A Practical Attack on Broadcast RC4

Itsik Mantin and Adi Shamir

Computer Science Department, The Weizmann Institute, Rehovot 76100, Israel.
{itsik,shamir}@wisdom.weizmann.ac.il

Abstract. RC4 is the most widely deployed stream cipher in software applications. In this paper we describe a major statistical weakness in RC4, which makes it trivial to distinguish between short outputs of RC4 and random strings by analyzing their second bytes. This weakness can be used to mount a practical ciphertext-only attack on RC4 in some broadcast applications, in which the same plaintext is sent to multiple recipients under different keys.

1 Introduction

A large number of stream ciphers were proposed and implemented over the last twenty years. Most of these ciphers were based on various combinations of linear feedback shift registers, which were easy to implement in hardware, but relatively slow in software. In 1987 Ron Rivest designed the RC4 stream cipher, which was based on a different and more software friendly paradigm. It was integrated into Microsoft Windows, Lotus Notes, Apple AOCE, Oracle Secure SQL, and many other applications, and has thus become the most widely used software-based stream cipher. In addition, it was chosen to be part of the Cellular Digital Packet Data specification. Its design was kept a trade secret until 1994, when someone anonymously posted its source code to the Cypherpunks mailing list. The correctness of this unofficial description was confirmed by comparing its outputs to those produced by licensed implementations.

RC4 has a secret internal state which is a permutation of all the $N = 2^n$ possible n bits words. The initial state is derived from a variable size key by a key scheduling algorithm, and then RC4 alternately modifies the state (by exchanging two out of the N values) and produces an output (by picking one of the N values).

In practical applications n is typically chosen as 8, and thus RC4 has a huge state of $log_2(2^8!) \approx 1684$ bits. It is thus impractical to guess even a small part of this state, or to use standard time/memory/data tradeoff attacks. In addition, the state evolves in a complex non-linear way, and thus it is difficult to combine partial information about states which are far away in time. Consequently, all the techniques developed to attack stream ciphers based on linear feedback shift registers seem to be inapplicable to RC4.

Since RC4 is such a widely used stream cipher, it had attracted considerable attention in the research community, but so far no one had found an attack on

M. Matsui (Ed.): FSE 2001, LNCS 2355, pp. 152–164, 2002.
© Springer-Verlag Berlin Heidelberg 2002

RC4 which is even close to being practical: For $n = 8$ and sufficiently long keys, the best known attack requires more than 2^{700} time to find its initial state.

Our main result in this paper is the discovery of a trivial distinguisher between RC4 and random ciphers, which needs only two output words under several hundred unknown and unrelated keys to make a reliable decision (the best previously published distinguisher requires more than a billion output words). This is surprising, since the algorithm had been analyzed internally for more than 13 years and externally for more than 6 years, and this obviously nonrandom behavior should have been discovered long ago. Since RC4 has such a unique "fingerprint", it is easy to recognize its use in given black box encryption devices. In addition, the nonrandom behavior of RC4 makes it possible to deduce partial information about the plaintext by analyzing a small number of ciphertexts produced from the same plaintext under unrelated and arbitrarily long keys. This provides a practical ciphertext-only attack on RC4 in many applications which allow data broadcasting, such as groupware and email.

The paper is organized in the following way: In section 2 we describe RC4 and survey previous results about its security. In section 3 we show that some RC4 outputs are highly nonuniform, and analyze the number of outputs required to turn this nonuniformity into a reliable distinguisher. In addition, we show how to apply a practical ciphertext-only attack on RC4 in broadcast applications. In section 4 we generalize our particular observation, and show that such biases are created by *predictive* states, which are related to (but more general than) the fortuitous states introduced in [FM00]. We end the section with several examples of such states, along with a theoretical analysis of the size of biases they can generate and the complexity of the distinguishers they give rise to.

2 RC4 and Its Security

2.1 Description of RC4

The RC4 stream cipher is still considered a trade secret, and the following unofficial description is taken from Schneier's book [Sch96]. It consists of a key scheduling part (which turns a random key whose typical size is 40-256 bits into an initial permutation S of $\{0, \ldots, N-1\}$, and is ignored in this paper) and an output generation part which is described in figure 1. It 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 pointed to by i and j, and output the value of S pointed to by $S[i] + S[j]$[1]. Note that every entry of S is swapped at least once (possibly with itself) within any N consecutive rounds, and thus the permutation S evolves fairly rapidly during the output generation process.

