INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

MORE RANDOMNESS OF IMPROVED RC4 (IRC4) THAN ORIGINAL RC4

HEMANTA DEY[1], DR. UTTAM KUMAR ROY[2]

1. Department of Computer Application, Techno India College of Technology, Kolkata, W.B., India.
2. Dept. of Information Technology, Jadavpur University, Salt Lake City, Kolkata – 700098, India

Accepted Date: 27/02/2016; Published Date: 01/03/2016

Abstract: - "RC4 is one of the simple and popular stream in the cryptography family." It is robust enough and trusted by many organizations after become public. Eminent researchers like Roos and et. al claimed that RC4 has some weakness and biasness in its internal states. Original RC4 contains two components: KSA and PRGA. KSA is related with the supply key and initial arrangements of the s-boxes and second one PRGA is the random number generators. So in the architecture of RC4, some researchers pointed out the problem in swap function at the initial stages. In this paper, we used Blum-Micali algorithm of random number generator to eliminate the biasness of KSA and improved s-boxes are the input of next stage in PRGA generates better results than original RC4. The original RC4 and the Improved RC4 (IRC4) are tested with the NIST Statistical Test Suite and it has been found that the IRC4 is giving a better randomness in the ciphertexts, hence giving a better security.

Keywords: Improved RC4; Improved KSA; Random S-Box; NIST Test Suite;

Corresponding Author: MR. HEMANTA DEY

Access Online On:

www.ijpret.com

How to Cite This Article:

Hemanta Dey, IJPRET, 2016; Volume 4 (7): 149-157

PAPER-QR CODE

1. INTRODUCTION

RC4 is one of the best simple, popular, efficient, fast and easy-to-implement stream ciphers in cryptographic society. RC4 is widely used in many commercial products and standards, as it is designed for fast software and hardware implementations. The core of the algorithm is an internal state table and a table-shuffling principle, which is basically a swap function. RC4 contains two parts one is an initialization routine i.e., Key-Scheduling Algorithm (KSA) and another one is random number generator i.e., Pseudo-Random Generation Algorithm (PRGA) where the random values are selected from the s-boxes and two elements are swapped for every byte reported. RC4 cryptanalysis has been mainly devoted to the statistical analysis of the output sequence, or to the initialization weaknesses.

In RC4, KSA turns an identity permutation into a random-looking permutation and the PRGA generates the random keystream bytes, which are XORed with the plaintext to generate ciphertext. RC4 contains a secret array S of size N (generally, 256), in which integers (0 to N–1) are swapped, depending upon two index pointers i and j, where i is deterministic and j is pseudo-random. There is significant interest in the cryptographic community for RC4, for which there exist several works on strength and weakness of this cipher. It has been argued that there are many biases in the PRGA due to propagation of biases in the KSA.

In this paper, we have replaced the KSA using Blum-Micali algorithm to generate random elements of S-Boxes. Also, the initial value of j in PRGA has been calculated from the key values, not from 0 as stated in the original algorithm – thus giving a more dynamic value to j.

NIST (National Institute of Standards and Technology), USA recommended some criteria and statistical tests for characterizing the security of cryptographic algorithms. The NIST Statistical Test Suite is a statistical package of 15 tests to verify randomness of long (order of 106) binary sequences, which focuses on the randomness of a sequence in many ways, useful as a first step to check whether a generator is suitable for a cryptographic application. We compared and analyzed this variant of RC4 statistically with the original RC4, following the guidelines given by NIST in their Test Suite, coded by us. It has been found that though RC4 itself is quite secured to use, even after so many years of its primary design, the new variant is able to prove itself more efficient.

```
KSA
Input: Secret Key K
for i = 0, … N – 1
        S[i] = i;
next i
j = 0;
for i = 0, …, N – 1
        { j = j + S[i] + K[i]
          swap(S[i], S[j]);
        }
next i
Output: S-Box S generated by K
```

```
PRGA
Input: S-Box S – The o/p of KSA

i = 0; j = 0;
while TRUE
        { i = i + 1
          j = j + S[i]
          swap(S[i], S[j]);
          z = S[S[i] + S[j]];
        }

Output: Random Stream Z
```

Table A. The RC4 Algorithm

2. MOTIVATION

RC4 has gone through tremendous analysis since it has become public. Roos, 1995 showed strong correlation between the secret key bytes and the final key stream generated. He showed some weakness in KSA and defined several classes of weak keys for RC4 which bias the technical results.

