Securing 802.11 (WiFi) networks

Claude Castelluccia- INRIA
(some slides borrowed from Srdjan Čapkun)
IEEE 802.11 Protocol Primer
802.11 protocol

- 802.11 is a wireless LAN standard
  - 2 frequency bands: 2.4GHz and 5GHz.
- The standard is referring to the link and physical layers
- This is the most widely deployed wireless LAN technology.
- It is composed of a Access point(s) and mobile devices
802.11 protocol

- IEEE802.11 has two modes of operations:
  - Infrastructure mode (ESS)
    - All communications (even between two mobile devices of same network) are through the access point (AP)
    - The AP is coordinating the communications.
  - Infrastructure-less mode (IBSS) or ad-hoc mode
    - Mobile devices (STA) can communicate directly
    - No AP is necessary!

- Most networks operate in infrastructure mode...
Basics of Operation in Infrastructure Mode

- Let’s consider an access point (AP) and a mobile station (STA)
- The AP uses several radio frequencies (called **channels**) to communicate with the STAs
- The AP advertises its presence, on each channel, by transmitting short wireless messages at regular intervals (10 times a second)
  - These messages are called **beacons**
- The STA must then tune into each channel and listen for beacon messages
  - This process is called **scanning**
  - This process can be accelerated by probing (i.e. the STA sends a request)
- The STA may discover several APs in a large network and must decide to which it intends to connect (based on signal strength, security policy, roaming agreement, SSID,...)
Basics of Operation in Infrastructure Mode (2)

• When the STA is ready to connect to the AP, it first sends an *authenticate* request message to the AP.
• The AP immediately responds by sending an *authenticate response message* indicating acceptance (*no security is used in this example*)
• The STA sends an *association request* message
• The AP responds with an *association reply* message...
• The **STA is then connected** (associated in the term) and can send data through the AP!

• There are 3 types of messages:
  – **Control**: short msgs that tell devices when to start or stop transmission
  – **Management**: messages use to negotiate and control the association.
  – **Data**: messages that contain the data...
Introduction to WiFi

- beacon
- MAC header
- timestamp
- beacon interval
- capability info
- SSID (network name)
- supported data rates
- radio parameters
- power slave flags

STA → AP
association request

"connected" → scanning on each channel

STA

association response

AP
802.11 MAC header (DATA message)

<table>
<thead>
<tr>
<th>Bytes:</th>
<th>2</th>
<th>2</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>6</th>
<th>0-2312</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Control</td>
<td>Duration</td>
<td>ID</td>
<td>Addr 1</td>
<td>Addr 2</td>
<td>Addr 3</td>
<td>Sequence</td>
<td>Control</td>
<td>Addr 4</td>
<td>Frame Body</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRC</td>
</tr>
</tbody>
</table>

Frame Control Field

Bits: 2 2 4 1 1 1 1 1 1 1 1

<table>
<thead>
<tr>
<th>Protocol Version</th>
<th>Type</th>
<th>SubType</th>
<th>To DS</th>
<th>From DS</th>
<th>More Frag</th>
<th>Retry</th>
<th>Pwr Mgt</th>
<th>More Data</th>
<th>WEP</th>
<th>Rsvd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>To DS</th>
<th>From DS</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>DA</td>
<td>SA</td>
<td>BSSID</td>
<td>N/A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>DA</td>
<td>BSSID</td>
<td>SA</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>BSSID</td>
<td>SA</td>
<td>DA</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RA</td>
<td>TA</td>
<td>DA</td>
<td>SA</td>
</tr>
</tbody>
</table>

- DA: destination address
- SA: source address
- RA: receiver address (AP)
Management Frames

802.11 defines several management frames

- Beacon (notify)
- Probe (request and response)
- Authenticate (request and response)
- Associate (request and response)
- Deassociate (notify)
- Deauthenticate (notify)

- “notify” means that no response is expected.
Management Frames/Beacon Frame

Each frame contains specific information/field

For example, the Beacon frame contains:

- MAC header
- Timestamp: time since set-up of AP
- Beacon interval: time between 2 beacons
- Capability info.: defines AP optional features
- SSID (Service Set Identifier): name of the network
- Supported Data Rates: indicates what speed the AP can support i.e. 1, 2, 5.5, 11Mbps or up to 54 Mbps.
- Radio Parameters: exact radio frequency used by AP. The STA might be listening on a nearby frequency...
- Power Save Flags: are used to tell sleepy wireless devices that there is data waiting for them.
IEEE 802.11 Security Solutions
Access mechanisms

open network (no protection)
  • assumption: there are no unauthorized users in the range of the network
  • problems: range is hard to determine (unpredictable propagation of the signals, directional antennas, ...)

closed network
  • using SSIDs for authentication (Service Set Identifier)
  • MAC filtering
  • shared keys
  • authentication servers
MAC filtering

- MAC address filtering
  - only devices with certain MAC addresses are allowed to associate
  - needs pre-registration of all device at the AP

- MAC can be sniffed and forged
  - sent in clear text in each packet (can be sniffed)
  - can be forged
Device identification – MAC addresses

- “Hardcoded” addresses in WiFi cards (“unique device identifiers”)
  - all devices have different addresses

