Predicting, Decrypting, and Abusing WPA2/802.11 Group Keys

Mathy Vanhoef and Frank Piessens, iMinds-DistriNet, KU Leuven
USENIX Security 2016
Security of Wi-Fi group keys?

Protect broadcast and multicast Wi-Fi frames:

- All clients share a copy of the group key

Security of groups keys not yet properly investigated!

- In contrast with preshared & pairwise keys …

Analyze security of group key during its full lifetime!
Contributions: Security of Group Keys

- Flawed generation
- Force RC4 in handshake
- Inject & decrypt all traffic
- New Wi-Fi tailored RNG
Contributions: Security of Group Keys

- Flawed generation
- Force RC4 in handshake
- Inject & decrypt all traffic
- New Wi-Fi tailored RNG
How are group keys generated?

Group key hierarchy:
- AP generates public counter and secret master key
- Derive group temporal keys (GTKs)

Entropy only introduced at boot
- If master key is leaked, all group keys become known
How are random numbers generated?

802.11 standard has example Random Number Generator

- §11.1.6a: “… can generate cryptographic-quality randomness”
- Annex M.5: “This solution is expository only”

Inconsistent description of RNG’s security guarantees!

- How secure is the design of the 802.11 RNG?
- How many platforms implement this RNG?
802.11 RNG: Main Design

The 802.11 RNG is a stateless function returning 32 bytes

- Collects entropy on-demand
- Entropy extracted from frame arrival times and clock jitter

Deviates from traditional RNG design:

- No entropy pools being maintained
- Entropy only extracted from events when the RNG is being invoked
802.11 RNG: Entropy sources

Frame arrival times:
- Collected by repeatedly starting & aborting 4-way handshake
- Problem: AP is blacklisted after several handshake failures

Clock jitter and drift:
- Note: Router’s current time is leaked in beacons
- Problem: No minimum time resolution → small clock jitter
Surely no one implemented this...?

Weakened 802.11 RNG

Depends on OS
Surely no one implemented this…?

Weakened 802.11 RNG

Depends on OS
MediaTek RNG: Linux-based APs

Uses custom Linux drivers:

- Implements 802.11’s RNG using only clock jitter
- Uses *jiffies* for current time: at best millisecond accuracy

RT-AC51U

OpenCL

~3 mins

GMK & GTK
Contributions: Security of Group Keys

- Flawed generation
- Force RC4 in handshake
- Inject & decrypt all traffic
- New Wi-Fi tailored RNG
Simplified 4-way handshake

Client

Beacons: supported ciphers in IEs

Select cipher

Association Request: chosen cipher

Msg1: ANonce

PTK

Msg2: SNonce

Msg3: IEs + GTK

verify IEs

Msg4: ACK

Access Point

PTK
Simplified 4-way handshake

- Beacons: supported ciphers in IEs
- Select cipher
- Association Request: chosen cipher
- Msg1: ANonce
- PTK
- Msg2: SNonce
- Msg3: IEs + GTK
- verify IEs
- Msg4: ACK

Group key encrypted and transmitted ...

... before downgrade attack detection!
Simplified 4-way handshake

Client

Access Point

Beacons: supported ciphers in IEs

Select cipher

Association Request: chosen cipher

Pairwise Cipher

WPA-TKIP

AES-CCMP

GTK encryption

RC4

AES Key Wrap

Group key encrypted and transmitted ...

... before downgrade attack detection!

Msg2: GTK

Msg3: IEs + GTK

Msg4: ACK

verify IEs

PTK
**Downgrade attack**

1. Rogue AP: Only advertise WPA-TKIP

2. Client picks WPA-TKIP

   **Select cipher**

   Association Request: chosen cipher

   **Msg1**: ANonce

   **PTK**

   **Msg2**: S2TK

   **Msg3**: IEs + GTK

   **verify IEs**

   **Msg4**: ACK

3. Encrypted with RC4!

4. Rogue AP detected
Attacking RC4 encryption of GTK

- RC4 Key: 16-byte IV || 16-byte secret key
- First 256 keystream bytes are dropped

Recover repeated encryptions of GTK:
- Requires $\sim 2^{31}$ handshakes: takes $>50$ years

Countermeasures:
- Disable WPA-TKIP & RC4
- Send GTK after handshake
Contributions: Security of Group Keys

- Flawed generation
- Force RC4 in handshake
- Inject & decrypt all traffic
- New Wi-Fi tailored RNG
Abusing the group key: Hole 196?

- Inject unicast IP packet in broadcast Wi-Fi frame
- Detected by “Hole 196” check

Hole 196 check done at network-layer…
… but an AP works at link-layer!
Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

<table>
<thead>
<tr>
<th>Sender</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
</table>

Victim | Sender

Attacker | Data

AP | Destination
Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:
1. Inject as group frame to AP

<table>
<thead>
<tr>
<th>Flags</th>
<th>Receiver</th>
<th>Sender</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>To AP</td>
<td>FF:...:FF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

802.11 specific

Data

Encrypted using group key
Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:
1. Inject as group frame to AP
2. AP processes and routes frame

<table>
<thead>
<tr>
<th>Flags</th>
<th>Receiver</th>
<th>To AP</th>
<th>FF::FF</th>
<th>Sender</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>802.11 specific</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decrypted using group key
Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:
1. Inject as group frame to AP
2. AP processes and routes frame
3. AP transmits it to destination

Flags | Receiver
--- | ---
To STA | Destination
Sender | Destination
802.11 specific

Data
Encrypted using pairwise key
Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:
1. Inject as group frame to AP
2. AP processes and routes frame
3. AP transmits it to destination
4. Victim sees normal unicast frame

Flags  Receiver
To STA  Destination  Sender  Destination  Data
802.11 specific

Decrypted using pairwise key
## Forging unicast frames using group key

### Abuse AP to bypass Hole 196 check:

1. Inject as group frame to AP
2. AP processes and routes frame
3. AP transmits it to destination
4. Victim sees normal unicast frame

<table>
<thead>
<tr>
<th>Flags</th>
<th>Receiver</th>
<th>To STA</th>
<th>Destination</th>
<th>Sender</th>
<th>Destination</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>802.11 specific</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decrypted using pairwise key
Decrypting all traffic

ARP poison to broadcast MAC address
- Poison both router and clients
- Targets network-layer protocols: IPv4, IPv6, ...

Countermeasure:
- AP should ignore frames received on broadcast or multicast MAC address.
Contributions: Security of Group Keys

- Flawed generation
- Force RC4 in handshake
- Inject & decrypt all traffic
- New Wi-Fi tailored RNG
An improved 802.11 RNG

Entropy present on all Wi-Fi chips?
- Wi-Fi signals & background noise

Spectral scan feature in commodity chips:
- Can generate 3 million samples / second
- First XOR samples in firmware
- Extract & manage resulting entropy using known approaches

Additional research needed: performance under jamming?
Conclusion: lessons learned

1. Use a proper RNG
2. Let AP ignore group-addressed frames
3. Don’t put “expository” security algos in a specification
4. Don’t transmit sensitive data before downgrade detection

Questions?