

RC4-2S: RC4 Stream Cipher with Two State Tables

Maytham M. Hammood, Kenji Yoshigoe and Ali M. Sagheer

Abstract One of the most important symmetric cryptographic algorithms is Rivest Cipher 4 (RC4) stream cipher which can be applied to many security applications in real time security. However, RC4 cipher shows some weaknesses including a correlation problem between the public known outputs of the internal state. We propose RC4 stream cipher with two state tables (RC4-2S) as an enhancement to RC4. RC4-2S stream cipher system solves the correlation problem between the public known outputs of the internal state using permutation between state 1 (S_1) and state 2 (S_2). Furthermore, key generation time of the RC4-2S is faster than that of the original RC4 due to less number of operations per a key generation required by the former. The experimental results confirm that the output streams generated by the RC4-2S are more random than that generated by RC4 while requiring less time than RC4. Moreover, RC4-2S's high resistivity protects against many attacks vulnerable to RC4 and solves several weaknesses of RC4 such as distinguishing attack.

Keywords Stream cipher • RC4 • Pseudo-random number generator

This work is based in part, upon research supported by the National Science Foundation (under Grant Nos. CNS-0855248 and EPS-0918970). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author (s) and do not necessarily reflect the views of the funding agencies or those of the employers.

M. M. Hammood

Applied Science, University of Arkansas at Little Rock, Little Rock, USA

e-mail: mmhammood@ualr.edu

K. Yoshigoe (✉)

Computer Science, University of Arkansas at Little Rock, Little Rock, AR, USA

e-mail: kxyoshigoe@ualr.edu

A. M. Sagheer

College of Computer, University of Anbar, Anbar, Iraq

e-mail: ali_makki@ieee.org

1 Introduction

Cryptographic algorithms that can provide fast implementation, small size, low complexity, and high security for resource-constrained devices such as wireless sensor devices are imperative. Conventional cryptographic algorithms are very complex and consume significant amount of energy when used by resource-constrained devices for the provision of secure communication, and public key algorithms are still not feasible in sensor networks for several reasons including limited storage and excessive energy usage [1]. Therefore, security schemes should rely on a symmetric key cryptography especially when systems have limited hardware resources.

There are a number of stream cipher algorithms presented to implement high performance software including IDEA, ORYX, LEVIATHAN, MUGI, RC4, Helix, SEAL, SOBER, and SNOW. One-time pad, which is stronger than all these aforementioned algorithms, is unbreakable since it never uses the same key more than once. Consequently, it is impossible to deduce any information about the key from ciphertext by an attacker. The robustness of stream ciphers depends on Pseudo-Random Number Generator (PRNG) which has proved to be resistant to the attacks if it passes the statistical tests.

RC4 is a proprietary stream cipher which was designed in 1987 by Ron Rivest. RC4 is widely used in security software based on stream cipher including one in the encryption of traffic to and from secure web sites such as Transport Layer Security (TLS), Secure Socket Layer (SSL), and Wired Equivalent Privacy (WEP) implementations. RC4 is fast in comparison to other algorithms and it has a simple design hardware implementation [2]. For instance, RC4 is five times faster than Data Encryption Standard (DES) and fifteen times faster than Triple-DES [3].

Many stream ciphers use de-facto RC4 standard for meeting specific requirements such as limited storage size and power of devices; however, there are many weaknesses found in stream ciphers caused by mathematical relations between the key, ciphertext, and plaintext with which attackers can assess the security of the cryptographic algorithms via cryptanalysis. Thus the goal is to create a sequence of keys that approaches to true randomness [4]. The rest of the paper is organized as follows: [Section 2](#) reviews related works. [Section 3](#) presents the description of RC4, and [Sect. 4](#) shows some weaknesses of RC4. We then present our proposed algorithm to enhance RC4 technique in [Sect. 5](#), followed by description of randomness testing and NIST in [Sect. 6](#). [Section 7](#) presents analysis and implementation followed by the conclusion in [Sect. 8](#).

