Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 183 (2021) 456–461

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Conference of Information and Communication Technology (ICICT-2020)

A new form of initialization vectors in the FMS attack of RC4 in WEP

Teng Guo[a,*], YuanZhe Feng[a], YuHan Fu[b]

[a]*School of Information Science and Technology, University of International Relations, Poshang Cun 12, Haidian District, Beijing 100091, China*
[b]*China Industrial Control Systems Cyber Emergency Response Team, No. 35, Lugu Rd., Shijingshan District, Beijing 100040, China*

Abstract

S. Fluhrer, I. Mantin and A. Shamir (FMS) had put forward a fascinating attack of the key of RC4 in Wired Equivalent Privacy (WEP) protocol on the basis of special initialization vectors (IVs) of pattern (*3, 255, v*). Inspired by the FMS attack, this paper tries to find new pattern of IVs that can be used for recovering the key of RC4 in WEP. We discovered that IVs of new pattern (*v, 257-v, 255*) can also be used in the FMS attack in a very similar way. Combining the new pattern with the previous pattern (*3, 255, v*) could result in a higher proportion of IVs that can be exploited in the FMS attack.

Keywords: RC4; WEP; Initialization Vector; FMS attack

1. Introduction

As a popular stream cipher, RC4 is put forward by Ron Rivest in 1987 and is in the service of many different security applications: 1, the SSL/TLS protocol, in which it serves as the default cipher [1]; 2, the wired equivalent privacy (WEP) protocol, in which it serves as a cryptographic primitive [2]. Due to its simplicity and straightforwardness, RC4 has drawn lots of attention from many scientific researchers, and many weaknesses are revealed ceaselessly.

* Corresponding author. Tel.: +86-010-62861449; fax: +86-010-62861449.
*E-mail address:*guoteng.cas@gmail.com

10.1016/j.procs.2021.02.084

In [3], S. Fluhrer, I. Mantin and A. Shamir showed a known plaintext attack of the usage of RC4 in WEP protocol based on a special pattern of IVs, where their method is well known as the FMS attack. Inspired by their method, we try to find new pattern of IVs that still can be used in the FMS attack. Our main contribution includes: 1, the discovery of IVs of new pattern (*v, 257-v, 255*) meet our expectations; 2, the complete verification of the FMS attack with the newly discovered IVs. Besides, there are still many researches that either aimed to point out more weaknesses of RC4 and its usage [4, 5, 6, 7, 8] or tried to propose remedial solutions [6, 9].

The paper includes the following parts: In section 2, we present the RC4 algorithm and the FMS attack of its usage in WEP protocol; In section 3, we consider a novel pattern of IVs that can also be used in the FMS attack in a similar way; In section 4, our work is summarized.

2. Preliminary

In this section, we first present the RC4 algorithm in the form of C code. Then we present the FMS attack in [3].

2.1. RC4 algorithm

The RC4 algorithm consists of two parts: *1*, Key-Scheduling Algorithm (KSA); *2*, Pseudo-Random Generation Algorithm (PRGA). We first describe the KSA, which takes the long-term key K as input and use it to scramble a table T that contains a permutation of *0,1,...,255* and will serve as a basis to output keystream in PRGA. Suppose the key K contains N bytes. The C code of KSA is as follows.

```
KSA(K):
  Stage 1:
    for a = 0 to 255
      T[a] = a
      K[a] = key[a (mod N)]
    b = 0
  Stage 2:
    for a = 0 to 255
      b = (b + T[a] + K[a]) (mod 256)
      exchange(T[a],T[b])
```

The PRGA interchanges elements of table T and chooses a byte from the table as a keystream byte, which is presented as follows.

```
PRGA(K)
  Stage 1:
    a =b = 0
  Stage 2:
    a = (a + 1) (mod 256)
    b = (b + T[a]) (mod 256)
    exchange (T[a], T[b])
    t = (T[a] + T[b]) (mod 256)
    keystream = T[t]
```

2.2. FMS attack

WEP protocol employs a long-term key K while every data packet has to be encoded by a different seed key. The seed key for each packet is in the pattern of (IV, K), and the IV is transmitted in plaintext. Each packet obtains a new IV that always has *24*-bit (*3* byte) and is prepended to K. The seed key is recorded as bytes (K_0, K_1, K_2, K_3, K_4, K_5,..., K_{N+2}), where (K_0, K_1, K_2) is known by the attacker as IV and the long-term key K=(K_3, K_4, K_5,..., K_{N+2}) is kept secret. Given enough IVs, FMS demonstrated that the attacker can decode the long-term key K based on the first keystream byte. Their method belongs to known plaintext attack and uses the plaintext and its corresponding ciphertext to recover the keystream, from which the first byte of K can be computed.

