Statistical Analysis of the Alleged RC4 Keystream Generator

Scott R. Fluhrer and David A. McGrew

Cisco Systems, Inc.
170 West Tasman Drive, San Jose, CA 95134
{sfluhrer, mcgrew}@cisco.com

Abstract. The alleged RC4 keystream generator is examined, and a method of explicitly computing digraph probabilities is given. Using this method, we demonstrate a method for distinguishing 8-bit RC4 from randomness. Our method requires less keystream output than currently published attacks, requiring only $2^{30.6}$ bytes of output. In addition, we observe that an attacker can, on occasion, determine portions of the internal state with nontrivial probability. However, we are currently unable to extend this observation to a full attack.

1 Introduction

We show an algorithm for deriving the exact probability of a digraph in the output of the alleged RC4 stream cipher. This algorithm has a running time of approximately 2^{6n}, where n is the number of bits in a single output. Using the computed probabilities of each digraph for the case that $n = 5$, we discern which digraphs have probabilities furthest from the value expected from a uniform random distribution of digraphs. Extrapolating this knowledge, we show how to distinguish the output of the alleged RC4 cipher with $n = 8$ from randomness with $2^{30.6}$ outputs. This result improves on the best known method of distinguishing that cipher from a truly random source. In addition, heuristic arguments about the cause of the observed anomalies in the digraph distribution are offered.

The irregularities in the digraph distribution that we observed allow the recovery of n and i parameters (defined in Section 2) if the attacker happens not to know them. Also, an attacker can use this information in a ciphertext-only attack, to reduce the uncertainty in a highly redundant unknown plaintext.

We also observe how an attacker can learn, with nontrivial probability, the value of some internal variables at certain points by observing large portions of the keystream. We are unable to derive the entire state from this observation, though with more study, this insight might lead to an exploitable weakness in the cipher.

This paper is structured as follows. In Section 2, the alleged RC4 cipher is described, and previous analysis and results are summarized. Section 3 presents our analysis of that cipher, and Section 4 investigates the mechanisms behind the

B. Schneier (Ed.): FSE 2000, LNCS 1978, pp. 19–30, 2001.

statistical anomalies that we observe in that cipher. Section 5 examines fortuitous states, which allow the attacker to deduce parts of the internal state. In Section 6, extensions of our analysis and directions for future work are discussed. Section 7 summarizes our conclusions. Lastly, the Appendix summarizes the results from information theory that are needed to put a strong bound the effectiveness of tests based on the statistical anomalies, and presents those bounds for our work and for previous work.

2 Description of the Alleged RC4 Cipher and Other Work

The alleged RC4 keystream generator is an algorithm for generating an arbitrarily long pseudorandom sequence based on a variable length key. The pseudorandom sequence is conjectured to be cryptographically secure for use in a stream cipher. The algorithm is parameterized by the number of bits n within a permutation element, which is also the number of bits that are output by a single iteration of the next state function of the cipher. The value of $n = 8$ is of greatest interest, as this is the value used by all known RC4 applications.

The RC4 keystream generator was created by RSA Data Security, Inc. [6]. An anonymous source claimed to have reverse-engineered this algorithm, and published an alleged specification of it in 1994 [8]. Although public confirmation of the validity of this specification is still lacking, we abbreviate the name 'alleged RC4' to 'RC4' in the remainder of this paper. We also denote n-bit RC4 as RC4/n.

A summary of the RC4 operations is given in Table 1. Note that in this table, and throughout this paper, all additions and increments are done modulo 2^n.

Table 1. The RC4 next state function. i and j are elements of $\mathbb{Z}/2^n$, and S is a permutation of integers between zero and $2^n - 1$. All increments and sums are modulo 2^n.

1. Increment i by 1
2. Increment j by $S[i]$
3. Swap $S[i]$ and $S[j]$
4. Output $S[S[i] + S[j]]$

2.1 Previous Analysis of RC4

The best previously known result for distinguishing the output of RC4 from that of a truly random source was found by Golić [3,2], who presents a statistical defect that he estimates will allow an attacker to distinguish RC4/8 from randomness with approximately 2^{40} successive outputs. However, this result appears to be somewhat optimistic. We use the information theoretic lower bound

on the number of bytes needed to distinguish RC4 from randomness, for a given statistical anomaly, and use this to measure the effectiveness of Golić's anomaly and our own anomalies (see the Appendix). The number of bytes of RC4/8 output needed to reduce the false positive and false negative rates to 10% is $2^{44.7}$, using Golić's anomaly, while the irregularities in the digraph distribution that we found require just $2^{30.6}$ bytes to achieve the same result.

