RC4
RC4

- Invented by Ron Rivest
  - “RC” is “Ron’s Code” or “Rivest Cipher”
- A stream cipher
- Generate keystream `byte` at a step
  - Efficient in software
  - Simple and elegant
  - Diffie: RC4 is “too good to be true”
- Used lots of places: SSL, WEP, etc., etc.
- Most popular stream cipher in existence
**RC4 Initialization**

- **Array** `key` contains `N` bytes of key
- **Array** `S` always has a permutation of 0,1,...,255

```plaintext
for i = 0 to 255
    S[i] = i
    K[i] = key[i (mod N)]
next i
j = 0
for i = 0 to 255
    j = (j + S[i] + K[i]) (mod 256)
    swap(S[i],S[j])
next i
i = j = 0
```
For each keystream byte, swap elements of array $S$ and select a byte from the array:

\[
i = (i + 1) \mod 256
\]
\[
j = (j + S[i]) \mod 256
\]
\[
\text{swap}(S[i], S[j])
\]
\[
t = (S[i] + S[j]) \mod 256
\]
\[
\text{keystreamByte} = S[t]
\]

Use keystream bytes like a one-time pad
- XOR to encrypt or decrypt
WEP

- WEP == Wired Equivalent Privacy
- The stated goal of WEP is to make wireless LAN as secure as a wired LAN
- According to Tanenbaum:
  - “The 802.11 standard prescribes a data link-level security protocol called WEP (Wired Equivalent Privacy), which is designed to make the security of a wireless LAN as good as that of a wired LAN. Since the default for a wired LAN is no security at all, this goal is easy to achieve, and WEP achieves it as we shall see.”
WEP

- Wired Equivalent Privacy
- WEP uses RC4 for confidentiality
  - Considered a strong cipher
  - But WEP introduces a subtle flaw
- WEP uses CRC for "integrity"
  - Should have used a crypto hash instead
  - CRC is for error detection, not cryptographic integrity
WEP Integrity Problems

- WEP “integrity” does not provide integrity
  - CRC is linear, so is stream cipher XOR
  - Can change ciphertext and CRC so that checksum remains correct
  - Such introduced errors go undetected
  - This requires no knowledge of the plaintext!
  - Even worse if plaintext is known

- CRC does not provide a cryptographic integrity check!
  - CRC designed to detect random errors
  - Not designed to detect intelligent changes
WEP Key

- WEP uses a long-term secret key: K
- RC4 is a stream cipher, so each packet must be encrypted using a different key
  - Initialization Vector (IV) sent with packet
  - Sent in the clear (IV is **not** secret)
  - IV has similar purpose as “MI” in WWII ciphers
- Actual RC4 key for packet is (IV,K)
  - That is, IV is **pre-pended** to K
 Initialization Vector “Issue”

- WEP uses 24-bit (3 byte) IV
  - Each packet gets a new IV
  - RC4 packet key: IV pre-pended to long-term key, K
- Long term key K seldom changes
- If long-term key and IV are same, then same keystream is used
  - This is bad!
  - It is at least as bad as reuse of one-time pad
Initialization Vector “Issue”

- Assume 1500 byte packets, 11 Mbps link
- Suppose IVs generated in sequence
  - Then $1500 \cdot \frac{8}{(11 \cdot 10^6)} \cdot 2^{24} = 18,000$ seconds
  - Implies IV must repeat in about 5 hours
- Suppose IVs generated at random
  - By birthday problem, some IV repeats in seconds
- Again, repeated IV (with same $K$) is bad!
WEP Active Attacks

- WEP: “Swiss cheese” of security protocols
- If Trudy can insert traffic and observe corresponding ciphertext
  - Then she will know keystream for that IV
  - And she can decrypt next msg that uses that IV
- Spse Trudy knows destination IP address
  - She can change IP address in ciphertext
  - And modify CRC so it is correct
  - Then access point will decrypt and forward packet to the Trudy’s selected IP address!
  - Requires no knowledge of the key $K$!
WEP Cryptanalytic Attack

- WEP data encrypted using RC4
  - Packet key is IV and long-term key K
  - 3-byte IV is pre-pended to K
  - Packet key is (IV,K)
- IV is sent in the clear (not secret)
  - New IV sent with every packet
  - Long-term key K seldom changes (maybe never)
- Assume Trudy knows IVs and ciphertext
- Trudy wants to find the key K
RC4 in WEP

- 3-byte IV pre-pended to key
- We denote the RC4 key *bytes*...
  - ...as $K_0, K_1, K_2, K_3, K_4, K_5, ...$
  - Where IV = $(K_0, K_1, K_2)$, which Trudy knows
  - Trudy wants to find $K_3, K_4, K_5, ...$
- Given enough IVs, we show that Trudy can recover the long-term key
  - Regardless of the length of the key!
  - Provided Trudy knows first keystream byte
  - Known plaintext attack (1st byte of each packet)
Recall that RC4 initialization is...

