6TiSCH Security Architectural Considerations
draft-struik-6tisch-security-considerations-01

Abstract

This document describes 6TiSCH security architectural elements with high level requirements and the security framework that are relevant for the design of the 6TiSCH security solution.

Status of This Memo

Copyright Notice

Table of Contents

1. Join Protocol Behavior

1.1. MAC Behavior

1. The joining node has to transgress from the so-called "embryonic stage", where it does not have shared keying material with any network nodes yet, to the stage where it has shared keying material with the security manager of the network (who hands out a network wide key, amongst other things). In many cases, the security manager will be the PAN coordinator.

2. Initially, the joining node listens to an enhanced beacon sent by its neighbor node. If this beacon is secured, it can still extract the visible portion of the enhanced beacon frame (which includes all frame fields before these were secured by the neighbor node if the frame was authenticated and which includes only the header fields, including potential header information elements, otherwise). With 802.15.4-2011, the passive scan procedure supports this (see 5.1.2.1.2). In either case, the joining node stores the PAN Descriptor. Note that it cannot rely on the authenticity of the PAN Descriptor, since the beacon frame is either not secured, or it was secured and the joining node did not have a shared key. Either way, it has to accept the PAN Descriptor "on face value".

3. The neighbor node, if it operates securely, normally does not accept incoming frames from the joining node, since these would not be properly secured with the correct keying material. However, the 802.15.4 specification allows one exception to this: it also accepts incoming messages from specifically identified devices that have diplomatic immunity (have so-called "exempt status"). This mechanism can be used to facilitate communication between a joining node and a neighbor node till they have

established shared keying material (whereby the joining node can emerge out of its initial embryonic stage). This can be done as follows:

* The neighbor node can temporarily give the joining node "exempt status", e.g., after failed incoming security processing (thereby, allowing subsequent unsecured data frames from this joining node to be accepted *from this specific device*). It can also populate the table with exempt devices via other means.

* The higher layer can switch on/off this "exempt status" facility for specific joining nodes based on local criteria (one joining node at the time; device open for enrollment of devices or not, pre-populated table, etc.)

* The higher layer of the neighbor node should ensure that this facility is only used for MAC data frames that correspond to initial join messages.

* The higher layer can use this "exempt status" flag for outgoing messages back to the joining node (where this indicates "please send message unsecured" (since message to newbee joining node with diplomatic immunity status).

4. Once the joining node and the neighbor node have established a shared key, the neighbor node can lift the diplomatic immunity status of the joining node (by removing the "exempt status" flag corresponding to this device), after which it may only accept incoming messages from the joining node if these are properly secured. Conversely, the joining node can now update its security policy settings, after which it may only accept properly secured messages received from the neighbor node. Note that from that moment on, the communications between the joining node and the neighbor node can all be authenticated, including time corrections that are very important for proper operation of TSCH (where, e.g., neighbor node is time "clock tower" for joining node).

5. Conceptually, the use of the "exempt flag" could be considered as a mechanism for forming a temporary two-node "join network" (consisting of the joining node and its neighbor node), in which join-related messages are allowed to flow unsecurely. This does not mean, however, that these nodes operate in a separate PAN, though, since incoming frame processing relies on filtering on a single destination PAN Identifier (see 802.15.4-2011, 5.1.6.2), which implies that the neighbor node can only be part of a single PAN (802.15.4-2011 does not know the concept of "multiple PAN

instances"). This also implies that there is no mechanism within 802.15.4 to designate frames for "join" purposes or other special uses (as Wireless HART seems to do with enhanced beacon frames). Of course, there are ways to still artificially realize this, e.g., based on context information (overloading semantics of schedules) or based on yet-to-be-defined information elements (so as to make these act as frame "sub-types"), should one wish to emulate this behavior. Emulating any of this would require changes to 802.15.4 security processing. Currently, there does not seem to be a need for this additional complexity.)

