

ESC 2013

# Near-colliding Keys in RC4

Willi Meier

(joint work with M. Lehmann, FHNW, S. Maitra, G. Paul,  
S. Sarkar, Indian Statistical Institute, Kolkata)



University of Applied Sciences Northwestern Switzerland  
School of Engineering

# Overview

- ▶ Motivation
- ▶ RC4 (Description)
- ▶ Conditional differentials in the key scheduling
- ▶ Near-colliding states
- ▶ Related key streams
- ▶ A related key stream distinguisher
- ▶ Conclusion

# Motivation

Study of key scheduling (KSA) of RC4 stream cipher.

KSA of RC4 transforms  $\ell$ -byte key into  $N$ -byte initial state,  $N = 256$ .

KSA has good differentials for  $\ell = 256$ -byte keys (Grosul-Wallach, 2000).

Differences in last two key bytes of form  $(+\delta, -\delta)$  for some non-zero  $\delta$ .

Key streams same in first few bytes with high probability.

Experiments for  $\ell = 16$ : Similar state tables and related keys (Demirci-Ayaz-Selçuk, 2006).

# Motivation

Are there key collisions, i.e., two different keys that produce same initial state (and thus same key stream)?

Reduced effective key size?

Initial considerations for  $\ell = 32$  by Biham-Dunkelman, 2007.

Practical key collisions for  $\ell = 24$  bytes (Matsui, FSE 2009) and  $\ell = 22$  (Chen-Miyaji, ISC 2011).

**Question:** Practical (near)-collisions for key size  $\ell = 16$  bytes, as is most common in implementations of RC4?

Two types of near-collisions:

- ▶ In initial state
- ▶ In the output (related key stream)

## RC4 (Description)

$\ell$ -byte key  $k$  is expanded into  $N$ -byte array  $K[0\dots(N - 1)]$ :

$K[y] = k[y \bmod \ell]$  for any  $y$ ,  $0 \leq y \leq N - 1$ .

---

### Algorithm 1 KSA

---

**for**  $i = 0$  to  $N - 1$  in steps of 1 **do**

$S[i] = i$

**end for**

$j = 0$

**for**  $i = 0$  to  $N - 1$  in steps of 1 **do**

$j = (j + S[i] + K[i])$

    Swap( $S[i], S[j]$ )

**end for**

---

# RC4 (Description)

---

## Algorithm 2 PRGA

---

$i = j = 0$

*Key Stream Generation Loop:*

$i = i + 1$

$j = j + S[i]$

Swap( $S[i], S[j]$ )

$t = S[i] + S[j]$

**return**  $z = S[t]$

---

# Conditional differentials in KSA

Method for studying key scheduling of stream ciphers:

Conditional differentials, message modification (borrowed from analysis of hash functions and block ciphers).

Various differences conceivable for key pairs.

Differences in a single byte or in two consecutive bytes preferable:

Difference trails more likely to be controllable; conditions allow for trails of weight as low as 2 (or 3) throughout KSA.

Colliding key pairs  $(k_1, k_2)$  by Matsui, Chen-Miyaji, use single byte difference:

Bytes coincide, except  $k_2[d] = k_1[d] + 1$  for some  $d$  in  $0 \leq d \leq \ell - 1$ .

# Conditional differentials in KSA

During KSA, key repeated  $n = \lceil (N + \ell - 1 - d) / \ell \rceil$  times, called periods.

Complexity of collision search proportional to number of conditions, i.e., number of periods.

Effort significant for  $\ell = 24$  or  $\ell = 22$ .

Finding exact collisions for 16-byte key, even with refinements of Chen-Miyaji, infeasible in practice.

Similar method applicable to near-collisions in state.

However: Near-collisions don't always provide strongly related key streams.



# Conditional differentials in KSA

For matching key stream with random key pairs:

Difference  $(-\delta, +\delta)$  in two (consecutive) bytes of key more appropriate than difference in single byte:

- ▶ Less conditions to follow differential trail.
- ▶ More than one set of conditions possible.
- ▶ Sum of all key bytes  $\pmod N$  remains unchanged.

Let

$$k_2[i] = k_1[i], \text{ all } i \in [0, \ell - 3],$$

$$k_2[d] = k_1[d] - 1, k_2[d + 1] = k_1[d + 1] + 1, \text{ where } d = \ell - 2.$$

Same condition in each period:

$$j = \ell - 2 \text{ if } i = d \pmod{\ell}.$$

If satisfied, weight of difference remains 2 for all  $i \in [d, N - 1]$ .

# Near-colliding states

To satisfy conditions, use multi-key modification (Chen-Miyaji), inspired by collision search in hash functions.