[1] Here and in the rest of the paper all the additions are carried out modulu N

Initialization:
$i = 0$
$j = 0$
Generation loop:
$i = i + 1$
$j = j + S[i]$
Swap$(S[i], S[j])$
Output $z = S[S[i] + S[j]]$

Fig. 1. The Output Generation Algorithm

2.2 Previous Attacks on RC4

Interesting properties of RC4 were described in several papers. Finney specified in [Fin94] a class of states that RC4 can never enter. This class contains all the states for which $j = i + 1$ and $S[j] = 1$ (a fraction of approximately N^{-2} of the RC4 states are in this class, which is closed under RC4 round operation). These states are connected by short cycles of length $N(N - 1)$. Since the state transition in RC4 is invertible and the initial state ($i = j = 0$) is not of this type, RC4 can never enter these states for any key. Additional properties of the state transition graph of RC4 were analyzed by Mister and Tavares in [MT98].

Jenkins identified in [Jen] a probabilistic correlation between the secret information (S, j) and the public information (i, z) (in known message attacks), in which the events $S[j] = i - z$ and $j - S[i] = z$ occur with approximately twice their expected probability.

Andrew Roos noted in [Roo95] that for keys for which $K[0] + K[1] = 0$, the first output is equal to $K[2] + 3$ with probability $2^{-2.85}$. The cryptanalyst can use this fact to deduce two bytes of information about the key ($K[0] + K[1]$ and $K[2]$) with probability $2^{-10.85}$ instead of the trivial 2^{-16}, and thus reduce the effective key length in exhaustive search by about five bits.

Grosul and Wallach show in [GW00] that for large keys whose size is close to 2^n bytes, RC4 is vulnerable to a related key attack.

A branch and bound attack that is based on the "Guess on Demand" paradigm is analyzed in [MT98] and [KMP+98]. The attack simulates the generation process, and keeps track of all the known values in S which had been deduced so far. Whenever an unknown entry in S is needed in order to continue the simulation, the attacker tries all the possible values (their number is typically smaller than N since known values in the permutation S cannot be repeated). Actual outputs are used by the simulation to either deduce additional values in S (if the pointed value is unknown) or to backtrack (if the pointed value is known and different from the actual output). This tree search is simple to implement, and needs only $O(N)$ outputs. However, its enormous running time (which was analyzed both analytically and experimentally in [KMP+98]) makes

it worse than exhaustive search for typical key sizes, and completely impractical for any value of n above 5.

A different research direction was to analyze the statistical properties of RC4 outputs, and in particular to construct distinguishers between RC4 and truly random bit generators. Golić described in [Gol97] a linear statistical weakness of RC4, caused by a positive correlation between the second binary derivative of the lsb and 1. This weakness implies that RC4 outputs of length $2^{6n-7.8}$ ($2^{40.2}$ for the typical $n = 8$) can be reliably distinguished from random strings. This result was subsequently improved by Fluhrer and McGrew in [FM00]. They analyzed the distribution of triplets consisting of the two outputs produced at times t and $t + 1$ and the known value of $i \equiv t(modN)$, and found small biases in the distribution of $(7N - 8)$ of these N^3 triplets: some of these probabilities were $2^{-3n}(1 + 2^{-n})$ (with a small positive bias), and some of these probabilities were $2^{-3n}(1 - 2^{-n})$ (with a small negative bias). They used information theoretic methods to prove that these biases can be used to distinguish between $RC4$ and a truly random source by analyzing sequences of $2^{30.6}$ output words. They also identified a special class of states, which they denoted as fortuitous states, that are the source of most of these biases. These states can be used to extract part of the internal state with non-trivial probability, but since RC4 states are huge this does not lead to practical attacks on RC4 for $n > 5$.

