RC4: An analysis with respect to Randomness

Shradha Gautam[*%] Kewal Krishna[* +] Anisur Rahman[*]

[*]Computer Science & Engineering
National Institute of Science & Technology
Berhampur, Orrisa

[%]shradhagautam15@gmail.com [+]kewal07@gmail.com [#]anisur.rahman96@gmail.com

***Abstract*— the paper aims to study one of the most widely used stream cipher, RC4. The bytes generated by the Psedo Random Generator Algorithm (PRGA) of RC4 are evaluated to check the randomness. Some widely known and accepted statistical measures are applied for this purpose. This is then followed by modifying the initial primitive key fed to RC4 using the system clock and compared with the initially generated bytes.**

Keywords—*RC4, PRGA, Randomness, System Clock*

I. Introduction

RC4 is the most widely used stream cipher. It is used in many popular applications such as Secure Socket Layer and Wired Equivalent Privacy. The cipher was designed by Ron Rivest in 1987. However it was kept a trade secret until September 1994 when a description of it was anonymously posted to the Cypherpunks mailing list. The leaked code was confirmed to be genuine as its output was found to match that of proprietary software using licensed RC4. The main factors in RC4's success are its speed and simplicity; efficient implementations in both software and hardware are very easy to develop [1,2]. RC4 is a shared key stream cipher algorithm requiring a secure exchange of the secret key. The algorithm is used identically for encryption and decryption as the data stream is simply XORed with the generated key sequence. The algorithm is identical in the sense that as it is a symmetric key cipher, the key for encryption and decryption are the same. The algorithm is serial as it requires successive exchanges of state entries based on the key sequence.

This paper aims to study the bytes generated by the PRGA of RC4. The initial seed provided by the user is fed to the algorithm to generate the final key for encryption which may be greater in length than that given by the user. A fundamental requirement of the generated key stream is to be sufficiently random in nature to prevent the cipher text from being deciphered by an adversary. Intensive studies over the past two decades have shown that the generated key stream is not random as one would like it to be. This paper performs some basic randomness tests on the key stream and observes the results in a comparative fashion with those generated from the initial random seed modified with the system clock. The different biases observed in the rich literature of RC4 till now and explanation of them is outside the scope of this paper; the interested reader may look into [2,3,4] and the references therein for further details.

II. Description of RC4

RC4 uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. This table is used for generation of pseudo-random bytes. This work is performed in the first phase of the algorithm known as the Key Scheduling Algorithm (KSA). The output from the KSA is then given as input to the second phase known as the Pseudo Random Generation Algorithm (PRGA). The PRGA generates the key stream which is then XORed with the message to obtain the cipher text. This is equivalent in some respect to the Vernom cipher.

The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits [5].

A. *The Key Scheduling Algorithm (KSA)*

The key-scheduling algorithm initializes the permutation in the state table, S. The number of bytes in the key, K, can be in the range 1 to 256, typically between 5 and 16 bytes. S is initialized to the identity permutation which is then processed for 256 iterations swapping the values at different indices between them using the key. The KSA provides the S table which is used in the PRGA to get the final key. Figure 1 lays out the algorithm in detail.

B. *The Pseudo Random Generator Algorithm (PRGA)*

The S box generated from KSA is swapped within itself using a known index and a random index. The random index is generated successively using the values of the same from the previous iteration. The S table is then swapped using these values. The output byte is generated taking the modular addition of the values at the index pointers. Figure 1 lays out the algorithm in detail.

```
KSA $(K, S)$
for $i = 0$ to $N - 1$
    $S[i] = i$
$j = 0$

for $i = 0$ to $N - 1$
    $j = (j + S[i] + K[i \bmod l]) \bmod N$
    Swap$(S[i], S[j])$
```

```
PRGA$(S)$
$i = 0$
$j = 0$
Output Generation loop
    $i = (i + 1) \bmod N$
    $j = (j + S[i]) \bmod N$
    Swap$(S[i], S[j])$
    Output=$S[(S[i] + S[j]) \bmod N]$
```

Figure 1: The KSA & PRGA

III. Modified Seed fed to RC4

The key stream generated by the algorithm of Figure 1 is not as random as one would like it to be. Section 4 gives some observations in support of this. To get a better key stream we modify the secret key. The initial random seed provided by the user is treated as plaintext. The system clock is used as the encryption key. We use the concept of Vernam cipher to get a more random key which is then used by subsequent phases of the algorithm to get a more random key stream. The generated key stream is then XORed with the original plain text. Figure 2 lays out the detailed methodology.

