Enhancing Security of Improved RC4 Stream Cipher by Converting into Product Cipher

Nishith Sinha
Dept. of Computer Science & Engineering
Manipal Institute of Technology
Manipal University, Manipal

Mallika Chawda
Dept. of Computer Science & Engineering
Manipal Institute of Technology
Manipal University, Manipal

Kishore Bhamidipati
Assistant Professor
Dept. of Computer Science & Engineering
Manipal Institute of Technology
Manipal University, Manipal

ABSTRACT
RC4 is one of the most widely used stream ciphers which finds its application in many security protocols such as Wi-Fi Protocol Access (WPA) and Wired Equivalence Privacy (WEP). RC4 algorithm has several weaknesses. In order to overcome those weaknesses and enhance its security, numerous modifications have been suggested. These amendments destroy the basis of various cryptanalysis attacks on RC4. One such significant modification was the algorithm proposed by Jian Xie et al [1]. In this paper, we propose an enhancement to this algorithm by converting it into a product cipher and thereby enhancing its security.

General Terms
Cipher, Security, Stream Ciphers, Product Ciphers

Keywords
Cryptanalysis, RC4

1. INTRODUCTION
Stream ciphers process one bit or one byte at a time for encryption or decryption. One of the cornerstones of a stream cipher is the pseudorandom bit generator. The pseudorandom bit generator takes a key as the input and produces a stream of random bits as the output using a deterministic algorithm. This stream of random bits is known as keystream. The keystream is then combined, one bit or one byte at a time with the plaintext to produce the corresponding ciphertext. RC4 is a prime example of stream cipher which is widely used in many security protocols such as Wi-Fi Protocol Access (WPA) and Wired Equivalence Privacy (WEP). These protocols use RC4 because it is fast, utilizes less resource and is easy to implement [2, 3].

The RC4 algorithm which was initially proposed in 1987 uses a variable length key and its operations are byte oriented. It uses a deterministic algorithm to produce a random permutation. The RC4 algorithm can be divided into two phases: Key Scheduling Algorithm (KSA) and Pseudo Random Generation Algorithm (PRGA). KSA makes use of the variable length key to initialize a 256 Bytes array S. This operation is known as the initialization of the S-block. The key is then used to produce a random permutation of the initialized array S. This marks the end of the KSA phase. Once the array S has been initialized, the key is no longer used. PRGA phase now begins. It produces a random sequence of words from the permutation in S known as the key stream. During the decryption process, the key stream is then XORed with the plaintext to produce the ciphertext. During decryption, the ciphertext is XORed with the keystream to produce the plaintext. The algorithm can be summarized as:

1.1 Key Scheduling Algorithm

```
for  i = 0 to 255
S [i] = i;
j = 0;
for i = 0 to 255
j = ( j + S [i] + K [i % key_length] ) % 256;
swap (S[i], S[j]);
```

1.2 Pseudo- Random Generation Algorithm

```
i = 0, j = 0;
while ( true )
i = ( i + 1 ) % 256;
j = ( j + S[ i ] ) % 256;
swap ( S[i], S[j] );
t = ( S[i] + S[j] ) % 256;
k= S[t];
```

For encryption, the keystream k is XORed with the next byte of plaintext to produce the ciphertext. In case of decryption, the keystream is XORed with the ciphertext to produce the plaintext.

However, this algorithm suffers from many weaknesses that have been exposed by various cryptanalysis attacks. The cryptanalysis of RC4 can be broadly divided into two categories: attacks focused on exploiting the randomness of KSA and attacks focused on exploiting the properties of the internal states of PRGA. Fluhrer et al. [4] discovered a major weakness in the RC4 algorithm i.e. it is possible to completely attack RC4 if some portion of the secret key is known. Paul and Maitra [5] detected that it is possible to derive the secret key from the initial state array using biases. Klein [6] identified the statistical relation in between the output byte generated and the value of S[j] at the time of output generation. A number of attempts were made to improve RC4, making it resistant to the weaknesses identified. Paul and Preneel [7] developed a new algorithm RC4A, which was resistant to most attacks that applied on RC4. However, even in RC4A, there existed certain relations between the internal states of the S-box. These relations were destroyed by "An Improved RC4 Algorithm", proposed by Jian Xie et al [1]. However, there is scope available to further enhance this algorithm because it only uses permutation for encryption. In this paper, we impose substitution to the Improved RC4

Algorithm making it a product cipher, thereby improving its security.