3 The New Attack

Our main observation is that the second output word of RC4 has a very strong bias: It takes on the value 0 with twice the expected probability (1/128 instead of 1/256 for $n = 8$). Other values of the second output and all the values of other outputs have almost uniform distributions.

3.1 Distinguishing between Distinguishers

One interesting question is why hasn't this strong bias been discovered long ago? RC4 was subjected to rigorous statistical tests, but they typically consisted of taking a single stream with billions of output words, and counting the number of times each value occurred along it. Even if this experiment is repeated many times with different keys, the strong bias of the second output word is not going to be visible in such an experiment.

The difference between these types of statistical experiments is not new. Goldreich described in [Gol99] two different notions of computational indistinguishability, *indistinguishability in polynomial time* and *indistinguishability in polynomial sampling*. The first definition deals with a probabilistic polynomial time machine (called a *weak distinguisher*) which is given a single output stream produced by one of two possible sources, whereas the second definition deals with a probabilistic polynomial time machine (called a *strong distinguisher*) which is given as a black box one of the two sources, and can reset and rerun it with

new random keys polynomially many times. These notions of indistinguishability are theoretically equivalent in the sense that if one of them can succeed with probability $\frac{1}{2} + \frac{1}{p(\cdot)}$ for some polynomial p, then the other can also succeed with probability $\frac{1}{2} + \frac{1}{q(\cdot)}$ for some other polynomial q. However, in practice there can be a large difference between the two biases, and in particular one of them can be a large constant whereas the other is a tiny polynomial fraction. This phenomenon is wonderfully demonstrated in the case of RC4, since the analysis of a single output stream makes the bias almost unnoticeable, whereas the analysis of a single output word across many executions amplifies the bias and makes it almost unmissable.

3.2 The Biased Second Output of RC4

Theorem 1 *Assume that the initial permutation S is randomly chosen from the set of all the possible permutations of $\{0, \ldots, N-1\}$. Then the probability that the second output word of RC4 is 0 is approximately $2/N$.*

Proof: Denote the permutation S after it has been updated in round t by S_t (S_0 is the initial permutation) and the output of this round as z_t. We show that when $S_0[2] = 0$ and $S_0[1] \neq 2$, the second output is 0 with probability 1 (see figure 2).

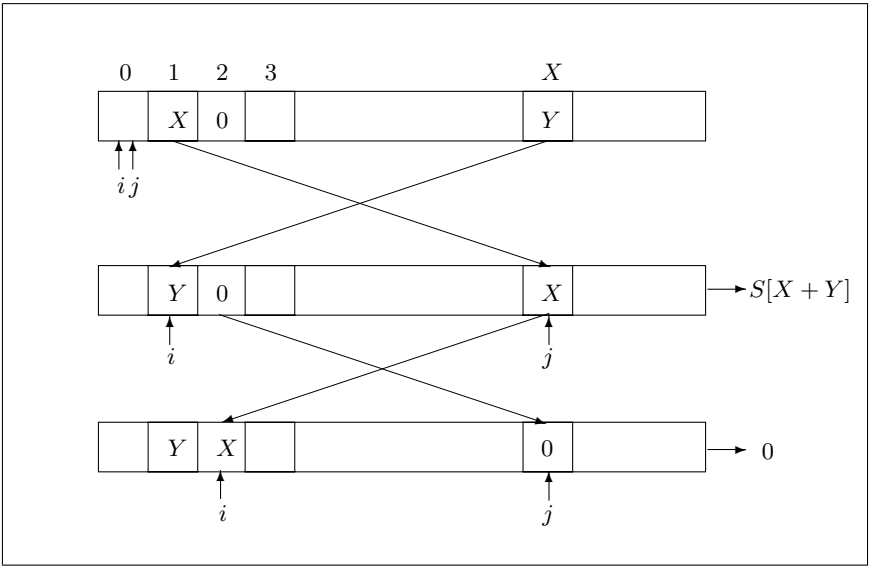

Fig. 2. The first 2 rounds of RC4 when $S_0[2] = 0$ and $S_0[1] \neq 2$

Initially, i and j are 0. If we denote $S_0[1]$ by X, then during the first round i is updated to 1 and j is updated to $0 + S_0[1] = X$, and the contents X and

