International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:2, 2009

A New Variant of RC4 Stream Cipher

Lae Lae Khine

Abstract**—**RC4 was used as an encryption algorithm in WEP(Wired Equivalent Privacy) protocol that is a standardized for 802.11 wireless network. A few attacks followed, indicating certain weakness in the design. In this paper, we proposed a new variant of RC4 stream cipher. The new version of the cipher does not only appear to be more secure, but its keystream also has large period, large complexity and good statistical properties.

Keywords**—**Cryptography, New variant, RC4, Stream Cipher.

I. Introduction

IN cryptography, s*ubstitution box* (S-box) constitutes a cornerstone component of symmetric key algorithm that maps input vectors non-linearly into output vectors. It is used in both block ciphers and stream ciphers. In modern software efficient stream cipher, RC4 is one of the most widely used stream ciphers based on *S-box*. It is used in two security schemes defined for IEEE 802.11 wireless LANS: *Wired Equivalent Privacy* (WEP) and *WiFi Protected Access* (WPA).

Since RC4 is such a widely used stream cipher, it had attracted considerable attention in the research community since it was first proposed. However, RC4 has been analyzed independently by many people and a number of weaknesses have been pointed out. Some of these weaknesses are minor and just of theoretical interest whilst others are severe and lead to practical attacks.

The RC4 *Key Scheduling algorithm* (KSA) is much more problematic, since it was designed to be unnecessarily simple. As a result, the first few output bytes of the PRNG are either biased or related to some key bytes, and thus the analysis of these bytes makes it possible to attack RC4 in some modes of operation. Andrew Roos[1] developed one of the first such attacks on RC4: a set of weak keys causing a bias in the initial output. Fluhrer, Mantin and Shamir [3] presented several weaknesses in the KSA. *Initialization Vector* (IV) weakness constitutes an interesting class of key schedule weaknesses that, in the case of RC4, lead to practical attacks. Knudsen's attack is a useful theoretical tool since it can be modified to take partial state information and some output bytes and provide a compete state. If probabilistic information about the state is known, this can be used to choose the order in which the algorithm branches, choosing the most likely branches first and accelerating the algorithm. Knudsen, Mier, Preneel, Rijimen and Verdoolaege [5] presented the swap operation makes the recovery of the table *S* very difficult. They developed an attack on simplified version of RC4, where the swap operation occurs less often. RC4 without the swap operation is useless as a key stream generator.

To prevent above biases, we proposed a new variant of RC4 stream cipher using *Random block*(R-block), which increases the internal complexity of the original algorithm. The larger internal state size of the proposed design increases the complexity of attacks and makes analysis more difficult. In addition, the keystream of modified RC4 stream cipher satisfies the properties of nonlinearity, long cyclic, and uniformly distributed.

We shall learn in description of RC4 in section 2. In section 3 we shall look at the weaknesses of RC4 to explore new possibilities for cipher design. Section 4 has more details of the new variant of RC4. Conclusion will be included in section 5.

Author is from the Department of Engineering Physics, Mandalay Technological University Mandalay 05052, Mandalay, Myanmar. (e-mail:1ai101.mi@gmail.com,).

II. Description of RC4

At the time of development of RC4 much of stream ciphers were focused on so-called linear feedback shift register or LSFR. The devices are easy to study from a mathematical point of view, making the analysis of their security in attractive topic. Unfortunately LSFRs are many bit operations, making the slow in software implementation. RC4 was designed by Ron Rivest in1987, co-developer RSA. In contrast to LSFRs, RC4 was byte operations that are friendlier to software implementations, especially on the 8-bit and 16-bit machines commonly available at the time.

The RC4 algorithm is fairly sample and is still secure enough for many application. The RC4 algorithm takes an input a random-seed of any length and outputs a pseudorandom number sequence. The random number sequence acts like a key stream for the encryption algorithm. RC4 consists of two parts, a *key scheduling algorithm* (KSA) which turns a random key (whose typical size is 40−256 bits) into an initial permutation of *S-box* of $\{0.....N-1\}$, where N is a power of 2 and is typically 256. The other part is a *pseudorandom number generator* (PRNG), which uses this permutation to generate a pseudo-random number sequence.

The PRNG initializes two indices i and j to 0 and then loops over four simple operations which increment i as a counter, increment j pseudo randomly, exchange the two values of *S-box* pointed to by i and j, and output the value of *S-box* pointed to by $S[i]+S[j]$.

The KSA consists of N loops that are similar to the PRNG round operation. It initializes S to be the identity permutation and i and j to 0, and applies the PRGA round operation N

times, stepping *i* across *S-box*, and updating *j* by adding *S*[*i*] and the next word of the key (in cyclic order).