Now we will review how they carry out the cryptanalysis that will recover K_3 (the first byte of K). The attacker keeps watch on the packet and records the packet with IVs of pattern *(3, 255, v),* where *v* could be any number between *0* and *255* and is publicly known by the attacker as a constant. In such a case, the actual RC4 key is (K_0=*3*, K_1=*255*, K_2=*v*, K_3, K_4, K_5,..., K_{N+2}) and the attacker aims to recover K_3.

Recall the RC4 initialization algorithm: KSA, and the array T is first set to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**254**	**255**
$\mathbf{T_a}$	0	1	2	3	4	5	…	254	255

Review KSA: set b = 0 and
for a = 0 to 255
$b = b + T_a + K_a$
exchange(T_a, T_b)

In step a = 0, we have
a = 0
$b = b + T_0 + K_0 = 0 + 0 + 3 = 3$
exchange (T_a, T_b) = exchange (T_0, T_3)
Now the array T is altered to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**254**	**255**
$\mathbf{T_a}$	3	1	2	0	4	5	…	254	255

In step a = 1, we have
a = 1
$b = b + T_1 + K_1 = 3 + 1 + 255 = 3 \pmod{256}$
exchange (T_a, T_b) = exchange (T_1, T_3)
Now the array T is altered to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**254**	**255**
$\mathbf{T_a}$	3	0	2	1	4	5	…	254	255

In step a = 2, we have
a = 2
$b = b + T_2 + K_2 = 3 + 2 + v = 5 + v$
exchange (T_a, T_b) = exchange (T_2, T_{5+v})
Now the array T is altered to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**5+*v***	**…**	**254**	**255**
$\mathbf{T_a}$	3	0	5+*v*	1	4	5	…	2	…	254	255

In step a = 3, we have
a = 3
$b = b + T_3 + K_3 = 5 + v + 1 + K_3 = 6 + v + K_3$
exchange (T_a, T_b) = exchange (T_3, T_{6+v+k3})
Now suppose $6+v+K_3 > 5+v \pmod{256}$, the array T is altered to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**5+*v***	**…**	**6+*v*+K_3**	**…**	**254**	**255**
T_a	3	0	5+*v*	6+*v*+K_3	4	5	…	2	…	1	…	254	255

Suppose that KSA comes to a halt after the step a = 3, and we analyze the RC4 keystream output algorithm: PRGA.

First set a = b = 0, and for every keystream byte
a = a + 1
b = b + T_a
exchange (T_a, T_b)
t = T_a + T_b
keystream = T_t
On basis of the table T just after the a = 3 step, we have
a = a + 1 = 1
b = b + T_1 = 0
t = T_a + T_b = T_1 + T_0 = 0 + 3 = 3
keystream= T_t = T_3 = 6 + *v* + K_3

Hence we can recover K_3 by computing K_3= keystream - 6 - *v* (mod 256), where keystream is the first byte of the key stream.

However the initialization does not stop at the step a = 3. Now we need to estimate the successs probability of the above cryptanalysis. Since the first keystream only depends on T_0, T_1 , T_3. If elements at positions 0, 1 and 3 are not interchanged in the rest of KSA steps, the above cryptanalysis works. For the rest of KSA steps, we have a = 4, 5, ..., 255, so index a will not affect elements at positions 0, 1 or 3. Based on the computing step of b in KSA: b = b + T_a + K_a, we may suppose that index b is chosen at random. For every step, the probability that $b \notin \{0, 1, 3\}$ is $\frac{253}{256}$. Since there are 252 residual steps, the probability that positions 0, 1 and 3 are not altered by index b after the step a = 3 is $\left(\frac{253}{256}\right)^{252} \approx 0.0513$. In a word, the attacker can recover K_3 based on a recorded data packet of IVs of pattern (3, 255, *v*) with approximate probability 0.0513.

3. Our work

In this section, we show that IVs of new pattern (*v, 257-v, 255*) can also be used in the FMS attack to recover K_3 in a very similar way.

Similarly review the RC4 initialization algorithm: KSA, and the array T is first set to

a	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**254**	**255**
T_a	0	1	2	3	4	5	…	254	255

In step a= 0, we have
a = 0
b = b + T_0 + K_0 = 0 + 0 + *v* = *v*
exchange (T_a, T_b) = exchange (T_0, T_v)
Now the array T is altered to

a	**0**	**1**	**2**	**3**	**…**	***v***	**…**	**254**	**255**
T_a	*v*	1	2	3	…	0	…	254	255

In step a = 1, we have
a = 1

$b = b + T_1 + K_1 = v + 1 + 257\text{-}v = 2 \pmod{256}$
exchange (T_a, T_b) = exchange (T_1, T_2)
Now the array T is altered to

a	**0**	**1**	**2**	**3**	**…**	***v***	**…**	**254**	**255**
$\mathbf{T_a}$	*v*	2	1	3	…	0	…	254	255