Y of locations 1 and X are exchanged. The first output is $S_1[X+Y]$, which can be any value with essentially uniform probability distribution. We now use the assumption that $S_0[2]=0$. During the second round, i is incremented to 2, and j is incremented to $X+0=X$, and thus we exchange the values 0 and X of locations 2 and X, and output $S_2[X+0]=S_2[X]=0$ (we have to assume that $S_0[1]\neq 2$ since otherwise 0 is swapped out before it is used as an output). This lucky sequence of events proves that for about $1/N$ of the keys, the second output is 0 with probability 1, whereas for the other $1-1/N$ of the keys, the second output is 0 with probability $1/N$ (since it is uniformly distributed). As a result, the total probability that the second output is 0 is

$$P[z_2=0]=P[z_2=0|S_0[2]=0]\cdot P[S_0[2]=0]+P[z_2=0|S_0[2]\neq 0]\cdot P[S_0[2]\neq 0]$$

$$\approx 1\cdot 1/N+(1-1/N)\cdot 1/N=1/N\cdot(1+1-1/N)\approx 2/N$$

which is twice its expected probability. □

One could expect to see a similar (but weaker) bias towards 0 at all the other outputs z_t with $t=0 \mod n$), since in $1/N^2$ of these cases $S_t[2]=0$ and $j=0$, which would give rise to the same situation. However, extensive experiments have shown that this weaker bias at later rounds does not exist. By carefully analyzing this situation one can show that for any $j\neq 0$ the output is zero with a slight *negative* bias, and the total contribution of these negative biases exactly cancels the positive bias derived from $j=0$. The only time we don't have this cancellation effect is at the beginning of the execution, when j always starts as 0 rather than as a uniformly distributed random value.

An interesting observation based on this bias, is that by applying Bayes rule to the equation $P[z_2=0|S_0[2]=0]\approx 2/N$, we get

$$P[S_0[2]=0|z_2=0]=\frac{P[S_0[2]=0]}{P[z_2=0]}\cdot P[z_2=0|S_0[2]=0]\approx\frac{1/N}{2/N}\cdot 1=\frac{1}{2}$$

Consequently, whenever the second output byte is 0 we can extract an entry of S with probability $1/2$, which significantly exceeds the trivial probability of $1/N$. This fact can be used to accelerate most of the known attacks by a factor of $N/2$, but this improvement does not suffice to mount a practical attack on $\mathrm{RC4}_{n>5}$.

3.3 Cryptanalytic Applications

The strong bias described in section 3.2 has several practical cryptanalytic applications.

Distinguishing RC4 from Random Sources. The best distinguisher mentioned in the literature ([FM00]), distinguishes *RC4* from a random source by analyzing $2^{30.6}$ output words. This distinguisher is *weak* (since it analyzes a single output stream), and is based on $7N-8$ independent biases of events that

happens with probability of $2^{-3n}(1 \pm 2^{-n})$ instead of the expected 2^{-3n}. Our new observation can be used to construct a *strong* distinguisher for RC4 which requires only $O(N)$ output words.

Theorem 2 *Let X, Y be distributions, and suppose that the event e happens in X with probability p and in Y with probability $p(1+q)$. Then for small p and q, $O(\frac{1}{pq^2})$ samples suffice to distinguish X from Y with a constant probability of success.*

Proof: Let X_e, Y_e be the random variables specifying the number of occurrences of e in t samples. Then X_e and Y_e have binomial distributions with parameters (t, p) and $(t, p(1+q))$, and their expectations, variances and standard deviations are:

$$E[X_e] = tp, E[Y_e] = tp(1+q)$$

$$V(X_e) = tp(1-p) \approx tp, V(Y_e) = tp(1+q)(1-p(1+q)) \approx tp(1+q)$$

$$\sigma(X_e) = \sqrt{V(X_e)} \approx \sqrt{tp}, \sigma(Y_e) = \sqrt{V(Y_e)} \approx \sqrt{tp(1+q)} \approx \sqrt{tp}$$

