The Perils of Repeating Patterns: Observation of Some Weak Keys in RC4

Joachim Strömbergson[1], Simon Josefsson[2]

Abstract

We describe some observed trivially weak keys for the stream cipher RC4.
Keys with repeating patterns are found to be key length invariant. The cause of the problem is the simplistic key dependent state permutation in the RC4 initialization.

Introduction

While writing the draft for RFC 6229 [1] and testing suitable test vectors, we observed that for some keys with different lengths, the stream cipher RC4 [2] generated identical keystreams.

Typical test patterns we tested were patterns where odd or even bits are set, as well as repeated sequences of byte values. What we saw was that these patterns generated the same keystream irrespectively of the key length used. The following two sets of keys illustrates the behaviour:

```
Key k1 = [0x55,0x55,0x55,0x55,0x55] (40 bit)
Key k2 = [0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55] (64 bit)
Generated keystream: 0x06,0xfe,0x68,0xd8,0x0,0xf9,...

Key k3 = [0x01,0x02,0x03,0x04] (32 bit)
Key k4 = [0x01,0x02,0x03,0x04,0x01,0x02,0x03,0x04] (64 bit)
Generated keystream: 0x1c,0xea,0x91,0x61,0xee,0xbc,...
```

Problem Description and Analysis

The RC4 stream cipher contains a Key Scheduling Algorithm (KSA) as given by the following pseudo code:

```
for i from 0 to 255                                          (1)
     S[i] = i                                                (2)
j = 0                                                        (3)
for i from 0 to 255                                          (4)
     j = (j + S[i] + key[i mod keylength]) mod 256           (5)
     swap(S[i], S[j])                                        (6)
```

The only key dependent operation of the KSA is the update of the *j* pointer in (5). Using the values of the bytes in the key, we add a displacement to *j,* which then affects the byte swap operation of the state *S* in (6).

Since the byte values in the key are accessed in sequence and cyclicly in (5), any key that consists

1 Joachim at secworks.se
2 Simon at josefsson.org

of a repeated pattern can be reduced to the length of the pattern. That is:

```
key A = [P, P, P, P, P]  = key B = [P]
where P is a pattern [p1, p2, p3, p4,...] of the length L.
```

In the asymptotic case that the pattern length `L` is one byte, that is the key consists of a number of equal bytes, the the operation in line (5) is reduced to a fixed displacement `C`:

```
j = (j + S[i] + C) mod 256                    (5')
```

This means that the for these types of keys, the key is length invariant. The effective key length for the key `n*P` (i.e a key with n instances of the pattern *P*) is not `n*L`, but `L` no matter how large the value of n is. A 128 bit key that consists of `0xdeaddeaddeaddead` is just `0xdead`.

Discussion

Since the observed behaviour and the simplistic, cyclic behaviour caused by the MOD operation is so obvious it seems improbable that this problem has not been seen before.

We have tried to survey the literature and research to see if the behaviour has previously been documented. There are a huge number of papers published on RC4 and applications such as the WEP wireless security standard. We therefore by no means claim to have done a complete survey.

The papers [3]...[27] describes analysis of and attacks on the KSA as well as the Pseudo Random Generator Algorithm (PRGA) part of RC4. The problems found can be divided into categories such as internal state recovery, secret key recovery, key and state leakage in the keystream as well as bias and patterns in the generated keystream.

But we have not been able to find any book or paper describing the behaviour we have observed. Neither have we found any documented recommendations that recommends users to avoid RC4 keys based on repeated patterns.

We note that the key dependent part of the KSA, where the key material is applied in a cyclic pattern is similar to the key dependent initialization of the P-array in the block cipher Blowfish. In the specification for Blowfish [28], Bruce Schneier notes that: *For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys*.

This is basically the same observation as what we have seen with RC4. Albeit we believe that the important thing is not that for a key with pattern *P*, there exists longer keys *PP*, *PPP* that are equivalent. No, the important thing is that *PP* and *PPP* can be reduced to *P*.

Conclusions

Pointing out flaws in RC4 might seem like flogging a dead horse. But RC4 is still widely used and we have also observed keys that consists of repeated patterns used in the wild. We therefore believe that this is a real problem that need to be properly documented.

If you are going to use RC4, use a key with a pattern length equal to the effective key length you need. In other words – don't use keys like `ABCDABCDABCDABCD` or `1111111111111111`.