- Concept taken over from ethernet addresses

```cmd
>ipconfig /all
```

*Ethernet adapter Wireless Network Connection:*

- Media State: Media disconnected
- Description: Intel(R) PRO/Wireless
- Physical Address: 00-13-02-B8-9A-1B
3 simple steps for overcoming MAC filtering

1. Put your card in promiscuous mode (accepts all packets).

2. Sniff the traffic and find out which MAC addresses are accepted by the AP

3. Change your MAC address (need a card that can do that)

```bash
# ifconfig ath0 hw ether <mac address of C>
```
SSID-based access control

- SSID = Service Set IDentifier (network name)
- a 32-character unique identifier
- attached to the header of packets
- acts as a password when a mobile device tries to connect to the WLAN
- SSID differentiates one WLAN from another
- all devices attempting to connect to a specific WLAN must use the same SSID
SSID-based access control

- SSIDs can be sniffed (e.g. http://www.ethereal.com)
  - advertised by the APs
  - contained in SSID response frames

- Overcoming SSID-based access control
  - Sniff SSID (either sent by the clients or advertised by the AP)
  - Set your SSID to the same value ...

- MAC/SSID access control: not a bad protection from unskilled neighbors (much better than no authentication/protection)

MAC/SSID sniffing and MAC forging will be shown in the exercises.
IEEE 802.11 WEP
Protected access using WEP

- WEP = Wired Equivalent Privacy
  - part of the IEEE 802.11 specification

- Goal
  - make the WiFi network \textit{at least as secure as a wired LAN} (that has no particular protection mechanisms)
  - WEP has never intended to achieve strong security
  - (at the end, it hasn’t achieved even weak security)

- Services
  - Mobile Authentications: access control to the network
  - Message confidentiality
  - Message integrity/authentication

- All these 3 services can be broken ;-)
The WEP Protocol

• Sender and receiver share a secret key $k$.

• Two classes of WEP implementation:
  – classic WEP as documented in standard (40-bit key)
  – extended version developed by some vendors (128-bit key)

• The payload of every packet is encrypted (confidentiality) with its CRC value (integrity)

• Authentication through ‘standard’ challenge-response authentication protocol ... using the shared key ...
WEP-authentication (1)

- Based on a shared key between the station and the AP (40 bit or 104 bit)
- Based on the RC4 symmetric stream cipher
- **24-bit** Initialization Vector (IV)
- Authentication through ‘standard’ challenge-response authentication protocol ... using the shared key ...
  - Challenge text sent in payload in cleartext (128 octets)
  - Response sent in payload encrypted
WEP-authentication (2)

probe request
(sent on all channels, trying to find AP with matching SSID)

response (if in range)

auth. request (choosing best AP)

challenge

response

auth. decision

.... if authentication successful

association request

association reply

user can now send data
WEP-authentication (3)

- Challenge text sent in payload in cleartext (128 octets), random IV used
- Response sent in payload encrypted with the key shared between the AP and the station

<table>
<thead>
<tr>
<th>Size in octets</th>
<th>Management Frame Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Control</td>
<td>Destination</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size in octets</th>
<th>Authentication Frame Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Number</td>
<td>Seq Num</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
WEP - Authentication frame format

- **MSDU** = MAC Service Data Unit
  - this is the message
- **ICV** = Integrity Check Value (encrypted CRC)
  - protects message integrity
- **IV** = Initialization Vector is used to expand the key
  - expands the key
WEP Confidentiality/Integrity Protocol

- In order to transmit a message M:
  - Compute $P = \langle M, c(M) \rangle$
  - pick IV $v$ and generate $\text{RC4}(v, k)$
  - compute $C = P \oplus \text{RC4}(v, k)$
  - Send $A \rightarrow B$: $v, (P \oplus \text{RC4}(v, k))$

- Upon receipt:
  - generate $\text{RC4}(v, k)$
  - Compute $P = C \oplus \text{RC4}(v, k) = P \oplus \text{RC4}(v, k) \oplus \text{RC4}(v, k)$
  - check if $c = c(M)$

- If so, accept the message $M$ as being the one transmitted
WEP, Pictorially

Plaintext

Message

CRC

XOR

Keystream = RC4(v,k)

v

Ciphertext

Transmitted Data
WEP confidentiality/integrity protection (1)

- RC4 Generates a key stream of a desired length from the key
- The key stream is XORed with plaintext data
- The result is ciphertext data

IV: Initialization vector
ICV: checksum
WEP confidentiality/integrity protection (2)

- RC4 is a stream cipher
  - given a short input key, it produces a pseudorandom sequence (key stream)
  - the key stream is always the same for the same key
  - A different IV (initialization vector) is therefore needed for each message!
    - The key stream is initialized for each message...
    - ...so it is tolerant to packet loss...
- The output of the key stream is XORed with the plaintext to obtain a ciphertext:
Stream Ciphers (a short review)

• Definition
  – encrypt individual characters of plaintext message one at a time, using encryption transformation which varies with time.

• Block vs. Stream
  – Block ciphers
    • process plaintext in relatively large blocks
    • The same function is used to encrypt successive blocks ⇒ memoryless
Stream Ciphers (2)

- Block vs. Stream
  - Block ciphers
    - ...
  - Stream ciphers
    - process plaintext in small blocks, and the encryption function may vary as plaintext is processed $\Rightarrow$ have memory
    - sometimes called state ciphers since encryption depends on not only the key and plaintext, but also on the current state.
    - Note that in WEP, the state is initialized for each packet!