We'll analyze the size of t that implies a difference of at least one standard deviation between the expectations of the two distributions:

$$E[Y_e] - E[X_e] \geq \sigma(X_e) \Leftrightarrow tp(1+q) - tp \geq \sqrt{tp} \Leftrightarrow tpq \geq \sqrt{tp} \Leftrightarrow t \geq \frac{1}{pq^2}$$

Consequently, $O(\frac{1}{pq^2})$ samples (The constant depends on the desired success probability) suffice for the distinguishing. □

Let X be the probability distribution of the second output in uniformly distributed streams, and let Y be the probability distribution of the second output in streams produced by RC4 for randomly chosen keys. The event e denotes an output value of 0, which happens with probability of $1/N$ in X and $2/N$ in Y. By using the previous theorem with $p = 1/N$ and $q = 1$, we can conclude that we need about $\frac{1}{pq^2} = N$ outputs to reliably distinguish the two distributions. To find the exact number, we carried out actual experiments, and found out that by analyzing just 200 streams we can correctly distinguish between RC4 outputs and random streams with probability of success which exceeds 0.64.

A Ciphertext-Only Attack on Broadcast RC4. A classical problem in distributed computing is to allow N Byzantine generals to coordinate their actions when up to one third of them can be traitors. The problem is solved by a multiround protocol in which each general broadcasts the same message (which initially consists of either "Attack" or "Retreat") to all the other generals, where each copy is encrypted under a different key agreed in advance between any two

generals. By using RC4, the generals will succeed in reaching a coordinated decision, but will probably fail in implementing it, since an enemy that collects all the ciphertexts can easily deduce which one of the two possible messages was broadcast by each general, regardless of the length of their keys. More formally, we show:

Theorem 3 *Let M be a plaintext, and let $C_1, C_2, \ldots, C_k$ be the RC4 encryptions of M under k uniformly distributed keys. Then if $k = \Omega(N)$, the second byte of M can be reliably extracted from $C_1, C_2, \ldots, C_k$.*

Proof: For every encryption key, $M[2]$ has probability $\frac{2}{N}$ to be XORed with 0, and probability $\frac{1}{N}$ to be XORed with each of the other possible bytes. Thus, a fraction of $\frac{2}{N}$ of the second ciphertext bytes are expected to have the same value as the second plaintext byte, and thus the most frequent character in $C_1[2], \ldots, C_k[2]$ is likely to be $M[2]$ itself.

□

The Byzantine empire no longer exists (did they use RC4?), but there are many other broadcasting protocols which are used today in a variety of applications. For example, many users send the same email message to multiple recipients (encrypted under different keys), and many groupware applications enable multiple users to synchronize their documents by broadcasting encrypted modification lists to all the other group members. All these applications are vulnerable to this attack.

4 Analysis

In this section we introduce the notion of *predictive* states, and show that it generalizes both our observation and the notion of fortuitous states which were defined in [FM00]. Each predictive state can give rise to some biased outputs, and we analyze the size of such biases as a function of certain parameters of these states. Finally, we show that such states can be used to improve the time and data complexities of branch and bound attacks on the internal state of RC4.

4.1 *a*-States and *b*-Predictiveness

Definition 1 *An* a-state *is a partially specified RC4 state, that includes i, j, and a (not necessarily consecutive) elements of S.*

Definition 2 *Let A be an a-state for some a. Suppose that all*[2] *the RC4 states that are compatible with A produce the same output word after r rounds. Then A is said to* predict *its rth output.*

[2] We can generalize the definition without affecting the analysis by allowing some exceptions due to rare coincidences.

Definition 3 *Let A be an a-state, and suppose that for some $r_1, \ldots, r_b \leq 2N$, A predicts the outputs of rounds $r_1, \ldots, r_b$. Then A is said to be* b-predictive[3]

Intuitively, a b-predictive a-state can be associated with a sequence of b specific (round, output) pairs. Many experiments (and strong intuition) indicate that an a-state can be b- predictive only when $a \geq b$, but formalizing a proof for this conjecture seems to be non-trivial.

