CryptAnalysis of the alleged $RC4$

Itsik Mantin

September 26, 2000

1 Introduction

The alleged $RC4$ is a stream cipher that is based on a simple pseudo-random generator that generates pseudo random numbers of the domain $\{0, \ldots, 2^n - 1\}$ (in most of the applications $n = 8$). The generator has a state, which includes 2 indices i and j, and a permutation of the numbers $\{0, \ldots, 2^n - 1\}$. The indices are initialized to 0. The permutation is generated from a key of 128 bit. Though, in this paper I investigated the pseudo-random generation of $RC4$ and therefore I ignored the key scheduling and assumed that the initial permutation is uniformly distributed over the space of permutations. The number of bits of the this pseudo-key can be estimated by Stirling formula to be $N(n - \lg e)$, e.g., for $n = 8$, the pseudo-key is of 1600 bits. The round operation is very simple and it includes swapping 2 positions (a counter and a random position) and outputing a value that depends of the values in these indices.

1. $i = i + 1$
2. $j = j + S_i$
3. $S_i \leftrightarrow S_j$
4. Output $S_{S_i + S_j}$

In this document I'll describe:

1. Previous results on the cryptanalysis of $RC4$
2. General results
3. Analysis of some weakened versions of $RC4$
4. My trials
5. My conclusions about the hardness of $RC4$

2 Previous results on the Cryptanalysis of $RC4$

The results are divided into 2 kinds, attacks and distinguishers

2.1 Attacks

Currently there are no practical attacks on $RC4$, even for the case when $n = 6$. The known attacks (detailed in [MT99] and [KMP$^+$98]) are trivial and based on "Branch and Bound" methods. At the beginning i and j are known, and S is unknown. Whenever we need a value,e.g., S_i on the first iteration, we guess it. If we reach a contradiction, we kill the branch (the guess) and try another guess. This method is practical for $n = 4$ and for special cases, for $n = 5$. These cases are when there is a lot of redundancy in a small portion of the output. In this case, the dependency between the guesses increases, and thus, wrong branches converge to contradiction faster than the regular case.

2.2 Distinguishers

The most interesting result on the cryptanalysis of $RC4$, is a Distinguisher of $RC4$ output from a random sequence, described by R. Fluhrer and D. Mcgrew in [FM00]. The Distinguisher is based on bioses in the distribution of the triplets $\langle i, z_i, z_{i+1}\rangle$, i.e., i and 2 output words. They recognized some bioses in the distribution of these triplets and designed a Distinguisher based on these bioses. One example of this distribution is that the for $i = 1$, the probability that the next output words will be 0,0, is $2^{-2n}(1 + 2^{-n+1})$ when in a random sequence, this probability of this event would have been 2^{-2n}. The Distinguisher is based on this gap and others, and is claimed to distinguish $RC4$ output from a real random sequence by analyzing 2^{30} output words.

3 General results

Lemma 1 *Suppose we have a bucket with n balls, each one is numbered* $1, \dots, n$. *Suppose on every trial we pull a ball from the bucket, check its number, and return it to the bucket. Let X be a random variable representing the number of trials we do until every ball was pulled at least once. Then* $E[X] = n \lg n$

Proof: Let the random variables $\{X_i\}_{i=1}^{n}$ represent the number of trials we have to do from the moment we have $i-1$ *distinct* balls until the i-th *new* ball is pulled out. Obviously, $X = \sum_{i=1}^{n} X_i$. I'll analyze the expectation of X_i. X_i is a random variable Geometrically distributed with $p = \frac{n-(i-1)}{n}$. Thus, $E[X_i] = \frac{1}{p} = \frac{n}{n-(i-1)}$. Because of the linearity of the expectation,

$$E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{n}{n-(i-1)} = n \sum_{i=1}^{n} \frac{1}{i} \approx n(C_E + \lg n)$$

for $C_E \approx 0.577$, the Euler constant $\Box$

Lemma 2 *Suppose $RC4$ has a random behavior. Then, in case we know all the j's, we can restore the initial permutation by analyzing approximately Nn output words.*