KSA
For $i = 0.....N-1$
$S[i] = i$
$j = 0$
Scrambling
$j = (j + S[i] + K[i \bmod l]) \bmod 256$
$S[i] \leftrightarrow S[j]$
PRNG
Generation loop:
$Q1 = (Q1 + 1) \bmod 256$
$Q2 = (Q2 + S[i]) \bmod 256$
$S[Q1] \leftrightarrow S[Q2]$
$T = (S[Q1] + S[Q2]) \bmod 256$
Output $Y = S[T]$

Fig. 1 The KSA and the PRNG Algorithms on RC4

III. Weaknesses of RC4

Since its public disclosure in 1994, RC4 has been analyzed independently by many people and a number of weaknesses have been pointed out. Some of these weaknesses are minor and just of theoretical interest whilst others are severe and lead to practical attacks. In this section we describe some known attacks on RC4.

A. Roos's Class of Weak Key

The weak keys of RC4 are discovered by Andrew Roos [1]. By examining the details of key schedule algorithm, Roos was able to find a strong bias in the initial state which in turn causes a bias in the first few output words. This bias is in effect for about $\frac{255}{256}$ of keys (the analysis was done for N=8), producing a class of weak keys.

The bias stems from the fact that a particular state element is indexed by *j* sometime during the key schedule with probability about $1-(\frac{255}{256})^{256} = 0.631$, assuming a uniform distribution on the values of *j*. This means that a particular element is swapped only once (when indexed by *i*) with probability about 0.37. The value with which it is swapped depends of the value of *j* and $S[j]$ at the time of the swap. Note that, early in the algorithm, the value of $S[j]$ probably has not been swapped out, so $S[j] = j$. Also, note that the $S[i]$, when added to *j*, has not previously been swapped by *i* and thus probably has not been swapped and satisfies $S[i] = i$. If this holds for all the necessary elements of *S-box* then we get $j = 1 + 2 + ... + i$ and $S[j] = i$. In this case, after the swap the value of $S[i]$ is

$$\frac{i(i+1)}{2} + \sum_{m=0}^{i} k[\bmod l]$$

For small *i* the probability that all the necessary events happen is quite high. Roos conducted experiments that show a steadily decreasing probability of 0.004.

In order to take advantage of the bias in the initial state, Roos constructs what are essentially predictive states: conditions on a small number of state elements that cause a particular output. In particular, we are interested in conditions on the first few elements of the state, since these have the highest bias. This bias caused in the first output byte is somewhat unique in that provides information can be used to speed up brute force attacks in cases where the biased output occurs. To avoid this bias we need to introduce additional non-linear dependencies into the state table, this would make analysis more difficult.

B. Key Schedule Invariance

Fluher, Martin, and Shimar [2] found the key schedule invariant attack, well-known attack of RC4. It is more sophisticated class of weak keys. This weakness depends on how *j* is changed in the key schedule. The actual line in this algorithm is

$$j = j + S[i] + K[i \bmod l] \bmod n$$

To determine the small part of secret key of RC4 can be calculated with the following assuming:

Let *S-box* be a permutation of $\{0,.....,N-1\}$, *x* be an index in *S-box* and *b* be some integer. Then if $S[i] \equiv i \bmod b$, the permutation in *S-box* is said to *b*-conserve the index *x*. On the other hand, it can say the permutation *S-box* is said to *b*-unconserved the index *x*. Denote the permutation *S-box* and the indices that permutation *b*-conserves as $I_b(S)$. For the sake of simplicity, we often write I_t instead of $I_b(S_t)$. The permutation *S-box* of $\{0,.....,N-1\}$ is b-conserving if $I_b(S) = n$, and is almost *b*-conserving if $I_b(S) \geq n-2$. If for any index, $K[t \bmod l] \equiv (1-t) \bmod b$ where *b* and *l* are integers and K be an *l* word key. In case $K[0] = 1$ and msb $K[1] = 1$, *K* is called a special *b* exact key. Note that in order to have a special *b*-exact key of length *i* it is necessary that *b* divides *i*.

KSA thus transform special patterns in the key into corresponding patterns in the initial permutation. The fraction of determined permutation bits is proportional to the fraction of fixed key bits. For example, applying this result to RC4 *n*=8, *l*=6 and *q*=1, 6 out of the 48 key bits completely determined 252 out of the 1684 permutation bits.

C. Initialization Vector Weakness

We shall now look at a particular key attack that will focus on the case where the IV precedes the secret key.