Maitra and Paul, 2008 modified the RC4 cipher and named it as RC4+, which avoids existing weaknesses of RC4. They revolved the non-uniformity in KSA and proposed for additional layers over the KSA and the PRGA. They presented three-layer architecture in a scrambling phase after the initialization phase to remove the weaknesses of KSA (KSA+). They also introduced some extra phases to improve the PRGA (PRGA+).

Paul and Preneel, 2004 described a new statistical weakness in the first two output bytes of RC4 key stream and recommended to drop at least the initial 2N bytes, where N is the size of the internal S-Box. They also presented a new statistical bias in the distribution of the first two output bytes of the RC4 key stream generator and also proposed a new key stream generator namely RC4A with much less operations per output byte. They proposed to introduce more random variables in PRGA to reduce the correlation between the internal and the external states.

Tomasevic and Bojanic, 2004 proposed a new technique to improve cryptanalytic attack on RC4, which is based on new information from the tree representation of RC4. Strategy has been used to favor more promising values that should be assigned to unknown entries in the RC4 table. They introduced an abstraction in the form of general conditions about the current state of RC4.

Akgün et al., 2008 detected a new bias in the KSA and proposed a new algorithm to retrieve the RC4 key in a faster way. Their framework significantly increases the success rate of key retrieval attack. They showed that KSA leaks information about the secret key if the initial state table is known.

Sen Gupta et al., 2013 implemented hardware architecture to generate two keystream bytes per clock cycle using the idea of loop unrolling and hardware pipelining. They thoroughly studied RC4 designing problem from the view point of throughput.

Nawaz et al., 2005 introduced a new 32-bit RC4 like faster key stream generator. It has a huge internal state and offers higher resistance against state recovery attacks. This is suitable for high speed software encryption.

Dey Das et al., 2014 avoided the KSA and used a robust PRBG, BBS, to fill-up the internal state array, thus eliminated the swap function of KSA, which has been found giving a better security.

3. IMPROVED RC4 (IRC4)

Roos and et.al showed the weakness of KSA and weak keys in RC4. Roos, 1995 argued that in KSA, only the line of swap directly affects the state table S while exchanging two elements and hence the previous line j = j + S[i] + K[i] is responsible for calculating the indexes. Here the variables i is deterministic and j is pseudo-random. Therefore, swap between two elements may happen once, more than once, or may not happen at all – thus brings a weakness in KSA. He showed that there is a high probability of about 37% for an element not to be swapped at all.

In this paper, we propose to introduce an IRC4, where we represent the initial key sets for the first S-Box which are taken from another set of random numbers generates from the Blum-Micali algorithm, before starting the Pseudo-random generation algorithm (PRGA) stage. Since Roos and et. al discussed about the weakness which is related the first few elements so that new approach will be overcome that weakness.

The random function generates 256 numbers for the KSA to make more disordered the elements in input S-Box for the PRGA stage. The index S[i]+S[j] evaluated on S to produces output. Here we calculated the initial values of i and j variables as initial starting points of the PRGA from the last result, not from 0 as stated in the original algorithm or in other variants of RC4, to give more dynamic values to these variables. The basic aim of the authors is to make the outputs of the algorithm more random to make it more difficult to the intruder to break it.

The outputs of RC4 (original and improved), have been tested statistically using the guidance of NIST, by the package NIST Statistical Test Suite. For both the algorithms, a same text file has been encrypted 500 times by using 500 same encryption keys, generating 500 ciphertexts for each algorithm, each of which contains at least 1342500 bits, as recommended by NIST. The two sets of ciphertexts are then tested statistically to find out if the security varies for the original and the improved algorithms.

4. THE NIST STATISTICAL TEST SUITE

NIST developed a Statistical Test Suite, which is an excellent and exhaustive document consisting of 15 tests developed to test various aspects of randomness in long binary sequences produced by RNGs and PRNGs [NIST, 2010; Kim, et al., 2004]. The tests are listed as follows:

Frequency (Mono-bit) Test: No. of 1’s and 0’s should be approximately the same, i.e., with probability ½.

Frequency Test within a Block: Whether frequency of 1’s in an M-bit block is approximately M/2.

Runs Test: Number of runs of 1’s and 0’s of various lengths is as expected for a random sequence.