- Stream ciphers are more efficient than Block ciphers
  - RC4 is 10 times faster than DES.
RC4 (Rivest Cipher 4) Stream Cipher

- Proprietary cipher owned by RSA
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input info processed one byte at a time

\[ M \rightarrow \text{RC4 Enc.} \rightarrow C \]
\[ K \rightarrow \text{RC4 Enc.} \rightarrow C \]
\[ C \rightarrow \text{RC4 Dec.} \rightarrow M \]

K: secret i.e. shared between Alice and Bod
RC4 Key Schedule

- Starts with an array $S$ of numbers: $0..255$
- $S$ forms **internal state** of the cipher
- given a key $k$ of length $l$ bytes (64 or 128 bits)

```plaintext
for i = 0 to 255 do
    S[i] = i

j = 0

for i = 0 to 255 do
    j = (j + S[i] + k[i mod l]) (mod 256)
    swap (S[i], S[j])
```
RC4 Key Schedule: An simple example

- Starts with an array $S$ of numbers: $0..10$

- given a key $k$ of length $l$ bytes (5)
RC4 Key Schedule: An simple example (2)

- $j = 0$
  - for $i = 0$ to $9$ do
    - $j = (j + S[i] + k[i \mod 1]) \mod 10$
    - swap $(S[i], S[j])$
  - $j = 0$
    - $i = 0$
      - $j = (0 + S[0] + k[0 \mod 5]) \mod (10) = (0 + 0 + 3) = 3$
      - Swap $(S[0], S[3])$ $\Rightarrow S[3] = 0; S[0] = 3$
    - $i = 1$
      - $j = (3 + S[1] + k[1]) = (3 + 1 + 9) = 4$
      - Swap $(S[1], S[4])$; $\Rightarrow S[1] = 4; S[4] = 1$
      - ...
    - $\Rightarrow$ At this stage: $S[0] = 3; S[1] = 4; S[2] = 2; S[3] = 0; S[4] = 1$
    - $\Rightarrow$ And so on until $i = 9$!
**RC4 Stream Generation**

- Encryption continues shuffling array values

\[
i = j = 0
\]

for each message byte \( M_i \)

\[
i = (i + 1) \pmod{256}
\]

\[
j = (j + S[i]) \pmod{256}
\]

\[
\text{swap}(S[i], S[j])
\]

\[
t = (S[i] + S[j]) \pmod{256}
\]

\[
C_i = M_i \ XOR \ S[t]
\]
RC4 Stream Generation: A simple example

- *Encryption continues shuffling array values*
  
  \[
  i = j = 0 \\
  \text{for each message byte } M_i \\
  i = (i + 1) \pmod{256} \\
  j = (j + S[i]) \pmod{256} \\
  \text{swap}(S[i], S[j]) \\
  t = (S[i] + S[j]) \pmod{256} \\
  C_i = M_i \text{ XOR } S[t]
  \]

- \( i=j=0; \)
- \( i=1; \)
- \( j=(0+S[1]) \pmod{256}, \) let’s say \( S[1]=112 \Rightarrow j = 112; \)
- \( \text{Swap}(S[1], S[112]) \Rightarrow \) let’s say \( S[112]=33 \Rightarrow S[1]=33; S[112]=112 \)
- \( t= (S[1]+S[112])= 33 + 112 = 145 \)
- \( C1 = M_i \text{ xor } 145; \)
Steam Cipher’s "Two-Time Pad" Problem

• You must **never** encrypt two messages with the same keystream $S$.

• **Problem 1:**
  – Suppose $P_1$ and $P_2$ are both encrypted with the same $S$.
    • then
      $$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$
  – So the adversary learns the XOR of two plaintexts!
  – Usually, just knowing the XOR of two plaintexts is enough to recover them.
    • For example, if the adversary knows $P_1$, he can get $P_2$…
The "Two-Time Pad" Problem

• *Problem 2:*
  – if an adversary knows a (P1,C1) pair
    • then $S = C_1 \text{xor} P_1$
    • the adversary can sent $C_2 = M_2 \text{xor} S$
      – $C_2$ will be successful decrypted by Bob!

• As a result in RC4,
  – The key must never be used twice!
• If not, many attacks are possible
  – See WEP (later in this class)!

• In WEP, the key between the AP and STA is fixed
  – Therefore WEP uses a different IV (initialization vector) per message
WEP confidentiality/integrity protection (bis)

Different for each packet (24 bits)

Fixed (40 or 104 bits)
WEP – Keys

- two kinds of keys are allowed by the standard
  - default key (also called shared key, group key, multicast key, broadcast key, key)
  - key mapping keys (also called individual key, per-station key, unique key)

- in practice, often only default keys are supported
  - the default key is manually installed in every STA and the AP
  - each STA uses the same shared secret key → in principle, STAs can decrypt each other’s messages
WEP – Management of default keys

- the default key is a group key, and group keys need to be changed when a member leaves the group
  - e.g., when someone leaves the company and shouldn’t have access to the network anymore
- it is practically impossible to change the default key in every device simultaneously
WEP – The key change process

1. abc* → abc* (* active key)

