Security in WiFi networks

Cryptographic Protocols (EIT ICT MSc)

Dr. Levente Buttyán
associate professor
BME Hálózati Rendszerek és Szolgáltatások Tanszék
Lab of Cryptography and System Security (CrySyS)
buttyan@hit.bme.hu, buttyan@crysys.hu
Outline

- reminder on the operation of WiFi networks
- MAC address filtering and SSID based access control
- WEP – operation and attacks
- 802.11i (WPA/TKIP, WPA2/AES-CCMP)
Brief reminder on WiFi networks

STA

AP

association request

association response

“connected”

scanning on each channel

STA

association request

association response

AP

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

scanning on each channel
SSID-based access control

- SSID = Service Set IDentifier (network name)
  - a 32-character unique identifier (differentiates one WLAN from another)
  - found in the header of packets and acts as a “password” when a mobile device tries to connect to the WLAN

- unfortunately, the SSID can be sniffed, and hence, this mechanism does not provide really secure access control
MAC filtering based access control

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

- unfortunately, MAC addresses can be sniffed and forged
  - sniffing
    - MAC address is sent in clear in each packet
    - put your WLAN adapter card in promiscuous mode (accepts all packets)
    - eavesdrop the traffic and find out which MAC addresses are accepted
  - forging
    • MAC address of certain WLAN adapter cards can be set by the user
    • example: # ifconfig ath0 hw ether <mac address of C>
Eavesdropping with Wireshark
Security problems in wireless

- no inherent physical protection
  - physical connections between devices are replaced by logical associations
  - sending and receiving messages do not need physical access to the network infrastructure (cables, hubs, routers, etc.)

- broadcast communications
  - wireless usually means radio, which has a broadcast nature
  - transmissions can be overheard by anyone in range
  - anyone can generate transmissions,
    - which will be received by other devices in range
    - which will interfere with other nearby transmissions and may prevent their correct reception (jamming)

- eavesdropping is easy
- injecting bogus messages into the network is easy
- replaying previously recorded messages is easy
- illegitimate access to the network and its services is easy
- denial of service is easily achieved by jamming
Security in WiFi networks

Wireless com sec requirements

- **confidentiality**
  - messages sent over wireless links must be encrypted

- **authenticity**
  - origin of messages received over wireless links must be verified

- **replay detection**
  - freshness of messages received over wireless links must be checked

- **integrity**
  - modifying messages on-the-fly (during radio transmission) is not so easy, but possible
  - integrity of messages received over wireless links must be verified

- **access control**
  - access to the network services should be provided only to legitimate entities
  - access control should be permanent
    - it is not enough to check the legitimacy of an entity only when it joins the network and its logical associations are established, because logical associations can be hijacked

- **protection against jamming**
WEP – Wired Equivalent Privacy

- part of the original IEEE 802.11 specification

- goal
  - make the WiFi network *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 has achieved no security at all

- services
  - access control to the network
  - message confidentiality
  - message authenticity/integrity
WEP – Access control

- before association, the STA needs to authenticate itself to the AP

- authentication is based on a simple challenge-response protocol:
  - STA $\rightarrow$ AP: authenticate request
  - AP $\rightarrow$ STA: authenticate challenge ($r$) // $r$ is 128 bits long
  - STA $\rightarrow$ AP: authenticate response ($e_K(r)$)
  - AP $\rightarrow$ STA: authenticate success/failure

- once authenticated, the STA can send an association request, and the AP will respond with an association response

- if authentication fails, no association is possible
WEP – Message conf and integrity

- WEP encryption is based on the RC4 stream cipher
  - it is essential that each message is encrypted with a different key stream
  - the RC4 cipher is initialized with a shared secret key and an IV (initial value)
    - shared secret key (40 or 104 bits) remains unchanged
    - 24-bit IV is changed for every message sent
  - RC4 produces a pseudo-random byte sequence (key stream), which is XORed to the message
  - reception is analogous

- WEP integrity protection is based on an encrypted CRC value
  - ICV (integrity check value) is computed and appended to the message
  - the message and the ICV are encrypted together as described above
WEP – Message conf and integrity