Proof: Having all the j's (and trivially all the i's), we have all the swapping operations. In this case we have the exact transformation of the initial permutation S_0 into the current permutation S. Since we know all the j's, we also know the sequence of differences between consecutive j's and thus, in every round we expose $S_i = S_0[i']$ for some i'. Assuming randomness of S on every round we expose one value of the initial permutation, perhaps an already known value. From lemma 1 the expected number of trials until we restore the whole permutation is $2^n \sum_{i=1}^{n} \frac{1}{i} \approx N \lg N = nN$ $\Box$

Unfortunately, lemma 2 is usually not enough, because the knowledge of the j's is usually a result of some specific pattern, which can't be assumed to have a random behavior. For example in the short cycles specified in [Fin94] we know all the j's, but this method can't be used. The exposed value is always 1 (j increases by 1 on every round), and thus, we can expose only the initial position of 1. For such cases I'll prove a stronger lemma

Lemma 3 *In case we know all the j's, we can restore the initial permutation by analyzing approximately N^2 output words.*

Proof: As in the previous case, we know all the swapping operations (and so, we have the exact transformation $S_0 \rightarrow S$) and all the S_i's. We will guess 2-3 values, and use these guesses in the following way. Let the guess be $S_l = k$. Then we'll locate the appearances of k in the output sequence. Since we know the transformation $S_0 \rightarrow S$, we know (for every such appearance) the new position (according to our guess) of k in each of these rounds, i.e., we know $S_i + S_j$. Since we know S_i (the difference of the j's) we also have S_j, and thus, we have the value of the corresponding position in S_0. The result of this new knowledge can be one of the following:

Contradiction This indication is inconsistent with a previous indication, i.e., the value in this position or the position of this value was already known and different than our indication. In this case we know the initial guess was wrong and kill the branch

Elaboration The value of this position and the position of this value were unknown. In this case our guess is elaborated

Confirmation The value and the position are already known to be linked together. In this case the only information we gain is that our guess is more probable to be right.

In case of Elaboration, we have another initial value, and we can repeat the process on its appearances, and so on. In N^2 output words, we are expected to have N appearances of each one of our guesses. Unfortunately, the probability that $3N$ pulls of the bucket will cover all the balls is not sufficient, but the probability that $3N$ independent guesses of N values will be consistent, i.e., won't lead to a contradiction, is negligible. Therefore, we can find the right guess by eliminating the wrong guesses. We can expect that the only guess

that will survive, will be the right one, and thus, we can continue with another guesses on the same output data. □
The result of lemma 3 can be strengthen. Experimental results I made showed that $2N$ output words suffice for the algorithm to expose the complete initial permutation.

4 Analysis of weakened versions

4.1 Short Cycles

Finney noted in [Fin94] that the $RC4$ function has a class of short cycles, based on the following observation. The state $i = a, j = a+1, S_{a+1} = 1$ is transformed by $RC4$ into $i = a', j = a'+1, S_{a'+1} = 1$ for $a' = a+1$. This cycle has a size of $N(N-1)$ states. A significant portion of the $RC4$ states space is included in cycles of this class. For every a (2^n possibilities) and for every $2^n - 1$ values of S in the $2^n - 1$ positions other than $a+1$ ($(2^n-1)!$ possibilities) there is such state. Thus, the fraction of these states is

$$\frac{2^n \cdot (2^n - 1)!}{2^n 2^n (2^n)!} = 2^{-2n}$$

$RC4$ has an initial state (with $i = j$) that doesn't have the structure of the states of this class.

I'll analyze the case when the initialization is changed

4.2 Initialize $i = 0, j = 1$

Notice that in this case the indices of the state are of the pattern $i = a, j = a+1$, and if by accident $S_j = 1$, i.e., $S_1 = 1$, we are in a short cycle. This event will happen with probability 2^{-n}. In case we know we are in such a cycle (and it is easy to know because the period of the output will be $N(N-1)$), we know all the j's ($j = i+1$ and i is known) and by lemma 3 we can restore the initial permutation with $\approx 2^{2n}$ output words.

4.3 Output S_j

S_j after the swap operation is exactly S_i before the swapping. Therefore, we know the difference between consecutive values of j and since we have the initialization of j, we have the values of j for all the rounds. By lemma 2 (we can expect the state to behave uniformly), we can restore the initial permutation with Nn output words.