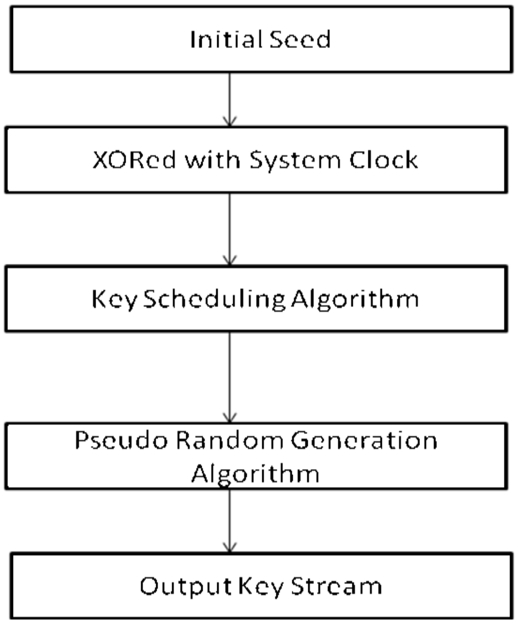

Figure 2: Flowchart for the modified seed fed RC4

IV. Observations & Discussions

We perform various tests to examine the randomness of the key stream generated by the original RC4 and the RC4 with the modified key stream. We have taken 8938 common words of 5 bytes each[6] in order to perform these tests and operate for 256 rounds.

A. *Mono Bit Test*

Mono bit test is used to count the number of 1's and 0's of the key stream. The statistic generated from all the words, if random, would follow a normal distribution. Figure 3 shows the plots for mono bit test for both the above stated algorithms. As is clear from the normal probability plot, the statistic values are not found lying on the true random line. However, Figure 3 makes it clear that the key stream generated using the modified key RC4 is somewhat more random. This needs to be verified mathematically because the curves lie close together.

B. *Serial Test*

Serial Test is used to determine whether the number of occurrences of 00, 01, 10, and 11 as subsequences are approximately the same, as would be expected for a random sequence. The plots for both the generated key streams are shown in Figure 4. Once again it is clear that the algorithm with system clock provides better randomness.

C. *Poker Test*

Poker Test Determines whether the sequences of length m each appear approximately the same number of times, as would be expected for a random sequence. We have taken m to be of 1 byte. The plots in Figure 5 show that key stream with the system clock provide better randomness.

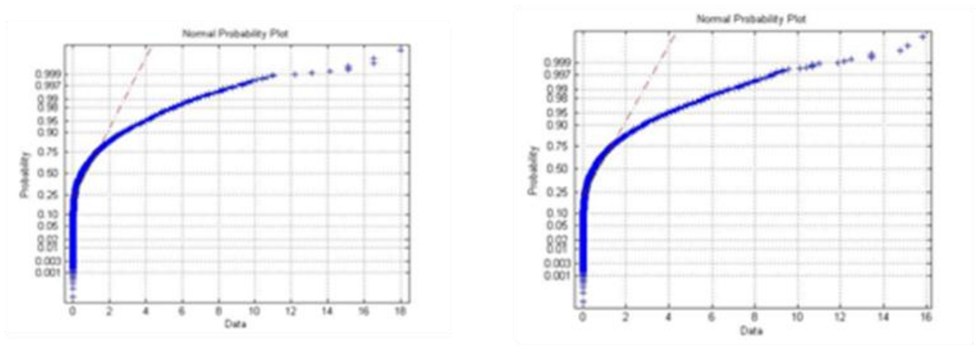

Figure 3: Normal Probability plots for frequency test for (i) original RC4 (ii) modified seed fed to RC4