Mister and Tavares analyzed the cycle structure of RC4 [5]. They observe that the state of the permutation can be recovered, given a significant fraction of the full keystream. In addition, they also present a backtracking algorithm that can recover the permutation from a short keystream output. Their analyses are supported by experimental results on RC4/n for $n < 6$, and show that an RC4/5 secret key can be recovered after only 2^{42} steps, though the nominal key size is about 160 bits.

Knudsen et. al. presented attacks on weakened versions of RC4 [4]. The weakened RC4 variants that they studied change their internal state less often than does RC4, though they change it in a similar way. Their basic attack backtracks through the internal state, guessing values of table entries that have not yet been observed, and backtracking upon contradictions. They present several variants of their attack, and analyze its runtime. They estimate that the complexity of their attack is less than the square root of the number of possible RC4 states.

3 Analysis of Digraph Probabilities

The probability with which each digraph (that is, each successive pair of n-bit outputs) will appear in the output of RC4 is directly computable, given some reasonable assumptions. The probability of each digraph for each value of the i index is also computable. By taking advantage of the information on i, rather than averaging over all values of i and allowing some of the detail about the statistical anomalies to wash away, it is possible to more effectively distinguish RC4 from randomness.

To simplify analysis, we idealize the key set up phase. We assume that the key setup will generate each possible permutation with equal probability, and will assign all possible values to j with equal probability. Then, after any fixed N steps, all states of j and the permutation will still have equally probability, because the next state function is invertible. This is an idealization; the actual RC4 key setup will initialize j to zero. Also, the RC4 key setup routine generates only 2^{n_k} different permutations, where n_k is the number of bits in the key, while there are $2^n!$ possible permutations. Intuitively, our idealization becomes a close approximation of the internal state after RC4 runs for a short period of time.

However, we leave in the assumption that the i pointer is initially zero after the key setup phase. Note that, since each step changes i in a predictable manner, the attacker can assume knowledge of the i pointer for each output.

We compute the exact digraph probabilities, under the assumptions given above, by counting the number of internal states consistent with each digraph. This approach works with RC4 because only a limited amount of the unknown

internal state actually affects each output, though the total amount of internal state is quite large.

Starting at step 4 of Table 1, we look at what controls the two successive outputs. The exhaustive list of everything on which those two outputs depend on is given in Table 2.

Table 2. The variables that control two successive outputs of RC4 and the cryptanalyst's knowledge of them.

Variable	**Cryptanalyst's knowledge**
i	known (increments regularly)
j	unknown
$S[i]$	unknown
$S[j]$	unknown
$S[S[i] + S[j]]$	known (first output)
$S[i+1]$	unknown
$S[j + S[i+1]]$	unknown
$S[j + S[i+1]]$ if $i + 1 = S[i+1] + S[j + S[i+1]]$ $S[i+1]$ if $j + S[i+1] = S[i+1] + S[j + S[i+1]]$ $S[S[i+1] + S[j + S[i+1]]]$ otherwise	known (second output)

As the next-state algorithm progresses, for each successive unknown value, any value that is consistent with the previously seen states is equally probable. Thus the probability of a digraph (a, b) for a particular value of i can be found by stepping through all possible values of all other variables, and counting the number of times that each output is consistent with the fixed values of i, a, and b. The consistency of a set of values is determined by the fact that S is a permutation. Because the start states were considered equally probable, this immediately gives us the exact value of the probability of $i, (a, b)$. This approach requires about 2^{5n} operations to compute the probability of a single digraph, for a given value of i, as there are five n-bit unknowns in Table 2. Approximately 2^{8n} operations are required to compute the probabilities of a digraph for all values of i. This puts the most interesting case of $n = 8$ out of immediate reach, with a computational cost of 2^{64} operations. However, we circumvented this difficulty by computing the exact $n = 3, 4, 5$ digraph distributions for all i, observing which digraphs have anomalous probabilities, and estimating the probabilities of the anomalous digraphs for RC4/8. This method is described in the next two subsections.

3.1 Anomalous RC4 Outputs

The full digraph distributions for $n = 3, 4$, and 5 are computable with about 2^{40} operations. We computed these, and found that the distributions were significantly different from a uniform distribution. In addition, there is a consistency (across different values of n) to the irregularities in the digraph probabilities. In

particular, one type of digraph is more probable than expected by a factor of approximately $1 + 2^{-n+1}$, seven types of digraphs are more probable than expected by approximately $1 + 2^{-n}$, and three types of digraphs are less probable than expected by approximately $1+2^{-n}$. These results are summarized in Table 3.