4.4 Output S_i

This case is a bit harder to analyze than the previous one because j can't be exposed as easily as before. Though, it is easy to see that a significant part of

S is known with high probability. Notice that the permutation changes very slowly. On every round, at most 2 values are changed. Thus, every position in the permutation has probability of $\frac{2^n-2}{2^n}$ to stay the same. By knowing the output S_i (after the swap), we can assume that this value (remember i is known) stays the same for at least (roughly estimated) $\frac{2^n}{4}$ rounds. Thus, the Branch and Bound method will work much faster and it can practically attack the case when $n = 8$.

4.5 Initialize i, j uniformly distributed

Despite it seems harder, this version is more vulnerable than the original one because of the short cycles specified in section 4.1. The initialization state will be in a short cycle with probability 2^{-2n} which is not small enough to be negligible, and therefore, this version isn't secure.

4.6 Determine i differently

In this version the change of i is more complicated, e.g., $i = i + S_j$. It seems that more randomness enters the round operation and the security increases, but deeper analysis of this version, shows that this is not the case. When i increases contiguously, all the positions in S are promised to be swapped at least once. There is a case that a value is swapped with itself, but the expected number of positions where such event happens is 1 in 2^n rounds and the probability that this position won't be swapped in none of the rounds (as j) is

$$(1-2^{-n})^{2^n} = ((1+(\frac{1}{-2^n}))^{-2^n})^{-1} \leq e^{-1}$$

not to be j in none of the 2^n rounds. Therefore, the expected number of unchanged positions in 2^n rounds differs in at most e^{-1} from uniform distribution and is small. In the changed version, the expected number of untouched positions is (assuming uniform distribution on S_i and S_j) can be analyzed, using the linearity of the expectation

$$2^n P[\text{No swapping in position } k] = 2^n(1-\frac{2}{2^n})^{2^n} = 2^n((1+\frac{1}{-2^{n-1}})^{-2^n-1})^{-2} \approx 2^n e^{-2} \approx \frac{2^n}{8}$$

which is significant (especially for large n's such as 8).

For the case of $i = i + S_j$, another weakness exists, which turns the cipher to be totally insecure. In the original version of $RC4$, we had a fraction of 2^{-2n} of the states in short cycles. In this version, the fraction of the cycles increases significantly, to be at least 2^{-n}. Furthermore, these cycles can't be avoided by setting some specific values to i and j. I'll analyze one kind of short cycles in this version, that has more significant fraction. The cycles are of the form $(0 \leq k \leq 2^n - 1)$:

$$i = a, j = a + k, S_j = k$$

I'll analyze the new round operation on these states:

Round	i	j	S_j
0	a	$a+k$	k
1	$a+k$	$a+2k$	k
2	$a+2k$	$a+3k$	k
3	$a+3k$	$a+4k$	k
$\vdots$			
2^{n-1}	$a+(2^n-1)k$	$a+2^nk=a$	k
2^n	$a+2^nk=a$	$a+k$	k
$\vdots$			

I'll analyze the fraction of these states in the space of the states. For every value of i (a) and for every value of j ($a+k$) the only restriction is that $S_j = i - j$. Thus, the fraction of these states is

$$\frac{2^n 2^n (2^n-1)!}{2^{2n}(2^n)!} = \frac{2^n}{2^{2n}} = 2^{-n}$$

There is another class of short cycles (almost disjoint of this class) when we replace the parts of i and j. Therefore, the fraction of the short cycles states can be roughly estimated as 2^{1-n}, which is huge. Another important property of these cycles, is that for every i and j, there is a significant fraction of permutations (for which $S_{i-j} = i - j$) that will complete a state in a short cycle and thus it is not trivial to find an initialization that will not be able to be in a short cycle.

4.7 Output $S[S[i]]$ or $S[S[j]]$

For this version I couldn't find an explicit attack. I succeeded to find an interesting property that can be used for a very efficient distinguisher and maybe for an attack. This property holds for both kinds of outputs.

