

# Cryptanalysis of RC4-like Ciphers

S. Mister<sup>1</sup> and S. E. Tavares<sup>2</sup>

<sup>1</sup> Security Technology Group, Entrust Technologies Limited  
750 Heron Road, Ottawa, Ontario, Canada, K1V 1A7  
`serge.mister@entrust.com`

<sup>2</sup> Department of Electrical and Computer Engineering  
Queen's University, Kingston, Ontario, Canada, K7L 3N6  
`tavares@ee.queensu.ca`

**Abstract.** RC4, a stream cipher designed by Rivest for RSA Data Security Inc., has found several commercial applications, but little public analysis has been done to date. In this paper, alleged RC4 (hereafter called RC4) is described and existing analysis outlined. The properties of RC4, and in particular its cycle structure, are discussed. Several variants of a basic “tracking” attack are described, and we provide experimental results on their success for scaled-down versions of RC4. This analysis shows that, although the full-size RC4 remains secure against known attacks, keystreams are distinguishable from randomly generated bit streams, and the RC4 key can be recovered if a significant fraction of the full cycle of keystream bits is generated (while recognizing that for a full-size system, the cycle length is too large for this to be practical). The tracking attacks discussed provide a significant improvement over the exhaustive search of the full RC4 keyspace. For example, the state of a 5 bit RC4-like cipher can be obtained from a portion of the keystream using  $2^{42}$  steps, while the nominal keyspace of the system is  $2^{160}$ . More work is necessary to improve these attacks in the case where a reduced keyspace is used.

## 1 Introduction

Stream ciphers are often used in applications where high speed and low delay are a requirement. Although many stream ciphers are based on linear feedback shift registers, the need for software-oriented stream ciphers has lead to several alternative proposals. One of the more promising algorithms, RC4[9], designed by R. Rivest for RSA Data Security Inc., has been incorporated into many commercial products including BSAFE and Lotus Notes, and is being considered in upcoming standards such as TLS[1].

In this paper, the RC4<sup>1</sup> algorithm is described and known attacks reviewed. A detailed discussion of “tracking” attacks is provided and estimates of the complexity of cryptanalysis for simplified versions of RC4 are given.

---

<sup>1</sup> While RC4 remains a trade secret of RSA Data Security Inc., the algorithm described in [11] is believed to be output-compatible with RC4. This paper discusses the algorithm given in [11], and is referred to as RC4 for convenience.

## 2 Background

The following description of RC4 is based on that given in [11]. It generalizes RC4 to use  $n$ -bit words, but  $n = 8$  is the most commonly used value. To use RC4, a key is first used to initialize the  $2^n$  word s-box  $S$  and counters  $i$  and  $j$  through Algorithm 1. The keystream  $\mathcal{K}$  is then generated using Algorithm 2. The s-box entries and the counters  $i$  and  $j$  are  $n$ -bit words.

**Algorithm 1 (RC4 Initialization).** *Let  $k_0 \dots k_{l-1}$  denote the user's key, a set of  $l$   $n$ -bit words.*

1. For  $z$  from 0 to  $2^n - 1$ 
  - (a) Set  $K_z = k_{z \bmod l}$ .
2. For  $z$  from 0 to  $2^n - 1$ 
  - (a) Set  $S_z = z$ .
3. Set  $j = 0$ .
4. For  $i$  from 0 to  $2^n - 1$ 
  - (a) Set  $j = j + S_i + K_i \bmod 2^n$ .
  - (b) Swap  $S_i$  and  $S_j$ .
5. Set  $i = 0$  and  $j = 0$ .

**Algorithm 2 (Keystream Generation).**

1. Set  $i = i + 1 \bmod 2^n$ .
2. Set  $j = j + S_i \bmod 2^n$ .
3. Swap  $S_i$  and  $S_j$ .
4. Output  $S_{S_i + S_j \bmod 2^n}$  as the next word in the keystream.

The RC4 keystream generation algorithm is depicted in Fig. 1.

The initialization algorithm is a key-dependent variant of the keystream generation algorithm, and is used to initialize the s-box  $S$  to a “randomly chosen” permutation. The nominal key length could be up to  $n \cdot 2^n$  bits, but since it is used to generate only a permutation of  $2^n$  values, the entropy provided by the key can be at most  $\log_2(2^{n!})$  bits, which will be referred to as the effective key length. Table 1 shows the nominal and effective key lengths for different values of  $n$ . In the remainder of this paper, the  $\bmod 2^n$  is sometimes omitted for brevity.

