2007－ＣＳＥＣ－38（32）
2007／7／20

Tews らによる WEP に対する鍵回復攻撃に関する考察

小篠　裕子[†]　藤川　香顕[‡]　大東　俊博[‡]　桑門　秀典[†]　森井　昌克[†]

† 神戸大学大学院工学研究科
〒657-8501 兵庫県神戸市灘区六甲台町 1-1
‡ 神戸大学大学院自然科学研究科
〒657-8501 兵庫県神戸市灘区六甲台町 1-1
E-mail: † ozasa@stu.kobe-u.ac.jp, {kuwakado,mmorii}@kobe-u.ac.jp,
‡ {y.fujikwa,ohigashi}@stu.kobe-u.ac.jp

あらまし　2007 年，Tews らは Klein による WEP に対する鍵回復攻撃を最適化した手法を提案し，40,000 パケットの観測によって 50%，85,000 パケットの観測によって 95%の秘密鍵復元を可能とした．本稿では，Tews らの攻撃を考察し，より少ないパケットの観測によって効率よく秘密鍵の情報を得られる手法を提案する．

キーワード　WEP，鍵回復攻撃，RC4

A Study on the Tews-Weinmann-Pyshkin Attack against WEP

Yuko OZASA[†], Yoshiaki FUJIKAWA[‡], Toshihiro OHIGASHI[‡],
Hidenori KUWAKADO[†], and Masakatu MORII[†]

† Graduate School of Engineering, Kobe University,
1-1Rokkodai, Nada-ku, Kobe-shi, Hyogo, 657-8501 Japan.
‡ Guraduate School of Science and Technology, Kobe University,
1-1Rokkodai, Nada-ku, Kobe-shi, Hyogo, 657-8501 Japan.
E-mail: † ozasa@stu.kobe-u.ac.jp, {kuwakado,mmorii}@kobe-u.ac.jp,
‡ {y.fujikwa,ohigashi}@stu.kobe-u.ac.jp

Abstract　Tews, Weinmann, and Phshkin have shown a key recovery attack against WEP (the TWP attack), which is the modification of Klein's key recovery attack against RC4. The TWP attack allows an attacker to recover a 104-bit secret key from 40,000 captured packets with probability 0.5 and from 85,000 captured packets with probability 0.95. In this paper, we improve the probability of recovering the key when the number of captured packets is less, efficiently. For example, if 10,000 packets are captured, then the probability of recovering it is three times as large as than that of the TWP attack. If 20,000 packets are done, then it is twice as large as than that of the TWP attack. Our attack can find the secret key in a few seconds.

Keyword　WEP, key recovery attack, RC4

1. Introduction

WEP[1] is a protocol for securing wireless LANs (WLANs). WEP stands for "Wired Equivalent Privacy" which means it should provide the level of protection a wired LAN has. WEP uses the RC4 stream cipher[2] to encrypt data which is transmitted over the air, using usually a single secret key (called a WEP key) of a length of 40 or 104 bits. A 104-bit WEP key is used with a 24-bit IV. The IV is a public value changed in each packet, and it is used for a part of the session key.

In 2001 Fluhrer, Mantin, and Shamir[3] published an analysis of the RC4. Some time later, it was shown that this attack can be applied to WEP and the secret key can be recovered from about 4,000,000 to 6,000,000 captured packets. In 2004 a hacker named KoreK[4] improved the attack: the complexity of recovering a 104-bit secret key was reduced to 500,000 to 2,000,000 captured packets. In 2005, Klein[5] presented another analysis of the RC4. Klein showed that there are more correlations between the RC4 keystream and the secret key than the ones found by Fluhrer, Mantin, and Shamir which can additionally be used to break WEP in WEP like usage modes.

Tews, Weinmann, and Pyshkin[6] extened Klein's attack and optimized it for WEP. We call this attack the TWP attack. Using the TWP attack, it is possible to recover a 104-bit WEP key with probability 50% using just 40,000 captured packets. For 60,000 captured data packets, the success probability is about 80% and for 85,000 data packets about 95%. The same attack can be used for 40-bit keys too with higher success probability.

