Cryptanalysis of a 32-bit RC4-like Stream Cipher

Hongjun Wu

Katholieke Universiteit Leuven, Dept. ESAT/COSIC
wu.hongjun@esat.kuleuven.be

Abstract. Nawaz, Gupta and Gong recently proposed a 32-bit RC4-like stream cipher. In this paper, we show that the keystream generated from their stream cipher is not random. The keystream can be distinguished from random with only about 100 outputs (3200 bits) in 2 milliseconds on Intel Centrino 1.6GHz processor.

1 Introduction

A 32-bit RC4-like stream cipher was recently proposed by Nawaz, Gupta and Gong [1]. It is very efficient in software and it is claimed that its the speed is about 3.5 times that of RC4.

In the rest of this paper, we refer this cipher as NGG stream cipher. And we will focus on the version which is fully specified, i.e., the version with 256 32-bit elements in the table.

In this paper, we point out that the NGG stream cipher is very weak. The keystream can be distinguished from random with only about 100 outputs and negligible amount of computation time. We will describe the NGG stream cipher in Section 2 and give our attack in Section 3.

2 The Stream Cipher NGG

NGG is a 32-bit RC4-like stream cipher. The table in NGG is with 256 32-bit elements. At each step, one 32-bit element is generated and one 32-bit output is updated.

We ignore the key schedule here. The keystream generation of NGG is given below. (The S is a table with 256 32-bit elements. a_i $(0 \leq i \leq 255)$ are some random constants, $N = 256$, $M = 2^{32}$).

$i = 0$
$j = 0$
iterate until enough keystream bits are generated
  $i = (i + 1) \bmod N$
  $j = (j + S[i]) \bmod N$
  Swap $(S[i], S[j])$
  output $= S[(S[i] + S[j]) \bmod N]$
  $S[(S[i] + S[j]) \bmod N] =$ (S[i]+S[j]) mod M

3 Distinguishing attack on the NGG stream cipher

We notice the value of each element is updated as the sum of two elements in the table. And each output of NGG is given as an element randomly chosen from the table. The flaw is that if $S[k]$ is updated as $S[k] = S[i] + S[j]$, and if $S[i]$, $S[j]$, and $S[k]$ are selected as three outputs out'_i, out'_j and out'_k, then $out'_k = out'_i + out'_j$ (Here the i', j' and k' are three random numbers).

In the attack, we do not know which elements are selected as outputs and their positions in the table and the keystream, but we are certain that for three outputs out_x, out_y and out_z, the probability that $out_x = out_y + out_z$ would increase due to the flaw indicated above.

We carried out the experiment to compute the probability that $out_x = out_y + out_z$. For n consecutive outputs out_b, out_{b+1}, out_{b+2}, ... , out_{b+n-1}, we carry out the experiment with the following pseudo code:

$counter_1 = 0$
$counter_2 = 0$
for $i = b$ to $b + n - 1$
  for $j = i$ to $b + n - 1$
    for $k = b$ to $b + n - 1$
      $count_1$++;
      if ($out_k == out_i + out_j$) $count_2$++;

The probability $p = \frac{count_2}{count_1}$

We mention here that in the above pseudo code, if $i = j = k$, then the values of $counter_1$ and $counter_2$ should remain unchanged. And this experiment is repeated for a number of times with different keystream to compute the average probability.

For $n = 100$, this probability is $2^{-18.7}$, which is much larger than the random probability 2^{-32}. It means that once we find the first $out_k = out_i + out_j$ in the experiment above, then we are almost certain that the keystream is distinguished from random. And in average, for every consecutive 100 outputs, we find about 1.1 cases that $out_x = out_y + out_z$. It is the same as to say that with 100 outputs, we can distinguish the keystream from random with probability close to 1.

Thus our attack can distinguish the output of NGG cipher from random with only 100 outputs. The attack requires only about 2 milliseconds on the Intel Centrino 1.6GHz processor.

We note that there are some other attacks that can also be applied to distinguish the keystream of NGG from random. For example, in the keystream generation process given in Section 2, we do the approximation $j \approx j + i$, and ($output$ mod 256) $\approx$ $(i + j)$ mod 256. The two expressions hold with chance larger than 2^{-8}, and it results in the probability that $(out_x - out_{x-1} - x -$

1)&255 $= 0$ larger than 2^{-8}. We ignore the details of those attacks in this report.

4 Discussion

The NGG stream cipher is very weak. The attack given in this paper can distinguish the keystream of NGG from random efficiently.

In [1], the designers of NGG stream cipher suggested using rotation in the cipher to enhance the security of the cipher. However, that improvement fails to resist the attack given in this paper.

References

1. Y. Nawaz, K.C. Gupta, and G. Gong, "A 32-bit RC4-like Keystream Generator", Available at http://eprint.iacr.org/2005/175