2. abc* → abc* → abc def* → abc def*

3. --- def* → --- def*

4. --- def* → --- def*
Breaking WEP Confidentiality
WEP Flaws- Breaking Confidentiality

• The keystream for WEP is $\text{RC4}(\text{IV}, k)$.
• $k$ is a fixed shared secret, that changes rarely, if ever (in many setups, every user shares the same $k$).
• If two packets ever get transmitted with the same value of $\text{IV}$, you reuse the keystream.
• Since $\text{IV}$ gets transmitted in clear for each packet, the adversary can even easily tell when a value of $\text{IV}$ is reused (a "collision").
WEP Flaws - Confidentiality (2)

- How many possible values of IV are there?
- IV only occupies 24 bits of the header = at most there are $2^{24}$ (about 16 million of IV).
- After 16 million packets, you have to repeat one!
- It is even worse than that!
  - All the 802.11 cards reset their IV counter to 0 every time they were activated, and incremented by 1 for each packet transmitted.
- This means that low IV values get reused at the beginning of every wireless session.
- Usually use the same secret k, and often many different people use the same k.
- So you can find collisions between packets sent by different people!
- This makes collisions much more common => birthday paradox!
Birthday Paradox/attack

- How many people must be in a room for the chance that someone share your birthday is larger than ½?
  - 253!
- How many people must be in a room for the chance that at least 2 of them share the same birthday is larger than ½?
  - Only 23!!!
- More generally, if a IV is 24 bit long and generated randomly...
  - After $2^{24/2} = 2^{12}$ messages, the probability that 2 IV collide is larger than ½!
- Using this result, the attacker can build a dictionary!
WEP Flaws - Confidentiality (3)

Decryption Dictionaries

- Adversary knows both the C and the P for some packets encrypted with a given IV v.
  - Easy if he knows the P (pings, ARP request/reply, or spam email!).
  - $\text{RC4}(k,v) = P \oplus C$
  - Note: no need to know the value of the shared secret k.
- Store keystream in a table, indexed by v.
- Next time a packet with an IV stored in the table passes by, look up the keystream, XOR it against the packet, and read the data!
- Table is at most $1500 \times 2^{24}$ bytes = 24 GB
  - 1500 is the max. frame size.
- If the cards that are being used have the IV-reset-to-0 property, then most IV's will be small, and the dictionary will be even smaller!
• weak RC4 keys
  – for some special keys (called weak keys), the beginning of the RC4 output is not really random
  – if a weak key is used, then the first few bytes of the output reveals a lot of information about the key → breaking the key is made easier
  – for this reason, crypto experts suggest to always throw away the first 256 bytes of the RC4 output, but WEP doesn’t do that!
  – due to the use of IVs, eventually a weak key will be used, and the attacker will know that, because the IV is sent in clear
  → WEP encryption can be broken by capturing a few million messages !!!
Breaking Confidentiality - Conclusion

- WEP confidentiality was broken because
  - IV is too small
  - IV is choosen randomly!
Breaking WEP Message Integrity
Breaking Message Integrity in WEP

- An 802.11 receiver will accept a packet if, after decryption, it contains a correct checksum of the plaintext.
- The checksum algorithm used is CRC-32.
- CRC's are used to detect random errors; they are useless against malicious errors.
- There is already a CRC at a lower layer of the protocol to detect random bit errors in transmission.
CRC-32 Properties

- It is independent of the shared secret and the IV.
- It is linear: \( c(M \oplus D) = c(M) \oplus c(D) \)
- We can make a controlled modification and get unnoticed.
  - Assume a message \( M \) was transmitted, and the ciphertext was \( C \) and the IV was \( v \) (i.e. \( C \) and \( v \) are known to the adversary).
Breaking Message Integrity

- An attacker can modify any message...

- \( C = \text{RC4}(v, k) \oplus <M, c(M)> \)

- A -> B: \(<v, C>\)

- Given \(<v, C>\), it is possible to find \(C'\) s.t. it decrypts to \(M'\) and \(M' = M \oplus \Delta\), \(\Delta = \) arbitrarily chosen by the attacker

- A -> B: \(<v, C'>\)
  Where \(C' = C \oplus <\Delta, c(\Delta)>\)
  
  \[
  \begin{align*}
  &= \text{RC4}(v, k) \oplus <M, c(M)> \oplus <\Delta, c(\Delta)> \\
  &= \text{RC4}(v, k) \oplus <M \oplus \Delta, c(M) \oplus c(\Delta)> \\
  &= \text{RC4}(v, k) \oplus <M', c(M' \oplus \Delta)> \\
  &= \text{RC4}(v, k) \oplus <M', c(M')> \\
  \end{align*}
  \]

- Receiver checks that \(c' = c(M')\)
- Accept message \(M'\) as the one transmitted!
Message Injection

• An Attacker can inject any bogus message $M$...

• The adversary just needs to know a single plaintext, and its corresponding encrypted packet (P and C).
  
  $P \oplus C = P \oplus RC4(v,k) \oplus P = RC4(v,k)$
  
  • Construct $M'$ and $P' = <M', c(M')>$
  
  $C' = RC4(v,k) \oplus P'$
  
  • “A” -> B: $<v, C'>$
  
• The IV is selected by A (i.e. the attacker in this case)!
  