Table 3. Positive and negative events. Here, i is the value of the index when the first symbol of the digraph the output. The top eight digraphs are in the set of positive events, and the bottom three digraphs are in the set of negative events, as defined in Section 3.1. The probabilities are approximate.

Digraph	**Value(s) of i**	**Probability**
$(0, 0)$	$i = 1$	$2^{-2n}(1 + 2^{-n+1})$
$(0, 0)$	$i \neq 1, 2^n - 1$	$2^{-2n}(1 + 2^{-n})$
$(0, 1)$	$i \neq 0, 1$	$2^{-2n}(1 + 2^{-n})$
$(i + 1, 2^n - 1)$	$i \neq 2^n - 2$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, i + 1)$	$i \neq 1, 2^n - 2$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, i + 2)$	$i \neq 0, 2^n - 1, 2^n - 2, 2^n - 3$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, 0)$	$i = 2^n - 2$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, 1)$	$i = 2^n - 1$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, 2)$	$i = 0, 1$	$2^{-2n}(1 + 2^{-n})$
$(2^{n-1} + 1, 2^{n-1} + 1)$	$i = 2$	$2^{-2n}(1 + 2^{-n})$
$(2^n - 1, 2^n - 1)$	$i \neq 2^n - 2$	$2^{-2n}(1 - 2^{-n})$
$(0, i + 1)$	$i \neq 0, 2^n - 1$	$2^{-2n}(1 - 2^{-n})$

We call the event that a digraph appears in the RC4 output at a given value of i a *positive event* when it is significantly more probable than expected. A *negative event* is similarly defined to be the appearance of a digraph at a given i that is significantly less probable than expected. An exhaustive list of positive and negative events is provided in Table 3.

In Section 4, we examine these particular digraphs to see why they are more or less likely than expected. Most of the positive events correspond to length 2 fortuitous states, which will be defined in Section 5. For the $(0, 1)$ and $(0, 0)$ positive events, and the negative events, a more complicated mechanism occurs, which is discussed in the next section.

3.2 Extrapolating to Higher Values of n

To apply our attack to higher values of n without directly computing the digraph probabilities, we computed the probabilities of positive events and negative events by running RC4/8 with several randomly selected keys and counting the occurances of those events in the RC4 output. The observed probabilities (derived using RC4/8 with 10 starting keys for a length of 2^{38} for each key), along with the computed expected probability from a truly random sequence, are given in Table 4. It is possible to distinguish between these two probability

distributions with a 10% false positive and false negative rate by observing $2^{30.6}$ successive outputs using the data in Table 3 (see the Appendix).

Table 4. Comparison of event probabilities between RC4/8 and a random keystream. The listed probabilities are the probability that two successive outputs are the specified event

	Positive Events	**Negative Events**
RC4/8	0.00007630	0.00003022
Random	0.00007600	0.00003034

In order to evaluate the effectiveness of our 'extrapolation' approach, we compute the amount of keystream needed using a test based on our observed positive and negative events, and compare that to the best possible test using the exact probabilities. For RC4/5 our selected positive/negative events require $2^{18.76}$ keystream outputs, while the optimal test using the exact probabilities of all digraphs requires $2^{18.62}$ keystream outputs. These numbers agree to within a small factor, suggesting that the extrapolation approach is close to optimal.

4 Understanding the Statistical Anomalies

In this section we analyze the next state function of RC4 and show mechanisms that cause the increased (or decreased) likelihood of some of the anomalous digraphs. The figures below show the internal state of RC4 immediately before state 4 in Table 1 of the first output of the digraph. The bottom line shows the state of the permutation. Those permutation elements with a specific value are labeled with that value. Elements that are of unspecified value are labeled with the 'wildcard' symbol $*$. Ellipsis indicate unspecified numbers of unspecified elements, and elements separated by ellipsis may actually be in opposite order within the permutation. The elements pointed to by i and j are indicated by the i and j symbols appearing above them.