References

[1] J. Strombergson, S. Josefsson. *Test vectors for the stream cipher RC4*. RFC 6229. Internet Engineering Task Force (IETF). April 2011. http://tools.ietf.org/html/rfc6229

[2] Wikipedia. *RC4*. Retrieved 2011-03-21. http://en.wikipedia.org/wiki/RC4

[3] Andrew Roos. *A Class of Weak Keys in the RC4 Stream Cipher*. 1995. http://www.impic.org/papers/WeakKeys-report.pdf

[4] S. R. Fluhrer, D. A. McGrew. *Statistical Analysis of the Alleged RC4 Keystream Generator*. FSE 2000. http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/FluhrerMcgrew.pdf

[5] S. Fluhrer, I. Mantin, A. Shamir. *Weaknesses in the Key Scheduling Algorithm of RC4*. Selected Areas in Cryptography. 2001. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.2652&rep=rep1&type=pdf

[6] I. Mantin, A. Shamir. *A Practical Attack on Broadcast RC4*. FSE 2001. Revised Papers from the 8th International Workshop on Fast Software Encryption. http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/bc_rc4.ps

[7] I. Mironov. *(Not so) Random Shuffles of RC4*. IACR Cryptology ePrint Archive. Report 2002/067. http://eprint.iacr.org/2002/067

[8] M. Pudrovkina. *Statistical weaknesses in the alleged RC4 keystream generator*. IACR Cryptology ePrint Archive. Report 2002/171. http://eprint.iacr.org/2002/171

[9] S. Paul, B. Preneel. *Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator*. INDOCRYPT 2003. http://www.cosic.esat.kuleuven.be/publications/article-86.pdf

[10] M. Pudrovkina. *The number of initial states of the RC4 cipher with the same cycle structure*. Journal of Information Security 2002. IACR Cryptology ePrint Archive. Report 2003/012. http://eprint.iacr.org/2003/012

[11] S. Paul, B. Preneel. *A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher*. FSE 2004. http://www.cosic.esat.kuleuven.be/publications/article-40.pdf

[12] H. Wu. *The Misuse of RC4 in Microsoft Word and Excel*. IACR Cryptology ePrint Archive. Report 2005/007. http://eprint.iacr.org/2005/007

[13] S. Doroshenko, B. Ryabko. *The experimental distinguishing attack on RC4*. IACR Cryptology ePrint Archive. Report 2006/070. http://eprint.iacr.org/2006/070

[14] A. Klein. *Attacks on the RC4 stream cipher.* Designs, Codes and Cryptography (2008) http://cage.ugent.be/~klein/RC4/RC4-en.ps

[15] G. Paul, S. Maitra. *RC4 State Information at Any Stage Reveals the Secret Key.* SAC 2007. Cryptology ePrint Archive. Report 2007/208.
http://eprint.iacr.org/2007/208

[16] A. Maximov, D. Khovratovich. *New State Recovery Attack on RC4.* Cryptology ePrint Archive. Report 2008/017.
http://eprint.iacr.org/2008/017

[17] R. Basu, S. Ganguly, S. Maitra, G. Paul. *A Complete Characterization of the Evolution of RC4 Pseudo Random Generation Algorithm*. Journal of Mathematical Cryptology. October, 2008.
http://www.reference-global.com/doi/abs/10.1515/JMC.2008.012

[18] E. Biham, Y. Carmeli. *Efficient Reconstruction of RC4 Keys from Internal States*.
FSE 2008.
http://www.iacr.org/archive/fse2008/50860272/50860272.pdf

[19] M. Akgun, P. Kavak, H. Demirci. *New Results on the Key Scheduling Algorithm of RC4*.
INDOCRYPT 2008.
http://www.springerlink.com/content/cr301963u2m26393/

[20] J. Golic, G. Morgari. *Iterative Probalistic Reconstruction of RC4 Internal States*.
Cryptology ePrint Archive. Report 2008/348.
http://eprint.iacr.org/2008/348

[21] S. Maitra, G. Paul. *Analysis of RC4 and Proposal of Additional Layers for Better Security Margin.* Cryptology ePrint Archive. Report 2008/396.
http://eprint.iacr.org/2008/396

[22] S. Maitra, G. Paul. *New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4*. FSE 2008
http://eprint.iacr.org/2007/261.pdf

[23] R. Basu, S. Maitra, G. Paul, T. Talukdar. *On Some Sequences of the Secret Pseudo-random Index j in RC4 Key Scheduling*. Springer. Lecture Notes in Computer Science. 2009.
http://www.springerlink.com/content/q365643q743t56w0/

[24] G. Paul. *Fast and Efficient Key Recovery from RC4 Permutation after KSA*. Presented at the Indo-Japan meeting, ISI, Kolkata, April 13, 2009.
http://www.rcis.aist.go.jp/files/project/JST-DST/2nd-visit-en/Lecture_5-rcjap.pdf

[25] *S. Gupta, S. Maitra, G. Paul, S. Sarkar. RC4: (Non-)Random Words from (Non-)Random Permutations.* IACR Cryptology ePrint Archive: Report 2011/448.
http://eprint.iacr.org/2011/448

[26] J. Lv, B. Zhang, D. Lin*. Distinguishing Attack on RC4 and A New Improvement of the Cipher.*
IACR Cryptology ePrint Archive: Report 2013/176.
http://eprint.iacr.org/2013/176

[27] M. A. Orumiehchiha, J. Pieprzyk, E. Shakour, R Steinfeld *Cryptanalysis of RC4(n,m) Stream Cipher.* IACR Cryptology ePrint Archive: Report 2013/176*.*
http://eprint.iacr.org/2013/178

[28] B. Schneier. *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*.
FSE 1994.
http://www.schneier.com/paper-blowfish-fse.html