6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2018

M. Tiloca
S. Duquennoy
RISE SICS
G. Dini
University of Pisa
June 29, 2018

Robust scheduling against selective jamming in 6TiSCH networks
draft-tiloca-6tisch-robust-scheduling-00

Abstract

This document defines a method to generate robust TSCH schedules in a 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4-2015) network, so as to protect network nodes against selective jamming attack. Network nodes independently compute the new schedule at each slotframe, by altering the one originally available from 6top or alternative protocols, while preserving a consistent and collision-free communication pattern. This method can be added on top of the minimal security framework for 6TiSCH.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2018.

Copyright Notice

Table of Contents

1. Introduction

Nodes in a 6TiSCH network communicate using the IEEE 802.15.4-2015 standard and its Timeslotted Channel Hopping (TSCH) mode. Some properties of TSCH make schedule units, i.e. cells, and their usage predictable, even if security services are used at the MAC layer.

This allows an external adversary to easy derive the communication pattern of a victim node. After that, the adversary can perform a selective jamming attack, by efficiently and effectively transmitting over the only exact cell(s) in the victim's schedule.

This document describes a method to counteract such an attack. At each slotframe, every node autonomously computes a TSCH schedule, as a pseudo-random permutation of the one originally available from 6top [I-D.ietf-6tisch-6top-protocol] or alternative protocols.

The resulting schedule is provided to TSCH and used to communicate during the next slotframe. In particular, the new communication pattern results unpredictable for an external adversary. Besides, since all nodes compute the same pseudo-random permutation, the new communication pattern remains consistent and collision-free.

Furthermore, this document specifies how this method can be added on top of the minimal security framework for 6TiSCH described in [I-D.ietf-6tisch-minimal-security].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with terms and concepts defined in [I-D.ietf-6tisch-minimal-security], [I-D.ietf-6tisch-terminology] and [RFC8152].

This document refers also to the following terminology:

- Permutation key. A cryptographic key shared by network nodes and used to permute schedules. Different keys are used to permute the utilization pattern of timeslots and of channelOffsets.

2. Properties of TSCH that simplify selective jamming

This section highlights a number of properties of the TSCH cell usage that greatly simplify the performance of the selective jamming attack described in Section 3.

Given:

- The channel 'f' to communicate at timeslot 's' with ASN and channelOffset 'chOff' computed as f = F[(ASN + chOff) mod N_C];

And assuming for simplicity that:

- N_S and N_C are coprime values;

- The channel hopping sequence is N_C in size and equal to {0, 1, ..., N_C - 1};

Then, the following properties hold:

- Periodicity property. The sequence of channels used for communication by a certain cell repeats with period (N_C x N_S) timeslots.

- Usage property. Within a period, every cell uses all the available channels, each of which only once.

- Offset property. All cells follow the same sequence of channels with a certain offset.

- Predictability property. For each cell, the sequence of channels is predictable. That is, by knowing the channel used by a cell in a given timeslot, it is possible to compute the remaining channel hopping sub-sequence.

  In fact, given a cell active on channel 'f' and timeslot 's' on slotframe 'T', and since ASN = (s + T x N_S), it holds that

  f = [(s + T x N_S + c) mod N_C] (Eq. 1)

  By solving this equation in 'c', one can predict the channels used by the cell in the next sloframes. Note that, in order to do that, one does not need to know the absolute number 'T' of the slotframe (and thus the exact ASN) in which timeslot 's' uses a certain channel 'f'. In fact, one can re-number slotframes starting from any arbitrarily assumed "starting-slotframe".

3. Attack example

This section describes how an external adversary can exploit the proterties in Section 2, and determine the full schedule of a victim node, even if security services at the MAC layer are used. It is also assumed that the victim node actually transmits/receives during all its allocated cells at each slotframe.

The example considers Figure 1, where N_S = 3, N_C = 4, and the

channel hopping sequence is {0,1,2,3}. The shown schedule refers to a network node that uses three cells 'L_1', 'L_2' and 'L_3', with {0,3}, {1,1} and {2,0} as pairs {timeslot, channelOffset}, respectively.