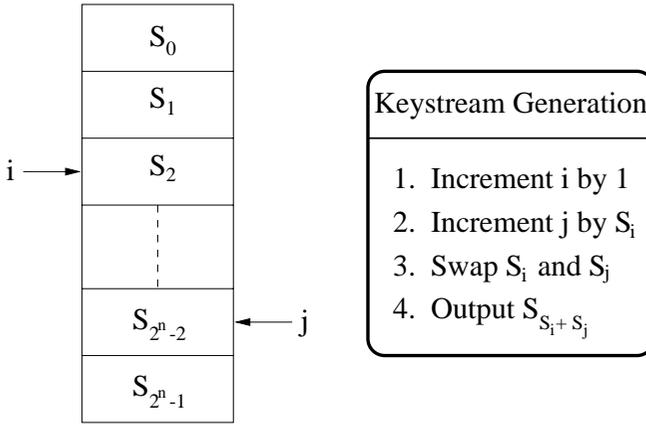
## 3 Published Results

This section is based on [7].

### 3.1 A Class of Weak Keys

In 1995, Andrew Roos posted a paper to the sci.crypt newsgroup[10] describing a class of weak keys, for which the initial byte of the keystream is highly correlated with the first few key bytes. The weak keys are those satisfying

$$k_0 + k_1 \equiv 0 \bmod 2^n .$$



**Fig. 1.** RC4 Keystream Generation

**Table 1.** Nominal and Effective Key Sizes for RC4- $n$

| RC4 Word Size | Nominal Key Length (bits) | Effective Key Length (bits) |
|---------------|---------------------------|-----------------------------|
| 2             | 8                         | 4.58                        |
| 3             | 24                        | 15.30                       |
| 4             | 64                        | 44.25                       |
| 5             | 160                       | 117.66                      |
| 6             | 384                       | 296.00                      |
| 7             | 896                       | 716.16                      |
| 8             | 2048                      | 1684.00                     |
| 9             | 4608                      | 3875.17                     |

The weak keys occur because the keystream initialization algorithm swaps a given entry of the s-box exactly once (corresponding to when the pointer  $i$  points to the entry) with probability  $1/e$ . In addition, for low values of  $i$ , it is likely that  $S_j = j$  during the initialization. The reduction in search effort from this attack is  $2^{5.1}$ , but if linearly related session keys are used, the reduction in effort increases to  $2^{18}$ .

### 3.2 Linear Statistical Weaknesses in RC4

In [3], the author derives a linear model of RC4 using the linear sequential circuit approximation (LSCA) method. The model has correlation coefficient  $15 \cdot 2^{-3n}$ , and requires  $64^n/225$  keystream words. The model is successful in part because the s-box evolves slowly.

### 3.3 A Set of Short Cycles

Suppose that  $i = a$ ,  $j = a+1$ , and  $S_{a+1} = 1$  for some  $a$ . Then, after one iteration,  $i = a+1$ ,  $j = a+2$ , and  $S_{a+2} = 1$ . Thus, the original relationship is preserved. Each such cycle has length  $2^n \cdot (2^n - 1)$ , and  $(2^n - 2)!$  such cycles exist. Note however that, because RC4 is initialized to  $i = j = 0$ , these cycles never occur in practice. These observations were first made in [2] and outlined in [5].

## 4 Cycle Structures in RC4

### 4.1 Comparison with Randomly Chosen Invertible Mappings

The state of RC4 is fully determined by the two  $n$ -bit counters  $i$  and  $j$  and the s-box  $S$ . Since the number of states is finite, it must ultimately be periodic as the keystream generation function is iterated. Because the keystream generation function is invertible, the sequence of states is periodic. The length of the period depends on the word size  $n$  and the particular starting state, as illustrated in Table 2 for  $n = 2$  and  $n = 3$ . For each period, the number of distinct cycles of that period is listed, followed by the number of initial states in each cycle, expressed as a formal sum. The last three columns will be explained in the next section. For comparison, Fig. 2 plots the expected cycle lengths for a randomly chosen permutation and those observed for RC4. For the randomly chosen permutation, the minimum and maximum lengths observed for the  $k$ th longest cycle in a set of 1500 arbitrarily chosen permutations is plotted.