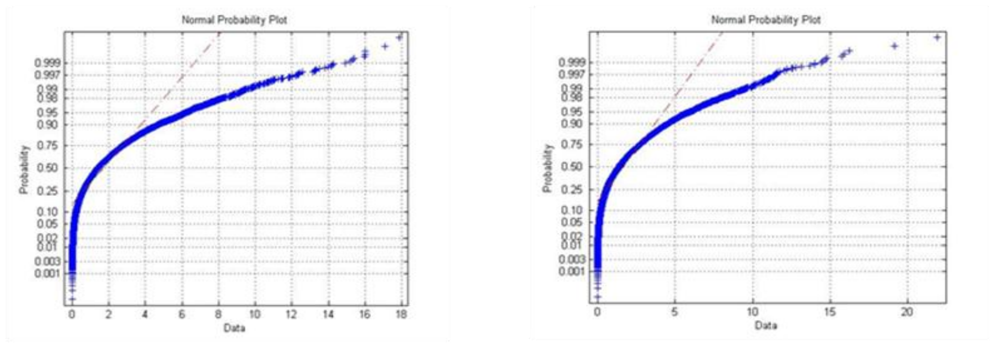

Figure 4: Normal Probability plots for serial test for (i) original RC4 (ii) modified seed fed to RC4

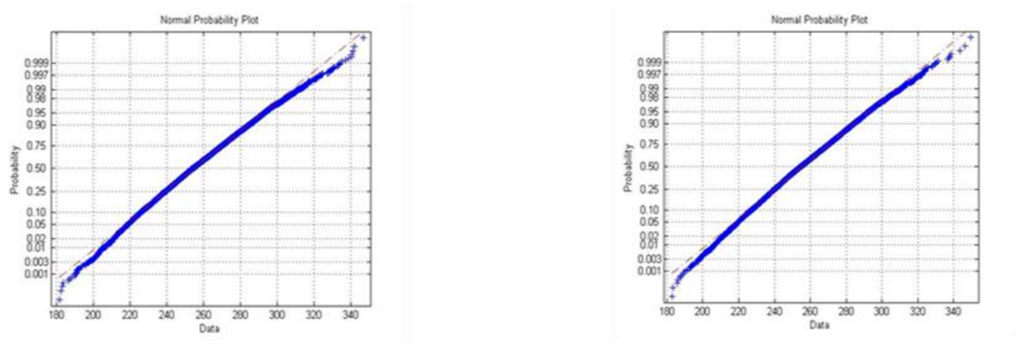

Figure 5: Normal Probability plots for poker test for (i) original RC4 (ii) modified seed fed to RC4

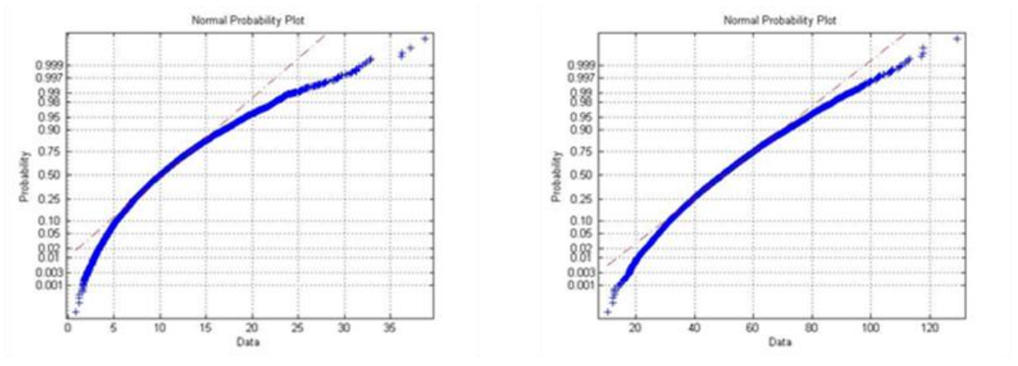

Figure 6: Normal Probability plots for runs test for (i) original RC4 (ii) modified seed fed to RC4

D. *Runs Test*

The purpose of runs test is to determine whether the number of runs of either 0s or 1s of various lengths in the sequence is as expected for a random sequence. The plots for both the generated key streams are shown in Figure 6. An interesting observation is that while the norm plots seem in favor of modified RC4, when tested with the statistical parameters of chi square tests,[7,8] the modified seed fed algorithm fails miserably as shown in Appendix.