4.2 Distinguishers Based on Predictive States

In general, any a-state that is b-predictive can induce some bias in the output distribution. Consider the events E_A (that the current state is compatible with the a-state A), and E_B (that the outputs in rounds $r_1, \ldots, r_b$ are those predicted by A). E_A includes $a+2$ constraints (i, j and a elements of S) and thus has a probability of $\frac{1}{N^2}\frac{N!}{(N-a)!} \approx N^{-(a+2)}$. Whenever E_A occurs, E_B occurs with probability 1, and whenever E_A does not occur, E_B typically happens with the trivial probability of $N^{-(b+1)}$ (based on the choice of i and the b outputs). Thus the probability of E_B is computed as follows

$$P[E_B] = P[E_B|E_A] \cdot P[E_A] + P[E_B|\neg E_A] \cdot P[\neg E_A] \approx$$

$$\approx 1 \cdot N^{-(a+2)} + N^{-(b+1)} \cdot (1 - N^{-(a+2)}) =$$

$$= N^{-(b+1)}(1 - N^{-(a+2)} + N^{b+1-(a+2)}) \approx N^{-(b+1)}(1 + N^{b-a-1})$$

Applying the result of Theorem 2 to these probabilities yields the following corollary

Corollary 1 *An a-state that is b-predictive implies the existence of a distinguisher for RC4 which requires $O(N^{2a-b+3})$ output words.*

Proof: In terms of Theorem 2, $p = N^{-(b+1)}$ and $q = N^{b-a-1}$. Thus, the number of output words required to reliably distinguish RC4 from random sources is $O(\frac{1}{pq^2}) = O(N^{2a-b+3})$ □

Our major observation in this paper was caused by a 1-predictive 1-state, $i_A = 0, j_A = 0, S_A[2] = 0$ which predicted an output of zero in the second round. Since at the beginning of the generation process i (which is part of the definition of p) and j (which is part of the definition of q) are fixed, we actually get a better distinguisher than the one implied by the general corollary: both p and q (from Theorem 2) increase by a factor of N, and thus the distinguisher needs only $O(N)$ output words.

[3] without the bound on the r_i's, this definition might include degenerate cases since the output of RC4 eventually cycles.

4.3 An Attack Based on Predictive States

Assuming the uniformity of RC4 internal state and outputs (their slight biases are negligible in the current context), a b-predictive a-state can be used to extract some partial information on the internal state with non-trivial probability. By applying Bayes Rule to the probabilities of the events E_A and E_B from section 4.2) we can calculate

$$P[E_A|E_B] = \frac{P[E_A]}{P[E_B]} P[E_B|E_A] \approx \frac{N^{-(a+2)}}{N^{-(b+1)}} \cdot 1 = N^{b-a-1}$$

Since E_A is internal and E_B is external, we can use the external event E_B as an indicator for the internal event E_A with a success probability of N^{b-a-1}. In other words, if we see N^{a+1-b} occurrences of E_B, one of them is likely to be a real occurrence of E_A. Having j and a values of S, we can apply the branch and bound attack from [KMP+98] to reveal the rest of the secret state of RC4.

Intuitively, a determines the quantity of the revealed information, but increasing a reduces the probability for E_A and increases the required number of outputs. Consequently, the attack is limited to small values of a. The value of $b - a$ determines the number of false candidates that have to be examined, and reducing it is essential to reducing the time complexity. Under the conjecture that $b \leq a$, the best states for this attack are a-predictive a-states. In order to reduce the required number of outputs, the attack can use a database of many a-predictive a-states sorted by outputs, and lookup all the output sequences in it.

The attack requires an efficient way to find in a preprocessing stage all the a-predictive a-states of RC4 for small values of a. This is a non-trivial problem whose complexity is not well understood at the moment. A subset of predictive states that is easier to enumerate is the class of fortuitous states mentioned in [FM00], which includes all the a-predictive a-states in which the predicted outputs appear as a contiguous prefix of the output sequence (note that our 1-predictive 1-state with its strong bias is not a fortuitous state, since we can predict the second output but not the first one). In fortuitous states, all the j's and $(S[i]+S[j])$'s in the first a rounds must be inside an interval of a consecutive known values (otherwise, these outputs can't be specified) and thus we can find these states via an exhaustive search on all the possible values of short segments in S.