1. **Message + ICV**
   - **IV**
   - secret key
   - **RC4**

2. **Encode**
   - **IV**
   - **message + ICV**

3. **Decode**
   - **IV**
   - secret key
   - **RC4**

   - **message + ICV**
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

- hence, WEP supports multiple default keys to help the smooth change of keys
  - one of the keys is called the active key
  - the active key is used to encrypt messages
  - any key can be used to decrypt messages
  - the message header contains a key ID that allows the receiver to find out which key should be used to decrypt the message
WEP – The key change process

1. abc*  →  abc*  
   ---  →  ---  

2. abc*  →  abc*  →  abc  
   ---  →  def  →  def*  

3. ---  →  abc  
   def*  →  def*  

4. ---  →  ---  
   def*  →  def*  

* active key
WEP flaws – Auth and access control

- authentication is one-way only
  - AP is not authenticated to STA
  - STA may associate to a rogue AP which can perform a Man-in-the-Middle attack

- the same shared secret key is used for authentication and encryption
  - weaknesses in any of the two protocol 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 can 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
WEP flaws – Auth and access control

- recall that authentication is based on a challenge-response protocol:
  
  \[ \text{AP} \rightarrow \text{STA}: r \]
  
  \[ \text{STA} \rightarrow \text{AP}: \text{IV} | r \oplus K \]

  where \( K \) is a 128 bit RC4 output on IV and the shared secret

- an attacker can compute \( r \oplus (r \oplus K) = K \)

- she can use \( K \) (and the same IV) to impersonate STA later:
  
  \[ \text{AP} \rightarrow \text{attacker}: r' \]
  
  \[ \text{attacker} \rightarrow \text{AP}: \text{IV} | r' \oplus K \]

  ...
WEP flaws – Integrity and replay

- there’s no replay protection at all
  - IV is not mandated to be incremented after each message
  - receiver is not mandated to check the freshness of received IVs

- attacker can manipulate messages despite the ICV mechanism and encryption
  - CRC is a linear function wrt to XOR:
    \[
    \text{CRC}(X \oplus Y) = \text{CRC}(X) \oplus \text{CRC}(Y)
    \]
    - attacker observes \((M \mid \text{CRC}(M)) \oplus K\) where \(K\) is the RC4 output
    - for any \(\Delta M\), the attacker can compute \(\text{CRC}(\Delta M)\)
    - hence, the attacker can compute:
      \[
      ((M \mid \text{CRC}(M)) \oplus K) \oplus (\Delta M \mid \text{CRC}(\Delta M)) = \\
      ((M \oplus \Delta M) \mid (\text{CRC}(M) \oplus \text{CRC}(\Delta M))) \oplus K = \\
      ((M \oplus \Delta M) \mid \text{CRC}(M \oplus \Delta M)) \oplus K
      \]
WEP flaws – Confidentiality

- IV reuse
  - IV space is too small
    - IV size is only 24 bits → there are 16,777,216 possible IVs
    - after around 17 million messages, IVs are reused
    - a busy AP at 11 Mbps is capable for transmitting 700 packets per second → IV space is used up in around 7 hours
  - in many implementations IVs are initialized with 0 on startup and then incremented by one after each message
    - if several devices are switched on nearly at the same time, they all use the same sequence of IVs
    - if they all use the same default key (which is the common case), then IV collisions are readily available to an attacker

- exploiting weaknesses in the RC4 cipher
  - see next slides …
Operation of RC4

- **initialization:**
  
  ```
  for i = 0 to 255 do
    S[i] = i
  end

  j = 0
  for i = 0 to 255 do
    j = j+S[i]+K[i mod len(K)] mod 256
    swap(S, i, j)
  end

  i = 0
  j = 0
  ```

- **generation:**
  
  ```
  i = i+1 mod 256
  j = j+S[i] mod 256
  swap(S, i, j)
  return S[ S[i]+S[j] mod 256 ]
  ```
FMS (Fluhrer-Mantin-Shamir) attack

- notation:
  - \((S_k, i_k, j_k)\) is the state after the k-th step of the initialization algorithm
  - \((S_{256}, i_{256}=0, j_{256}=0)\) is the state before the generation starts

- in WEP, the attacker knows the first 3 bytes of K \((K[0..2] = IV)\)
- the attacker can simulate the execution of the RC4 initialization for the first three steps, and hence, knows \(S_3\) and \(j_3\)
- in the next step \((i_4 = 3)\):
  \[
  \]
  \[
  \]
- since S is a permutation, \(S[a] = b\) implies \(S^{-1}[b] = a\)
- thus, we have:
  \[
  \]
- if we knew \(S_4[3]\), we would be able to compute \(K[3]\) (first byte of the key)
FMS attack