Consider the case where we know the first *A* values of the secret key $(K[3].....K[A+2])$. Examine a series of IV's of the form $(A+3, -1, X)$, where X is random value. In the first step, *j* is advanced by $S_0[0] - A + 3$. On the next step, *i* is advanced, and the advance on *j* is computed to be $j_1 = j_0 + S_0[1] + K[1] = j_0 + 1 + (-1)$. Therefore, *j* is not moved. Then $S[i]$ and $S[j]$ are swapped. On the next step, *j* is advanced by $X + 2$. From here on until $i = A + 3$, the key schedule can be computed. Since all the X's are different, each IV acts differently. At this point $(i_{A+2} = A + 2)$, the value

of j_{A+2} and of the entire permutation S_{A+2} throws out this IV. We also know $S_{A+2}[A+3]$.The value that will be $S_{A+3}[A+3]$ depend on $j_{A+3}=j_{A+2}+S_{A+2}[A+3]+K[A+3]$, which is two known and one unknown. $S_{A+3}[A+3]=S_{A+2}[j_{A+3}]$. This is a state in a (pseudo) resolved condition. Therefore, with probability approximately 0.05, none of the three important values will be touched by the remaining key schedule, and the first output word will be $S[A+3]$. In this case, it is possible to calculate $K[A+3]$ as follow:

$$K[A+3]=S_{A+2}^{-1}[z_0]-j_{A+2}-S_{A+2}[A+3]$$

If we use approximately 60 IV's of the form above, we can calculate secret key byte *A*. If we iterate this attack over all the secret key bytes (*l* bytes), it takes approximately 60*l* chosen IV's and their corresponding first output byte to determine the entire secret key.

D. State Guessing

The idea behind this attack is to guess some of the initial state of the RC4 keystream generator, and by looking for some contradictions in the keystream generator, and by looking for contradictions in the keystream it is possible to detect incorrect guesses and to discover the rest of initial state.

In this discussion, let Q_{1t} and Q_{2t} be the value of the two counters at time *t*. Also, let be the state at time *t*. Therefore, the output of RC4 at time is

$$z_t = S_t[S[Q_{1t}]+S_t[Q_{2t}]] \qquad (1)$$

At any time *t*, it is always known possibly unknown are Q_{2t}, $S[Q_{2t}]$, and $S^{-1}[z_t]$. From any tree of these variables, the fourth can be calculated:

$$Q_{2t}=S^{-1}[z_t]-S[Q_{1t}] \qquad (2)$$
$$S[Q_{1t}]=S^{-1}[z_t]-S[Q_{2t}] \qquad (3)$$
$$S[Q_{2t}]=S^{-1}[z_t]-S[Q_{1t}] \qquad (4)$$
$$S^{-1}[z_t]=S[Q_{1t}]+S[Q_{2t}] \qquad (5)$$

The algorithm to recover *S-box* works as follows. Initially, guess a small subset of the values of *S-box*. Use equation 2-5 as time t progresses to derive additional values of *S-box*. If a contradiction arises, then the initial guess was incorrect. Repeat this process for all possible guesses.

E. Knudsen's attack

This attack is based on RC4 PRNG algorithm. First, the internal state at the beginning of algorithm has *i* and *j* fixed at zero and there *N* bytes contained in *S*. Thus, we expect that *N* bytes of output should contain enough information to reconstruct the state. If this is not the case then there must be some bias in the output that account for the lack of *S* in the first *N* outputs, causing these outputs to be depend on every elements of *S*. Based on this reasoning we expected to be able to construct an algorithm that determines the state given *N* words of output.

Knudsen's attack although impractical due to the high complexity is a useful theoretical tool since it can be modified to take partial state information and some output bytes and provide a complete state. Thus, if some other attack exposes some state information, it can be used in conjunction with this expose some state information, it can be used in conjunction with attack to provide complete break. Also, if probabilistic information about the state is known, this can be used to choose the order in which the algorithm branches, choosing the most likely branches first and accelerated the algorithm.

IV. Modifying RC4

A. Principles of R-block

The construction of R-block (stochastic transformation) is shown in figure (2). The principles of R-block are as follows. Firstly, the cells of *H* table are initialized by initialization vector. After that, the permutation of *H* table is performed by the swapping mechanism based on key information. $H[i]\leftrightarrow H[k_i]$. The resulting output used as the stochastic number is computed by using formula $R_H(A,B)=H((m_A+B)\bmod 2^n$, where A, B are input elements of registers, m_A is an address of cells containing code *A* in *H* table, i.e., $H(m_A)=A$. The output of R-block is the essence of reading the contents of cells in *H* table. The result $\mathrm{R_H(A,B)}$ of the stochastic transformation depends on the key information in *H* table and input parameter *B*.