Fluhrer and McGrew correctly stated that such an attack (see figure 3), based on fortuitous states, is impractical for the full version of RC4 with $n = 8$. However, they ignored the fact that it can lead to significant improvements in attacks on versions of RC4 with smaller values of n.

There are two possible benchmarks for the efficiency of such an attack: the time complexity and the number of required outputs. Denote the number of fortuitous states of order a by $F(a, N)$, and the probability that a (specific) fortuitous state happens as $pf(a, N) \approx N^{-(a+2)}$. The attack requires one fortuitous state to happen and thus the number of outputs needed for this attack is $\frac{1}{F(a,N)pf(a,N)}$. The time complexity of this attack is N (false candidates) times

```
FortAttack(Z^a, FDB^b)
    For each a-tuple ⟨x_1, ... , x_a⟩ of Z
        FList = Lookup(⟨x_1, ... , x_a⟩, FDB)
        If FList ≠ NULL
            For each state f ∈ FList
                Let S̄ be the partially guessed permutation according to f
                Let Z̄ be the next N words of Z
                S̄ = Complete^c(Z̄, S̄)
                If S̄ ≠ NULL
                    Return S̄
```

[a] the output stream
[b] table of fortuitous states
[c] Branch and Bound attack with apriori information

Fig. 3. A Fortuitous State Attack

the recursive function *complex* (defined in [KMP+98]), that describes the time complexity of the Branch and Bound attack with apriori information about a elements in S.

For small values of a the revealed part of S is insignificant, whereas for large values of a the output size increases dramatically. However, for RC4 with $n = 5$ this attack is feasible, and for other values of n it leads to considerable improvements in the time complexities of the best known attacks (see table 1)

		S									
i	j	-2	-1	0	1	2	3	$S[i]$	$S[j]$	$S[i]+S[j]$	$Output$
-3	-1	1	2	$*$	-1	$*$	$*$	/	/	/	/
-2	0	$*$	2	1	-1	$*$	$*$	$*$	1	$*$	$*$
-1	2	$*$	$*$	1	-1	2	$*$	$*$	2	$*$	$*$
0	3	$*$	$*$	$*$	-1	2	1	$*$	1	$*$	$*$
1	2	$*$	$*$	$*$	2	-1	1	2	-1	1	2
2	1	$*$	$*$	$*$	-1	2	1	2	-1	1	-1
3	2	$*$	$*$	$*$	-1	1	2	2	1	3	2

Fig. 4. A non-fortuitous 3-predictive 3-state

This attack can be further improved by finding predictive rather than fortuitous states. Every fortuitous state is predictive but the converse is clearly false, and thus we can get a better attack. However, we do not know how to estimate the approximate number of predictive states of various orders, and thus we cannot quantify the expected improvement. Small computational experiments had

Table 1. Performance of the fortuitous state attack (app. stands for approximation)

N	a	$F(a,N)$	$pf(a,N)$	Data	fp ratio	Time	Complex(0)
32	5	$2^{12.2}$	2^{-35}	$2^{22.8}$	$2^{4.5}$	$2^{41.7}$	2^{53}
	6	$2^{14.8}$	2^{-40}	$2^{25.2}$	$2^{4.3}$	$2^{38.6}$	
	7	2^{18}	2^{-45}	2^{27}	$2^{4.0}$	$2^{35.4}$	
	8	$2^{21.5}$	2^{-50}	$2^{28.5}$	$2^{3.6}$	$2^{32.3}$	
	9	(app.) $2^{24.5}$	2^{-55}	$2^{30.5}$	$2^{3.2}$	$2^{29.2}$	
64	5	$2^{12.9}$	2^{-42}	$2^{29.1}$	$2^{5.8}$	$2^{118.6}$	2^{132}
	6	$2^{15.2}$	2^{-48}	$2^{32.8}$	$2^{5.65}$	$2^{114.65}$	
	7	(app.)2^{17}	2^{-54}	2^{37}	$2^{5.5}$	$2^{110.7}$	
	8	(app.)2^{21}	2^{-60}	2^{39}	$2^{5.35}$	$2^{105.85}$	
	9	(app.)2^{24}	2^{-66}	2^{42}	$2^{5.2}$	2^{103}	
	10	(app.)2^{27}	2^{-72}	2^{45}	2^{5}	$2^{99.2}$	
128	4	$2^{11.7}$	2^{-42}	$2^{30.3}$	2^{7}	$2^{312.8}$	2^{324}
	8	(app.) 2^{24}	2^{-70}	2^{46}	2^{7}	$2^{287.5}$	
256	6	(app.) $2^{16.6}$	2^{-64}	$2^{47.4}$	2^{8}	$2^{755.2}$	2^{779}