2. PROPOSED ALGORITHM

The proposed algorithm is an extension of the Improved RC4 Stream Cipher Algorithm proposed by Jian Xie et al. [1]. It uses three secret keys- two secret keys K_1 and K_2 as seeds for Enhanced RC4 and K_3 as the key for Vigenère Cipher substitution. It also uses two S-Boxes S_1 and S_2. Both of them contain N elements from 0 to N-1. The Key Scheduling Algorithm is the same as original RC4 except that it uses two S-boxes instead of one, as proposed in the Enhanced RC4 Algorithm.

In PRGA two output streams are obtained from S_1 and S_2. The output streams are XORed with each other. The resulting stream is then XORed with the plaintext P, to obtain the intermediary ciphertext, X. This intermediary ciphertext is then fed as the input of Vigenère Cipher. In this final phase of the encryption process, substitution on the intermediary ciphertext X takes place using the key K_3. This gives us the final ciphertext C. The encryption process is stated below-

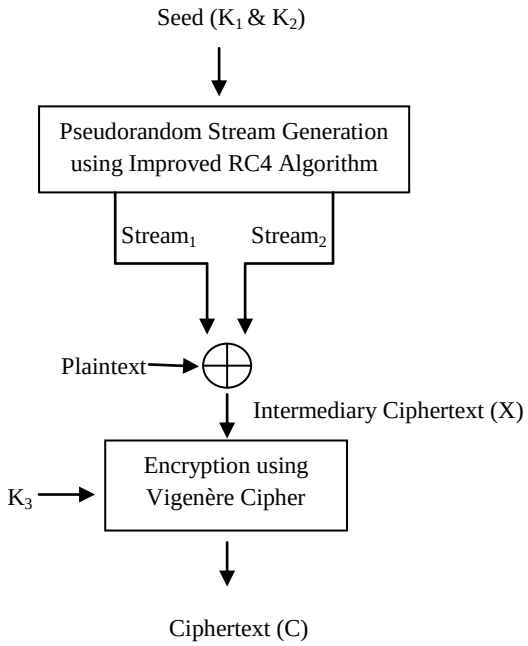

Fig 1: Encryption Process using Proposed Algorithm

In the algorithm proposed, Vigenère Cipher is used in the final phase of the encryption process to perform substitution. Vigenère Cipher is a polyalphabetic stream cipher. Each character of the intermediary ciphertext X is encrypted using K_3 as the key. This final phase of encrypting using Vigenère Cipher can be summarized as follows-

Encryption: $C_a = (X_a + k_a) \bmod 256$ – (Equation 1)

Where $C = C_0 \ldots C_n$ is the Ciphertext, $X = X_0 \ldots X_n$ is the Intermediary Ciphertext and $K_3 = k_0 \ldots k_m$ is the key used.

Decryption process is similar to encryption obeying the laws of symmetric cryptography algorithms. The ciphertext C is first decrypted using Vigenère Cipher with K_3 as the key. The output of this process is the intermediary plaintext Y. In the next phase, keys K_1 and K_2 are used as the seed for the Pseudorandom Stream Generator using Improved RC4. Two output streams are obtained as the output of the stream generation phase. The output streams are XORed with each other. The resulting stream is then XORed with the intermediary plaintext Y to give the final plaintext P. Decryption Process is stated below-