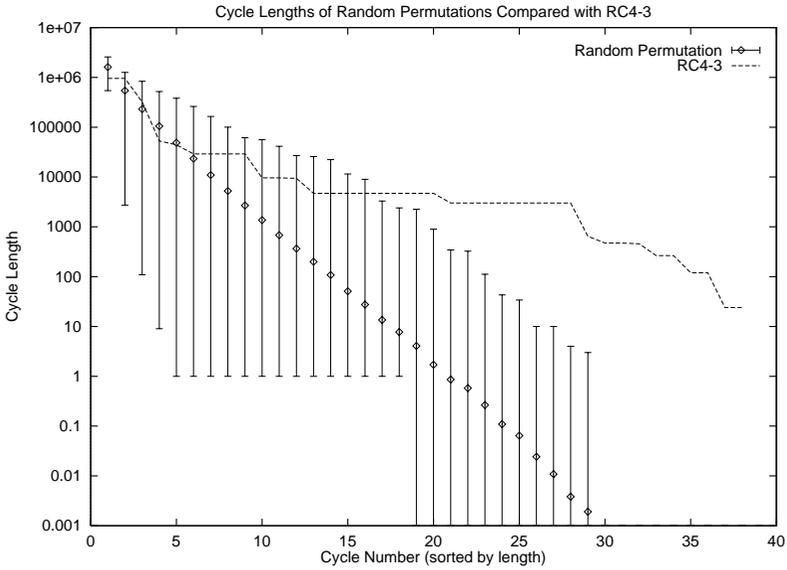
It has also been observed that RC4 keystream sequences are slightly biased[4]. Define the gap at  $i$  for a sequence  $s$  to be the smallest integer  $t \geq 0$  such that  $s_i = s_{i-t-1}$ . For a random sequence in which each element takes on one of  $2^n$  values, the probability that  $t = k$  is given by

$$\left(\frac{2^n - 1}{2^n}\right)^k \cdot \frac{1}{2^n}.$$

Table 3 shows the ratio of the actual to the expected gap probability, based on a sample of approximately  $2^{30}$  elements of an arbitrarily chosen RC4 keystream. For all values of  $n$ , gaps of length 0 are more likely than expected, and gaps of length 1 are less likely than expected. In support of this, it has also been observed that the probability that  $S_i = 0$  is lower than expected and that the probability that  $S_i = 2^n - 1$  is higher than expected after a gap of length 0.

**Table 2.** Possible Periods for RC4 with Word Length 2 and 3

| $n$ | Period | # of Cycles | Number of Initial States                          | Shift Generator | Offset | Shifts Found (indexes are right shifts)               |
|-----|--------|-------------|---|-----------------|--------|---|
| 2   | 196    | 1           | 12  | 1               | 49     | {0, 1, 2, 3}  |
|     | 164    | 1           | 12  | 1               | 41     | {0, 1, 2, 3}  |
| 3   | 955496 | 2           | $15010 + 15274 = 30284$                           | 2               | 238874 | {0, 2, 4, 6}, {0, 2, 4, 6}                            |
|     | 322120 | 1           | 5144  | 1               | 40265  | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 53000  | 1           | 816   | 1               | 6625   | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 44264  | 1           | 688   | 5               | 5533   | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 29032  | 4           | $482 + 505 + 488 + 457 = 1932$                    | 4               | 14516  | {0, 4}, {0, 4}, {0, 4}, {0, 4}                        |
|     | 9624   | 2           | $153 + 149 = 302$                                 | 6               | 2406   | {0, 2, 4, 6}, {0, 2, 4, 6}                            |
|     | 9432   | 1           | 140   | 3               | 3537   | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 4696   | 8           | $80 + 77 + 70 + 93$<br>$+61 + 77 + 81 + 83 = 622$ | 0               | 4696   | {0}, {0}, {0}, {0},<br>{0}, {0}, {0}, {0}             |
|     | 3008   | 8           | $50 + 41 + 32 + 35$<br>$+46 + 55 + 43 + 38 = 340$ | 0               | 3008   | {0}, {0}, {0}, {0},<br>{0}, {0}, {0}, {0}             |
|     | 648    | 1           | 8   | 1               | 81     | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 472    | 2           | $7 + 7 = 14$                                      | 6               | 118    | {0, 2, 4, 6}, {0, 2, 4, 6}                            |
|     | 456    | 1           | 12  | 1               | 57     | {0, 1, 2, 3, 4, 5, 6, 7}                              |
|     | 264    | 2           | $5 + 7 = 12$                                      | 2               | 66     | {0, 2, 4, 6}, {0, 2, 4, 6}                            |
|     | 120    | 2           | $2 + 2 = 4$                                       | 7               | 15     | {0, 1, 2, 3, 4, 5, 6, 7},<br>{0, 1, 2, 3, 4, 5, 6, 7} |
|     | 24     | 2           | $1 + 1 = 2$                                       | 4               | 12     | {0, 4}, {0, 4}  |