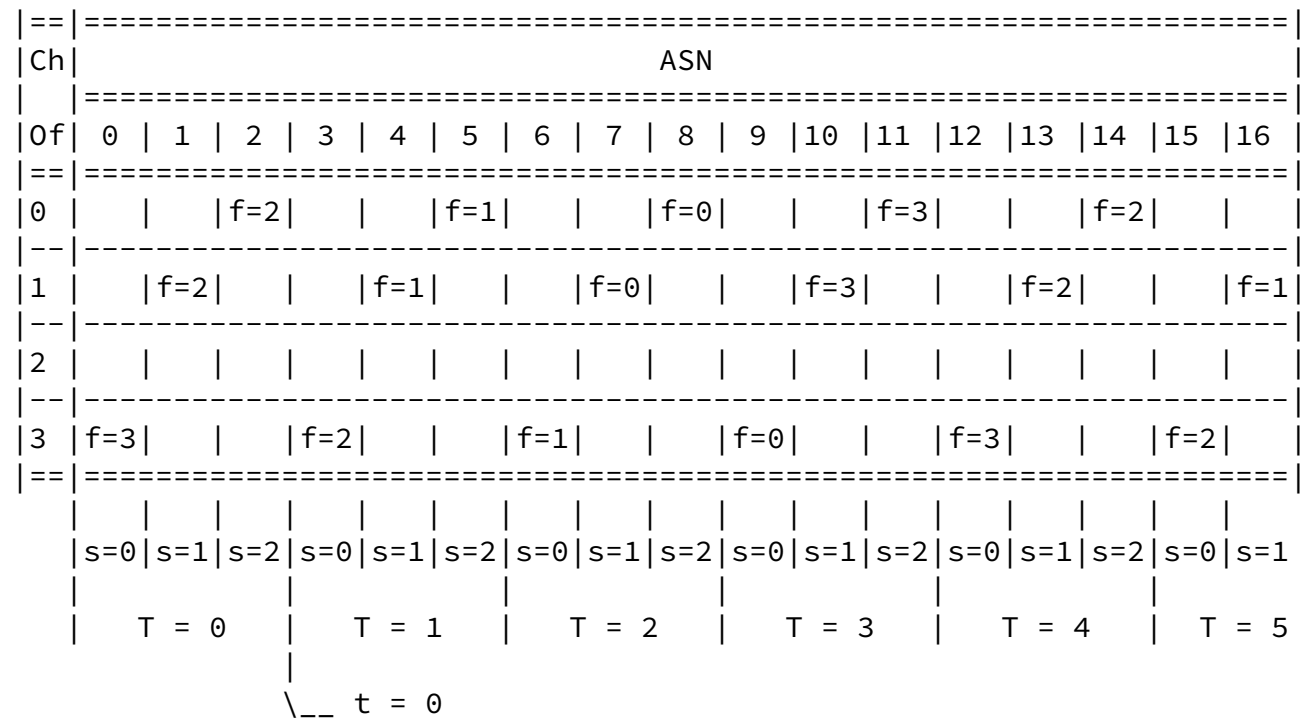

Figure 1: Attack example with slotframe re-numbering

1. The adversary starts the attack at absolute slotframe T = 1, which is assumed as "starting-slotframe" and thus renamed as slotframe t = 0. The renaming is possible due to the offset and predictability properties.

2. The adversary picks a channel 'f*' at random, and monitors it for N_C consecutive slotframes to determine the timeslots in which the victim node communicates on that channel. Due to the usage property, the number of such timeslots is equal to the number of cells assigned to the victim node.

   With reference to Figure 1, if, for example, f* = 1, the adversary determines that the victim node uses channel 'f*' in timeslots s = 1 and s = 2 of slotframe t = 0 and in timeslot s = 0 of slotframe t = 1. The adversary can then deduce that the victim node uses three different cells 'L_1', 'L_2' and 'L_3', in timeslots 0, 1 and 2, respectively.

3. The adversary determines the channels on which the victim node is going to transmit in the next slotframes, by exploiting the predictability property.

   That is, by instantiating Equation 1 for cell L_1, timeslot s = 0 and slotframe t = 1, one gets [1 = (3 + c_1) mod 4], which has solution for c_1 = 2. Hence, the function to predict the channel 'f_1' to be used by cell 'L_1' in a slotframe 't', t >= 1, is f_1

   = [(2 + 3 x t) mod 4], which produces the correct periodic sequence of channels {1, 0, 3, 2}. Similarly, one can instantiate Equation 1 for cells 'L_2' and 'L_3', so producing the respective periodic sequence of channels {1,0,3,2} and {1,0,3,2}.