The mechanism that leads to the $(0, 1)$ digraph starts in the state

$$\overset{}{*}, \ldots, \overset{}{*}, \overset{i}{1}, \overset{}{0}, \overset{}{*}, \ldots, \overset{}{*}, \overset{j}{AA}, \overset{}{*}, \ldots \text{ where } AA = i.$$

Following through the steps in the next-state function, the first output will be 0, and at the following step 4, be in the state

$$\overset{}{*}, \ldots, \overset{}{*}, \overset{\mathrm{i}}{1}, \overset{j}{AA}, \overset{}{*}, \ldots, \overset{}{*}, \overset{}{0}, \overset{}{*}, \ldots$$

and output an 1. This mechanism occurs approximately 2^{-3n} of the time, and since other mechanisms output a $(0, 1)$ 2^{-2n} of the time, this accounts for the observed increase over expected.

For the $(0, 0)$ positive events, the additional mechanism starts with the following state:

$$\overset{}{*}, \ldots, \overset{}{*}, \overset{i}{AA}, \overset{}{0}, \overset{}{*}, \ldots, \overset{}{*}, \overset{j}{BB}, \overset{}{*}, \ldots \text{ where } AA = i + 1 - j \text{ and } BB = j.$$

The negative events, on the other hand, correspond to mechanisms that normally contribute to the expected output which do not work in those particular cases. For example, a normal method of producing a repeated digraph (AA, AA) is[1]:

$$\overset{\quad i \quad j}{*, \ldots, *, BB, \text{-}1, *, \ldots, *, AA, *, \ldots}$$

Here the value AA occurs at location $BB - 1$. This outputs an AA, and steps into the state:

$$\overset{\quad j \quad i}{*, \ldots, *, \text{-}1, BB, *, \ldots, *, AA, *, \ldots}$$

This will output another AA, unless AA happens to be either BB or -1. In either case, this will output a BB. Since normal (AA, AA) pairs rely on this to achieve a near-expected rate, the lack of this mechanism for $(-1, -1)$ prevents the output once every approximately 2^{-3n} outputs, which accounts for the reduction of approximately a factor of 2^{-n} that we observe.

These mechanisms do not depend on the value of n, and so can be expected to operate in the $n = 8$ case. This supports our extrapolation approach, which assumes that the positive and negative events to still apply in that case.

5 Analysis of Fortuitous States

There are RC4 states in which only N elements of the permutation S are involved in the next N successive outputs. We call these states *fortuitous states*[2]. Since the variable i sweeps through the array regularly, it will always index N different array elements on N successive outputs (for $N \leq 2^n$). So, the necessary and sufficient condition for a fortuitous state is that the elements pointed to by j and pointed to by $S[i]+S[j]$ must come from the set of N array elements indexed by i.

An example of an $N = 3$ fortuitous state follows:

$$\overset{\text{i} \quad\quad j}{*, 255, 2, 1, *, *, \ldots,}$$

1. advance i to 1
2. advance j to 2
3. swap $S[1]$ and $S[2]$
4. output $S[1] = 2$

$$\overset{\quad i \quad j}{*, 2, 255, 1, *, *, \ldots,}$$

1. advance i to 2
2. advance j to 1
3. swap $S[2]$ and $S[1]$
4. output $S[1] = 255$

[1] The symbol -1 is used as shorthand for $2^n - 1$ here and throughout the paper.
[2] Observing such a state is fortuitous for a cryptanalyst.

$j \quad i$
$*, 255, 2, 1, *, *, \ldots,$
1. advance i to 3
2. advance j to 2
3. swap $S[3]$ and $S[2]$
4. output $S[3] = 2$

$j \quad i$
$*, 255, 1, 2, *, *, \ldots,$

If $i = 0$ at the first step, and assuming that all permutations and settings for j are equally probable, then the above initial conditions will hold with probability $1/(256 \cdot 256 \cdot 255 \cdot 254)$. When the initial conditions hold, the output sequence will always be $(2, 255, 2)$. If RC4 outputs all trigraphs with equal probability (the results in our previous section imply that it doesn't, but we will use that as an approximation), the sequence $(2, 255, 2)$ will occur at $i = 0$ with probability $1/(256 \cdot 256 \cdot 256)$. This implies that, when the output is the sequence $(2, 255, 2)$ when $i = 0$, then this scenario caused that output approximately $1/253$ of the time. In other words, if the attacker sees, at offset 0, the sequence $(2, 255, 2)$, he can guess that j was initially 3, and $S[1]$, $S[2]$, $S[3]$ was initially 255, 2 and 1, and be right a nontrivial portion of the time.