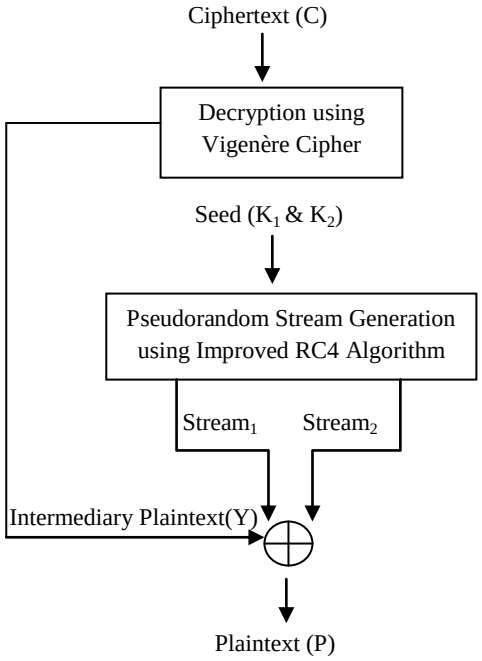

Fig 2: Decryption Process using Proposed Algorithm

The proposed algorithm can be summarized as follows-

Encryption:

```
for i=0 to 255
      S1[i]=i;
      S2[i]=i;

j1 = j2 = 0;
for i = 0 to 255
      j1 = ( j1 + S1[i] + K1[i] ) mod 256;
      swap ( S1[i], S1[ j1]);
      j2 = ( j2 + S2[i] + K2[i]) mod 256;
      swap(S2 [i], S2[j2]);

i = j1= j2=a=0;
while ( true )
      i = ( i+1 )%256;
      j1 = j1 + S1[i];
      swap ( S1[i], S1[ j1] );
      j2= j2 + S2[i];
      swap ( S2 [i], S2[ j2] )
      Stream1= S1 [ (S1 [i]+ S1[j1]) mod 256 ];
      Stream2= S2 [ (S2 [i]+ S2[ j2]) mod 256 ];
      swap ( S1 [S2 [j1]], S1 [S2 [j2]] );
      swap ( S2 [S1 [j1]], S2 [S1 [j2]] );
      X[a] = Stream1 XOR Stream2 XOR P[a];
      C[a] = (X[a] + K3 [a]) mod 256
      a=a+1;
```

Decryption:

```
for a=0 to length(Y) – 1

      if K3 [a] < C[a] then
         Y[a]  = (C[a] – K3[a]) mod 256
```

```
    else
        Y[a] = (256 + C[a] – K3[a]) mod 256

for i=0 to 255
    S1[i]=i;
    S2[i]=i;

j1 = j2 = 0;
for i = 0 to 255
    j1 = ( j1 + S1[i] + K1[i] ) mod 256;
    swap (S1[i], S1[ j1]);
    j2 = (j2 + S2[i] + K2[i]) mod 256;
    swap (S2 [i], S2[j2]);

i = jl= j2 = a= 0;
while ( true )
    i = ( i+1 )%256;
    jl = jl + S1[i];
    swap ( S1[i], S1[ j1] );
    j2=  j2 + S2[i];
    swap ( S2 [i], S2[ j2] )
    Stream1= S1 [ (S1 [i]+ S1[j1]) mod 256 ];
    Stream2= S2 [ (S2 [i]+ S2[ j2]) mod 256 ];
    swap( S1 [S2 [j1]], S1 [S2 [j2]] );
    swap( S2 [S1 [j1]], S2 [S1 [j2]] );
    P[a] = Stream1 XOR Stream2 XOR Y[a];
    a=a+1;
```

3. RESULTS

The fundamental intent behind the proposed algorithm is to convert the Improved RC4 Algorithm [1] which is a permutation cipher into a product cipher by imposing substitution using Vigenère Cipher. This would ensure higher levels of security compared to Improved RC4 algorithm, making it a better choice for confidentiality intrinsic applications. Another important factor to be considered is the time taken for encryption/decryption. Experimental results show that there is a 0.8% to 1% increase in the time required for encryption/decryption. This increase in time is negligible validating the practicality of the algorithm. The algorithm stated in the paper was implemented on Intel (R) Core (TM) i3-2310M CPU @ 2.10 GHz having 3.84 GB of usable RAM. Further in this section, we describe a few test cases and the time required for encryption/decryption for a given size of plaintext.

Table 1: Encryption for Plaintext (P) = TheQuickBrownFoxJumpedOverTheLazyDog

Plaintext	TheQuickBrownFoxJumpedOverTheLazyDog
K_1	cherryblossom
K_2	deception
X	'Ç=$"™bóK+ñièžùŠ{±ÂèÙÿVhØ×'õL2‡"
K_3	baskerville
Ciphertext	ˆ:¨‰„ kqÇU¬ž\ÎZböç$ILj»ÚN@"a±"mú

Table 2: Decryption for Ciphertext (C) = ˆ:¨‰„ kqÇU¬ž\ÎZböç$ILj»ÚN@"a±"mú

Ciphertext	ˆ:¨‰„ kqÇU¬ž\ÎZböç$ILj»ÚN@"a±"mú
K_1	cherryblossom
K_2	deception
K_3	baskerville
Y	'Ç=$"™bóK+ñièžùŠ{±ÂèÙÿVhØ×'õL2‡"
Plaintext	TheQuickBrownFoxJumpedOverTheLazyDog

Table 3: Encryption for Plaintext (P) = ONCEthereWASaDOGnamedROVER!

Plaintext	ONCEthereWASaDOGnamedROVER!
K_1	amnesia
K_2	trichotillophobia
X	ë9ž©¶i•.?Zßr-ÇD/"¼yaÓm´v
K_3	arachnoid
Ciphertext	]š$Øþ' …p½vMá–+¥¡ƒáÏBÖ×

Table 4: Decryption for Ciphertext (C) =]š$Øþ' …p½vMá–+¥¡ƒáÏBÖ×

Ciphertext	]š$Øþ' …p½vMá–+¥¡ƒáÏBÖ×
K_1	amnesia
K_2	trichotillophobia
K_3	arachnoid
Y	ë9ž©¶i•.?Zßr-ÇD/"¼yaÓm´v
Plaintext	ONCEthereWASaDOGnamedROVER!