The property is that the event when the output (for some i) is exactly i, is significantly more probable than this event in a random sequence. Notice that whenever $S[i] = i$ (probability 2^{-n}), the output will be $S[S[i]] = S[i] = i$ with probability 1. In case $S[i] = k \neq i$, there is a trivial probability (2^{-n}) that $S[k] = i$. Thus, in this version

$$P[S[S_i] = i] = P[S_i = i]P[S[S_i] = i | S_i = i] + P[S_i \neq i]P[S[S_i] = i | S_i \neq i] =$$

$$= 2^{-n} \cdot 1 + (1 - 2^{-n})2^{-n} = 2^{-n} + (1 - 2^{-n})2^{-n} = 2^{1-n} - 2^{-2n} \approx 2^{1-n}$$

when for $k \neq i$,

$$P[S[S_i] = k] = P[S_i = i]P[S[S_i] = k | S_i = i] + P[S_i \neq i]P[S[S_i] = k | S_i \neq i] =$$

$$= 2^{-n} \cdot 0 + (1 - 2^{-n})2^{-n} = 2^{-n} - 2^{-2n} \approx 2^{-n}$$

Thus, a distinguisher that is based on a counter of the rounds when the output is equal to i, will distinguish $RC4$ output from a random sequence with $c \cdot 2^n$ output words with failure probability that decreases exponentially in c.

A full attack is more problematic but the following process may lead to one. By applying Base rule we get

$$P[S_i = i | S[S_i] = i] = \frac{P[S_i] = i}{P[S[S_i] = i]} P[S[S_i] = i | S_i = i] = \frac{P[S_i] = i}{P[S[S_i] = i]} = \frac{2^{-n}}{2^{1-n} - 2^{-2n}} = \frac{1}{2 - 2^{-n}} \approx \frac{1}{2}$$

Thus, when we see an output that is equal to i, we can guess that $S_i = i$ and be right with probability of $\frac{1}{2}$. After 2^{nk} outputs, we are expected to meet an event when a consecutive sequence of $2k$ output words are of this format. In this case we can assume that all these outputs are a result of fixed points of S, and it will be a good starting point in trying to recover the full permutation in this state.

4.8 Output $S_i + S_j$

This version of $RC4$ seems to be almost as hard as the original one. I found a possible attack on this version. The attack reveals about $2^{\frac{n}{2}} \sqrt{\ln \frac{1}{p}}$ when p is the probability of the attack to be possible. For $p = \frac{1}{2}, n = 8$ the attack reveals $2^4 \sqrt{\ln 2} \approx 13$ values of S. For $p = \frac{1}{400} \approx e^{-6}$ the attack reveals $16\sqrt{\ln 400} \approx 16\sqrt{6} \approx 40$ values of S. For $p = 2^{-30} \approx e^{-20}$ the attack reveals $16\sqrt{20} \approx 71$ values of S. Having such a large portion of the permutation S, it seems feasible to complete S by Branch and Bound process (though it might be still not feasible).

First I'll analyze the output of the crypto system ($S_{i,j}$ is the value of S in position j after round i):

Round	i	j	*Out*
	0	0	/
1	1	$S_{0,1}$	$1 + S_{0,1}$
2	2	$S_{0,1} + S_{1,2}$	$2 + S_{0,1} + S_{1,2}$
3	3	$S_{0,1} + S_{1,2} + S_{2,3}$	$3 + S_{0,1} + S_{1,2} + S_{2,3}$
	$\vdots$		
k	k	$\sum_{l=0}^{k-1} S_{l,l+1}$	$k - 1 + \sum_{l=0}^{k-1} S_{k,k+1}$

I'll analyze the probability that all the values of S are those of S_0, i.e., $\forall l \leq k-1\ S_{l,l+1} = S_{0,l+1}$ as a function of k. Notice that this is exactly the probability that the first k j's won't participate in any of the swapping operation *before* they become j's (after they become j's they do participate in a swapping operation). This will happen if and only if these j's will differ from all the i's and j's that were in the first k rounds. The probability of the l-th j to differ from the previous i's and j's can be estimated as (there are $2l$ bad values and the rest are good)

$$\frac{2^n - 2l}{2^n} = 1 - \frac{2l}{2^n} = (1 + \frac{1}{\frac{-2^n}{2l}}) = ((1 + \frac{1}{\frac{-2^n}{2l}})^{\frac{-2^n}{2l}})^{\frac{1}{\frac{-2^n}{2l}}} \approx e^{-\frac{2l}{2^n}}$$