\[ S_i = i \text{ for } i = 0,1,2,...,255 \]

\[ j = 0 \]

\[ \text{for } i = 0 \text{ to } 255 \]

\[ j = j + S_i + K_i \]

\[ \text{swap}(S_i, S_j) \]

\[ \text{next } i \]
RC4/WEP Attack

- Attack due to Fluher, Mantin and Shamir
- Trudy watches IVs until she sees 3-byte IV of the form: $IV = (3, 255, V)$
- Where $V$ can be anything (Trudy knows $V$)
- Then RC4 key for this packet is $key = (3, 255, V, K_3, K_4, K_5, ...)$
- Trudy wants to find $(K_3, K_4, K_5, ...)$
RC4/WEP Attack

- $IV = (3, 255, V)$
- $Key = (3, 255, V, K_3, K_4, \ldots)$
- Trudy knows $K_0 = 3, K_1 = 255, K_2 = V$
- Other $K_i$ are long-term key
  - Which is unknown to Trudy
- Recall RC4 initialization: first, set $S$ to...

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
RC4/WEP Attack

- IV = (3, 255, V)
- Key = (3, 255, V, K_3, K_4, ...)
- RC4 initialization: let j = 0 then
  
  for i = 0 to 255
  
  j = j + S_i + K_i
  
  swap(S_i, S_j)
  
  next i

- At i = 0 step we have
  
  i = 0
  
  j = j + S_0 + K_0 = 0 + 0 + 3 = 3
  
  swap(S_i, S_j) = swap(S_0, S_3)
RC4/WEP Attack

- From previous slide...
- At $i = 0$ step we have
  
  $i = 0$
  
  $j = j + S_0 + K_0 = 0 + 0 + 3 = 3$
  
  $\text{swap}(S_i, S_j) = \text{swap}(S_0, S_3)$

- After this step, the table $S$ is...

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>...</td>
</tr>
</tbody>
</table>
RC4/WEP Attack

- **IV** = (3, 255, V)
- **Key** = (3, 255, V, K_3, K_4, ...)

**Continuing, at** \( i = 1 \) **step**

\[
i = 1 \\
\text{j} = \text{j} + S_1 + K_1 = 3 + 1 + 255 = 3 \pmod{256} \\
\text{swap}\left(S_i, S_j\right)
\]

- **After this step, the table** \( S \) **is**:

<table>
<thead>
<tr>
<th>( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_i )</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>...</td>
</tr>
</tbody>
</table>
RC4/WEP Attack

- IV = (3, 255, V)
- Key = (3, 255, V, K_3, K_4, ...)

- Continuing, at \( i = 2 \) step
  - \( i = 2 \)
  - \( j = j + S_2 + K_2 = 3 + 2 + V = 5 + V \)
  - \( \text{swap}(S_i, S_j) \)

- After this step, the table \( S \) is...

<table>
<thead>
<tr>
<th>( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>5 + V</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_i )</td>
<td>3</td>
<td>0</td>
<td>5 + V</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>...</td>
<td>2</td>
<td>...</td>
</tr>
</tbody>
</table>
RC4/WEP Attack

- $\text{IV} = (3, 255, V)$
- $\text{Key} = (3, 255, V, K_3, K_4, \ldots)$

- Continuing, at $i = 3$ step
  
  $i = 3$
  
  $j = j + S_3 + K_3 = 5 + V + 1 + K_3 = 6 + V + K_3$
  
  $\text{swap}(S_i, S_j)$

- Assuming $6 + V + K_3 > 5 + V \pmod{256}$, the table is

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>$5 + V$</th>
<th>\ldots</th>
<th>$6 + V + K_3$</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>3</td>
<td>0</td>
<td>$5 + V$</td>
<td>$6 + V + K_3$</td>
<td>4</td>
<td>5</td>
<td>\ldots</td>
<td>2</td>
<td>\ldots</td>
<td>1</td>
</tr>
</tbody>
</table>

- Otherwise $6 + V + K_3$ will be to the left of $5 + V$
RC4 Initialization

- Note that we have only considered the first 4 steps of initialization, $i = 0, 1, 2, 3$
  - In reality, there are 256 steps
- For now, assume that initialization stops after $i = 3$ step
- Then $S$ is

| $i$  | 0   | 1   | 2   | 3   | 4   | 5   | $5 + V$ | $6 + V + K_3$ | 4   | 5   | $6 + V + K_3$ | \ldots |
|------|-----|-----|-----|-----|-----|-----|---------|-------------|-----|-----|-------------| \ldots |
| $S_i$| 3   | 0   | 5 + V | 6 + V + K_3 | 4   | 5   | $5 + V$ | $6 + V + K_3$ | 4   | 5   | $6 + V + K_3$ | \ldots |

- Next, we consider RC4 keystream algorithm
RC4 Keystream

- After initialization, let $i = j = 0$
- Then for each keystream byte
  
  $i = i+1$
  
  $j = j+S_i$
  
  swap($S_i, S_j$)
  