**Fig. 2.** Comparison of Cycle Lengths for RC4 and Random Permutations

**Table 3.** Deviation of RC4 Gap Lengths from those of Random Keystream

| $n$ | Gap      |          |          |          |          |         |          |
|-----|----------|----------|----------|----------|----------|---------|----------|
|     | 0        | 1        | 2        | 3        | 4        | 5       | 6        |
| 2   | 1.04082  | 0.952381 | 0.834467 | 0.870748 | 0.902998 | 1.72    | 1.26133  |
| 3   | 1.01828  | 0.956577 | 1.01042  | 0.994535 | 1.02179  | 1.00909 | 1.00284  |
| 4   | 1.00365  | 0.993622 | 1.0009   | 1.00126  | 1.00276  | 1.00039 | 1.00059  |
| 5   | 1.00099  | 0.99859  | 0.999946 | 1.0009   | 1.00081  | 1.00024 | 1.00046  |
| 6   | 0.999762 | 0.999901 | 1.00024  | 1.00039  | 1.00036  | 1.00014 | 0.999714 |

## 4.2 Partitioning of RC4 Cycles

As observed independently in [5], individual RC4 cycles can be partitioned into pieces of “equivalent” subsets, as follows. Define by  $S^{\gg d}$  the s-box obtained by rotating the s-box entries to the right (or down) by  $d$  (formally,  $S'_t = S^{\gg d}$  if  $S'_t = S_{t-d \bmod 2^n}$ ). Let the right shift by  $d$  of an RC4 state  $(i, j, S)$  be defined by  $(i + d, j + d, S^{\gg d})$ . The following theorem holds:

**Theorem 1.** *Suppose that, for a given key, an RC4- $n$  system goes through the state  $(i', j', S')$  and that the cycle length for this key is  $T$ . Then any cycle going through one or more states of the form  $(i' + d, j' + d, S'^{\gg d})$  (where  $d$  is an integer) has period  $T$  and the shift relationship between the states is maintained as the two systems evolve. In addition, if the output sequences are compared word for word as the systems evolve beyond those states, the outputs will always differ if  $d \not\equiv 0 \pmod{2^n}$  and will always agree otherwise.*

*Proof.* Compare the state evolutions of the two systems  $(i', j', S')$  and  $(i'' = i' + d, j'' = j' + d, S'' = S'^{\gg d})$ . The steps for one round of keystream generation are:

1. Set  $i'' = i'' + 1 \pmod{2^n}$ .
2. Set  $j'' = j'' + S''_{i''} \pmod{2^n}$ .
3. Swap  $S''_{i''}$  and  $S''_{j''}$ .
4. Output  $S''_{S''_{i''} + S''_{j''} \pmod{2^n}}$  as the next word in the keystream.

or

1. Set  $i'' = i'' + 1 \pmod{2^n}$ .
2. Set  $j'' = j'' + S''_{i''-d} \pmod{2^n}$ .
3. Swap  $S''_{i''}$  and  $S''_{j''}$ .
4. Output  $S''_{S''_{i''-d} + S''_{j''-d} - d \pmod{2^n}}$  as the next word in the keystream.

which becomes:

1. Set  $i'' = i' + d + 1 \pmod{2^n}$ .
2. Set  $j'' = j' + d + S'_{i'}$   $\pmod{2^n}$ .
3. Swap  $S''_{i'+d}$  and  $S''_{j'+d}$ .
4. Output  $S'_{S'_{i'+d} + S'_{j'+d} - d \pmod{2^n}}$  as the next word in the keystream.