- let’s assume that
  - \( S_{256}[1] = S_{3}[1] \), \( S_{256}[S_{256}[1]] = S_{3}[S_{3}[1]] \), and \( S_{256}[3] = S_{4}[3] \) (i.e., \( S[1] \), \( S[S[1]] \), and \( S[3] \) do not participate in any more swaps in the rest of the initialization procedure)
  - \( S_{3}[1] < 3 \), \( S_{3}[1] + S_{3}[S_{3}[1]] = 3 \) (note that these can be tested, because \( S_{3} \) is known to the attacker)

- let’s compute the first key stream byte:
  - \( i = 1 \), \( j = S_{256}[1] \)
  - swap \( S_{256}[1] \) and \( S_{256}[S_{256}[1]] \)
    - their sum is still \( S_{256}[1] + S_{256}[S_{256}[1]] = S_{3}[1] + S_{3}[S_{3}[1]] = 3 \)
  - output \( X = S_{256}[3] = S_{4}[3] \)

- hence the first key stream byte will give us \( S_{4}[3] \), which reveals \( K[3] \)
FMS attack

- what is the probability that S[1], S[S[1]], and S[3] do not participate in any swaps in the remaining 253 steps of the initialization?
  - note that S_3[1] < 3 (by assumption), and i > 3 (we passed the first 3 steps)
  - S[1], S[S[1]], and S[3] may be swapped only if j takes the value 1, S[1], or 3, respectively (i will not take these values anymore)
  - if we assume that j changes randomly, then it takes any value with probability 1/256
  - hence, we get \( \left( \frac{253}{256} \right)^{253} \approx 0.0584 \)

- if the conditions S_3[1] < 3 and S_3[1] + S_3[S_3[1]] = 3 are satisfied, then we compute the correct value of K[3] with probability 0.0584
- otherwise the value that we compute for K[3] can be any value with probability 1/255 \( \approx 0.004 \)
- if we repeat this experiment, then the most frequent value that we compute for K[3] is the correct value!
Example: RC4 initialization

\[
\text{for } i = 0 \text{ to } 7 \text{ do }
\]
\[
j = j + S[i] + K[i] \mod 8
\]
\[
\text{swap } (S, i, j)
\]
\[
\text{end}
\]

\[
S_0 = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

\[
K = \begin{array}{cccccccc}
3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \\
\end{array}
\]

\[
\text{for } i = 0 \text{ to } 7 \text{ do }
\]
\[
i = 0
\]
\[
j_1 = 3
\]
\[
j = 0
\]
Example: RC4 initialization

\[
\text{for } i = 0 \text{ to } 7 \text{ do } \\
\quad j = j + S[i] + K[i] \mod 8 \\
\quad \text{swap } (S, i, j) \\
\text{end}
\]

\[
S_1 = \begin{bmatrix}
3 & 1 & 2 & 0 & 4 & 5 & 6 & 7
\end{bmatrix}
\]

\[
K = \begin{bmatrix}
3 & 7 & 1 & 5 & 7 & 1 & 0 & 3
\end{bmatrix}
\]
Example: RC4 initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```

\[ S_2 = \begin{array}{ccccccc}
3 & 0 & 2 & 1 & 4 & 5 & 6 & 7 \\
\end{array} \]

\[ K = \begin{array}{ccccccc}
3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \\
\end{array} \]

\[ \begin{align*}
S_2 &= \begin{array}{ccccccc}
3 & 0 & 2 & 1 & 4 & 5 & 6 & 7 \\
\end{array} \\
S_2 &= \begin{array}{ccccccc}
3 & 0 & 2 & 1 & 4 & 5 & 6 & 7 \\
\end{array} \\
S_2 &= \begin{array}{ccccccc}
3 & 0 & 2 & 1 & 4 & 5 & 6 & 7 \\
\end{array} \\
\end{align*} \]

\[ j = 0 \]

\[ i = 0 \quad j_1 = 3 \]

\[ i = 1 \quad j_2 = 3 \]

\[ i = 2 \quad j_3 = 6 \]
Example: RC4 initialization

for $i = 0$ to $7$ do
    $j = j + S[i] + K[i] \mod 8$
    swap ($S$, $i$, $j$)
end