The number of fortuitous states can be found using a state-counting algorithm similar to that given above. The numbers of such states, for small N, are given in Table 5. The table lists, for each N, the number of fortuitous states that exist of that length, the logarithm (base 2) of the expected time between occurrances of any fortuitous state of that length, and the expected number within that length of false hits. By false hit, we mean those output patterns that have identical patterns as a fortuitous state, but are not caused by a fortuitous state. For example, an attacker can expect to see, in a keystream of length $2^{35.2}$, one fortuitous state of length 4 and 250 output patterns that look like fortuitous states.

Table 5. The number of fortuitous states for RC4/8, their expected occurrance rates, and their expected false hit rates.

Length	Number	Lg(Expected)	Expected False Hits
2	516	22.9	255
3	290	31.8	253
4	6540	35.2	250
5	25,419	41.3	246
6	101,819	47.2	241

It is not immediately clear how an attacker can use this information. What saves RC4 from an immediate break is that the state space is so huge that an attacker who directly guess 56 bits (which is approximately what you get with a length 6 fortuitous pattern) still has so many bits unguessed that there is

no obvious way to proceed. However, it does appear to be a weakness that the attacker can guess significant portions of internal state at times with nontrivial probability.

It may be possible to improve the backtracking approaches to deriving RC4 state [5,4] using fortuitous states. For example, a backtracking algorithm can be started at the keystream location immediately after a fortuitous state, using the values of the internal state that are suggested by that state. This approach extends slightly the attack using 'special streams' presented in Section 4.3 of [4].

6 Directions for Future Work

Some extensions of our current work are possible. One direct extension is to compute the exact digraph probabilities for the case that $n = 8$, and other cases for $n > 5$. Since RC4/n is actually a complex combinatorial object, it may happen that the results for these cases are significantly different than what might be expected.

Another worthwhile direction is to investigate the statistics of trigraphs (that is, the three consecutive output symbols). The exact trigraph probabilities can be computed using an algorithm similar to that outlined in Section 3. The computational cost to compute the complete trigraph distribution, for all i, is 2^{11n}. We have computed this for RC4/4, and found that the length of outputs required to distinguish that cipher from randomness using trigraphs is about one-seventh that required when using digraphs. This result is encouraging, though it does not guarantee that trigraph statistics will be equally as effective for larger values of n. It must be considered that with $n = 4$, there are only $2^4 = 16$ entries in the table S, and that three consecutive output symbols typically uses half of the state in this cipher.

The computational cost of computing the complete trigraph distribution motivates the consideration of lagged digraphs, that is, two symbols of fixed value separated by some symbols of non-fixed value. We call the number of intervening symbols of non-fixed value the *lag*. For example, adapting the notation used above, $(1, *, 2)$ is a lag one digraph with initial value 1 and final value 2. Here we use the 'wildcard' symbol $*$ to indicate that the middle symbol can take on any possible value. Lagged digraphs are far easier to compute than trigraphs, because it is not necessary to individually count the states that are used only to determine the middle symbols. In general, the computational effort to compute the distribution of RC4/n lag L digraphs, for all i, requires about $2^{(8+L)n}$ operations.

Another approach to computing a digraph probability is to list the possible situations that can occur within the RC4 cipher when producing that digraph, generate the equations that must hold among the internal elements, and use algebraic means to enumerate the solutions to those equations. The number of solutions corresponds to the number of states that lead to that digraph. This approach could lead to a method to compute the exact digraph probability in a time independent of n.

Another direction would be to eliminate some of the assumptions made in our analysis. For example, the assumption that S and j are uniformly random is false, and it is especially wrong immediately after key setup. In particular, j is initially set to zero during the key setup. We venture that an analysis of fortuitous states that takes the key setup into consideration may lead to a method for deriving some information about the secret key.

7 Conclusions

We presented a method for computing exact digraph probabilities for RC4/n under reasonable assumptions, used this method to compute the exact distributions for small n, observed consistency in the digraph statistics across all values of n, and presented a simple method to extrapolate our knowledge to higher values of n. The minimum amount of RC4 output needed to distinguish that cipher from randomness was derived using information theoretic bounds, and this method was used to compare the effectiveness of our attack to those in the literature. Our methods provide the best known way to distinguish RC4/8, requiring only $2^{30.6}$ bytes of output.

While we cannot extend either attack to find the original key or the entire internal state of the cipher, further research may be able to extend these observations into an attack that is more efficient than exhaustive search.

Appendix: Information Theoretic Bounds on Distinguishing RC4 from Randomness

Information theory provides a lower bound on the number of outputs that are needed to distinguish RC4 output from a truly random sequence. We derive this bound for the case that with false positive and false negative rates of 10%, for our results and for those in the literature.