In search algorithm, random key is manipulated in order to satisfy conditions.

Notation:  $j_y$  and  $S_y$  denote the index  $j$  and the permutation  $S$  after  $i = y$ .

Class 1 and Class 2 conditions on index  $j$ :

Class 1 conditions set so that difference follows desired trail.

Class 2 conditions assure that  $j$  does not point to index within the period in question.

## Near-colliding states

For passing first two periods, two Class 1 conditions on key are set (target value is  $j_d = d$ ):

In first period, update of  $j$  is  $j_d = j_{d-1} + S_{d-1}[d] + K[d] = target$ .  
Hence, a (deterministic) condition is

$$K[d] = target - j_{d-1} - S_{d-1}[d].$$

## Near-colliding states

To pass second period, require

$$j_{\ell+d} = j_d + K[d+1] + \sum_{m=0}^d K[m] + \sum_{m=d+1}^{\ell+d} S_{m-1}[m] = \text{target} = j_d.$$

Condition satisfied (with high probability) by modifying key as

$$K[d+1] = -\left(\sum_{m=0}^d K[m] + \sum_{m=d+1}^{\ell+d} S_d[m]\right).$$

For each period to follow, two other key bytes to be set at prescribed positions.

## Near-colliding states

Denote  $t$  the first period not passed and  
 $(t - 1) * l + d \geq r \geq (t - 2) * l + d + 1$ .

Value of  $j$  is given by  $j_r = j_{r-1} + S_{r-1}[r] + K[r]$ .

$D = j_r - target$ . Hence

Value of  $j_r$  needs to be decreased by  $D$ .

Therefore we need

$$S'_{r-1}[r] = S_{r-1}[r] - D = S_{r-1}[r] - j_r + target = \Delta_r.$$

## Near-colliding states

This can be achieved by setting

$$j'_{\Delta_r} = j_{\Delta_r-1} + S_{\Delta_r-1}[\Delta_r] + K'[\Delta_r] = r.$$

Take the difference  $j'_{\Delta_r} - j_{\Delta_r} = K'[\Delta_r] - K[\Delta_r] = r - j_{\Delta_r}$ .

Thus we modify the key as follows:

$$K[\Delta_r] = K[\Delta_r] + r - j_{\Delta_r}, \quad K[\Delta_r + 1] = K[\Delta_r + 1] - r + j_{\Delta_r}$$

Works only if  $\Delta_r \leq (t - 2) \cdot \ell + d$ .

# Near-colliding states

Modified  $j_{(t-1)\ell+d}$  will have correct value with some probability.

For key of  $\ell = 16$  bytes, only limited freedom for choosing  $\Delta_r$ .

In experiments, algorithm didn't pass more than six (out of 16) periods.

Conditions  $j = \ell - 2$  if  $i = d \pmod{\ell}$  are in favor of near-colliding states, but they are not necessary.

# Near-colliding states

Random experiments for near-collisions in state with  $(-1, +1)$ -difference in key show:

Conditions are not strict values that  $j_{(x-1)\ell+d}$ ,  $1 \leq x \leq n$ , should take.

Values  $j_{(x-1)\ell+d}$ ,  $x > 1$ , merely have to touch a difference.

Effect: Existing differences are not spread further but just shifted.

Best near-collision found has 230 (out of 256) matching state bytes after KSA.



# Near-colliding states

Other parameters:

Can study near collisions for  $(-\delta, +\delta)$ -difference for  $\delta \neq 1$  as well.

Instead of  $d = \ell - 2$ , any  $d$ ,  $0 \leq d \leq \ell - 2$ , conceivable for position of difference.

Best results however for parameters as chosen.

# Related key streams

Key pairs that produce (large number of) colliding bytes in key stream?

Near-collisions in state:

Key pair with  $\ell = 20$  and difference  $+1$  in one byte:

Near-collision of weight two in state (Matsui);

Leads to match of only 20 bytes in first 256 key stream bytes.

Different approach:

Search directly for random key pairs with  $(-1, +1)$ -difference that produce 20 or more colliding bytes within first 256 key stream bytes.

## Related key streams

Two structural properties are imposed (hold except in few cases):

P1. Position of first mismatch in state comes as late as possible.

P2. Number of mismatches in state is as low as possible.

Both properties in favor of colliding key stream bytes:

P1.: With significant probability, colliding state bytes are addressed in output function.

P2.:  $j$ -values coincide in both states with good probability:  
Updated state bytes remain the same.

# Related key streams

Property P2. at the basis of phenomena related to (near-) collisions in state and key stream.

P2. can be studied independently.

Experiments for related key streams: Number of mismatches in state around 48.