$S_3 = \begin{bmatrix} 3 & 0 & 6 & 1 & 4 & 5 & 2 & 7 \end{bmatrix}$

$K = \begin{bmatrix} 3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \end{bmatrix}$

$j = 0$

$i = 0 \quad j_1 = 3$

$i = 1 \quad j_2 = 3$

$i = 2 \quad j_3 = 6$

$i = 3 \quad j_4 = 4$
Example: RC4 initialization

\[
\begin{align*}
\text{for } i &= 0 \text{ to } 7 \text{ do} \\
& \quad j = j + S[i] + K[i] \mod 8 \\
& \quad \text{swap } (S, i, j) \\
\end{align*}
\]

\(S_4 = \begin{bmatrix} 3 & 0 & 6 & 4 & 1 & 5 & 2 & 7 \end{bmatrix}\)

\(K = \begin{bmatrix} 3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \end{bmatrix}\)

\(i = 0 \quad j_1 = 3\)

\(i = 1 \quad j_2 = 3\)

\(i = 2 \quad j_3 = 6\)

\(i = 3 \quad j_4 = 4\)

\(i = 4 \quad j_5 = 4\)
Example: RC4 initialization

\[
\begin{align*}
S_5 &= \begin{bmatrix} 3 & 0 & 6 & 4 & 1 & 5 & 2 & 7 \end{bmatrix} \\
K &= \begin{bmatrix} 3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{for } i &= 0 \text{ to } 7 \text{ do} \\
& \quad j = j + S[i] + K[i] \mod 8 \\
& \quad \text{swap } (S, i, j) \\
\text{end}
\end{align*}
\]
Example: RC4 initialization

\[
\begin{align*}
S_6 &= [3, 0, 5, 4, 1, 6, 2, 7] \\
K &= [3, 7, 1, 5, 7, 1, 0, 3]
\end{align*}
\]

\[
\begin{align*}
&\text{for } i = 0 \text{ to } 7 \text{ do} \\
&\quad j = j + S[i] + K[i] \mod 8 \\
&\quad \text{swap } (S, i, j) \\
&\text{end}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{c}
\text{i = 0 } \quad j = 0 \\
\text{i = 1 } \quad j_1 = 3 \\
\text{i = 2 } \quad j_2 = 3 \\
\text{i = 3 } \quad j_3 = 6 \\
\text{i = 4 } \quad j_4 = 4 \\
\text{i = 5 } \quad j_5 = 4 \\
\text{i = 6 } \quad j_6 = 2 \\
\text{i = 7 } \quad j_7 = 4
\end{array}
\end{align*}
\]
Example: RC4 initialization

for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
Example: RC4 generation

\[ S_8 = \begin{bmatrix} 3 & 0 & 5 & 4 & 2 & 6 & 7 & 1 \end{bmatrix} \]

\[ K = \begin{bmatrix} 3 & 7 & 1 & 5 & 7 & 1 & 0 & 3 \end{bmatrix} \]

\[ \text{output } X = 4 \]

\[ K[3] = S_3^{-1}[X] - j_3 - S_3[3] = 4 - 6 - 1 = 5 \]
FMS attack – the full monty

- Assume that we know $K[0..c]$
- Observe the traffic and pick messages for which the FMS conditions ($S_c[1] < c$, $S_c[1] + S_c[S_c[1]] = c$) hold
- For each message, guess the first byte of the message (and hence the first byte $X$ of the key stream used for encrypting the message)
- Compute a candidate value for $K[c]$ as $S_c^{-1}[X] - j_c - S_c[c]$
- Select the most frequent candidate $\rightarrow K[c]$
- Increment $c$ and repeat

- When you have a full $K$, try it by decrypting the eavesdropped messages
- If it does not work, then go back and select the second most frequent candidate for $K[c]$ in the last step
- …
FMS attack – success rate

Chopchop attack (KoreK)

- allows an attacker to interactively decrypt the last $m$ bytes of plaintext of a WEP encrypted packet by sending $128m$ crafted packets on average to the network, and observing if they are accepted or not (ICV is correct or not)
- does not reveal the root key!
- not based on any special properties of the RC4 stream cipher (protocol flaw!)
Chopchop attack – background