We extend the TWP attack. We focus on the attack in small amount of packets, about 10,000 captured packets. Using our attack, it is possible to recover a 104-bit WEP key three times the number using 10,000 captured packets and twice the number using 20,000 captured packets than the TWP attack.

The structure of the paper is as follows: In Section 2 we explain WEP, in Section 3 we introduce the notation, in Section 4 we present a summary of the TWP attack against WEP, in Section 5 we specialize the TWP Attack to WEP, and describes extension of the attack, in Section 6 we gives experimental results.

2. WEP

2.1. Generation of the session key

In WEP, a secret key ***K'*** is pre-shared between an access point and a mobile node. The length of ***K'*** is ether 40-bit or 104-bit. A session key ***K*** is generated according to ***K***=***IV***||***K'***, where ***IV*** is 24-bit IV and || is concatenation The IV is transmitted plain text and changed for every packet. In this paper, we discuss the case the length of ***K'*** is 104 bits.

The encryption of WEP is done by the RC4 algorithm with ***K***. So, we introduce the RC4 algorithm in next session to explain how the RC4 algorithm is used in WEP.

2.2. RC4

We follow the description of RC4 as given in [2]. RC4 comprises the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). The initial state is made by the session key *K* in the KSA. A keystream is generated from the initial state in the PRGA. The plaintext is XOR-ed with the keystream to obtain the ciphertext.

2.2.1. Key scheduling algorithm

In the KSA the initial state of RC4 at time t consists of a permutation table $S_t=S_t[x]$ where $x=0,1,\ldots 256$, S_t is initialized $S_0[x]=x$. Two 8-bit word pointers i_t and j_t at time t are used, and these are initialized $i_0=j_0=0$. In the KSA, following Eqs. (1), (2), and (3) is executed at time $t=1,2,\ldots 256$.

$$j_t=(j_{t-1}+S_{t-1}[i_{t-1}]+K[i_{t-1} \bmod 16]) \bmod 256, \tag{1}$$

$$S_t[i]=\begin{cases} S_{t-1}[j_t] & i=i_{t-1}, \\ S_{t-1}[i_{t-1}] & i=j_t, \\ S_{t-1}[i\ ] & i\neq i_{t-1}, j_t, \end{cases} \tag{2}$$

$$i_t=(i_{t-1}+1) \bmod 256. \tag{3}$$

In WEP, $K[0]$, $K[1]$ and $K[2]$ are the IV entries and $K[y](y=0,1,\ldots,15)$ are the session keys.

2.2.2. Pseudo random generation algorithm

The initial state of RC4 at time t in the PRGA consists of permutation $S_t'=S_t'[x]$ $(x=0,1,\ldots 255)$. S_0' is initialized to $S_0'[x]=S_{256}[x]$. Two 8-bit word pointers i_t' and j_t' at time t are used, and these are initialized $i_0'=j_0'=0$. Let Z_t denote the output 8-bit word of RC4 at time t. Then, the next state and output functions of RC4 for every t are defined as follow:

$$i_t'=(i_{t-1}'+1) \bmod 256, \tag{4}$$

$$j_t'=(j_{t-1}'+S_{t-1}'[i_t']) \bmod 256, \tag{5}$$

$$S_t'[i]=\begin{cases} S_{t-1}'[j_t'] & i=i_t', \\ S_{t-1}'[i_t'] & i=j_t', \\ S_{t-1}'[i_t] & i\neq i_t', j_t', \end{cases} \tag{6}$$

$$Z_t=S_t'[(S_t'[i_t']+S_t'[j_t']) \bmod 256]. \tag{7}$$

3. The TWP attack against WEP

Tews, Weinmann, and Pyshkin extend Klein's attack against RC4 and optimize it for usage against WEP. This attack is straightforward to Klein's attack. First we introduce Klein's attack, and next we introduce how they extended Klein's attack against WEP in this section.