  $t = S_i+S_j$
  
  keystreamByte = $S_t$
RC4/WEP Attack

- Suppose initialization stopped with

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>5+V</th>
<th>...</th>
<th>6+V+K_3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_i</td>
<td>3</td>
<td>0</td>
<td>5+V</td>
<td>6+V+K_3</td>
<td>4</td>
<td>5</td>
<td>...</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

- First keystream byte
- Let i = j = 0

- Then

  \[ i = i+1 = 1 \]
  \[ j = j+S_1 = 0 \]
  \[ t = S_i+S_j = S_1+S_0 = 0+3 = 3 \]
  \[ \text{keystreamByte} = S_t = S_3 = 6+V+K_3 \]
RC4/WEP Attack

- **Note:** keystreamByte = 6+V+K₃
- **If keystreamByte is known, we can solve for K₃ since**
  
  \[ K₃ = (\text{keystreamByte} - 6 - V) \mod 256 \]
- **But initialization does not stop at i=3**
- **So can this “attack” really work?**
RC4/WEP Attack

- After $i=3$ initialization step, $S$ is

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>5 + V</th>
<th>...</th>
<th>6 + V + K_3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>3</td>
<td>0</td>
<td>5 + V</td>
<td>6 + V + K_3</td>
<td>4</td>
<td>5</td>
<td>...</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

- If elements at 0, 1 and 3 not swapped in remaining initialization steps, attack works

- For remaining initialization steps...
  - We have $i = 4, 5, 6, ...$ so index $i$ will not affect anything at indices 0, 1 or 3
  - But what about index $j$?
RC4/WEP Attack

- Pretend index $j$ selected at random
  - At each step, probability is $\frac{253}{256}$ that $j \notin \{0,1,3\}$
  - There are 252 steps after $i = 3$

- Probability that 0, 1 and 3 not affected by $j$ index after $i=3$ step is
  \[
  \left(\frac{253}{256}\right)^{252} = 0.0513
  \]
RC4/WEP Attack

- Can be shown that with about 60 IVs of the form \((3, 255, v)\) can find \(K_3\)
  - Not so easy to prove that 60 is correct
  - Easy to verify empirically

- This is enough to show that a shortcut attack on WEP/RC4 exists

- Can Trudy really recover the key?

- If she sees enough IVs she gets \(K_3\)
RC4/WEP Attack

- Suppose Trudy has found $K_3$
- Then how to find $K_4$?
- Consider IVs of the form: $IV = (4, 255, V)$
- Then after initialization step $i=4$, we have

\[
\begin{array}{ccccccc}
  i & 0 & 1 & 2 & 3 & 4 & 5 & \ldots \\
  S_i & 4 & 0 & 6+V & 9+V+K_3 & 10+V+K_3+K_4 & 5 & \ldots \\
\end{array}
\]

\[
\begin{array}{ccccccc}
  i & \ldots & 6+V & \ldots & 9+V+K_3 & \ldots & 10+V+K_3+K_4 & \ldots \\
  S_i & \ldots & 2 & \ldots & 3 & \ldots & 1 & \ldots \\
\end{array}
\]
RC4/WEP Attack

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>4</td>
<td>0</td>
<td>$6 + V$</td>
<td>$9 + V + K_3$</td>
<td>$10 + V + K_3 + K_4$</td>
<td>5</td>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i$</th>
<th>…</th>
<th>6 + $V$</th>
<th>…</th>
<th>9 + $V + K_3$</th>
<th>…</th>
<th>10 + $V + K_3 + K_4$</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>…</td>
<td>2</td>
<td>…</td>
<td>3</td>
<td>…</td>
<td>1</td>
<td>…</td>
</tr>
</tbody>
</table>

- If we now generate first keystream byte
  
  $i = 1$
  
  $j = S_i = 0$
  
  $t = S_1 + S_0 = 4$
  
  keystreamByte = $S_4 = 10 + V + K_3 + K_4$

- Then $K_4 = (\text{keystreamByte} - 10 - V - K_3) \mod 256$

- Probability of this is also about 0.05
RC4/WEP Attack

- If enough IVs are available
  - And corresponding 1st keystreamBytes are known
- Then Trudy can recover the key
  - Finds $K_3$ then $K_4$ then $K_5$ and so on...
- Get entire key, regardless of length!
RC4/WEP Attack

- Can reduce number of IVs Trudy needs
- Consider again key $K_3$
- Suppose IV = (2, 253, 0)
- Then after $i=3$ initialization step

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>3 + $K_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3 + $K_3$</td>
<td>4</td>
<td>...</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- IVs other than (3, 255, V) can work!
- Easy to determine which IVs are useful
RC4/WEP Conclusions

- This attack is practical!
- This attack has been used to recover keys from real WEP traffic
- How to prevent this attack?
  - Discard first 256 bytes of keystream
- This attack on RC4 is just one of many security flaws in WEP