- CRC verification mechanism:
  - (message, ICV) is represented as a binary polynomial $P$
  - if $P \mod R_{CRC} = P_{ONE}$, then the message is accepted, where
    - $R_{CRC}$ is the given CRC polynomial
    - $P_{ONE}$ is polynomial with all coefficients equal to one

- we can write $P$ as $Qx^8 + L$, where
  - $Q$ represents the one byte shortened packet
  - $L$ represents the last byte

- if $P$ verifies correctly then how do we need to modify $Q$ such that it verifies correctly too?

$$
\begin{align*}
P \mod R_{CRC} &= P_{ONE} \\
Qx^8 \mod R_{CRC} &= P_{ONE} + L \mod R_{CRC} \\
Q \mod R_{CRC} &= (P_{ONE} + L)(x^8)^{-1} \mod R_{CRC} \\
\Delta Q &\leftarrow P_{ONE} + (P_{ONE} + L)(x^8)^{-1}
\end{align*}
$$
Chopchop attack – background

- Q+ΔQ verifies correctly and ΔQ depends only on L
  \[ Q + \Delta Q \mod R_{CRC} = Q + P_{ONE} + (P_{ONE} + L)(x^8)^{-1} \mod R_{CRC} = P_{ONE} \]
  because \( Q + (P_{ONE} + L)(x^8)^{-1} \mod R_{CRC} = 0 \)

- due to RC4 stream ciphering, anything can be added to an encrypted packet without knowing the encryption key
Chopchop attack – the full monty

- eavesdrop a WEP encrypted packet $P + K$
  - we know that $P$ verifies correctly, but we don’t know $P$
- guess the last byte $L$ of $P$
  - there are only 256 possibilities!
- chop the last byte of the encrypted packet and XOR in $\Delta Q$ to get $(Q + K') + \Delta Q = (Q + \Delta Q) + K'$
- send $(Q + \Delta Q) + K'$ to the AP and observe if it is accepted
  - send the packet from a station not yet associated to the AP
  - if the packet is correct, the AP will send a message telling the station that it needs to rejoin the network, otherwise the packet is discarded
- if unsuccessful, try another candidate for $L$
  - on average, after 128 trials you have the correct value for $L$
- repeat the procedure by chopping $Q + \Delta Q$ …
Tools

- there are many available on the web
- look at http://www.aircrack-ng.org for a comprehensive cracking toolbox
WEP – Lessons learnt

1. engineering security protocols is a very risky business
   • you may combine otherwise strong 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 in a way done in WEP
   • example:
     – encrypting a message digest to obtain an ICV is acceptable
     – but it doesn’t work if the message digest function is linear wrt to the encryption function
   • using an expert 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)
Overview of 802.11i

- after the collapse of WEP, IEEE started to develop a new security architecture → 802.11i (now integrated in 802.11)

- 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 shared 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
  - replay protection is added
  - encryption function is improved
802.11i, WPA, and WPA2

- WPA (WiFi protected access)
  - industrial name for 802.11i TKIP (Temporal Key Integrity Protocol)
  - 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)

- WPA2
  - industrial name for 802.11i AES-CCMP (Counter mode and CBC-MAC Protocol)
  - integrity protection and encryption is based on AES (in CCMP mode)
  - nice solution, but needs new hardware
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
Protocols: EAP, EAPOL, RADIUS

- **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 authentication server to the supplicant
    - EAP response – carries messages from the supplicant to the authentication server
    - EAP success – signals successful authentication
    - EAP failure – signals authentication failure
  - authenticator doesn’t understand what is inside the EAP messages, it recognizes only EAP success and failure

- **EAPOL (EAP over LAN)** [802.1X]
  - used to encapsulate EAP messages into LAN protocols (e.g., Ethernet)
  - EAPOL is used to carry EAP messages between the STA and the AP

- **RADIUS (Remote Access Dial-In User Service)** [RFC 2865-2869, RFC 2548]
  - used to carry EAP messages between the AP and the auth server
  - MS-MPPE-Recv-Key attribute is used to transport the session key from the auth server to the AP
  - RADIUS is mandated by WPA and optional for WPA2
EAP in action

**Diagram:**

- STA (Station)
  - EAP Request (Identity)
  - EAP Response (Identity)
  - EAP Request 1
  - EAP Response 1
  - EAP Request n
  - EAP Response n
  - EAP Success

