data Encryption,"International Journal of Computer Science and Application, Vol.3, No.2, June 2006.

[3] R.L.Rivest, "The RC4 Encryption Algorithm," RSA Data Security, Inc. March 1992.

[4] N.Chandra, "Enhancing RC4 Algorithm for WEP Protocol using FAKE character Insertions and compression Techniques," 2005.

[5] C.Pu , W.Y.Chung, " Group key update method for improving RC4 cipher stream in Wireless Sensor Network,", International Conference on convergence Information Technology, 2007.

[6]

**Vishesh Raimugia,** B.E. in Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India.

**Rahul Shah,** B.E. in Computer Engineering, Dwarkadas J Sanghvi College of Engineering, Mumbai, India.

**Kunal Sheth,** B.E. in Computer Engineering, Dwarkadas J Sanghvi College of Engineering, Mumbai, India.