Thus, the shift relationship between the two systems is preserved, and only the output is different provided  $d \not\equiv 0 \pmod{2^n}$ . Because the systems are identical except for the output, the periods of the two systems must also be the same.  $\square$

Consider a cycle of period  $T$ , and an arbitrary RC4 state  $(i', j', S')$  in that cycle. Then all shifts of this state belong to a cycle of length  $T$ . If there are only a few cycles of length  $T$  (as will be the case if  $T$  is large), then more than one may appear in the same cycle. The following theorem holds:

**Theorem 2 (Cycle Partitioning).** *Let  $\gamma$  be a cycle of period  $T$  and let  $D = (i', j', S')$  be any state in the cycle. Let  $d_1, \dots, d_{k-1}$  ( $k < 2^n$ ) be the right shifts of  $D$  in the order they appear as RC4 evolves from state  $D$  ( $d_0 = 0$  is understood). Then the distance (expressed as the number of encryptions) between successive shifts is given by  $T/k$ , and for any other state,  $D'$ , in the same cycle, the right shifts of  $D'$   $d_1, \dots, d_{k-1}$  are the only right shifts of  $D'$  appearing in the cycle, and appear in that order. Figure 3 illustrates this partitioning, with  $\alpha = T/k$  and  $k = 4$ .*

*Proof.* Denote by  $D_t$  the right shift by  $d_t$  of  $D$ , and by  $D(s)$  the RC4 state obtained by performing  $s$  encryptions starting at state  $D$ . Let  $l$  be the greatest distance between two consecutive shifts and denote the corresponding shifts  $d_a$  and  $d_{a+1 \bmod k}$ . Suppose that for some  $b$ , the distance  $s$  between  $d_b$  and  $d_{b+1 \bmod k}$  was smaller than  $l$ . By Theorem 1,  $D_a$  remains a right shift of  $D_b$  as the systems evolve. Since  $D_b(s)$  is a right shift of  $D_b$ ,  $D_a(s)$  must be a right shift of  $D_a$ . Thus, the distance between  $D_a$  and  $D_{a+1 \bmod k}$  is less than or equal to  $s$ . But that distance is  $l$ , contradicting the assumption that  $s < l$ . Therefore, no smaller distance exists, and the shifts are at equal distances from each other. The second part of the theorem follows by Theorem 1 and the fact that any state in the cycle can be obtained by repeated encryption starting at any state  $D$ .  $\square$

In fact, only certain orderings of the shifts present in a given cycle are possible because Theorem 2 implies that  $d_{i+1} - d_i$  is constant in a cycle.  $d_1$  must then be a generator for the shifts in the cycle, and  $d_i = i \cdot d_1 \bmod 2^n$ . The last three columns of Table 2 confirm this statement for RC4-2 and RC4-3. In this table, the “Shift Generator” entry is the value of  $d_1$ , and the entry “Offset” is the distance between successive shifts. Finally, the “Shifts Found” table enumerates the right shifts of the initial state found in each cycle. All of these results were obtained experimentally. Note that in all cases,  $T/k = \text{Offset}$  as required by the theorem. For example, for the cycles of length 472, a distance of 118 exists between shifts, the shifts appear in the order  $\{0, 6, 4, 2\}$ , and  $472/4 = 118$ . The entry  $\{0\}$  in the “Shifts Found” column indicates that no shifts of the initial state appear in the cycle.

## 5 Tracking Analysis

Algorithm 3 below outlines a basic attack which can be mounted against RC4. In essence, the algorithm keeps track of all states which RC4 could be in, given that a particular keystream has been generated.

