draft-tiloca-6tisch-robust-scheduling-02

Authors: Marco Tiloca
Simon Duquennoy
Gianluca Dini

Recap

- Attack:
  - Selectively jamming exact cells of the victim's schedule
  - Effective, stealthy, targeted and low-power
- Preventive solution:
  - Efficient pseudo-random shuffling of cells, at each slotframe
  - No communication overhead (only local computation)
- Resulting new schedule:
  - Collision-free and consistent
  - Unpredictable to the adversary

After IETF 104

- Reviews on the list
  - Pascal, Tengfei, Michael – Thanks!
- More comments on the list
  - Michael, Mališa
- Summary of updates
  - Reviews have been addressed
  - Clarifications and editorial improvements

Updates from -01 (1/2)

- [Editorial – Tengfei]
  - Clarifications on symbols used for equation
- [Informational – Michael]
  - More on attack motivations (e.g., harm competitor's network)
- [Fixed cost of shuffling]
  - An array of N elements is shuffled with (N - 1) swaps – was N
  - Slower growth of counters N_C and N_S
  - Aligned with modern Fisher-Yates version

Updates from -02 (2/2)

- [Recommended use – Mališa]
  - Not explicitly recommending anymore the time offset shuffling
  - Still describing the pros & cons of doing or not doing it

- [Output example – Michael]
  - New Appendix A – Full step-by-step execution on a single node (2 slotframes)
  - Schedule encoding; exact shuffling with pseudo-random number generation
  - Configuration: original schedule; permutation keys; permutation cipher
  - Both time and channel offset are shuffled; show intermediate and final permuted schedule
  - *https://gitlab.com/crimson84/draft-tiloca-6tisch-robust-scheduling/tree/master/test*

Key management (1/2)

- Renewal of permutation keys with CoJP
  - Provide them again together with the network keys
  - New parameter in the CoJP Join Response message

```
Configuration = {
    ? 2    : [ +Link_Layer_Key ],
    ? 3    : Short_Identifier,
    ? 4    : bstr,
    ? 6    : [ *bstr ],
    ? 7    : uint,
    ? TBD  : [ +Permutation_Key ],
    ? TBD  : Permutation_Cipher
}

Permutation_Key = (
        key_value              : bstr
(
```

- Temporary misalignment
  - Different nodes may get the new keys at different point in times
  - Risk of both old and new shuffling performed in the network (for a while)

- How/when switching to the new permutation keys?

Key management (2/2)

- Possible way forward (input: Michael, Mališa)
  - Add a new parameter ‘Permutation_Index’ in the CoJP Join response
  - Act as unique identifier of the distributed permutation key set
  - Incremented upon every renewal of the permutation keys

- Key switching
  - Signal the new permutation key set in a EB, in a newly allocated IE value
  - The EB is sent by the trusted 6LBR, and verifiable with the new network keys
  - A 6LN node able to verify the EB will also switch to the new permutation keys

Summary

- Addressed reviews and comments from IETF 104

- Main open point
  - Switching to new permutation keys

- WG adoption?

Thank you!

Comments/questions?

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