  – So it can be replayed!
WEP Message Integrity: Conclusions

- *WEP Message Integrity was broken because*
  - CRC was used instead of a MAC!
  - The IV is selected by A (i.e. the attacker in this case)!
    - So it can be replayed!
Breaking WEP Access Control/User Authentication
WEP flaws – Breaking Authentication and access control

- authentication is one-way only
  - AP is not authenticated to STA
  - STA may associate to a rogue AP
- the same shared secret key is used for authentication and encryption
  - weaknesses in any of the two protocols can be used to break the key
  - different keys for different functions are desirable
- no session key is established during authentication
  - access control is not continuous
  - once a STA has authenticated and associated to the AP, an attacker send messages using the MAC address of STA
  - correctly encrypted messages cannot be produced by the attacker, but replay of STA messages is still possible
- STA can be impersonated
  - ... next slide
Breaking STA Authentication Protocol

- **Authentication Goal**: the base station verifies that a client joining the network really knows the shared secret key $k$.
- The base station sends a challenge string to the client $B \rightarrow A: M$
- The client sends encrypted challenge: $A \rightarrow B: v, <M,c(M)> \oplus RC4(v,k)$
- The base station checks if the challenge is correctly encrypted, and if so, accepts the client.

- Adversary sees a challenge/response pair for a given key $k$; he can extract $v$ and $RC4(v,k)$.
  - $RC4(v,k)=C \oplus P$;
- Adversary can execute authentication protocol himself!
  - See next slide…
Authentication Spoofing

- Adversary connects to the network himself:
- The base station sends a challenge string $M'$ to the adversary.
- The adversary replies with $v, <M',c(M')> \oplus RC4(v,k)$
- This is the correct response, so the base station accepts the adversary.
- Success even though he never did learn the value of $k$!
Authentication: Conclusion

- Challenge-response protocols do not work with stream ciphers!
Recovering WEP Key
Recovering the key (in less than 60 seconds)

- End of 2007 a new attack was demonstrated by Klein and then Tews and al.
- With this attack, a 104-bit WEP key can be recovered with less than 40,000 frames with a probability of 50%!
- With 85,000 frames, the probability increases to 95%!
- It works for any random IV and keys (even “strong” key!)
- It is based on a correlation between the RC4 keystream and the secret key K
Recovering the key in less than 60 seconds (2)

- Klein has shown that if we know:
  - $K[0]$, first byte of the key, and $X[1]$, second byte of the keystream we can find $K[1]$ with a high probability!
  - If we know $K[0]$, $K[1]$, and $X[2]...K[2]$ can be computed with a high probability!
  - ...
    - If we know $K[0]$, $K[1]$, $K[2],...,K[p-1]$, $X[p]$, $K[p]$ can be computed with a high probability...
  - Eventually the whole key can be retrieved with less than 85,000 frames!
- See “Breaking 104 bit WEP in less than 60 seconds” for more details!
Recovering the key in less than 60 seconds (3)

- In WEP, the 3 first bytes of the secret key is the IV
  - i.e. K[0], K[1] and K[2] are public...
- By identifying the encrypted ARP request and ARP reply messages...
  - Only Packets of size 68 bytes
- We can get many (P,C) pairs for different IV
  - 16 first bytes of ARP packets are well known...
- And therefore derive many keystream bytes X[i]
  - For i=1 to 16
Recovering the key in less than 60 seconds (4)

- So by capturing several packets with different IV...
- We can compute for the different frames and their IV (i.e. K[0], K[1], K[2]) and X[3], the corresponding K'[3]
  - The K'[3] which appears the most frequently is the correct one, K[3]!
  - Note that K[0], K[1] and K[2] are the 3 bytes of the IV and will differ from one frame to the other...
  - ...but K[3], K[4], ..., K[16] are always the same!
- K'[4] can then be computed for the different frames using IV[0], IV[1], IV[2], K[3] and X[4]
  - The K'[4] which appears the most frequently is the correct one, K[4]!
- And so on...
Recovering the key in less than 60 seconds (5)

- We know 16 bytes of the ARP messages...
- which is enough to break keys up to 128 bits (24 of IV + 104 secret key)
- If more bytes are necessary (because the key is larger)
  - There exists another attack that can be used in combination
  - "The final nail in WEP's coffin"
- It allows to retrieve larger (keypad, plaintext) pair using a fragmentation property of 802.11
  - AP re-assembles messages before reforwarding them...
  - So A can send many fragments to B (on same subnet)...
  - AP will re-assemble them before forwarding to B...
Recovering the key in less than 60 seconds (6)

- This is an attack on RC4 ...
  - but WEP facilitates it since 3 first bytes are public
  - Does not work with WPA/TKIP since each per-message key does not share a common suffix.
  - Usually first byte $K[0]$ is not known...but could be guessed!

- This is the algorithm that is implemented in the Aircrack-ng tool that you’ll use next week!
Aircrack-ng

- Airedump to collect data...

- Aircrack-ng to crack the key...
WEP: what went wrong?