1.2. MAC Security Considerations

1. With 802.15.4-2011, incoming security processing requires access to device-specific information of the originating device (stored on the recipient device in the so-called device descriptor table). This includes the extended address of the originating device, the "lowest" unseen frame counter for that device, and its "exempt status". Successful incoming security processing of a secured frame results in a state change of this device-specific information (since this updates, e.g., the frame counter).

2. Successful incoming security processing of a secured or unsecured frame may result in other state changes as well if only because the device simply "acts" on the received frame or, e.g., due to side effects of the successful receipt hereof. Examples of such side effects include actions triggered by information elements contained in the received frame, such as time corrections to the local clock (which are very important for proper operation of TSCH).

3. 802.15.4-2011 uses the AEAD scheme CCM for frame security, where the nonce is derived from the frame counter and other information. The security of this scheme (or other nonce-based authenticated encryption scheme) is void if nonces are ever reused with the same key. We give an example illustrating how nonce reuse breaks confidentiality: one can derive from two ciphertexts the xor of the corresponding plaintext (or the segment with the size of the shorter ciphertext). From this information and side information on the plaintext (e.g., redundancy), one can often recover both plaintexts (with virtually no remaining ambiguity).

4. Since successful incoming security processing induces a state change, it is imperative that all cryptographic keys used are, indeed, real keys. In particular, this implies that one shall never use 802.15.4 with "default" keys (fake keys with an easy to guess, low-entropy value).

5. If a device wants to communicate with a corresponding party with which it does not share cryptographic keying material yet (e.g., because it is a joining node in embryonic stage), it should send unsecured frames and *not* frames *obscured* (via security through obscurity techniques) using "fake" keys, if only because of avoidance of undesirable side effects: if a recipient accepts an unsecured frame (e.g, because the originator has "exempt status"), this does *not* trigger a state change of security-relevant parameters, whereas if a recipient accepts an obscured frame (secured using a "fake" key), this *does* trigger a state change of security-relevant parameters.

6. TSCH security with 802.15.4e-2012 relies on nonces that are derived from the absolute slot number (ASN), rather than from the frame counter in the device descriptor. Successul processing of a secured incoming frame depends on both originator and recipient of the frame having synchronized "world views" of the ASN entry. The ASN is also used for communication purposes, since indicates scheduling information. This "mixed" use (both for communication and security) is somewhat problematic, since changes to this parameter for either use has spill-over effects on the other use: any changes to the ASN as a communication parameter now might have side effects on security-critical parameters that could, worst case, entirely break security; conversely, any changes to the ASN as a security parameter, e.g., resulting from its inadverent use with a compromised key (or, equivalently, a "fake" key), could result in unreliability of this parameter for indicating scheduling information. Impact of ASN manipulation on security may include reuse of nonces (resulting in compromise of the AEAD cipher's properties), denial-of-service attacks on sender or recipient (e.g., due to putting the ASN entry "out-of-sync" on either end), or frame counter reuse (since 802.15.4e-2012 does not inspect the frame counter in the device descriptor, but solely relies on the ASN entry). Thus, ASN entries are very fragile and their use should happen with extreme care.

7. As already mentioned, ASN anomalies may seriously impact security. If any device's ASN state is out-of-synch with other devices, this may result in that device not being able to communicate in the network any more. With network-wide keys, the remedy may include a combination of rekeying all devices (a costly proposition) and resetting ASN entries of the impacted device.

8. The security provisions in 802.15.4-2011 and 802.15.4e-2012 leave some room for potential Denial of Service (DoS) attacks. We only discuss "accidental" DoS attacks for now, which we define as

those triggered without active involvement of an adversarial network element (active DoS attacks are considered separately).

