WEP Weak IVs Revisited

Kazukuni Kobara and Hideki Imai
IIS, Univ. of Tokyo
RCIS, AIST
Outline

- Available options for securing WLAN access
- WEP and its key recovery attack
- Condition to recover the WEP key
- Good and bad strategies to trace the condition back to the patterns of IVs and WEP keys
- Conclusion
Available Options for Securing WLAN Access

- Channel Protection (& Authentication)
  - AES-CCM
  - TKIP
  - (Weak-IV skipping WEP)
  - WEP

- Filtering
  - Filtering with MAC address

- (Authentication &)
  - Key-Establishment
  - EAP-TLS
  - EAP-TTLS, PEAP
  - EAP-MD5, LEAP
  - PSK
**Current Status**

- **AES-CCM**
  - Fully investigated and no serious attack has been identified

- **TKIP**
  - Insecure even against casual attacks

- **(Weak-IV skipping) WEP**
  - Not fully investigated

- **(Conventional) WEP**
  - Insecure even against casual attacks

- **Filtering with MAC address**

**Advantage:**
- Compatible with WEP
- Old WLAN cards and APs may support easily

**Disadvantage:**
- Old WLAN cards and APs cannot support them
WEP: Wired Equivalent Privacy

A specification for securing wireless access, especially of 802.11

Note: WEP (as well as TKIP and AES-CCM) give protection only for wireless part, but not for the wired part.
History of battles over WEP

This work: reviews the attacks and identifies more advanced patterns of IVs and WEP keys to skip

2001~: New specs, TKIP and AES (Not interoperable with WEP)

2001~: Some chip makers started skipping certain IVs, but this is still incomplete

1999: WEP was standardized

2001: The key recovery attack was identified by FMS, and then implemented

Keys can be recovered

Cracking tools are being improved

Prevention

Attack
WEP:
Wired Equivalent Privacy

IV, (m||CRC(m)) + RC4(IV||K')

Pre-Shared Key: K'

IV: Initial Value  m: message
+: exclusive-or  ||: concatenation
WEP:
Wired Equivalent Privacy

IV, \((m||CRC(m)) + RC4(IV||K')\)

Integrity check
Encryption with RC4 key stream

+: exclusive-or
RC4 Stream Cipher

**Key (seed)**

- **K**

**Key stream (pseudo random sequence)**

- RC4(K)
  - 011010010111

**Message**

- m

**Ciphertext**

- c
KSA: Key Scheduling Algorithm
PRGA: Pseudo Random Generator Algorithm

**RC4**

256 byte buffer for $n=8$

KSA: shuffles it byte wise according to the key

PRGA: outputs key stream while swapping the buffer

Key stream (pseudo random sequence)

KSA: Key Scheduling Algorithm
PRGA: Pseudo Random Generator Algorithm
**Input:** byte size $n$, a $l$ byte key $K$

**Output:** a $2^n$ byte buffer $S$

$$S := (0, 1, \cdots, 2^n - 1)$$
$$j := 0$$

For($i = 0; i < 2^n; i + +)${
$$j := j + S[i] + K[i \mod l] \mod 2^n$$
Swap $S[i]$ and $S[j]$
}

Return $S$
Input: output sequence size $s$ and a $2^n$ byte buffer $S$
Output: output sequence $Z$

\[
\begin{align*}
    j &:= 0 \\
    \text{For}(i' = 1; i' \leq s; i' + +)\{ \\
        i &:= i' \mod 2^n \\
        j &:= j + S[i] \mod 2^n \\
        \text{Swap } S[i] \text{ and } S[j] \\
        Z[i' - 1] &:= S[S[i] + S[j] \mod 2^n] \\
        \text{Return } Z[i' - 1] \} \\
\end{align*}
\]
The KSA (Key Scheduling Algorithm) is a process that takes in an IV (Initial Vector) and a key, and outputs a shuffled buffer. The algorithm is iterative, with each iteration involving the following steps:

1. For each iteration, $i = 0, 1, 2, \ldots, l-1$, the value of $j$ is calculated as $j = j_{i-1} + S_i[i] + K[i \mod l]$, where $S_i[i]$ is a permutation function and $K[i \mod l]$ is a key element.