- IV is too short: 56-64 bits should be used...
- IV should be set to random value, not zero when reset!
- AP must keep a list of used value IV
  - Require some memory
  - => Key must be changed periodically...
- A MAC should be used for message integrity instead of a checksum
- Although RC4 is believed to be secure
  - It is easy to make mistakes
  - And build insecure systems
- This is often the case
  - Attacks are on the protocols not on the crypto. Algorithms...
Summary: WEP – Lessons learnt

1. engineering security protocols is a **very** risky business
   - you may combine otherwise secure building blocks in a wrong way and obtain an insecure system at the end
     • example:
       - stream ciphers alone are OK
       - challenge-response protocols for entity authentication are OK
       - but they shouldn’t be combined
     • example:
       - encrypting a message digest to obtain an ICV is a good principle
       - but it doesn’t work if the message digest function is linear wrt to the encryption function
WEP – Lessons learnt (2)

– don’t do it alone (even if you are a security expert)
– using external experts in the design phase pays out (fixing the system after deployment will be much more expensive)
  • experts will not guarantee that your system is 100% secure
  • but at least they know many pitfalls that you don’t
  • they know the details of crypto algorithms better than you do

2. avoid the use of WEP (as much as possible)
WEP conclusion/references (RC4 vulnerabilities)

- Scott R. Fluhrer, Itsik Mantin, Adi Shamir, Weaknesses in the Key Scheduling Algorithm of RC4 Proceedings of SAC 01

- Implemented in the AirSnort application

WEP-crack will be shown during the LAB exercises.
IEEE 802.11i (WPA and WPA2)
Overview of 802.11i

- after the collapse of WEP, IEEE started to develop a new security architecture → 802.11i
- main novelties in 802.11i wrt to WEP
  - access control model is based on 802.1X
  - flexible authentication framework (based on EAP)
    - authentication can be based on strong protocols (e.g., TLS)
  - authentication process results in a session key (which prevents session hijacking)
  - different functions (encryption, integrity) use different keys derived from the session key using a one-way function
  - integrity protection is improved
  - encryption function is improved
  - Pairwise key is enforced (in WEP, mobile uses same key)
Overview of 802.11i

- 802.11i defines the concept of RSN (Robust Security Network)
  - integrity protection and encryption is based on AES (in CCMP mode)
  - nice solution, but needs new hardware → cannot be adopted immediately

- 802.11i also defines an optional protocol called TKIP
  - integrity protection is based on Michael
  - encryption is based on RC4, but WEP’s problems have been avoided
  - ugly solution, but runs on old hardware (after software upgrade)

- industrial names
  - TKIP → WPA (WiFi Protected Access)
  - RSN/AES-CCMP → WPA2
802.11i overview

Security capabilities discovery

802.1X authentication

802.1X key management

RADIUS key distribution

DATA protection
(TKIP, AES-CCMP)
802.11i overview

802.1x authentication
- Mutually authenticate STA and AS
- Generate Master Key as a side effect of authentication
- Generate 4 keys for encryption/integrity

*We will detail this phase later....*

Data protection
- Provides data confidentiality and integrity
- 2 possible schemes
  - TKIP (optional)
  - AES-CCMP

*Let’s consider for now that the STA and AS share a master key (we’ll see later on how to do that)...and let’s look at TKIP...*
TKIP/WPA1
Temporary Key Integrity Protocol (WPA)

- TKIP is a secure and available as an upgrade to WEP systems.
- The implementation of WEP almost depends on the hardware assist functions.
  - **RC4 is implemented on hard inside the card chip**
- The hardware assist functions in these earlier systems cannot support AES-CCMP.
  - Implementing AES-CCMP means changing the cards!
- **TKIP uses existing RC4 and upgrades the firmware.**
- Provides confidentiality and integrity.
- Ugly, but works with existing hardware.
- Usually used with manually configured master key (Pre-Shared Key mode, although 802.1x could be used)
Component of a Wi-Fi LAN Adapter

Host Interface

Medium Access Control (WEP)

Modem

Radio Frequency

Microprocessor

Firmware

Hardware Assist (RC4)

RAM
Changes from WEP to TKIP/WPA

- **IV selection and use**: as counter (sequence no)
- **Increase the size of IV**: to avoid ever reusing the same IV.
- **Per-packet key Mixing**: change the key for every frame
- **Message integrity**: add a message integrity protocol.
- **Key management**: add a mechanism to distribute and change the broadcast keys.

*The same encryption scheme than WEP is used: RC4*
IV Selection and Use

• In WEP
  – IV was too short...
  – IV was not specific to a Station and can be used by multiple devices (increase vulnerability)
  – Prepending the IV to the secret is susceptible to the Fluhrer-Mantin-Shamir attack (not covered in this class) that uses “weak” keys.