As is clear from the table given in Appendix, the resuls are much better in case of seed modified with the system clock except that for runs test. This is a unique phenomenon because intuition as well as literature suggests otherwise. Table I shows the chi square hypothesis test results for randomness with the hypothesis being that the sequence is random. The significance level α is the probability of rejecting the hypothesis when it is true. To achieve a significance level of α, a threshold value X_α is chosen corresponding to the degree of freedom using the chi square distribution table. If the value of the statistic X_s of the output sequence satisfies $X_s > X_\alpha$ then the sequence fails the test; otherwise it passes the test.

V. Conclusion & Future Work

The modified key RC4 performs better than the simple RC4. It yields more random key stream than that of the one without using system clock. It is found that for Poker test the generated key stream is most close to being random. This is an interesting observation because the size of m we took was 1 byte. Also, the observation for the runs tests is peculiar. While the norm plot lies approximately on the ideal line, the same when taken for chi square tests deviate almost completely. The reason behind this is to be investigated. The effect of different sizes and number of iterations on the same is a prospective for future work. The same key stream when tested using mono bit and serial test shows significant deviations as compared to the Poker test. In mono bit test, the plots from both the algorithms were identical to an extent greater than that of other tests. These issues, along with the modification of generated keystream instead of the initial seed using the system clock and overcoming the limitation of storing the keystream for decryption purposes as it keeps on changing with time constitute the authors' future work.

References

[1] Cypherpunks mailing list., "*Thank You Bob Anderson*" 1994-09-09. Retrieved 2007-05-28

[2] Souradyuti Paul and Bart Preenel, "A New Weakness in the RC4 Key Stream Generator and an Approach to Improve the Security of the Cipher", FSE 2004, LNCS, Springer-Verlag, pp. 245-259, 2004

[3] Itsik Mantin and Adi Shamir, "A Practical Attack on Broadcast RC4", FSE 2001, LNCS, Springer-Verlag, pp. 152-164, 2001

[4] S Maitra, G Paul and S Sen Gupta, "Attack on Broadcast Revisited", FSE 2011,LNCS, Springer-Verlag,, Denmark

[5] RC4 data sheet, VOCAL Technologies, Ltd., 2011

[6] TWL2006 and CSW2007, available online at, "http://www.poslarchive.com/math/scrabble/lists/common-5.html"

[7] A Menezes, P Van Oorschot, S Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

[8] Donald E Knuth, The Art of Computer Programming, Volume 2, 3rd edition, ADDISON-WESLEY, 1999

Appendix

TABLE I. Chi square tests

α	Test	Deg of freedom	X_α	No of $X_s > X_\alpha$	% Failure
0.0057	Mono Bit Figure1	1	7.8794	51	0.5705
	Mono Bit Figure2			44	0.4922
	Serial Figure1	2	10.5966	54	0.6041
	Serial Figure2			47	0.5258
	Poker Figure1	255	316.9194	62	0.6936
	Poker Figure2			49	0.5482
	Runs Figure1	10	25.1882	79	0.8838
	Runs Figure2			8664	96.9344
0.01	Mono Bit Figure1	1	6.6349	100	1.1188
	Mono Bit Figure2			78	0.8726
	Serial Figure1	2	9.2103	108	1.2083
	Serial Figure2			87	0.9733
	Poker Figure1	255	310.4574	94	1.0506
	Poker Figure2			108	1.2083
	Runs Figure1	10	23.2093	132	1.4768
	Runs Figure2			8764	98.0532
0.025	Mono Bit Figure1	1	5.0239	261	2.9201
	Mono Bit Figure2			240	2.6851
	Serial Figure1	2	7.3778	258	2.8865
	Serial Figure2			209	2.3383
	Poker Figure1	255	301.1250	252	2.8194
	Poker Figure2			244	2.7299
	Runs Figure1	10	20.4832	319	3.5690
	Runs Figure2			8841	98.914
0.05	Mono Bit Figure1	1	3.8415	469	5.2472
	Mono Bit Figure2			452	5.0570
	Serial Figure1	2	5.9915	502	5.6164
	Serial Figure2			461	5.1577
	Poker Figure1	255	293.2478	482	5.3927
	Poker Figure2			463	5.1801
	Runs Figure1	10	18.307	605	6.7688
	Runs Figure2			8890	99.4629