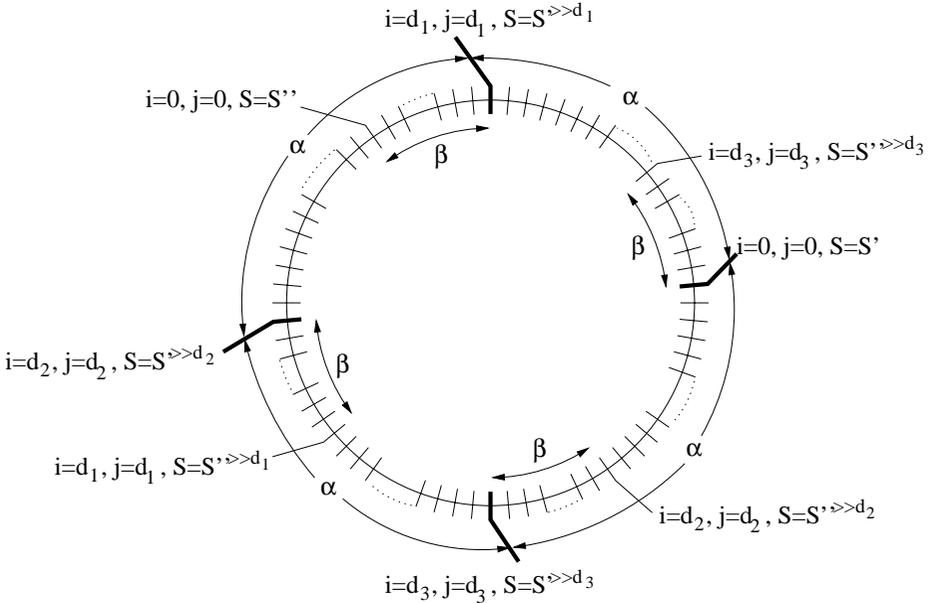


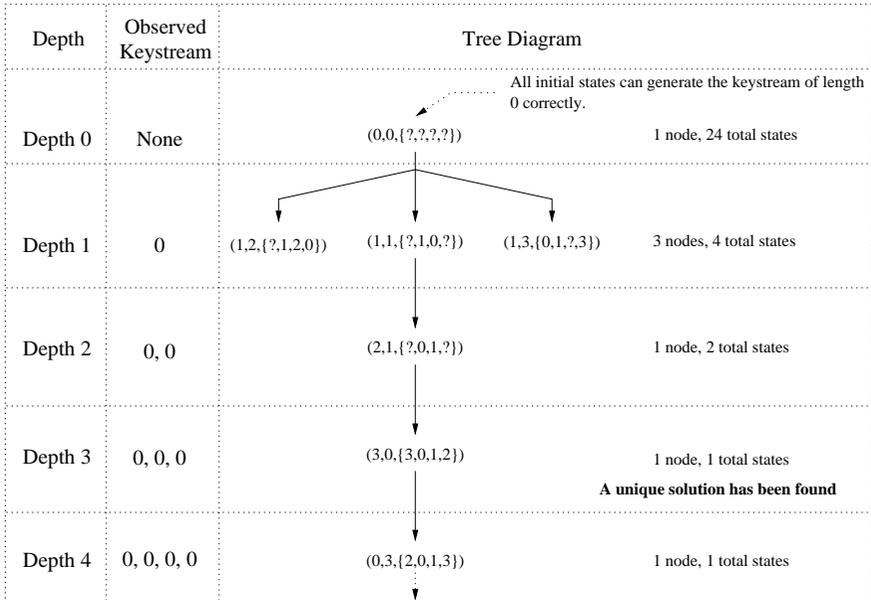
Fig. 3. Partitioning of an RC4 Cycle

**Algorithm 3 (Forward Tracking).**

1. Mark all entries  $S_t$  as unassigned.
2. Set  $i = 0, j = 0,$  and  $z = 0.$
3. Repeat:
  - (a) Set  $i = i + 1 \bmod 2^n.$
  - (b) If  $S_i$  is unassigned, continue with the remainder of the algorithm for each possible assignment of  $S_i.$
  - (c) Set  $j = j + S_i \bmod 2^n.$
  - (d) If  $S_j$  is unassigned, continue with the remainder of the algorithm for each possible assignment of  $S_j.$
  - (e) Swap  $S_i$  and  $S_j.$
  - (f) Set  $t = S_i + S_j \bmod 2^n.$
  - (g) If  $S_t$  is unassigned and  $\mathcal{K}_z$  does not yet appear in the s-box, set  $S_t = \mathcal{K}_z.$
  - (h) If  $S_t \neq \mathcal{K}_z,$  the state information is incorrect. Terminate this round.
  - (i) Increment  $z.$
  - (j) If  $z$  is equal to the length of the keystream, output the current state as a solution and terminate the run.