• In TKIP,
  – IV is increased from 24 bits to 48 bits
  – IV is incremented as a counter...not a random value
    • This reduces collisions (birthday attack paradox)
    • Prevent replay attack.
  – IV is contructed to avoid certain “weak keys”
IV as a Sequence Counter

• WEP had not protection against replay attack
  – An enemy could record a valid packet and play it back later
  – In TKIP, the AP can store the latest IV used value and detect replay attacks!
    • Since IV is a counter!
• In WEP, there were not requirements on how to generate IV
  – Many vendors picked a random value for IV for each packet...
  – Which seems to be a good idea...but is not!
  – ..because of birthday paradox!
Per-Packet Key Mixing

- A solution to protect against the previous attacks is to change the keys frequently.
- In TKIP, the encryption key is changed for every packet sent!
- In WEP, there was a single key used for everything.
  - If compromised...everything is compromised!
- TKIP uses multiple keys
  - The session keys (renewed periodically)
  - derived from a single master key
- The per-packet key mixing derives a key for each packet
  - The session keys and master keys do not change at every packet!
Per-Packet Key Mixing (2)

Efficient Hash functions

Different for each frame!
Per-Packet Key Mixing against weak key (3)

To avoid weak key
TKIP – Integrity

- Replaces ICV (Integrity Check Value) with MIC
  - MIC – Message Integrity Code or Message Authentication Code (MAC)
    - Computed using a non-reversible function and a secret key
    - Protects against bit-flip attacks by adding tamper-proof hash to messages
- TKIP uses MICHAEL a newly designed scheme
  - Can run on low power processor and without hardware support
  - Compute on 8-byte check value
  - Used with a secret key!
    - Part of firmware
Summary

• TKIP
  – uses RC4 → runs on old hardware
  – corrects WEP’s flaws
  – mandatory in WPA, optional in RSN (WPA2)
  – Temporary solution... but will probably be around for awhile ;(-)
802.11i/WPA1: a comprehensive redesign of WiFi security

- Robust Security Network (RSN) for establishing secure communications
  - Uses 802.1x for authentication
  - Replaces TKIP
- AES replaces RC4 w/TKIP,
  - Counter Mode with Cipher Block Chaining (CCM)
    (CCM=counter mode + CBC MAC)
    - Counter mode for encryption
    - CBC-MAC provides data integrity/authentication
      - 128-bit keys, 48-bit IV
      - CCMP mandatory with RSN
      - Ensures data confidentiality and integrity
- Dubbed “WPA2” by WiFi Alliance
AES: Advance Encryption Standard

- Block Cipher:
  - Message is decomposed into blocks
  - Each block is encrypted independently
  - Used the Rijndael Algorithm
  - Allows different block sizes and key sizes
    - 128, 192 and 256 bits.
AES – Encryption: Counter Mode of Operation

- AES can be used differently: mode of operation
- Counter Mode of Operation
  - Message divided into blocks
  - A counter i is encrypted
  - $E(i) \oplus E(Bi)$ produces the encrypted message block
AES-MAC: CBC-MAC mode of operation

- AES is also used to compute MAC
  - One algorithm for encryption and MAC!
- CBC is used to compute a MIC (Message Integrity Code)
  - 1. Take the first block and encrypt it using AES
  - 2. XOR the result with the second block and then encrypt the result
  - 3. XOR the result with next block and encrypt that...and so on!
How is CCM Used in RSN....
802.11i overview

Station -> Access Point

Security capabilities discovery

802.1X authentication

802.1X key management

RADIUS key distribution

DATA protection (TKIP, AES-CCMP)

Auth. Server/Radius Server
802.11i protocol steps

Discovery
   – AP advertises network security capabilities to STAs

802.1x authentication
   – **Mutually** authenticate STA and AS
     • Using password, Challenge/response, TLS,…
     • Generate Master Key as a side effect of authentication
     • Generate Pairwise MK (PMK) as an access authorization token
   – STA and AP Generate 4 keys for encryption/integrity from PMK
   – STA authenticates AP!
802.1X

- Implement access control
- It was not originally designed for wireless networks
- It was designed to allow an unauthorized device to be physically attached to a LAN infrastructure.
802.1X authentication model

- the **supplicant** requests access to the services (wants to connect to the network)
- the **authenticator** controls access to the services (controls the state of a port)
- the **authentication server** authorizes access to the services
  - the supplicant authenticates itself to the authentication server
  - if the authentication is successful, the authentication server instructs the authenticator to switch the port on
  - the authentication server informs the supplicant that access is allowed
Mapping the 802.1X model to WiFi

• supplicant → mobile device (STA)
• authenticator → access point (AP)
• authentication server → server application running on the AP or on a dedicated machine
• port → logical state implemented in software in the AP
• one more thing is added to the basic 802.1X model in 802.11i:
  – successful authentication results not only in switching the port on, but also in a session key between the mobile device and the authentication server
  – the session key is sent to the AP in a secure way
    • this assumes a shared key between the AP and the auth server
    • this key is usually set up manually
802.11i/RSN key Hierarchy

1. authenticate and derive Master Key

2. derive Pairwise Master Key (PMK)

3. Use PMK to enforce 802.11 channel access: derive and use PTK

- MK ≠ PMK or AP could make access control decision instead of AS
- MK is fresh and bound to the session between STA and AS
- PMK is bound to this STA and this AP
Key derivation

- STA and AS derive a PMK (Pairwise Master Key)
  - As a result of EAP protocols (TLS for example)
  - From a manually configured master key (MK)... 
- AS then sends PMK to the AP
  - Via Radius protocol 
- Four separate keys (the PTK: Pairwise Transient Keys) are then derived by AP and STA 
  - Data Encryption key 
  - Data Integrity key 
  - EAPOL-Key Encryption key 
  - EAPOL-Key Integrity key