4. The adversary has discovered the full schedule of the victim node and can proceed with the actual selective jamming attack. That is, according to the found schedule, the adversary transmits over the exact cells used by the victim node for transmission/ reception, while staying quiet and saving energy otherwise. This results in a highly effective, highly efficient and hard to detect attack against communications of network nodes.

4. Building robust schedules

This section defines a method to protect network nodes against the selective jamming attack described in Section 3. The proposed method alters the communication pattern of all network nodes at every slotframe, in a way unpredictable for an external adversary.

At each slotframe 'T', network nodes autonomously compute the communication pattern for the next slotframe 'T+1' as a pseudo-random permutation of the one originally available. In order to ensure that the new communication pattern remains consistent and collision-free, all nodes compute the same permutation of the original one. In particular, at every slotframe, each node separately and independently permutes its timeslot utilization pattern (optionally) as well as its channelOffset utilization pattern.

To perform the required permutations, all network nodes rely on a same secure pseudo-random number generator (SPRNG) as shown in Figure 2, where E(x,y) denotes a cipher which encrypts a plaintext 'y' by means of a key 'x'. Network nodes MUST support the AES-CCM-16-64-128 algorithm from [RFC8152].

```
unsigned random(unsigned K, unsigned z) {
    unsigned val = E(K,z);
    return val;
}
```

Figure 2: Secure Pseudo-Random Number Generator

All network nodes share the same following pieces of information.

- K_s, a permutation key used to permute the timeslot utilization pattern, and used as input to the random() function in Figure 2. K_s is provided upon joining the network, and MAY be provided as described in Section 5.

- K_c, a permutation key used to permute the channelOffset utilization pattern, and used as input to the random() function in Figure 2. K_c is provided upon joining the network, and MAY be provided as described in Section 5.

- z_s, a counter used to permute the timeslot utilization pattern, and used as input to the random() function in Figure 2. At the beginning of each slotframe, z_s is equal to the ASN value of the

first timeslot of that slotframe. Then, z_s grows by N_S from the beginning of a slotframe to the beginning of the next one.

- z_c, a counter used to permute the channelOffset utilization pattern, and used as input to the random() function in Figure 2. At the beginning of each slotframe, z_c is equal to [N_C x floor(ASN* / N_S)], where ASN* is the ASN value of the first timeslot of that slotframe. Then, z_c grows by N_C from the beginning of a slotframe to the beginning of the next one.

Then, at every slotframe, each network node takes the following steps, and generates its own permuted communication schedule to be used at the following slotframe. The actual permutation of cells relies on the well-known Fisher-Yates algorithm, that requires to generate n pseudo-random numbers in order to pseudo-randomly shuffle a vector of n elements.

1. First, a pseudo-random permutation is performed on the timeslot dimension of the slotframe. This requires N_S invocations of random(K,z), consistently with the Fisher-Yates algorithm. In particular, K = K_s, while z_s is passed as second argument and is incremented by 1 after each invocation. The result of this step is a permuted timeslot utilization pattern, while the channelOffset utilization pattern is not permuted yet.

2. Second, a pseudo-random permutation is performed on the channelOffset dimension of the slotframe. This requires N_C invocations of random(K,z), consistently with the Fisher-Yates algorithm. In particular, K = K_c, while z_c is passed as second argument and is incremented by 1 after each invocation. The result of this step is a fully shuffled communication pattern.

The resulting schedule is then provided to TSCH and considered for sending/receiving traffic during the next slotframe.

As further discussed in Section 6.3, it is possible, although NOT RECOMMENDED, to skip step 1 above, and hence permute only the channeOffset utilization pattern, while keeping a static timeslot utilization pattern.

Note for implementation: the process described above can be

practically implemented by using two vectors, i.e. one for shuffling the timeslot utilization pattern and one for shuffling the channelOffset utilization pattern.

5. Adaptation to the 6TiSCH minimal security framework

The security mechanism described in this specification can be added on top of the minimal security framework for 6TiSCH [I-D.ietf-6tisch-minimal-security].

That is, the two permutation keys K_s and K_c can be provided to a pledge when performing 6TiSCH Join Protocol (6JP). To this end, the payload of the Join Response defined in Section 9.2 of [I-D.ietf-6tisch-minimal-security] is extended with a further COSE_KeySet specified in [RFC8152].