- AP (Access Point)
  - EAP Request 1
  - EAP Response 1
  - EAP Request n
  - EAP Response n
  - EAP Success

- Auth Server
  - EAP Request (Identity)
  - EAP Response (Identity)
  - EAP Request 1
  - EAP Response 1
  - EAP Request n
  - EAP Response n
  - EAP Success

- EAP requests and responses are encapsulated in EAPOL and RADIUS.
Protocols: EAP-TLS, EAP-TTLS, …

- **EAP-TLS (TLS over EAP) [RFC 5216]**
  - 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 WPA2

- **EAP-TTLS (Tunneled TLS over EAP) [RFC 5281]**
  - phase 1: TLS Handshake possibly without client authentication
  - phase 2: legacy client authentication (e.g., password based) protected by the secure tunnel established in phase 1
    - eavesdropping and man-in-the-middle attacks are prevented
    - privacy is improved (user name is also encrypted)

- **EAP-PSK, EAP-FAST, EAP-PEAP, EAP-SIM, EAP-AKA, …**
Protocol architecture

- TLS (RFC 5246)
- EAP-TLS (RFC 5216)
- EAP (RFC 5247)
- EAPOL (802.1X)
- EAP over RADIUS (RFC 3579)
- RADIUS (RFC 2865)
- UDP/IP
- 802.3 or else

mobile device  AP  auth server
Key hierarchy overview

802.1X authentication

PMK (pairwise master key)
  - key derivation in STA and AP

PTK (pairwise transient keys)
  - key encryption key
  - key integrity key
  - data encryption key
  - data integrity key

GTK (group transient keys)
  - group encryption key
  - group integrity key

GMK (group master key)
  - key derivation in AP
  - transport to every STA

random generation in AP

unicast message trans. between STA and AP

broadcast messages trans. from AP to STAs

Protection

802.1X authentication

random generation in AP

Protection
Four way handshake (simplified)

Objective:
- Prove that AP also knows the PMK (result of the authentication between STA and Auth Server)
- Exchange random values to be used in the generation of PTK
- Distribution of GTK

STA

- ANonce, SeqNum
- Generate ANonce
- Generate SNonce, and compute PTK

AP

- ANonce, SeqNum
- Generate ANonce
- Compute PTK and GTK
- SeqNum+1, E_{KEK}(GTK), MIC_{KIK}(…)
- SeqNum+1, MIC_{KIK}(…)

MIC() and E(): HMAC-MD5 and RC4 or HMAC-SHA and AES
PTK and GTK computation for TKIP

PRF-512( PMK,
    “Pairwise key expansion”,
    MAC1 | MAC2 | Nonce1 | Nonce2 ) =
= KEK | KIK | DEK | DIK

PRF-256( GMK,
    “Group key expansion”,
    MAC | GNonce ) =
= GEK | GIK

see TLS slides for the definition of PRF …
PTK and GTK comp for AES-CCMP

PRF-384( PMK,
    “Pairwise key expansion”,
    MAC1 | MAC2 | Nonce1 | Nonce2 ) =
= KEK | KIK | DE&IK

PRF-128( GMK,
    “Group key expansion”,
    MAC | GNonce ) =
= GE&IK
TKIP

- runs on old hardware (supporting RC4), but …
- WEP weaknesses are corrected
  - new message integrity protection mechanism called Michael
    - old CRC based ICV is still there
  - IV is used as replay counter too
  - IV length is increased to 48 bits in order to prevent IV reuse
  - per-packet keys are used to prevent attacks similar to that of the FMS attack
TKIP – Michael

- severe design constraints
  - microprocessor inside most Wi-Fi cards is not powerful enough for strong cryptographic operations
  - moving the MIC computation from the hardware into software (device driver) is an option, but APs do not have powerful CPUs → MIC function should still be simple

- design decisions
  - Michael operates at SDU level (before message is fragmented into PDUs)
    - can be implemented in the device driver (software update)
    - reduced overhead (no MIC computation for each and every PDU)
  - Michael uses simple operations (substitution, rotation, XOR)
    - fits in existing APs
  - Michael is not very strong (equivalent to 20-bit security), hence, vulnerable to brute force attacks
    - if a MIC failure is detected, then a MIC failure report frame is sent
    - if two MIC failures are detected within less than 60 seconds, the communication is shut down, and keys are renegotiated after a 60 second penalty period
TKIP – RC4 seed generation