3.1. Klein's attack against WEP

Suppose w key streams were generated by RC4 using packet keys with a fixed root key and different initialization vectors. Denote by $K_u = (K_u[0],...,K_u[m]) = (IV|R_k)$ the u-th packet key and by $Z_u = (Z_u[0],...,Z_u[m - l])$ the first m bytes of the u-th key stream, where $1 \leq u \leq w$. Assume that an attacker knows the pairs (IV_u, Z_u), we shall refer to them as samples , and tries to find R_k.

If the first i bytes of a packet key are known, then the internal permutation S_{i-1} and the index j at the $(i - 1)$th step of the RC4 key setup algorithm can be found. We have

$$\begin{aligned} K[i] &= (j_{i+1} - j_i - S[i]) \bmod 256, \\ j_i &= (i - Z_i) \bmod 256. \end{aligned} \tag{8}$$

3.2. Extension to multiple key bytes

With Klein's attack, it is possible to iteratively compute all secret key bytes if enough samples are available. This iterative approach has a significant disadvantage: In this case the key streams and IVs need to be saved and processed for every key byte. Additionally correcting falsely guessed key byte is expensive, because the computations for all key bytes following $K[i]$ needs to be repeated if $K[i]$ was incorrect. They extend the attack such that is it possible to compute key bytes independently of each other and thus make efficient use of the attack possible by using key ranking techniques. Klein's attack is based on the the fact that Eq. (8) shows.

j_{i+1} may be written as $j_i + S_i[i] + K[i]$. By replacing j_{i+1} in Eq. (8), there is an approximation for $K[i] + K[i + 1]$:

$$K[i] + K[i+1] = j_{i+2} - j_i - S[i] - S[i+1]. \tag{9}$$

By repeatedly replacing j_{i+k}, there is an approximation for $K[3]+K[4]+...+K[3+i]$. Because we are mostly interested in $K[3]+K[4]+...+K[3+i]=R_k[0]+R_k[1]+...+R_k[i]$ in a WEP scenario, we will use the symbol A_i for this sum. A_i depends on the key bytes $K[3]$ to $K[i - 1]$.By replacing them with S_3, there is an approximation of A_i , which only depends on $K[0]$ to $K[2]$.

$$A_i = K[3] + K[4] + ... + K[i] = j_{i+1} - j_3 - \sum_{l=3}^{i} S_3[l]. \tag{10}$$

3.3. Key ranking

If only a low number of samples is available, the correct value for A_i is not always the most voted one in the table but tends to be one of the most voted. Instead of collecting more samples, they use another method for finding the correct key. It is got from just generating a key stream using an IV and a guessed key, and comparing it with the collected one. If the method used for key stream recovery did not always guess the key stream right, the correct value just needs to match a certain fraction of some key streams. For every key byte $K[i]$, they define a set M_i of possible values A_i might have. At the beginning, M_i is only initialized with the top voted value for A_i from the table. Until the correct key is found, they look for an entry for A_i in all tables having a minimum distance to the top voted entry in table i. They then add A_i to M_i and test all keys which can now be constructed from the sets M that have not been tested previously.

3.4. Handling strong keys

For Eq. (10), S_3 is assumed to be a approximation of S_{3+i}. This assumption is wrong for a fraction of the keyspace. They call these keys strong keys. For these keys, the value for j_{l+3} is most times taken by j in a iteration before i + 3 and after 3. This results in $S[j_l+3]$ being swapped with an unknown value, depending on the previous key bytes and the IV. In iteration i + 3, this value instead of $S_3[j_l+3]$ is now swapped with $S[i]$. More formally, let Rk be a key and $R_k[i]$ a key byte of R_k. $R_k[i]$ is a strong key byte, if there is an integer $l = \{1,...,i\}$ where

$$\sum_{k=l}^{i} (Rk[k] + 3 + k) \bmod 256 = 0. \tag{11}$$

A key R_k is a strong key, if at least one of its key bytes is a strong key byte. On the contrary, key bytes that are not strong key bytes are called normal key bytes and keys in which not a single strong key byte occurs are called normal keys. Assuming that S is still the identity permutation, the value 0 will be added to j_l+3 from iteration l + 3 to i + 3, making j_l+3 taking his previous value j_{l+3}. This results in the probability of that A_i has a correct value close to 0 and Prob(A_i takes correct value) is very close to $1/n$. An alternative way must be used to determine the correct value for this key byte.