*Note: In PSK mode (pre-shared key) mode, the AS=AP and the Master key is enter manually + shared by all users.*
1. Client sends request for association and security negotiation to AP.
2. AP passes request to Authentication Server (RADIUS).
3. RADIUS authenticates client.
4. AS (Radius server) and client initiate 4 way key negotiation to create unique session key (PMK). AS pushes key, which is AES encrypted to AP. AES encrypts all data traffic.
RADIUS: An overview

- Remote Access Dial-In User Service
- Was designed for Dial-in Users authentication...not 802.11
- Dial-In User
  - PPP (Point-to-Point protocol) is used between user and ISP modem (POP: point of presence)
  - PPP contains authentication methods...
  - but since ISP have many POP, they don’t want to keep a copy of their database at every PoP...
  - they create a central database (RADIUS server or authorizer)
RADIUS: An overview (2)

- 802.11i uses the same concept but
  - NAS/POP is the access point
  - PPP is replaced by EAP that is more flexible (see next slide)
STA communicates with AS using **EAP (Extensible Authentication Protocol)**

- **EAP (Extensible Authentication Protocol) [RFC 3748]**
  - carrier protocol designed to transport the messages of “real” authentication protocols (e.g., TLS)
  - very simple, four types of messages:
    - EAP request – carries messages from the supplicant to the authentication server
    - EAP response – carries messages from the authentication server to the supplicant
    - EAP success – signals successful authentication
    - EAP failure – signals authentication failure
- Authenticator (AP) doesn’t understand what is inside the EAP messages, it recognizes only EAP success and failure
106

802.11i/RSN: the protocols (RADIUS)

- AP and AS communicates using **RADIUS protocol**
  - RADIUS (Remote Access Dial-In User Service) [RFC 2865-2869, RFC 2548]
  - used to carry EAP messages between the AP and the AS
  - RADIUS is mandated by WPA and optional for RSN
EAP in action

STA

encrypted in EAPOL

EAP Request (Identity)

EAP Response (Identity)

EAP Request 1

EAP Response 1

.....

EAP Request n

EAP Response n

EAP Success

AP

encrypted in RADIUS

EAP Request 1

EAP Response 1

.....

EAP Request n

EAP Response n

EAP Success

auth server
Protocols – LEAP, EAP-TLS, PEAP, EAP-SIM

• LEAP (Light EAP)
  – developed by Cisco
  – similar to MS-CHAP extended with session key transport

• EAP-TLS (TLS over EAP)
  – only the TLS Handshake Protocol is used
  – server and client authentication, generation of master secret
  – TLS master secret becomes the session key
  – mandated by WPA, optional in RSN
• PEAP (Protected EAP)
  – phase 1: TLS Handshake without client authentication
  – phase 2: client authentication protected by the secure channel established in phase 1
• EAP-SIM
  – extended GSM authentication in WiFi context
  – protocol (simplified):
    STA $\rightarrow$ AP: EAP res ID (IMSI / pseudonym)
    STA $\rightarrow$ AP: EAP res (nonce)
    AP: [gets two auth triplets from the mobile operator’s AuC]
    AP $\rightarrow$ STA: EAP req (2*RAND | MIC$_{2*Kc}$ | {new pseudonym}$_{2*Kc}$)
    STA $\rightarrow$ AP: EAP res (2*SRES)
    AP $\rightarrow$ STA: EAP success
Summary of the protocol architecture

- TLS (RFC 2246)
- EAP-TLS (RFC 2716)
- EAP (RFC 3748)
- EAPOL (802.1X)
- EAP over RADIUS (RFC 3579)
- RADIUS (RFC 2865)
- TCP/IP
- 802.3 or else

mobile device  AP  auth server
## In summary: WEP vs. WPA vs. WPA2

<table>
<thead>
<tr>
<th></th>
<th>WEP</th>
<th>WPA</th>
<th>WPA2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encryption</strong></td>
<td>RC4</td>
<td>RC4</td>
<td>AES</td>
</tr>
<tr>
<td><strong>Key rotation</strong></td>
<td>None</td>
<td>Dynamic</td>
<td>Dynamic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>session keys</td>
<td>session keys</td>
</tr>
<tr>
<td><strong>Key distribution</strong></td>
<td>Manually</td>
<td>Automatic</td>
<td>Automatic</td>
</tr>
<tr>
<td></td>
<td>typed into</td>
<td>distribution</td>
<td>distribution</td>
</tr>
<tr>
<td></td>
<td>each device</td>
<td>available</td>
<td>available</td>
</tr>
<tr>
<td><strong>Authentication</strong></td>
<td>Uses WEP</td>
<td>Can use 802.1x</td>
<td>Can use 802.1x</td>
</tr>
<tr>
<td></td>
<td>key as AuthC</td>
<td>&amp; Pre-shared</td>
<td>&amp; Pre-shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>key</td>
<td>key</td>
</tr>
</tbody>
</table>
Summary

- the new security standard for WiFi is 802.11i
  - access control model is based on 802.1X
  - flexible authentication based on EAP and upper layer authentication protocols (e.g., TLS, GSM authentication)
  - improved key management
  - TKIP
    - uses RC4 → runs on old hardware
    - corrects WEP’s flaws
    - mandatory in WPA, optional in RSN (WPA2)
  - AES-CCMP
    - uses AES in CCMP mode (CTR mode and CBC-MAC)
    - needs new hardware that supports AES
Recommended readings