Test for Longest-Run-of-Ones in a Block: Whether the length of the longest run of 1’s within the tested sequence (M-bit blocks) is consistent with the length of the longest run of 1’s as expected.

Binary Matrix Rank Test: Checks for linear dependence among fixed length sub-strings, by finding the rank of disjoint sub-matrices of the sequence.

Discrete Fourier Transform Test: Detects periodic features in the sequence by focusing on the peak heights in the DFT of the sequence.

Non-overlapping Template Matching Test: Occurrences of a non-periodic pattern in a sequence, using a non-overlapping m-bit sliding window.

Overlapping Template Matching Test: Occurrences of a non-periodic pattern in a sequence, using an overlapping m-bit sliding window.

Maurer's Universal Statistical Test: Whether or not the sequence can be significantly compressed without loss of information, by focusing on the no. of bits between matching patterns.

Linear Complexity Test: Finds the length of a Linear Feedback Shift Register (LFSR) to generate the sequence – longer LFSRs imply better randomness.

Serial Test: Determines number of occurrences of the 2m m-bit overlapping patterns across the sequence – every pattern has the same chance of appearing as of others.

Approximate Entropy Test: Compares the frequency of all possible overlapping blocks of two consecutive / adjacent lengths (m and m + 1).

Cumulative Sums Test: Finds whether the cumulative sum of a sequence is too large or small – focuses on maximal excursion (from 0) of random walks defined, which should be near 0.

Random Excursions Test: Finds whether number of visits to a state within a cycle deviates from expected value, calculates the number of cycles having exactly K visits in a cumulative sum random walk.

Random Excursions Variant Test: Deviations from the expected visits to various states in the random walk, calculates the number of times that a state is visited in a cumulative sum random walk.

In each test, for a bit sequence, NIST adopted different procedures to calculate the P-values (probability values) from the observed and expected results under the assumption of randomness [NIST, 2010; Kim, et al., 2004]. The Test Suite has been coded by us and used to study the randomness features of RC4 and the modified RC4.

5. RESULTS AND DISCUSSIONS

After analyzing the outputs of the original RC4 and improved RC4, using the NIST Statistical Test Suite, as described above, it has been found that the improved algorithm creates a tweak in RC4 to increase its security. The final analysis and comparison is displayed in Table 1, where the POP (Proportion of Passing) status and uniformity distribution of NIST tests for these two algorithms are displayed and compared. The best values of a particular test for each algorithm are shaded (in rows) and then the numbers of shaded cells for each algorithm are counted (in columns). The higher count (here for the modified RC4) gives a better result for that particular algorithm, which shows that this one has a better security than the other, at least for this particular data-set.

Table 1. Comparison of POP Status and Uniformity Distribution Generated by the 15 NIST Tests for RC4 and the Modified RC4

Test↓	POP Status		Uniformity Distribution	
	Improved RC4	RC4	Improved RC4	RC4
1	0.990000	0.978000	6.781714^{-01}	4.034218^{-01}
2	0.939000	0.989000	2.095909^{-01}	4.614834^{-01}
3	0.992000	0.992000	5.120061^{-01}	8.780263^{-01}
4	0.984000	0.982000	2.343734^{-01}	5.790211^{-01}
5	0.985000	0.984000	2.399292^{-01}	2.091899^{-01}
6	0.980000	0.980000	4.319058^{-01}	4.120781^{-02}
7	0.976000	0.990000	4.319056^{-01}	8.221094^{-01}
8	0.976000	0.991000	9.013823^{-01}	2.201804^{-01}
9	0.992000	0.978000	3.789489^{-01}	3.276416^{-02}
10	0.985000	0.991000	7.123170^{-01}	5.892832^{-01}
11	0.990000	0.980000	5.712923^{-01}	1.613807^{-01}
12	0.9860000	0.992000	7.145391^{-01}	2.982018^{-01}
13	0.993000	0.993000	9.284378^{-03}	8.207835^{-01}
14	0.985000	0.981500	6.203653^{-01}	6.778885^{-01}
15	0.986667	0.985589	5.288562^{-01}	4.313075^{-02}
Total:	10	8	9	6