The approach can be devided into two steps:

1. Find out which key bytes are strong key bytes by

using the Eq. (11).

2. Find the correct values for these key bytes.

Assuming that $Rk[i]$ is a strong key byte and all values for $Rk[0],\dots,Rk[i-1]$ are known, $Rk[i]$ is calculated from the following equation , which can made from Eq. (11).

$$Rk[i]=[-3-i-\sum_{k=l}^{i}(Rk[k]+3+k)]\bmod 256. \quad (12)$$

Because there at most i possible values for l, it is possible to try every possible value for l and restrict $Rk[i]$ to at most i possible values. Instead of taking possible values for A_l from the top voted value in the table for key byte i, the table can be ignored. The values calculated with Eq. (12) for $Rk[i]$ can be used and it is possible to assume that $A_{l-1}+Rk[i]$ was top voted in the table. Possible values for A_l for all assumed to be normal key bytes are still taken from the top voted values in their tables.

4. Our attack on WEP

The key recovery of the TWP attack is much effective at the attack using over 40,000 captured packets, but using under 40,000 captured packets, its performance falls considerably. Usually, the performance of the key recovery attack using several thousands to 10,000 of captured packets is interested. So we focuse on the key recovery attack using about several thousands of packets, extend the TWP attack more effectively and improved the performance of the attack.

The basic attack is straightforward. Initially, we extended the TWP attack described in Section 3.2 and 3.3. We may recover a 104-bit WEP key by using half amount of captured packets Tews-Weinmann-Pyshkin used in their attack.

4.1. Extention to more key bytes

In the TWP attack, the only information of the keystream in 16 bytes from head of a packet is used. We propose an attack which we can use the information of the keystream in over 17 bytes from head of a packet. Using our attack, we can recover a WEP key using less number of captured packets. Concretely we enhanced the value where A_l can be taken $A_{15},A_{16},\dots,A_{24}$. This method may increase the probability for the event that A_l calculated in Eq. (10) takes correct value twice as that of the TWP attack.

Calculating $A_{16},A_{17},\dots,A_{24}$ by using Eq. (10), we can obtain A_{15} from them. Then, add this A_{15} to the vote for key ranking described in Section 3.3, we may obtain almost twice the data packets Tews, Weinmann and Pyshkin can obtain. Our approach can be divided in two cases:

Case 1. Using the value of A_i (i = 16, 17, 18)

The value A_{15} can be obtained from A_{16} to A_{18} by calculating from Eq. (10) and the following equations:

$$K[16]=K[0],\ K[17]=K[1],\ K[18]=K[2]. \quad (13)$$

Where IV:($K[0],K[1],K[2]$), too. So we have

$$A_{15}=\begin{cases} A_{16}-K[16]=A_{16}-K[0] & (i=16), \\ A_{17}-K[16]-K[17]=A_{17}-K[0]-K[1] & (i=17), \\ A_{18}-K[16]-K[17]-K[18]=A_{18}-K[0]-K[1]-K[2] & (i=18). \end{cases} \quad (14)$$

We add these values of A_{15} to the vote for key ranking described in Section 3.3, and then we recover a 104 bit WEP key at the same way to the TWP key recovery attack. Using this method, we may use the packets four times more than the TWP attack.