2 Related Work

Numerous researchers attempt to enhance the RC4 and create variant algorithms. Paul and Preneel [5] explored a new statistical weakness in the first two output bytes of the RC4 key stream generator. They showed that the number of outputs

required distinguishing the output of the RC4 of the random sequence with the presence of which bias is 128, and they recommended using 256 to overcome this bias. The authors were also creating a new pseudo-random number generator, RC4A, which is characterized by being more resistant against most attacks that apply to RC4 especially the weakness of distribution in the first two output bytes. However, the RC4A did not completely remove the autocorrelations, and it was broken with a distinguishing attack by Maximov [6]. Zoltak [4] proposed Variably Modified Permutation Composition (VMPC) which was designed to be efficient in software implementations and solving a weakness found in the RC4 Key Scheduling Algorithm (KSA) that was described by Fluhrer et al. in [7]. The structure of Pseudo-Random Generation Algorithm (PRGA) in VMPC was more complex in comparison with the RC4 that makes it more resistant against attacks. However, it was again broken by distinguishing attack [6].

Yu and Zhang [8] presented RC4 state combined with the hash function without affecting the simplicity and efficiency. The RC4 state based on hash function can generate Message Authentication Code (MAC). The enhancement includes the offset, forward, and backward properties of RC4 states where the authors use offset to ignore the first few bytes of the key and started encrypt the data in determine position which has led to increase the time of execution. Pardeep and Pateriya [9] proposed Pardeep Cipher (PC-RC4) which is adding a new improvements and extension to RC4 algorithm. In the PC-RC4, randomness in KSA and PRGA are improved to make it stronger but it again increased the time of execution [9]. Kadry and Smaili [10] presented Vigenère RC4 (VRC4) which is a combination of the RC4 and the poly alphabetic cipher Vigenère. The plain text encrypted by using the classic RC4 cipher followed by re-encrypting by Vigenère which results in increased time of execution. Mousa and Hamad examined the analysis of the effect of different parameters of the RC4 algorithm such as the execution time and the file size where they conclude that the speed of encryption and decryption time is affected by length of encryption key and the size of data file [11]. Yao, et al. presented analysis and enhancement of the security of RC4 algorithm by using public key encryption with RC4 which has led to increase the size of the system and the time of execution [12]. Hammood et al. proposed an RRC4 random initial state algorithm in which a new enhancement of RC4, and its improved randomness compared against the traditional RC4. However, the issue of the speed was not addressed [13].

3 Description of RC4

RC4 design avoids the use of Linear Feedback Shift Registers (LFSRs) where many stream cipher algorithms depend on it, especially in hardware. It attempts to achieve highest randomness via swapping of elements in arrays. The RC4 algorithm has variable key length which ranges between 0 to 255 bytes for initializing 256-byte array in initial state by elements from $S[0]$ to $S[255]$. As recommended in [7, 14], RC4 must use a key longer than 128 bytes. The algorithm consists of

KSA and PRGA which are executed sequentially. RC4 Key is initialized by KSA while the pseudo-random number is produced by PRGA.

The pseudo codes for the two parts of RC4 algorithm are shown in Algorithm 1 and Algorithm 2 where m is the message length of a plain text, L is the length of the initial key in bytes, N is the size of the array or state S , and i and j are index pointers. Such a self-modifying lookup table is simple and has been applied efficiently in software [15]. The output of the PRGA algorithm is a key sequence that will be XOR-ed with plaintext (or ciphertext) to get ciphertext (or plaintext).

Sharif and Mansoor provided comparison between different encryption algorithms using different data sizes and key sizes. The simulation results clarified preponderance of RC4 algorithm over other algorithms in terms of speed and throughput [16].

Algorithm1. KSA for RC4

INPUT: $K [k_1, k_2, \dots, k_L], m$
OUTPUT: S
 1. $S[i] = i$, for $i=0, 1, 2, \dots, 255$
 2. $j \leftarrow 0$;
 3. For $i \leftarrow 0$ to 255 Do
 3.1. $j \leftarrow (j + S[i] + K[i \bmod L]) \bmod 256$
 4. Swap $S[i]$ with $S[j]$
 5. Return (S)

Algorithm 2. PRGA for RC4

INPUT: State S
OUTPUT: Key sequence $Kseq$
 1. $j \leftarrow 0$
 2. $i \leftarrow 0$
 3. While not end of sequence Do
 3.1. $i \leftarrow (i+1) \bmod 256$
 3.2. $j \leftarrow (j + S[i]) \bmod 256$
 3.3. Swap $S[i]$ with $S[j]$
 3.4. $Kseq \leftarrow S[(S[i] + S[j]) \bmod 256]$
 4. Return ($Kseq$)