2. The value of $j$ is then used to swap elements in the shuffled buffer. For example, in the initial state, $j=0$, the buffer is:

   0 1 2 3 4 5

   After $j=0$, the buffer becomes:

   5 1 2 3 4 0

   and this process continues for subsequent iterations.

The final shuffled buffer is used as input for the encryption process. The key elements $K[4]$ and $K[l-1]$ are also used in the algorithm, as shown in the diagram.
i=1  

\[ j = 0 \]  

\[ i = 1 \]  

\[ j = 4 \]  

\( j_i = j_{i-1} + S_i[i] \)

\[ S_i[i] + S_i[j_i] \]

\( i = 2 \)  

\[ j = 5 \]  

\( i = 3 \)  

\[ j = 4 \]  

output sequence
Gap between WEP and others

WEP

known \quad \text{unknown}

IV, \text{RC4( } IV \text{ || key } \text{ )}

Key is recoverable

While the gap might be small, it made a big difference!!

[FMS01][SIR01]

SSL/TLS etc

unknown

\text{RC4( } \text{key } \text{ )}

key is not recoverable
Idea of Key Recovery Attack

For certain IVs called "Weak IVs" the correlation between the first output byte and one byte of the key becomes higher than the average 1/256=0.004.

Typical prob. is 0.05
The famous weak IVs identified by FMS

IV

WEP key

17


\[ t = 3 \text{ to } 15 \]

\[ t: \text{ target key byte to crack} \]
Notations

- Known byte
- Known and untouchable byte (should not be referred to by index $j_i$ for $i > t'$)
- Target byte (which depends on $K[t]$ and should not be referred to by $j_i$ for $i > t'$ except $i=t$)
- Unknown byte

$t' : (\# \text{ of known bytes in } K[]) - 1$
KSA PRGA depends on $K[3]$

IV WEP key

$t=3$

$$Pr = \frac{1-2}{256} \times \left(1-\frac{3}{256}\right)^{256-4} = 0.05$$
Relationship Among Weak IVs

Some of the current chips skip a little wider area

Current WEP cracking tools collect more wide area using general condition

Famous weak IVs

\((IV[0],IV[1],IV[2])=(t,255,*)\)

\(0 \leq S[1] \leq t'\) and 
\(S[1]+S[S[1]]=t\)

\((IV[0],IV[1],IV[2])=?\)

Convert the condition into the patterns of IVs and WEP keys so that the more advanced patterns to skip can be identified.
The difficult part

- S[ ] depends not only on IVs, but also on WEP keys, \( k[3] \) to \( k[t'] \)
  - i.e. by exhaustive searching \( k[3] \) to \( k[t'] \), a lot of key-dependent weak IVs are available
  - (and skipping key-dependent weak IVs only is not enough!!)

- Listing up all the combinations of IVs and WEP keys with exhaustive search is computationally infeasible

Note: \((k[0], k[1], k[2]) = (IV[0], IV[1], IV[2])\)
Another Naive Approach

- Skip IVs meeting the condition but only for the currently set WEP key
  - This is feasible, but

- This causes another vulnerability
  - the information on the WEP key is revealed from the skipped patterns
  - since most of the weak IVs depend on the WEP key
We took the approach

- to trace the condition back to the patterns of IVs and WEP keys theoretically

- We are now summarizing the results and will open them soon
Our Contribution

Security level

- Secure against WEP cracking tools
- Insecure against WEP cracking tools

This work

- More advanced versions of weak-IV-skipping WEP
- Current versions of weak-IV-skipping WEP
- Original WEP (no IV skip)