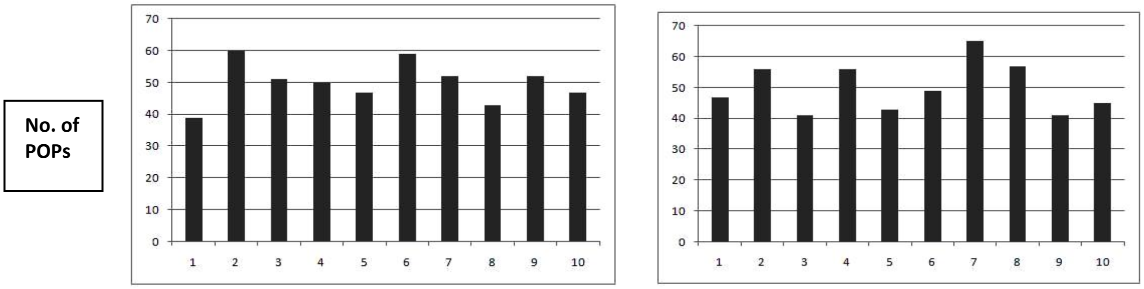

Fig. 1(a) & 1(b). Histograms for POP Distribution of Tests 4 & 8 for Modified RC4

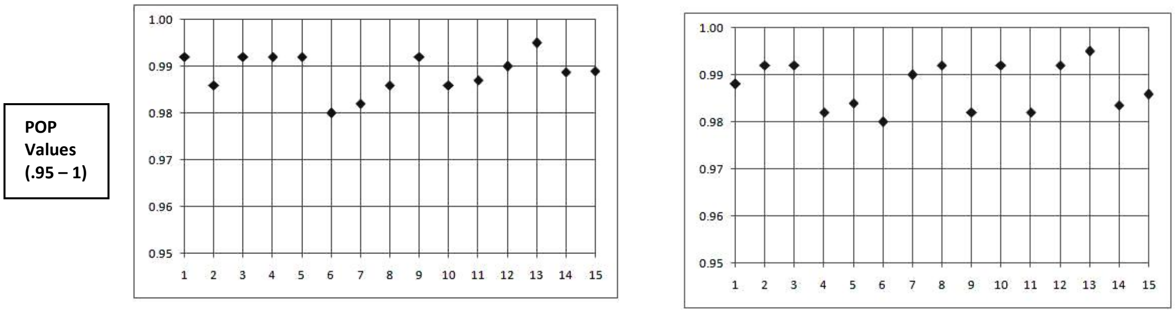

Fig. 2(a) & 2(b). Histograms for POP Distribution of Tests 4 & 8 for RC4

Finally, it has been observed that besides using the original KSA, a suitable mathematical model can be used to generate a secured internal S-array for RC4, which along with a modified PRGA gives better randomization in the cipher texts.

6. CONCLUSION

The improved RC4 is found to stand in a better merit list comparing to the standard RC4. It seems that security in RC4 will be enhanced by driving random number generators to generate the internal S-array using Blum-Micali algorithm. Also, the user can choose and generate any random state array S by using user own choice of modulus from a large set of options. In the case of suspicion of a trapdoor in the cipher text, an S-array might be replaced by another one by the user. Also it has been found that improved PRGA helps to generate more random output for RC4.

7. REFERENCES

1. Akgün, M., Kavak, P., Demicri, H.: New Results on the Key Scheduling Algorithm of RC4, INDOCRYPT, 5365, 40-52, (2008) Lecture Notes in Comp. Science, Springer.

2. Dey, H. and Roy, U: An approach to find out the optimal randomness using parallel s-boxes in RC4, ICBIM-2016 NIT Durgapur, India. 978-1-5090-1228-2/16/$31.00 ©2016 IEEE

3. Foruzan, B., Cryptography and Network Security, Tata McGraw-Hill, N. Delhi, Spl. Ind. Ed., 2007.

4. Maitra, S., Paul, G.: Analysis of RC4 and Proposal of Additional Layers for Better Security Margin. INDOCRYPT, 5365, 40-52, (2008) Lecture Notes in Computer Science, Springer.

5. National Institute of Standard & Technology (NIST), Technology Administration, U.S. Dept. of Commerce, A Statistical Test Suite for RNGs & PRNGs for Cryptographic Applications, 2010.

6. Paul, S., Preneel, B.: A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. In: FSE 2004, LNCS, vol. 3017, pp. 245-259, Springer, Heidelberg (2004).

7. Roos, A.: A Class of Weak Keys in the RC4 Stream Cipher. Post in sci.crypt, 1995)