4 The Weaknesses of RC4

Several weaknesses in RC4 were identified. Some of these weaknesses are simple and can be solved easily, but others are critical because they can be exploited by the attackers. Two of the problems of RC4 are 1) the weakness of the KSA and 2) the weakness of relations between the S-box in different time. Initial state of the PRGA achieves a pretty good efficiency against a number of attempted attacks. In [17] Mantin and Shamir, distinguished statistical weakness inside RC4 algorithm that the probability of the second round output byte generated is zero output byte is twice than other values, which can exploited this weakness by practical ciphertext only attack. This ciphertext only attack is limited to broadcast applications where different keys are used to encrypt the same plaintext for multiple recipients. There are many other attacks described in [7] such as subkey guessing attack, linear consistency attack, inversion attack, etc. In addition, an algebraic attack is a new type of higher order correlation attack. Since a fairly straight forward approach such as brute force attack infers the internal state of the PRGA, increased internal state size is recommended, yet it results in increased encryption and decryption time.

5 RC4 Stream Cipher With Two State Tables

As discussed in the previous section, RC4 has significant number of weaknesses in the phases of KSA and PRGA. In this section, we propose RC4 stream cipher with two state tables (RC4-2S) algorithm as an improvement to RC4. It is one of the RC4 stream cipher algorithm family proposed to reduce the correlation problem between the public known outputs of the internal state while improving the speed of the encryption and decryption. The new algorithm consists of initialization phase (KSA) and output phase (PRGA) as shown in Algorithm 3 and 4, respectively. All addition operations are carried out modulo N . KSA takes a key, k , consisting of 16 n -bit words. After the setup, the round algorithm is executed once for each word output. In reality, all practical applications of the developed RC4 is implemented with $n = 8$ in which case all entries of S along with i and j are bytes.

In the first phase of KSA, S_1 is filled from 0 to $(N/2)-1$ and S_2 gets the remaining $N/2$ numbers from $N/2$ to $N-1$. The input secret key, k , is used as a seed for the two states S_1 and S_2 . k is used to make permutation and swapping of the elements of S_1 and S_2 . Therefore S_1 and S_2 become two secret random inputs for second phase.

In the second phase, S_1 and S_2 produce two keys in each loop cycle instead of one as with the standard RC4. In this algorithm, there are more elements to be swapped between S_1 and S_2 by three pointers: i , $j_1 = j_1 + S_1[i]$, and $j_2 = j_2 + S_2[i]$ in the S -box. S_1 and S_2 in PRGA are used to produce the sequence of output stream which is XOR-ed with plaintext (or ciphertext) to generate ciphertext (or plaintext).

Algorithm3. KSA for RC4-2S

```

INPUT:  $k, m$ 
OUTPUT:  $S_1, S_2$ 
1. For  $i \leftarrow 0$  to  $N/2 - 1$  Do
     $S_1[i] \leftarrow i$ 
2. For  $i \leftarrow N/2$  to  $N - 1$  Do
     $S_2[i - N/2] \leftarrow i$ 
3.  $j \leftarrow 0$ .
4. For  $i \leftarrow 0$  to  $N/2 - 1$  Do {
    4.1.  $j \leftarrow (j + S_1[(i + k[i \bmod L]) \bmod N/2] + k[i \bmod L]) \bmod N/2$ 
    4.2. Swap  $S_1[i]$  with  $S_1[j]$  }
5.  $j \leftarrow 0$ .
6. For  $i \leftarrow 0$  to  $N/2 - 1$  Do {
    6.1.  $j \leftarrow (j + S_2[i] + k[i \bmod L]) \bmod N/2$ 
    6.2. Swap  $S_2[i]$  with  $S_2[j]$ 
Return ( $S_1$  and  $S_2$ )

```

Algorithm4. PRGA for RC4-2S

```

INPUT:  $S_1, S_2$ 
OUTPUT: Key sequence  $Kseq$ 
1.  $i, j_1, j_2 \leftarrow 0$ 
2. While not end of half sequence Do
    2.1  $i \leftarrow (i+1) \bmod N/2$ 
    2.2  $j_1 \leftarrow (j_1 + S_1[i]) \bmod N/2$ 
    2.3 Swap  $S_1[i]$  with  $S_2[j_1]$ 
    2.4  $t_1 \leftarrow S_1[(S_1[i] + S_1[j_1]) \bmod N/2]$ 
    2.5  $j_2 = j_2 + S_2[i] \bmod N/2$ 
    2.6 Swap ( $S_2[i]$  with  $S_1[j_2]$ )
    2.7  $t_2 = S_2[(S_2[i] + S_2[j_2]) \bmod N/2]$ 
    2.8  $Kseq = [t_1, t_2]$ 
Return ( $Kseq$ )

```

RC4-2S is faster than RC4 because RC4-2S requires two swaps and five modulo functions to generate two bytes of key per iteration in the PRGA algorithm

while RC4 requires one swap and three modulo functions to generate only one byte of key.