The forward tracking algorithm is illustrated in Fig. 4. In this diagram, the observed keystream is the all zero sequence, and the system is RC4-2. Figure 5 shows the number of nodes visited by the tracking algorithm as a function

of depth for various word sizes  $n$ . Two cases are considered for the observed keystream; an arbitrarily chosen nonzero keystream, and a zero keystream. The zero keystream can be analysed more quickly than a more general keystream. To obtain approximate data for the  $n = 5$  case in Fig. 5, at a given depth the depth-first search was only carried out for selected nodes. The total number of nodes visited was then calculated assuming that the number of nodes in each subtree would be the same for all nodes. Similar work has been done by Luke O'Connor[8].



**Fig. 4.** Forward Tracking Algorithm for  $n = 2$ , and a 0 Keystream of Length 4

Several variations of this algorithm are possible. Backtracking, in which the keystream is processed in reverse, appears to be easier to implement efficiently because s-box entries are fixed sooner. Probabilistic variants, which use truncated tracking analysis or other information to determine the “best” node to follow in the tracking analysis, may be able to analyse more keystream, avoiding keystreams which result in a more difficult search. These variants are discussed in [6].

The performance of these algorithms can be used to provide an upper bound on the complexity of RC4 cryptanalysis. Table 4 shows the attack complexity observed to date. All of the results are based on backtracking, except the nonzero keystream  $n = 5$  entry, based on an estimate from a probabilistic backtracking attack.

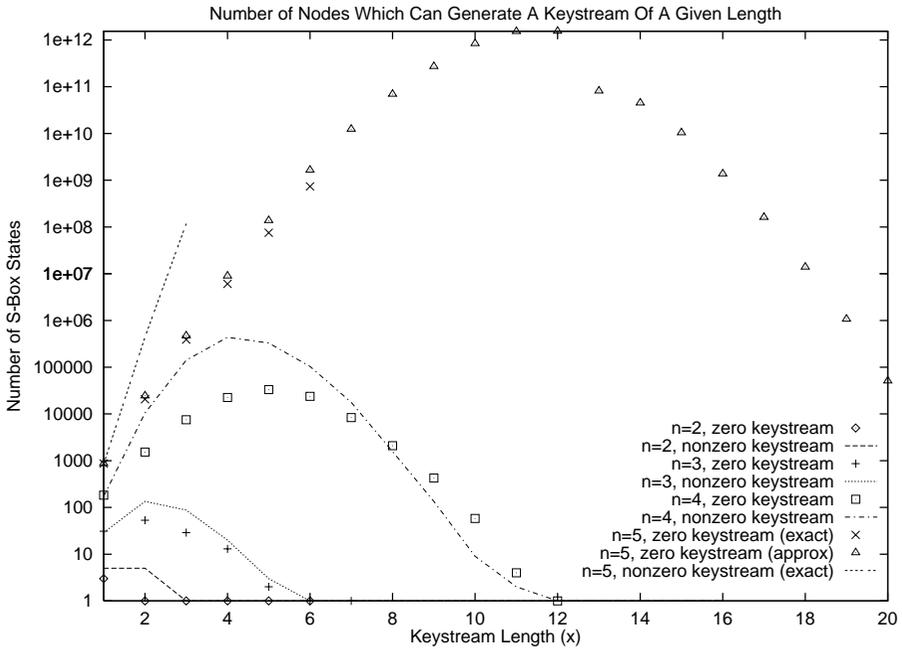


Fig. 5. Number of Nodes During Forward Tracking

## 6 An Attack on a Weakened Version of RC4

Suppose that RC4 was modified by replacing its initialization function with the following:

**Algorithm 4 (Weak RC4 Initialization).** Let  $k_0 \dots k_{l-1}$  denote the user's key.

1. Calculate  $\gamma$  such that  $\log_2(\gamma!) = 8 \cdot l$ .
2. For  $z$  from 0 to  $2^n - 1$ 
  - (a) Set  $K_z = k_{z \bmod l}$ .
3. For  $z$  from 0 to  $2^n - 1$ 
  - (a) Set  $S_z = z$ .
4. Set  $j = 0$ .
5. For  $i$  from 0 to  $2^n - 1$ 
  - (a) Set  $j = j + S_i + K_i \bmod 2^n$ .
  - (b) Swap  $S_{i \bmod \gamma}$  and  $S_{j \bmod \gamma}$ .
6. Set  $i = 0$  and  $j = 0$ .