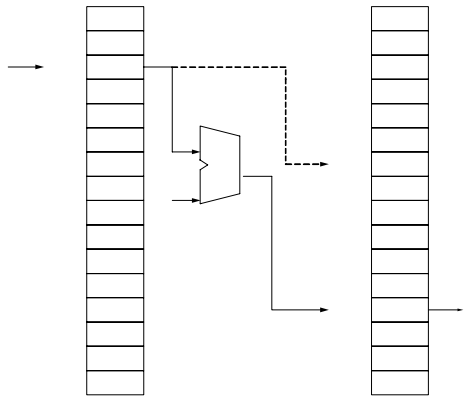

Fig. 2. The construction of R-block

B. The New Variant of RC4

The goal is to increase security primarily by increasing the internal complexity of the algorithm. By increasing the number of variables involved in the output the size of predictive states increased, reducing biases. In this proposed stream cipher, we applied double swap mechanism and R-block algorithm which has one-way property to make analysis more difficult. The larger internal state size of the proposed design increases the complexity of attacks mentioned in

section (3). The cost is a larger in the memory requirements as well as set-up time, but the proposed stream cipher is simple, efficient and secure design.

1) Key Scheduling Algorithm

The new KSA as shown in figure (8) is straightforward modification of original KSA that attempts to gain security by replacing the *j* update. The algorithm used to update *j* is *R-block* algorithm as a random function. The original KSA is used to permute the elements of the S-box in a key dependent way. In this step, the swapping mechanism is based on the counter *i* and the random number *j*. According to the weakness and attacks mentioned in the section (3), they depend on how *j* is changed in the key schedule. By using *R-block* algorithm in the new KSA, the random number *j* is more complex and the probability to predict output state is decreased. Therefore, the new KSA is more secure than the original KSA.

New KSA
For $i = 0$ to $N-1$
$S[i] = i$
$j = 0$
For $i = 0$ to $N-1$
$j = (j + Rblock(S[i], k[i])) \bmod N$
$S[i] \leftrightarrow S[j]$

2) Pseudorandom Number Generator Algorithm

The original PRNG was modified by double swap mechanism through R-block algorithm. This has two affects: the attacks mentioned in section3
cannot guess some of the initial state of the RC4 keysteam generator; the new PRNG can generate the pseudorandom number sequence with uniformly distribution and large period.

New PRNG
$Q_1 = (Q_1 + 1) \bmod N$
$Q_2 = (Q_2 + S[Q_1]) \bmod N$
$S[Q_1] \leftrightarrow S[Q_2]$
$T_1 = (S[Q_1] + S[Q_2]) \bmod N$
$T_2 = Rblock(S[Q_1], S[Q_2])$
$S[T_1] \leftrightarrow S[T_2]$
$T = (S[T_1] + S[T_2]) \bmod N$
$OutputY = S[T]$

C. Statistical Analysis

Most of the practical random number generators have some built in regularities which, sometimes, also become their deficiencies. The use of these generators in cryptographic applications will need more security requirements than other applications. There are mathematical safety measures that offer some protections against this issue. One safety measure is to assess generators by theoretical tests and the other is to run empirical tests. Since the year 2000 NIST has been developing a string of statistical tests to predict efficiently the randomness of a string of bits. A total of 16 tests were developed, implemented and tested as those of today. All of these tests were implemented in C program by NIST. By using these statistical tests, we measured the randomness of sequence generated by proposed stream cipher.

V. Conclusion

In this paper, we present the design and principles of proposed stream cipher. The proposed stream cipher provides added security against the weaknesses and attacks mentioned in section (3). In addition, it is simple, secure in design and efficient for software and hardware implementation. The results of statistical analysis proved that the proposed stream cipher had got the good statistical properties and the key stream of this cipher exceeded the length of the original stream cipher.

References

[1]. Andrew Roos, "A Class of Weak Keys in the RC4 Stream Cipher". Preliminary Draft, 22 September, 1995.

[2]. Fhlurer, I Martin, A Shamir, "Weakness in the key Scheduling Algorithm of RC4" Proceedings Selected Areas in Cryptography 201, SAC'01, LNCS vol. 2559, pp.1-24, Springer-Verlag, 2001.

[3]. Grosul A.L., Wallach D.S. "A related key cryptanalysis of RC4", 2000.

[4]. Mantin I. Shamir A. "A practical attack on broadcast RC4", Proceeding of FSE. 2001.

[5]. Matthew E. McKague. "Design and Analysis of RC4 like Stream Ciphers". Matthew E.McKague, 2005.

[6]. NIST Special Publication 800-22, "A Stastical Test Suite for Random and Pseudorandom Number Generator for Cryptographic Applications", May, 2001.

[7]. S. Mister and S. E. , Cryptanalysis of RC4-like ciphers , Selected Tavares Areas in Cryptography (Kingston , ON, 1998), Lecture Notes in Comput. Sci., vol. 1556, pp. 131–143, Springer, Berlin, MR MR1715807, 1999.