- how to fit 48+128 bits into a 128 bit RC4 seed supported by the hardware?
- a crypto hash function would be too expensive → special mix function (based on simple operations)
- two phases for efficiency reasons
  - result of the first phase can be used for \(2^{16}\) packets
- MAC address
  - same IV may be used by another device with no problem
- dummy byte
  - repeat of the first byte, except that bit 5 is forced to 1 and bit 4 is forced to 0
  - prevents the generation of the major class of weak RC4 keys
Attacks on TKIP

- in a recent paper (ACM WiSec, Zurich, March 16-18 2009), Beck and Tews published a chopcop attack against WPA

- background
  - Michael is not a one-way function: given a (message, MIC) pair, it is not difficult to recover the MIC key \(\rightarrow\) encryption of the MIC in WPA is essential
  - old CRC-based ICV mechanism is still used
    - last 12 bytes of a plaintext packet is the 8-byte MIC and the 4-byte CRC
    - if ICV is not correct, the packet is discarded silently
    - if ICV is correct, but MIC verification fails, a MIC failure report is sent
    - IV value is not incremented in any of these cases
  \(\rightarrow\) stations can be used as CRC oracles!

- attacker can break the MIC key and recover a key stream (RC4 output) for some IV (not chosen by the attacker)
BT attack on WPA – details

 assumptions:
• IPv4 is used, and the attacker knows most bytes of the used IP range (e.g., 192.168.0.X)
• TKIP re-keying period is sufficiently long

 attacker observes an encrypted ARP packet (sent by the AP)
• can identify ARP packets from their characteristic length
• ARP requests are sent to the broadcast address

 note that attacker knows large part of the plaintext; what she does not know is the last byte of the IP addresses, and the MIC and the ICV values (12 bytes)

 attacker chops the last byte, and using a guess for the last plaintext byte, modifies the chopped packet like in the KoreK chopchop attack against WEP

 modified packet is sent to the station, and response is observed
• no response means wrong guess (CRC is incorrect)
• MIC failure means right guess

 repeat until last byte is figured out (no need to wait between trials !)
BT attack on WPA – details

- wait 1 minute after the MIC failure, and repeat the procedure with the remaining (chopped) packet
  → this gives the last 12 bytes (MIC and ICV)

- missing parts of the IP addresses are guessed
  • each candidate is verified against the known CRC

- at this point, the entire ARP packet, the MIC, and the CRC are recovered → this gives the key stream

- from the (message, MIC) pair, the attacker can also recover the MIC key

- now, the attacker can craft a message, compute its MIC and ICV, encrypt it with the key stream, and send it to the station (with the previously observed IV)
  • however, key stream is IV specific, which functions as a sequence counter
  • therefore, the attacker must prevent the station from receiving the original ARP packet and any other packet during the attack (e.g., by constant jamming)
AES-CCMP

- CCMP means CTR mode and CBC-MAC Protocol
  - authenticated encryption mode
  - integrity protection is based on CBC-MAC (using AES)
  - encryption is based on CTR mode (using AES)

- CBC-MAC
  - computed over the MAC header, CCMP header, and the MPDU (fragmented data)
  - mutable fields are set to zero
  - input is padded with zeros if length is not multiple of 128 (bits)
  - CBC-MAC initial block:
    - flag (8)
    - priority (8)
    - source address (48)
    - packet number (48)
    - data length (16)
  - final 128-bit block of CBC encryption is truncated to (upper) 64 bits to get the CBC-MAC value

- CTR mode encryption
  - MPDU and CBC-MAC value is encrypted, MAC and CCMP headers are not
  - format of the counter is similar to the CBC-MAC initial block
    - “data length” is replaced by “counter”
    - counter is initialized with 1 and incremented after each encrypted block
Summary

- security has always been considered important for WiFi
- early solution was based on WEP
  - seriously flawed!
  - not recommended to use anymore
- the new security standard for WiFi is 802.11i (WPA and WPA2)
  - 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
  - WPA (TKIP)
    - uses RC4 → runs on old hardware, but corrects (most of) WEP’s flaws
    - WPA has been recently broken! (paper at ACM WiSec 2009, March 16-18)
  - AES-CCMP
    - uses AES in CCMP mode (an authenticated encryption mode)
    - needs new hardware that supports AES
Recommended readings


