draft-tiloca-6tisch-robust-scheduling-00

Authors:  Marco Tiloca
          Simon Duquennoy
          Gianluca Dini
Motivation

• Cell utilization patterns are predictable in TSCH
  • Even if security is used at the link layer

• An external adversary can easily:
  • Derive the communication pattern of a victim node
  • Selectively jam the exact cells of the victim’s schedule

• The attack is:
  • Easy and efficient to perform
  • Highly effective and with very low exposure
What makes the attack easy?

- Periodicity property --- Every cell re-uses the same sequence of channels, with period $(N_C \times N_S)$
- Usage property --- Within each period, all cells use all channels, once each
- Offset property --- All cells follow the same sequence of channels, with a certain offset
- Predictability property --- The sequence of channel is predictable, given a pair (timeslot, channel)
  - Timeslots repeat periodically on a same channel
  - One can compute the remaining channel hopping sub-sequence
- Attack rationale
  - ASN = $(s + T \times N_S) \mod N_C$ ; a cell uses channel $f$ and timeslot $s$ on slotframe $T$
  - Solve $f = [((s + T \times N_S + c) \mod N_C) \mod N_C]$ in $c$ (Equation 1)
  - Find the channels used by the cell in the next slotframes
  - The exact ASN is not needed! One can re-number slotframes from an arbitrary one
Attack outline

• Start the attack at a starting-slotframe $t = 0$

• Determine the timeslots in which the victim transmits
  • Pick a channel $f^*$ at random
  • Listen on $f^*$ for $N_C$ consecutive slotframes

• Find the channels used by the victim in the next slotframes
  • Solve Equation 1 in $c$ for each found timeslot
  • Now $f$ can be computed for any $t > 0$ and every timeslot $s$

• The adversary knows the full victim schedule
  • Selective jamming against the victim cells only
  • Staying quiet otherwise
Attack example

- Starting-slotframe $t = 0$ is slotframe $T = 1$

- Listen to $f^* = 1$ for 4 slotframes
  - Used in $s = 1$ and $s = 2$ of $t = 0$
  - Used in $s = 0$ of $t = 1$
  - All used timeslot are found

- Solve Equation 1 in $c$ for each $s$ found
  - Able to derive future $f$ from $s$ and $t$

- At each slotframe $t > 3$, i.e. $T > 4$
  - Derive $f$ for each timeslot $s$
  - Jam traffic during $s$ over $f$

<table>
<thead>
<tr>
<th>Ch</th>
<th>-----------</th>
<th>ASN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Of</td>
<td>0</td>
<td>f=2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>f=2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>f=3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s=0</td>
<td>s=1</td>
</tr>
<tr>
<td></td>
<td>T = 0</td>
<td>T = 1</td>
</tr>
</tbody>
</table>

Equation 1: $f = [(s + t \times 3 + c) \mod 4]$

6TiSCH - IETF103 - Bangkok
Solution – Overview

• Prevent the attack by construction
  • Alter the communication pattern of nodes at every slotframe
  • The resulting used pattern must be unpredictable for the adversary

• At each slotframe $T$:
  • All nodes pseudo-randomly permute the original schedule for $T + 1$
  • Separate permutation of timeslot usage (optional) and channel offset usage
  • All nodes locally compute the same permutation
  • The resulting schedule is consistent and collision-free

• Pseudo-random number generator
  • $val = \text{random}(K, z) = E(K, z)$ - Encrypt a fresh value $z$ with a key $K$
  • AES-CCM-16-64-128 must be supported
Solution – Key material

• Permutation key $K_s$
  • Used to permute the timeslot utilization pattern
  • Provided upon joining, e.g. using the 6TiSCH Join Protocol (CoJP)

• Permutation key $K_c$
  • Used to permute the channelOffset utilization pattern
  • Provided upon joining, e.g. using the 6TiSCH Join Protocol (CoJP)

• Counter $z_s$
  • Used to permute the timeslot utilization pattern
  • At the beginning of $T$, $z_s$ is equal to the ASN of the first timeslot of $T$

• Counter $z_c$
  • Used to permute the channelOffset utilization pattern
  • At the beginning of $T$, $z_c$ is computed from the ASN of the first timeslot of $T$
Solution steps

• At the beginning of each slotframe, each node:
  • Takes the original schedule for the next slotframe
  • Performs the steps below to permute the schedule using the Fisher-Yates algorithm
  • Provides the permuted schedule to TSCH, to send/receive traffic in the next slotframe

• Step 1 – Permute the timeslot utilization pattern (optional)
  • $N_s$ invocations of random($K, z$)
  • $K = K_s$ ; $z = z_s$
  • $z_s$ incremented after each invocation

• Step 2 – Permute the channelOffset utilization pattern
  • $N_c$ invocations of random($K, z$)
  • $K = K_c$ ; $z = z_c$
  • $z_c$ incremented after each invocation
Key provisioning

- $K_s$ and $K_c$ MAY be provisioned with the minimal security framework
  - The JRC provides the pledge with $K_s$ and $K_c$ upon joining

- Additional COSE_KeySet in the Join Response
  - If two keys are present, the first key is $K_s$ and the second key is $K_c$
  - If one key is present, it is $K_c$ (permute only channelOffset utilization patterns)

- Details need to be updated to the latest Join Response format
  - A dedicated COSE_KeySet seems still the best option
Summary and next steps

• Preventive approach against selective jamming
  • Agnostic of the specific scheduling algorithm
  • Preserve collision-free and consistent schedules
  • Efficient pseudo-random shuffling of cells
  • No communication overhead

• Next steps
  • Get comments and feedback
  • Align the text on key provisioning with the latest *draft-ietf-6tisch-minimal-security*
Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-6tisch-robust-scheduling