This technique combines increasing randomness of the initial internal states with permutation of the two state tables during key generation to solve the correlation problem between the public known outputs of the internal.

6 Randomness Test

The binary generated sequences are tested by National Institute of Standards and Technology (NIST) Test Suite which is a statistical package for random number generation test which consists of 16 statistical tests to measure the randomness of the output sequences of true random number generators or pseudo-random number generators.

7 Result Analysis

The design of the RC4-2S was done using MATLAB and the tests of this PRNG were done using NIST STS-1.6 [18]. Key generation time of RC4-2S was faster than that of the original RC4 by approximately 20 % for various amount of keys generated as shown in Table 1. This is expected because RC4-2S, which can generate two bytes of key per iteration in the PRGA algorithm, requires two swaps and five modulo functions while RC4, which generates only one byte of key, requires one swap and three modulo functions.

We check the produced binary sequence from RC4-2S by NIST statistical tests. The probability of a good or bad random number generator is represented by the p value. Testing process compared p -value to 0.01. If the p -value is more than 0.01, then the process accepts the sequence, else rejects the sequence because the sequence is non-randomness. Conversely, some tests accept large sizes of sequence and fail in the small size as well as other tests accept both sizes. In our program, we use a large size 134,000 bytes (1,072,000 bits); generated by each keys and these sequences were tested, and subsequently calculated the average of the p -values result from these tests. As shown in Table 2, the p -values are acceptable when greater than 0.01, and the produced sequence can be deemed random, uniformly distributed, and suitable for cryptography.

Table 1 Key generation time for RC4 and RC4 -2S

Amount of keys in KB	RC4 (ms)	RC4-2S (ms)
100	232.726	189.4
500	1372.36	970.195
1000	2407.425	1932.196

Table 2 Result of running the NIST suite over the set data produced by the proposed RC4-2S and standard RC4

Test no.	Statistical test name	RC4		RC4-2S	
		p-value	Conclusion	p-value	Conclusion
1	Approximate entropy	0.444380	SUCCESS	0.273897	SUCCESS
2	Block frequency	0.453253	SUCCESS	0.529301	SUCCESS
3	Cumulative sums (Forward)	0.406619	SUCCESS	0.544359	SUCCESS
4	Cumulative sum (Reverse)	0.394456	SUCCESS	0.448504	SUCCESS
5	FFT	0.492610	SUCCESS	0.451043	SUCCESS
6	Frequency	0.426195	SUCCESS	0.502214	SUCCESS
7	Lempel–Ziv compression	1.000000	SUCCESS	1.000000	SUCCESS
8	Linear complexity	0.425838	SUCCESS	0.631859	SUCCESS
9	Longest runs	0.462975	SUCCESS	0.460664	SUCCESS
10	Non periodic templates	0.501925	SUCCESS	0.485355	SUCCESS
11	Overlapping template	0.395097	SUCCESS	0.418021	SUCCESS
12	Random excursions	0.500386	SUCCESS	0.527889	SUCCESS
13	Random excursions variant	0.530319	SUCCESS	0.439212	SUCCESS
14	Rank	0.451984	SUCCESS	0.382616	SUCCESS
15	Runs	0.575766	SUCCESS	0.604976	SUCCESS
16	Serial	0.496741	SUCCESS	0.386188	SUCCESS
17	Universal statistical	0.356271	SUCCESS	0.542075	SUCCESS

If the tests give p-value asymptotically to 1, then the sequence appears to have perfect randomness. A p-value of zero indicates that the sequence appears to be completely nonrandom. The SUCCESS indicates the sequence is acceptable and has good randomness, where the FAILURE mean the sequence is not acceptable due to non-randomness. There are some statistical tests of PRBG that are very common and must be included in test suite such as *Runs* test, *Frequency* test, and *Universal* test [19]. In these tests, the p-values of the proposed RC4-2S algorithm are greater than the p-values of the standard RC4 as shown in the Table 2. Moreover, RC4-2S is better than RC4 in most of the other tests.