Following [1], we define the discrimination $L(p, q)$ between two probability distributions p and q as $L(p, q) = \sum_s p(s) \lg \frac{p(s)}{q(s)}$, where the sum is over all of the states in the distributions. The discrimination is the expected value of the log-likelihood ratio (with respect to the distribution p), and can be used to provide bounds on the effectiveness of hypothesis testing. A useful fact about discrimination is that in the case that l independent observations are made from the same set of states, the total discrimination is equal to l times the discrimination of a single observation.

We consider a test $\mathcal{T}$ that predicts (with some likelihood of success) whether or not a particular input string of l symbols, each of which is in $\mathbb{Z}/2^n$, was generated by n-bit RC4 or by a truly random process. If the input string was generated by RC4, the test $\mathcal{T}$ returns a 'yes' with probability $1 - \beta$. If the input string was generated by a truly random process, then $\mathcal{T}$ returns 'no' with probability $1 - \alpha$. In other words, α is the false positive rate, and β is the false negative rate. These rates can be related to the discrimination between

the probability distribution p_r generated by a truly random process and the distribution p_{RC4} generated by n-bit RC4 (with a randomly selected key), where the distributions are over all possible input strings. From [1], the discrimination is related to α and β by the inequality

$$L(p_r, p_{RC4}) \geq \beta \lg \frac{\beta}{1-\alpha} + (1-\beta) \lg \frac{1-\beta}{\alpha}. \quad (1)$$

Equality can be met by using an information-theoretic optimal test, such as a Neyman-Pearson test [1]. We expect our cryptanalyst to use such a test, and we regard Equation 1 as an equality, though the implementation of such tests are outside the scope of this paper.

Applying this result to use the RC4 digraph distribution ρ from the uniform random distribution ϕ,

$$L(\phi, \rho) = l \sum_{d \in \mathcal{D}} 2^{-2n} \lg \frac{1}{2^{2n}\rho(d)} = \beta \lg \frac{\beta}{1-\alpha} + (1-\beta) \lg \frac{1-\beta}{\alpha}, \quad (2)$$

where $\mathcal{D}$ is the set of digraphs, and $\rho(d)$ is the probability of digraph d with respect to the distribution ρ. Solving this equation for l, we get the number of RC4 outputs needed to distinguish that cipher.

To distinguish RC4 from randomness in the case that we only know the probabilities of the positive and negative events defined in Section 3.1, we consider only the states N, P and Q, where N is the occurrance of negative event, P is the occurrance of a positive event, and Q is the occurrance of any digraph that is neither a positive nor negative event. Then the discrimination is given by Equation 1, where the sum is over these three states. Solving this equation for the number l of outputs ,with $\alpha = \beta = 0.1$ and the data from Table 4 gives $2^{30.6}$.

The linear model of RC4 derived by Golić demonstrates a bias in RC4/8 with correlation coefficient 3.05×10^{-7} [3,2]. In other words, an event that occurs after each symbol output with probability $0.5 + 1.52 \cdot 10^{-7}$ in a keystream generated by RC4, and with probability 0.5 in a keystream generated by a truly random source. Using Equation 1 with $\alpha = \beta = 0.1$, we find that at least $2^{44.7}$ bytes are required.

References

1. Blahut, R., "Principles and Practice of Information Theory", Addison-Wesley, 1983.
2. Golić, J., "Linear Models for a Time-Variant Permutation Generator", IEEE Transactions on Information Theory, vol. 45, No. 7, pp. 2374-2382, Nov. 1999
3. Golić, J., "Linear Statistical Weakness of Alleged RC4 Keystream Generator", Proceedings of EUROCRYPT '97, Springer-Verlag.
4. Knudsen, L., Meier, W., Preneel, B., Rijmen, V., and Verdoolaege, S., "Analysis Methods for (Alleged) RC4", Proceedings of ASIACRYPT '99, Springer-Verlag.

5. Mister, S. and Tavares, S., “Cryptanalysis of RC4-like Ciphers”, in the Workshop Record of the Workshop on Selected Areas in Cryptography (SAC ’98), Aug. 17-18, 1998, pp. 136-148.
6. Rivest, R., ”The RC4 encryption algorithm”, RSA Data Security, Inc, Mar. 1992
7. RSA Laboratories FAQ, Question 3.6.3, http://www.rsasecurity.com/rsalabs/faq/3-6-3.html.
8. Schneier, B., “Applied Cryptography”, New York: Wiley, 1996.