The probability of all these events to happen (I assume independency) can be estimated as:

$$\prod_{l=0}^{k} e^{-\frac{2l}{2^n}} = e^{\sum_{l=0}^{k} -\frac{2l}{2^n}} = e^{-\frac{2}{2^n}\sum_{l=0}^{k} l} = e^{-\frac{2}{2^n}\frac{k(k+1)}{2}} = e^{-\frac{k(k+1)}{2^n}} \approx e^{-\frac{k^2}{2^n}} = p(k)$$

We got $p(k) = e^{-\frac{k^2}{2^n}}$. I'll analyze k as a function of p

$$e^{-\frac{k^2}{2^n}} = p$$

$$-\frac{k^2}{2^n} = \ln p$$

$$k = p^{-1}(p) = \sqrt{2^n \ln \frac{1}{p}} = 2^{\frac{n}{2}}\sqrt{\ln \frac{1}{p}}$$

Thus,

$$k(p) = p^{-1}(p) = 2^{\frac{n}{2}}\sqrt{\ln \frac{1}{p}}$$

In case this event happens, all the values that we see on the output can be translated into values of the initial permutation. Notice that the difference between consecutive outputs is $1 + S_{0,l}$ for a known l and thus, we can extract the value $S_{0,l}$. Notice that we might have bad luck and after 2,3 rounds there will be a redundancy in the i's j's values. But this method can be applied in the middle of the pseudo-random generation too. In this case we loose the knowledge of j, but we still can analyze the differences between the output words. If we wait $e^{16} \approx 2^{23}$ output words and guess every time that our event happened (with $k(e^{-16}) = 16 \cdot 4 = 64$), we are expected to be right at least once and have a significant portion of the permutation. A Branch and Bound technique can be used to eliminate bad events and to complete the permutation in the good events.

5 Trials

5.1 Analysis of the gaps between appearances of specific values

My assumption was that the gaps between the appearances of some values, will differ from uniform distribution. I expect that some hidden periodicity of $RC4$ will cause such bioses. I tried to find 3 kinds of bioses:

1. The number of gaps of some specific size, e.g., 2^n. The result was negative. For all the gaps sizes, the behavior was as it was a uniform distribution.
2. The maximal (or the average) gap size. I found it hard to analyze the maximal gaps size because of its large variance. As far as I can tell, there is no bios.
3. The values that achieved large gaps. In these trials, I noted an interesting phenomenon. For small n's (4-5) the value 0 usually achieved one/some of the largest gaps. For larger n's, this phenomenon disappeared. The reason might be that this event happens with some significantly biosed probability, that depends exponentially in n and therefore, I had to do some thousands of tests in order to see this bios.
4. The values that achieved large gaps as a function of the initial permutation (this one could have led to an explicit attack). I tried to find a special pattern that causes large gaps and failed.
5. I noted that the gap of size 1, i.e. "double", was achieved significantly more times with 0 (00) than with others. Fluhrer in [FM00] claims that this digraph (for almost every i) is more probable than others, and my results strengthen this phenomenon. Though, it seems that a distinguisher can work in much less than Fluhrer's 2^{30} output words.

The result of this trial was that these gaps might hide some trivial properties that can be useful for the cryptanalyst, but it seems to be very hard to find them.

5.2 Triplets analysis

Fluhrer in [FM00] analyzed the triplets, i, z_i, z_{i+1} and found bioses in their distribution. I decided to look for bioses in the triplets that include j. Although we probably won't be able to use such bioses for distinguishers, it can help understanding the structure of this cipher. The results of this trial were very interesting. I succeeded in finding a significant bios in the distribution of these triplets, and I found the reason for this bios and another bioses that might be very helpful in understanding the distribution of $RC4$ state. Actually, I have a non-trivial probabilistic dependency of the internal state, as a function of the known values, i.e., i and the output word.