The algorithm was run for a large number of test cases. The time required for encrypting/decrypting was observed for plaintext varying between 100kB to 1000kB. The values obtained were compared against values obtained by encrypting using Improved RC4 Algorithm [1] as shown in Fig. 1, Appendix I. The increase in the time consumed in this while comparing against the Improved RC4 Algorithm is measured in Fig. 2, Appendix I. An increase of 0.8% to 1% was observed. Considering the computational capabilities of modern computers, this increase is negligible.

4. CONCLUSION

The algorithm proposed in the paper enhances the security of Improved RC4 algorithm by imposing substitution, thereby converting it into a product cipher. Time taken for encryption/decryption using the proposed algorithm is marginally more than the Improved RC4 Algorithm. Experimental results show that there is a mere 0.8% - 1% increase in the time required for encryption/ decryption. These characteristics of the proposed algorithm make it a better candidate for practical applications as compared to the Improved RC4 Algorithm.

The algorithm suggested in the paper uses Vigenère Cipher for substitution. One of the weaknesses associated with using Vigenère Cipher is that it is not resistant to Kasiski test [8]. Security of the proposed algorithm can be compromised because Kasiski test can help determine the length of the key used for Vigenère Cipher [9]. As a part of future research work, researchers can look at improving the proposed algorithm making it resistant to Kasiski test.

Another possible scope of future research work can be introducing parallelism in the proposed algorithm. The decryption process consists of two major parts – Pseudorandom Stream Generation using Improved RC4 algorithm and decryption using Vigenère Cipher. Both these processes are independent of each other. Hence, we can exploit parallelism by concurrent execution of these two processes. This would reduce the time required for decryption, thereby enhancing its efficiency.

5. REFERENCES

[1] Jian Xie, Xiaozhong Pan, "An Improved RC4 Stream Cipher", International Conference on Computer Application and System Modeling (ICCASM), 2010

[2] Suhaila Omer Sharif, S.P. Mansoor, "Performance analysis of Stream Cipher algorithms", 3rd International Conference on Advanced Computer Theory and Engineering (ICATE), 2010..

[3] C.S Lamba, "Design and Analysis of Stream Cipher for Network Security", 2nd International Conference on Communication Software and Networks,2010

[4] S.Fluthrer, I. Mantin, A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", SAC2001 (S. Vaudenay, A. Youssef,eds.), col. 2259 of LNCS, pp. 1-24, springer-Verlag, 2001

[5] G. Paul, S.Maitra, "RC4 state in formation at Any Stage Reveals the Secret Key ", presented in the 14th Annual Workshop on Selected Areas in Cryptography, SAC 2007, August 16-17, Ottawa, Canada, LNCS (Springer) pages 360-377.

[6] A Klein, Attacks on the RC4 Stream Cipher, cage.ugent.be/~klein/RC4/RC4-en.ps

[7] S. Paul, B. Preneel, ―A New Weakness in the RC4 Key stream Generator and an Approach to Improve the Security of the Cipher‖, Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers, vol.3017, no., pp.245,259, 2004.

[8] William Stallings: "Cryptography and Network Security: Principles and Practices" 4th Edition,

[9] Cryptanalysis of Vigenère Cipher- http://www.nku.edu/~christensen/section%2012%20vigenere%20cryptanalysis.pdf