The system still has  $8 \cdot l$  bits of entropy. However, because the tracking analysis can easily be confined to searching the reduced key space, it is likely to succeed

**Table 4.** Estimated Upper Bound on the Complexity of Cryptanalysis of RC4- $n$

| RC4<br>Word Size | Nominal<br>Key Space | Effective<br>Keyspace | Attack<br>Complexity<br>(arbitrary keystream) | Attack<br>Complexity<br>(zero keystream) |
|------------------|----------------------|-----------------------|---|--|
| 2                | $2^8$                | $2^{4.58}$            | $2^4$   | $2^3$                                    |
| 3                | $2^{24}$             | $2^{15.30}$           | $2^8$   | $2^7$                                    |
| 4                | $2^{64}$             | $2^{44.25}$           | $2^{20}$                                      | $2^{17}$                                 |
| 5                | $2^{160}$            | $2^{117.66}$          | $2^{69}$                                      | $2^{42}$                                 |
| 6                | $2^{384}$            | $2^{296.00}$          | ?   | ?  |
| 7                | $2^{896}$            | $2^{716.16}$          | ?   | ?  |
| 8                | $2^{2048}$           | $2^{1684.00}$         | ?   | ?  |

very quickly even for the full-size cipher. Table 5 summarizes the performance of the tracking attack for this weakened variant of RC4. Table 5 was obtained by performing a tracking attack on 20 keystreams generated with randomly chosen keys for each value of  $\gamma$ . The maximum observed complexity is reported. This result shows that RC4 depends heavily on its key schedule for its security. The attack complexity indicated in Table 5 is not monotonically increasing because it is often the case that the attack succeeds in substantially less than the maximum number of steps.

**Table 5.** Tracking Attack Complexity for RC4 with a Weakened Key Schedule ( $n = 8$ )

| $\gamma$ | Effective Keyspace | Attack Complexity |
|----------|--------------------|-------------------|
| 15       | $2^{40}$           | $2^{14}$          |
| 20       | $2^{61}$           | $2^{19}$          |
| 25       | $2^{83}$           | $2^{23}$          |
| 30       | $2^{107}$          | $2^{17}$          |
| 31       | $2^{112}$          | $2^{28}$          |
| 32       | $2^{117}$          | $2^{26}$          |
| 33       | $2^{122}$          | $2^{26}$          |

## 7 Conclusion

RC4 remains a secure cipher for practical applications. Several theoretical attacks exist but none have been successful against commonly used key lengths. Nonetheless, tracking analysis does substantially reduce the complexity of cryptanalysis compared to the maximum key length which could be specified. Tracking analysis would show promise if it were possible to use knowledge of the actual key length to limit the state space to be searched. In this regard, RC4's resilience is mainly due to the fact that the key schedule effectively prevents partial knowledge of the s-box state from providing information about the key. If this were not the case, tracking analysis would be successful even for the full-size cipher.

## References

1. T. Dierks and C. Allen. The TLS protocol version 1.0. Internet Draft, <ftp://ftp.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt>, November 1997.
2. H. Finney. An RC4 cycle that can't happen. Posting to sci.crypt, Sept. 1994.
3. J. D. Golić. Linear statistical weakness of alleged RC4 keystream generator. In Walter Fumy, editor, *LNCS 1233, Advances in Cryptology - EUROCRYPT '97*, pages 226–238, Germany, 1997. Springer.
4. R. J. Jenkins Jr. Re: RC4? Posting to sci.crypt, Sept 1994.
5. R. J. Jenkins Jr. ISAAC and RC4. Internet document at [http://ourworld.compuserve.com/homepages/bob\\_jenkins/isaac.htm](http://ourworld.compuserve.com/homepages/bob_jenkins/isaac.htm), 1996.
6. S. Mister. Cryptanalysis of RC4-like stream ciphers. Master's thesis, Queen's University, Kingston, Ontario, 1998.
7. S. Mister and S. E. Tavares. Some results on the cryptanalysis of RC4. In *Proceedings of the 19th Biennial Symposium on Communications*, pages 393–397, Kingston, Ontario, June 1-3, 1998.
8. L. O'Connor. Private communication, August 1998.
9. R. L. Rivest. The RC4 encryption algorithm. RSA Data Security Inc., March 1992.
10. A. Roos. A class of weak keys in the RC4 stream cipher. Posting to sci.crypt, Sept. 1995.
11. B. Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., Toronto, Canada, 2nd edition, 1996.