Specifically, this COSE_KeySet contains one or two permutations keys and is interpreted as follows. If two keys are present, they are used as K_s and K_c to permute the timeslot utilization pattern and the channelOffset utilization pattern, respectively, as per Section 4. Instead, if only one key is present, it is used as K_c to permute the channelOffset utilization pattern as per Section 4.

The resulting payload of Join Responses becomes as follows:

```
response_payload = [
   COSE_KeySet,
   short_address,
   ? JRC_address : bstr,
   ? COSE_KeySet,
]
```

6. Security Considerations

With reference to Section 3.9 of [RFC7554], this specification achieves an additional "Secure Communication" objective, namely it defines a mechanism to build and enforce a TSCH schedule which is robust against selective jamming attack, while at the same time consistent and collision-free.

Furthermore, the same security considerations from the minimal security framework for 6TiSCH [I-D.ietf-6tisch-minimal-security] hold for this document. The rest of this section discusses a number of additional security considerations.

6.1. Effectiveness of schedule shuffling

The countermeasure defined in Section 4 practically makes each node's schedule look random to an external observer. Hence, it prevents the adversary from performing the attack described in Section 3.

Then, a still available strategy for the adversary is to jam a number of cells selected at random, possibly on a per-slotframe basis. This considerably reduces the attack effectiveness in successfully jeopardizing victims' communications.

At the same time, nodes using different cells than the intended victims' would experience an overall slightly higher fraction of corrupted messages. In fact, the communications of such accidental victims might be corrupted by the adversary, when they occur during a jammed timeslot and exactly over the channelOffset chosen at random.

6.2. Renewal of key material

It is RECOMMENDED that the two permutation keys K_s and K_c are revoked and renewed every time a node leaves the network. This prevents a leaving node to keep the permutation keys, which may be exploited to selectively jam communications in the network.

This rekeying operation is supposed to be performed anyway upon every change of network membership, in order to preserve backward and forward security. In particular, new IEEE 802.15.4 link-layer keys are expected to be distributed before a new pledge can join the network, or after one ore more nodes have left the network.

The specific approach to renew the two permutation keys, possibly together with other security material, is out of the scope of this specification.

6.3. Static timeslot allocations

As mentioned in Section 4 and Section 5, it is possible to permute only the channelOffset utilization pattern, while preserving the originally scheduled timeslot utilization pattern. This can be desirable, or even unavoidable in some scenarios, in order to guarantee end-to-end latencies in multi-hop networks, as per accordingly designed schedules.

However, it is NOT RECOMMENDED to preserve a static timeslot utilization pattern, as this would considerably increase the attack surface for a random jammer adversary. That is, the adversary would

immediately learn the timeslot utilization pattern of a victim node,

and would have a chance to successfully jam a victim's cell equal to (1/N_C), where N_C is the number of available channelOffsets.

6.4. Network joining through randez-vous cells

As described in [I-D.ietf-6tisch-minimal-security], a pledge joins a 6TiSCH network through a Join Proxy (JP), according to 6TiSCH Join Protocol (6JP) and based on the information conveyed in Enhanced Beacons (EBs). In particular, the pledge will communicate with the JP over indicated randez-vous cells. In practice, such cells are typically part of a dedicate slotframe sequence, which is different from the slotframe sequence used for EB and data transmission.

In order to keep the join process deterministic, the solution described in this specification can not be applied to the slotframe sequence with the randez-vous cells. That is, an adversary would remain able to selectively jam the randez-vous cells, so potentially jeopardizing the 6JP and preventing pledges to join altogether.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgments

The authors sincerely thank Malisa Vucinic for the initial discussion about this document.

The work on this document has been partly supported by the EIT-Digital High Impact Initiative ACTIVE.

9. References

9.1. Normative References

[I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-06 (work in progress), May 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.ietf-6tisch-6top-protocol] Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer Protocol (6P)", draft-ietf-6tisch-6top-protocol-12 (work in progress), June 2018.

[I-D.ietf-6tisch-terminology] Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terms Used in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-10 (work in progress), March 2018.

[RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <https://www.rfc-editor.org/info/rfc7554>.

Authors' Addresses

Marco Tiloca
RISE SICS
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: marco.tiloca@ri.se

Simon Duquennoy
RISE SICS
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: simon.duquennoy@ri.se

Gianluca Dini
University of Pisa
Largo L. Lazzarino 2
Pisa 56122
Italy

Email: gianluca.dini@unipi.it