Claim 1 *Let k be a non-zero parameter of the domain $0, \ldots, 2^n - 1$. Suppose on some stage of the generation, $S_j = (k-1)S_i$. Let z be the output word of this round. Then $P[i = kz] \approx 2^{1-n}$*

Proof: The proof is based on the property of $RC4$ that I used for the weakened version when the output is $S[S_i]$. Suppose that $S_j = (k-1)S_i$. Assuming uniform distribution on i, S_i, S_j, the probability of this pattern is

$$\begin{aligned}
P[i = kz] &= \\
&= P[kS_i = i]P[i = kz|kS_i = i] + P[kS_i \neq i]P[i = kz|kS_i \neq i] = \\
&= 2^{-n}P[i = kz|kS_i = i] + (1 - 2^{-n})2^{-n} = \\
&= 2^{-n}P[i = kS_{S_i+(k-1)S_i}|kS_i = i] + (1 - 2^{-n})2^{-n} = \\
&= 2^{-n}P[i = kS_{kS_i}|kS_i = i] + (1 - 2^{-n})2^{-n} = \\
&= 2^{-n}P[i = kS_i|kS_i = i] + (1 - 2^{-n})2^{-n} = \\
&= 2^{-n} \cdot 1 + (1 - 2^{-n})2^{-n} = \\
&= 2^{-n}(2 - 2^{-n}) \approx \\
&= 2^{1-n}
\end{aligned}$$

□

Having the claim that $P[i = kz|S_j = (k-1)S_i] \approx 2^{1-n}$

Claim 2 *Let k be a non-zero parameter of the domain $0, \ldots, 2^n - 1$. Let z be the output word of round i. Then $P[S_j = (k-1)S_i|i = kz] \approx 2^{1-n}$*

Proof: I'll use Base rule in order to calculate the probability. I'll assume uniform distribution on i, z, S_i, S_j

$$\begin{aligned}
P[S_j = (k-1)S_i|i = kz] &= \\
&= \frac{P[i = kz|S_j = (k-1)S_i] \cdot P[S_j = (k-1)S_i]}{P[i = kz|S_j \neq (k-1)S_i] \cdot P[S_j \neq (k-1)S_i]} \approx \\
&\approx \frac{2^{1-n} \cdot 2^{-n}}{2^{-n} \cdot (1 - 2^{-n})} = \\
&\approx \frac{2^{1-n}}{(1 - 2^{-n})} \approx \\
&\approx 2^{1-n}
\end{aligned}$$

□

The result of claim 2 is very interesting. It gives an explicit probabilistic connection between the public information (index and output) and the secret information (the internal state). We actually know something about the distribution of the internal state, as a function of the public information.

The use of this fact in designing a complete attack on $RC4$ is not trivial. Though, it is the first analysis of the distribution of the internal state of $RC4$, and it seems that no serious attack can work without considering this distribution.

5.3 Improved Branch and Bound attack

My method of defining *useful* states and using them to improve the Branch and Bound attacks on $RC4$ is detailed in a separate paper. Here I'll give a short description of the method. The method is based on identifying a sequence of consecutive *useful* states, extracting some information from them, and invoking the Branch and Bound algorithm on the partial solution.

6 Conclusions

The previous results on the cryptanalysis of $RC4$, doesn't threat the security of $RC4$. The attacks that are described in [MT99] and in [KMP+98] are trivial and doesn't expose structural information about $RC4$.

I think that the most important aspect of my first result, is the first analysis of the distribution of the secret part of the cipher as a function of the known part. I believe that a practical attack on $RC4$ must use such information in order to expose the secret part. It seems that the known part of the cipher, contains a lot of important information on the distribution of the hidden part, and this information can be used in almost every attack on the cipher.

The second result is the best known attack on this cipher. This attack is practical for $n = 5$, but the more important aspect of it, is the definition of the *useful* states and the methods of using them. It seems that this attack can be improved by deeper analysis of the properties of *useful* states, and by finding a non-trivial way for identifying them.

I believe that my results opens a window into the real weaknesses of RC4, and therefore, they poses a new direction of research on the cryptanalysis of this cipher.

References

[Fin94] H. Finney. Rc4 cycle that can't happen. September 1994.

[FM00] S. R. Fluhrer and D. McGrew. Statistical analysis of the alleged rc4 keystream generator. pages 19–30, 2000.

[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.

[MT99] S. Mister and S. E. Tavares. Cryptanalysis of RC4-like ciphers. *Lecture Notes in Computer Science*, 1556:131–143, 1999.