8 Conclusion

The RC4 cipher system is an important encryption algorithm that can be used to protect the information on the common channel as its implementation is simpler and its cryptographic function is faster than that of DES. The proposed RC4 stream cipher with two state tables (RC4-2S) offers an enhanced randomness in the generated key sequences while the key generation time of RC4-2S is faster than that of RC4. The generated output sequences of RC4-2S have passed the NIST suite of statistical tests. The suggested RC4-2S algorithm is not a complicated one. Thus, it can be implemented in either software or hardware.

References

1. Sharma K, Ghose MK, Kumar D, Singh RPK, Pandey VK (2010) A comparative study of various security approaches used in wireless sensor networks. *Int J Adv Sci Technol* 177(77)
2. Gupta SS, Chattopadhyay A, Sinha K, Maitra S, Sinha B (2013) High-performance hardware implementation for RC4 stream cipher. *IEEE Trans Comput* 62(4):730–743
3. Ahmad S, Beg MR, Abbas Q, Ahmad J, Atif S (2010) Comparative study between stream cipher and block cipher using RC4 and Hill Cipher. *Int J Comput Appl* (0975–8887), 1(25)
4. Zoltak B (2004) VMPC one-way function and stream cipher. *Fast software encrypt FSE 2004, LNCS 3017*. Springer-Verlag, New York, pp 210–225
5. Paul S, Preneel B (2004) A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher in fast software encrypt. *FSE 2004, LNCS 3017*. Springer-Verlag, New York, pp 245–259
6. Maximov A (2005) Two linear distinguishing attacks on VMPC and RC4A and weakness of the RC4 family of stream ciphers. *Fast software encryption, FSE*
7. Fluhrer S, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of RC4. In: *Proceedings of annual workshop on selected areas in cryptography*, vol 2259, Springer, Toronto, pp 1–24
8. Yu Q, Zhang C (2011) RC4 state and its applications. In: *Ninth annual international conference on privacy, security and trust*, pp 264–269
9. Pardeep Pateriya P (2012) PC-RC4 algorithm: an enhancement over standard RC4 algorithm. *Int J Comput Sci Netw* 1(3)
10. Kadry S, Smaili M (2010) An improvement of RC4 cipher using vigenère cipher. *Int J Comput Intell Inform Secur* 1(3)
11. Mousa A, Hamad A (2006) Evaluation of the RC4 algorithm for data encryption. *Int J Comput Sci Appl* 3(2)
12. Yao Y, Chong J, Xingwei W (2010) Enhancing RC4 algorithm for WLAN WEP protocol. In: *control and decision conference (CCDC)*, IEEE, pp 3623–3627
13. Hammood MM, Yoshigoe K, Sagheer AM (2013) RC4 stream cipher with a random initial state. In: *Proceedings of 10th FTRA international conference on secure and trust computing, data management, and applications (STA'13)*. Lecture notes in electrical engineering, Springer, Heidelberg
14. Grosul A, Wallach D (2000) A related-key cryptanalysis of RC4. Department of computer science, Rice University, Technical report TR-00-358, June 2000
15. Stamp M (2006) *Information security principles and practice*. Wiley, New York
16. Sharif SO, Mansoor SP (2010) Performance analysis of stream and block cipher algorithms. In: *3rd international conference on advanced computer theory and engineering (ICACTE)*, IEEE vol 1, pp 522–525
17. Mantin I, Shamir A (2001) A practical attack on broadcast RC4. In: *8th international workshop, FSE*, PP 152–164
18. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J, Vo S (2001) A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National institute of standards and technology (NIST), Gaithersburg. <http://csrc.nist.gov/rng/>
19. Stallings W (2011) *Cryptography and network security principles and practices*, 5th edn. Prentice Hall Pearson, New Jersey

Information Technology Convergence

Security, Robotics, Automations and Communication

Park, J.H.; Barolli, L.; Xhafa, F.; Jeong, H.-Y. (Eds.)

2013, XLVIII, 1075 p. 510 illus. In 2 volumes, not

available separately., Hardcover

ISBN: 978-94-007-6995-3