Probability for difference in state of weight 48 or less: Around 0.0025%.

Unusually high for key size  $\ell = 16$  bytes and state of  $N = 256$  bytes.

## Related key streams

Explanation:

Consider evolving difference period after period.

Initial  $(-1, +1)$ -difference in key causes either difference 2 or 3 at position  $i = d$ .

Difference in later 15 periods should not grow in steps of more than 3:

$i$  must not touch a difference (if it does, the two updated values of  $j$  will be different).

Modelling of this situation: Confirms experimental probability 0.0025% (that number of mismatches in state after KSA is no more than 48).

With modified Chen-Miyaji algorithm, this probability can be increased to 0.042%.

# Related key streams

## Results:

Using modified Chen-Miyaji algorithm easy to find key pairs with  $(-1, +1)$ -difference with more than 45 colliding key stream bytes ( $\ell = 16$ ):

25 key pairs in  $10^8$  trials.

Present record: 79 colliding bytes amongst the first 256 key stream bytes.

## Related key streams

Average and maximum number of collisions in state (*nstate*) and key stream (*nstream*) for different differences and search methods:

Search strategy	nstate		nstream	
	avg	max	avg	max
Random diff. (+1 at $d$ )	4.94	42	1.002	9
Random diff. ( $\mp 1$ at $d, d + 1$ )	24.27	230	1.015	37
Chen-Miyaji, One-difference	4.78	18	0.980	6
Chen-Miyaji modif., $target = d$	47.02	226	1.059	77
Adaptive algorithm	47.6	228	1.063	79

Table: Experimental results related to near-collisions.

## Related key streams

Adaptive algorithm again loops over  $i$ .

Tradeoff between two conditions.

Conditions at positions  $i$  with  $i = d \bmod \ell$ :

If  $j \leq i$  no action, else apply key modification with *target* =  $d$ .

First condition in detail:

$$j_{(x-1)\cdot\ell+d} \leq i = (x-1)\cdot\ell + d, 1 \leq x \leq n.$$

Is related to Property P.2 and assures that  $i$  does not touch a difference at any step.

Condition  $j_{i=(x-1)\cdot\ell+d} \leq i$  satisfied with prob.  $\frac{i+1}{N}$ .

Condition *target* =  $d$  satisfied only with prob.  $\frac{1}{N}$ .



# Related key stream distinguisher

Mount a distinguisher on RC4 using related keys.

Assume the attacker can only modify the secret key and rerun key stream generation as a black box several times.

He chooses random key pairs with two-byte difference,  
 $k_2[d] = k_1[d] - 1, k_2[d + 1] = k_1[d + 1] + 1, d = \ell - 2.$

First key stream bytes match with probability significantly higher than random.

## Related key stream distinguisher

Probability that first key stream bytes match is about 0.011.  
This is close to  $3/N$ .

Compares favorably with probability  $2/N$  that second key stream byte is zero (Mantin-Shamir (FSE 2001)).

Have got very efficient distinguisher for RC4.

Bias even larger if key size  $\ell > 16$ , e.g., for  $\ell = 30$ , bias is about 0.044.

## Related key stream distinguisher

For  $\ell = 16$  and success probability of about 70%, 138 sample key pairs are needed.

Use formula

$$\frac{\left(\sqrt{1-p} + \sqrt{(1+q)(1-p(1+q))}\right)^2}{4pq^2},$$

where  $p$  is probability of event in random stream, and in RC4 (or in any algorithm) with probability  $p(1+q)$ .

## Related key stream distinguisher

Explanation for distinguisher:

For first key stream byte to collide, need for the two states  $S_1$  and  $S_2$ :

$$S_1[S_1[1] + S_1[S_1[1]]] = S_2[S_2[1] + S_2[S_2[1]]].$$

Collision: Requires that if  $j_1 \neq j_2$ , they don't take values 1,  $S[1]$ ,  $S[S[1]]$  in  $S_1, S_2$ .

For first few periods,  $j_1$  and  $j_2$  collide with high probability (except for  $i = d, d + \ell, d + 2\ell, \dots$ , where  $j_1 \neq j_2$ ).

Thus for  $\ell = 16$ :

$$p \approx \left(1 - \frac{3}{N}\right)^{2(N-3\ell-d)} \approx 0.01$$

# Conclusion

- ▶ Have put various results regarding collisions and related keys in RC4 in perspective.
- ▶ Efficient algorithms for finding key pairs with related key stream.
- ▶ Additional insight into key scheduling.
- ▶ New distinguisher for RC4.
- ▶ Results apply to key size 16 bytes as used in implementations of RC4.
- ▶ Open problem: Reduced-complexity key search?