In step a = 2, we have
$a = 2$
$b = b + T_2 + K_2 = 2 + 1 + 255 = 2 \pmod{256}$
exchange (T_a, T_b) = exchange (T_2, T_2)
Now the array T remains unchanged

a	**0**	**1**	**2**	**3**	**…**	***v***	**…**	**254**	**255**
$\mathbf{T_a}$	*v*	2	1	3	…	0	…	254	255

In step a = 3, we have
$a = 3$
$b = b + T_3 + K_3 = 2 + 3 + K_3 = 5 + K_3$
exchange (T_a, T_b) = exchange (T_3, T_{5+k3})
Now suppose $5 + K_3 > v \pmod{256}$, the array T is altered to

a	**0**	**1**	**2**	**3**	**…**	***v***	**…**	**5+K₃**	**…**	**254**	**255**
$\mathbf{T_a}$	*v*	2	1	$5 + K_3$	…	0	…	3	…	254	255

Similarly suppose that KSA stops after the step a = 3, and we analyze the RC4 keystream algorithm: PRGA. Based on the array T just after the step a = 3, we have

$a = a + 1 = 0 + 1 = 1$
$b = b + T_1 = 0 + 2 = 2$
$t = T_a + T_b = T_1 + T_2 = 2 + 1 = 3$
keystream = $T_t = T_3 = 5 + K_3$

Hence we can recover K_3 by computing K_3= keystream - 5 (mod 256), where keystream is the first byte of the key stream.

However the KSA does not stop at the step a = 3. Now we need to estimate the successs probability of our cryptanalysis. Since the first keystream only depends on T_1, T_2 , T_3. If elements at positions 1, 2 and 3 are not interchanged in the rest of KSA steps, the proposed cryptanalysis works. For the rest of KSA steps, we have a = 4, 5,..., 255, so index a will not affect elements at positions 1, 2 and 3. Based on the computing step of index b in KSA: $b = b + T_a + K_a$, we may suppose that index b is chosen at random. For every step, the probability that index $j \notin \{1, 2, 3\}$ is $\frac{253}{256}$. Since there are 252 residual steps, the probability that positions 1, 2 and 3 are not altered by index b after the step a=3 is $\left(\frac{253}{256}\right)^{252} \approx 0.0513$. In a word, the attacker can recover K_3 based on a data packet of IVs of pattern (*v, 257-v, 255*) with approximate probability 0.0513.

There are 256 IVs of previous pattern (3, 255, v) and 256 IVs of newly discovered pattern (v, 257-v, 255). So our findings combined with the previous findings in [1] improve the proportion of IVs that can be used in the FMS attack from $\frac{1}{256^2}$ to $\frac{2}{256^2}$.

4. Conclusions

In this paper, we demonstrate that IVs of new pattern *(v, 257-v, 255)* can be used in the FMS attack to recover the first byte of the long-term key. Our findings combined with the IVs of previously known pattern *(3, 255, v)* increase the proportion of IVs that can be exploited in the FMS attack.

Acknowledgement

This work was supported by Research Funds for NSD Construction, University of International Relations, grant No. 2019GA34 and supported by Fundamental Research Funds for the Central Universities, University of International Relations, grant No. 3262018T02.

References

1. Dierks T, Allen C. The TLS protocol, version 1.0, Internet Engineering Task Force, January 1999.
2. Stamp M, Low R. *Applied cryptanalysis - breaking ciphers in the real world*. New York:Wiley; 2007.
3. Fluhrer S, Mantin I, Shamir A. Weaknesses in the key scheduling algorithm of RC4 In: S. Vaudenay and A. Youssef, editors. SAC 2001, Springer-Verlag Berlin, Lecture Notes in Computer Science, Vol. 2259, 1–24.
4. Ohigashi T, Shiraishi Y, Morii M. New weakness in the key-scheduling algorithm of RC4. article. *IEICE TRANS.FUNDAMENTALS 2008*; **24E91-A(1)**:3-11.
5. Vaudenay S, Vuagnoux M. Passive–only key recovery attacks on RC4. In: C. Adams, A. Miri, and M. Wiener, editors. SAC 2007, Springer-Verlag Berlin, Lecture Notes in Computer Science, Vol. 4876, 344–359.
6. Paul S, Preneel B. A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In: Roy B, Meier W, editors. FSE 2004, Springer-Verlag, Lecture Notes in Computer Science, Vol. 3017, 245–259.
7. Tomašević V, Bojanić S, Nieto-Taladriz O. Finding an internal state of RC4 stream cipher. article. *Information Sciences 2007*; **177(7)**: 1715-1727.
8. Stošić L, Bogdanović M. RC4 stream cipher and possible attacks on WEP. article. *International Journal of Advanced Computer Science and Applications 2012*; **3(3)**:110-114.
9. Paul R, Mitra J, Dey H, Sau S, Baidya P, Ghosh R, Chakrabrti A. Secure multi-gigabit optical link design for high energy physics experiment with acceleration of more secure RC4 variant in reconfigurable platform, article. *Journal of Instrumentation 2020*; **15(10)**: 10024-10028