Case 2. Using the value of A_i (i = 19, 20,..., 24)

When A_{15} is obtained, we can obtain $A_3, A_4,\dots,A_8$ from calculation of $A_{19},A_{20},\dots,A_{24}$ by calculating from Eqs.(8) and (13). For example, we have

$$\begin{aligned} A_3 &= A_{19}-A_{15}-K[16]-K[17]-K[18] \\ &= A_{19}-A_{15}-K[0]-K[1]-K[2] \quad (i=19), \end{aligned} \quad (15)$$

$$\begin{aligned} A_4 &= A_{20}-A_{15}-K[16]-K[17]-K[18] \\ &= A_{20}-A_{15}-K[0]-K[1]-K[2] \quad (i=20). \end{aligned} \quad (16)$$

A_{15} is able to obtain by the Case 1 attack, but when the number of captured packets is smaller, the probability of obtaining correct A_{15} in Case 1 attack becomes smaller. And more, A_{15} is able to obtain by examining all patterns for 2^8 candidates of A_{15} too, although the probability of that A_{15} takes correct value is larger, this examining costs more time to obtain A_{15}. When we obtained A_{15}, then we recover a 104-bits WEP key by the same attack to the TWP attack described in Section 4. Using the Case 2 attack, we may use the packets four times more than the TWP attack. Though we can use over A_{25} to A_{33} in the same way, the probability of recovering the WEP key is lower in use when we have experimented.

5. Experimental results

We wrote an implementation using the parallelized computation described in Section 3.2 and 4, and the error correction methods described in Section 3.3 and 3.4. To compare the performance between TWP and our attack, we implement both attacks and perform the simulation experiments. We focus on the attack in small data packets, so we define the number of packets:10,000, 20,000, 40,000, and 80,000 in the experiments.

Table 1 and 2 show the experimental results of the TWP attack and our attack when the number of test is 100,000, 1,000,000. It is clear that we improve the probability of recovering a 104-bit WEP key when the number of captured packets is under 40,000. According to Table 1 and 2, we confirmed that the probability of recovering the WEP key using our attack is improved.

At the result of the Case 1 attack when 10,000 packets are captured, the probability is twice as large as than that of the TWP attack. At the same situation, the Case 2 attack is more effective than the TWP attack and the Case 1 attack. Especially, the probability of recovering a 104-bit WEP key by the Case 2 attack is three times as large as than that by the TWP attack. At the results of these experiments, we can confirm that our attack is more effective than the TWP attack when the captured packets is less.

Table.1. Experimental results when the number of tests is 100,000.

Number of captured packets	The TWP attack Number of recovered key	Our attack(Case 1) Number of recovered key	Our attack(Case 2) Number of recovered key
20,000	4,440	8,740	9,830
40,000	30,163	38,780	39,483
80,000	93,172	95,653	96,433

Table.2. Experimental results when the number of tests is 1,000,000.

Number of captured packets	The TWP attack Number of recovered key	Our attack(Case 1) Number of recovered key	Our attack(Case 2) Number of recovered key
10,000	5	12	15

6. Conclusion

We focused on the key recovery attack using about several thousands of packets, and extend the TWP attack more effectively and improved the performance of the attack using the smaller amount of packets. Using our attack, it is possible to recover a 104 bit WEP key three times the number using 10,000 captured packets and twice the number using 20,000 captured packets than the TWP attack. Our attack against WEP is much effective to the key recovery attack using the smaller amount of packets than the TWP attack is.

References

[1] IEEE Computer Society, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications", IEEE Std 802.11, 1999.

[2] B. Schneier, Applied Cryptography, Wiley, New York, 1996

[3] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4", Proc. SAC2001, Lecture Notes in Computer Science, vol.2259, pp.1-24, Springer-Verlag, 2001.

[4] KoreK, http://www.netstumbler.org/showthread.php?t=12489.

[5] A. Klein, "Attacks on the RC4 stream cipher. submitted to Designs", Codes and Cryptography, 2007.

[6] E. Tews, R. Weinmann, and A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds", Cryptology ePrint, available at http://eprint.iacr.org/2007/120.pdf