yielded several interesting predictive states which are not fortuitous , such as the 3-state 3-predictive state described in figure 4, in which the predicted values appear surprisingly late in the generated sequence (at steps 4, 5, and 6).

		S									
i	j	1	2	3	4	5	6	7	8	9	10
0	-1	3	$*$	2	-1	$*$	$*$	$*$	$*$	$*$	$*$
1	2	$*$	3	2	-1	$*$	$*$	$*$	$*$	$*$	$*$
2	5	$*$	$*$	2	-1	3	$*$	$*$	$*$	$*$	$*$
3	7	$*$	$*$	$*$	-1	3	$*$	2	$*$	$*$	$*$
4	6	$*$	$*$	$*$	$*$	3	-1	2	$*$	$*$	$*$
5	9	$*$	$*$	$*$	$*$	$*$	-1	2	$*$	3	$*$
6	8	$*$	$*$	$*$	$*$	$*$	$*$	2	-1	3	$*$
7	10	$*$	$*$	$*$	$*$	$*$	$*$	$*$	-1	3	2
8	9	$*$	$*$	$*$	$*$	$*$	$*$	$*$	3	-1	2
9	8	$*$	$*$	$*$	$*$	$*$	$*$	$*$	-1	3	2
10	10	$*$	$*$	$*$	$*$	$*$	$*$	$*$	-1	3	2

Fig. 5. A 3-state that determines j for 10 rounds

Intuitively, a state that does not determine the values of j during the next r rounds, cannot predict the output word of the last of these rounds. In case j is "lost", it seems that it cannot be recovered because of the dependency of j on its former value. Thus, a predictive state must determine the values of j from its occurrence up to the last predicted round. A possible weakening of the notion

of predictive states is thus to require only that a prefix of j values should be uniquely determined by the a-state, without insisting on any predicted outputs. Despite the loss of our ability to filter false candidates, we earn the ability to track j values for d rounds conditioned by a constraints, where in some instances $d >> a$ (e.g., see figure 5).

5 Discussion

In sections 3 we described a significant statistical weakness of RC4, which had some interesting cryptanalytic consequences. However, we would like to stress that RC4 should not be considered as being completely broken, since our attack does not recover the key, and in many applications the ability to predict a single plaintext byte is of little importance. However, we believe that as a result of our observation, all future implementations of RC4 should skip at least the first two (and preferably the first N) output words.

References

[Fin94] H. Finney. an RC4 cycle that can't happen. September 1994.

[FM00] Fluhrer and McGrew. Statistical analysis of the alleged RC4 keystream generator. In *FSE: Fast Software Encryption*, 2000.

[Gol97] Golić. Linear statistical weakness of alleged RC4 keystream generator. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1997.

[Gol99] O. Goldreich. *Foundations of Cryptogtaphy.* 1 edition, 1999.

[GW00] A. L. Grosul and D. S. Wallach. a related-key cryptanalysis of RC4. June 2000.

[Jen] Robert J. Jenkins. ISAAC and RC4. Published on the internet at http://burtleburtle.net/bob/rand/isaac.html.

[KMP+98] Knudsen, Meier, Preneel, Rijmen, and Verdoolaege. Analysis methods for (alleged) RC4. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology.* LNCS, Springer-Verlag, 1998.

[MT98] Mister and Tavares. Cryptanalysis of RC4-like ciphers. In *SAC: Annual International Workshop on Selected Areas in Cryptography.* LNCS, 1998.

[Roo95] A. Roos. A class of weak keys in the RC4 stream cipher. September 1995.

[Sch96] B. Schneier. *Applied Cryptography.* John Wiley & Sons, Inc, Toronto, Canada, 2 edition, 1996.