APPENDIX I

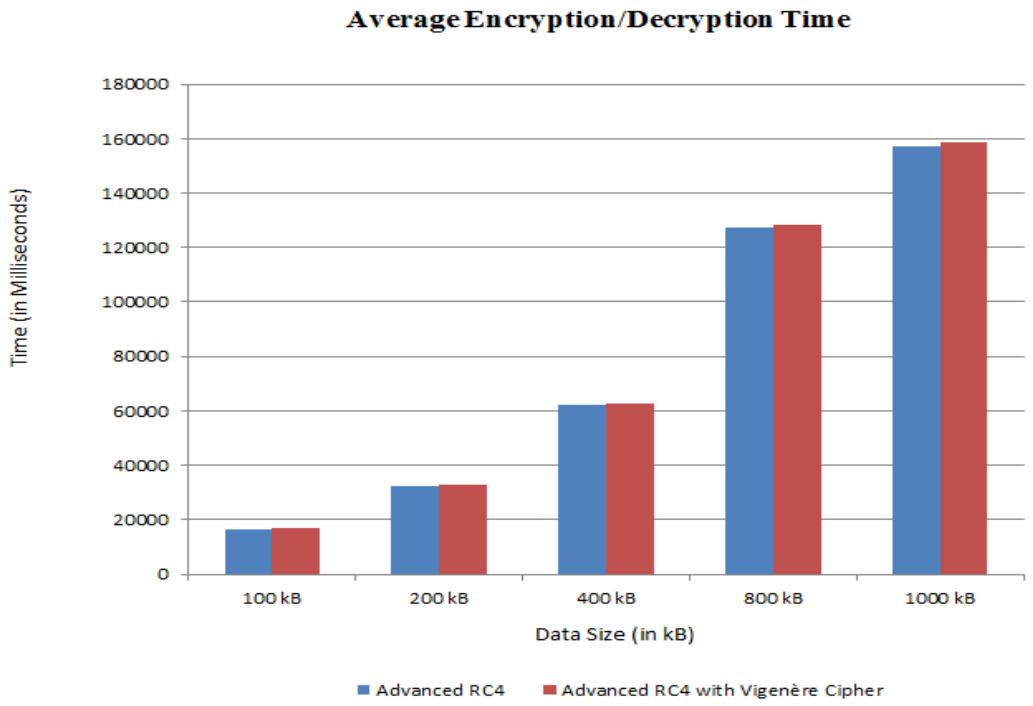

Fig. 1: Time required for Encryption using Proposed Algorithm and Improved RC4 Algorithm

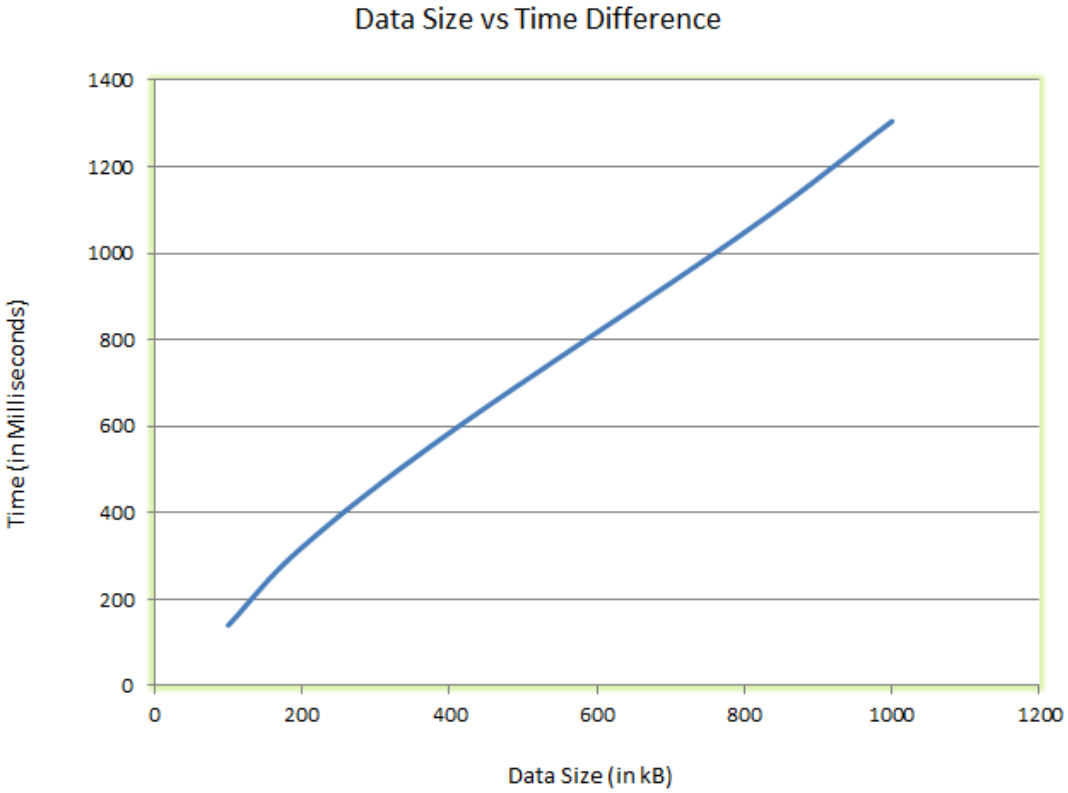

Fig. 2: Time increase using Proposed Algorithm compared against Improved RC4 Algorithm