* If a device acts on an incoming frame that is cryptographically secured, it has assurances that this frame originated from a device with access to the key. Here, processing a frame with a key provides a mechanism for network segregation, since proper incoming security processing (and assuming non-compromised locally stored security-relevant material and processes) allows one to draw conclusions as to whether originator and recipient belong to the same "group" (the key-sharing group). This propery holds if the incoming frame has an authenticity tag; in some cases, this may also hold if the frame was only encrypted, but not authenticated. This "network segregation" property holds independent of whether the key was actually a real key (cryptographic key); the number of groups created depends on the number of these group keys (perhaps, more properly termed "group identifiers" if of no cryptographic use) used.

* A joining node must make its decision to join the network based on information derived from processing an enhanced beacon. Since it is in embryonic stage, it has to take this information at face value (no matter whether this beacon was cryptographically secured or not). In theory, this may give rise to dilemmas of choice, i.e., how is a joining node to pick which beacon to act upon? As already said, one could realize network segregation using a "default" key, whereby the joining node and the beaconing device would be able to check membership of the same loosely defined group (this is the mechanism Wireless HART uses). However, as mentioned before, this could potentially adversely impact 802.15.4-2011 and 802.15.4e-2012 security. Even if one discards security concerns, this only establishes membership of a very crudely defined group (e.g., if one uses as "default" key the fixed value "6tisch-default-join", this would have any joining node accept any 6tisch-beacon). The same filtering mechanism could also, without any possible security side effects, be realized by partitioning the "language of well-formed frames" and, e.g., filtering enhanced beacons on the data object "6tisch-default-join" (e.g., when including this tag as a Header Information Element with the beacon). If one does not use such explicit "tags", one could conceivable also accept beacon frames that implement an alien protocol, rather than 802.15.4e-2012. It is, however, quite unlikely that a random alien frame will pass incoming frame filtering, since 802.15.4 incoming frame processing checks for well-formedness. Checking some built-in redundancy of well-formed frames

thereby most likely filters out virtually all unwanted alien frame types. Such filtering could, e.g., include a "language check" as to fixed fields in information elements. For enhanced beacon frames for TSCH, e.g., the header fields of the synchronization IE, timeslot IE, and header IE contained herein have fixed 2-octet values 0x1a06, 0x1c01, and 0x1d01, respectively, thereby providing up to 48 bits of redundancy. This provides similar filtering functionality as the explicit "6tisch-default-join" tag mentioned before, but without the need to introduce an explicit tag or to communicate this separately over the air.

It should be emphasized (again) that none of the mechanisms above protects against active attacks.

1.3. Join Protocol Behavior

1.3.1. Device Enrollment Phases

The join protocol consists of three phases, viz.

1. Device Authentication: The joining node and proxy network node authenticate each other and establish a shared key, so as to ensure on-going authenticated communications. This may involve a server as a third party.

2. Authorization: The proxy network node decides on whether/how to authorize a joining node (if denied, this may result in loss of bandwidth). Authorization decisions may involve other nodes in the network.

3. Configuration/Parameterization: The proxy network node distributes configuration information to the joined node, such as scheduling information, IP address assignment information, and network policies. This may originate from other network devices, for which it acts as proxy. This step may also include distribution of information from the joining node to the network node and other nodes in the network and, more generally, synchronization of information between these entities.

The device enrollment process is depicted in Figure Figure 1, where it is assumed that devices have access to certificates and where entities have access to the root CA keys of their communicating parties (initial set-up requirement). Under these assumptions, the authentication step of the device enrollment process does not require online involvement of a third party. Mutual authentication is performed between the joining node and the proxy using their certificates, which also results in a shared key between these two

entities. The proxy assists the joining node in mutual authentication with the server, which also results in a shared (end-to-end) key between those two entities. The server may arbitrage network authorization of the joining node (where the proxy will deny bandwidth if authorization is not successful) and may distribute network-specific configuration parameters (including network-wide keys) to the joining node. In its turn, the joining node may provide distribute/synchronize information (including, e.g., network statistics) to the server node.

The server functionality is a role and may be implemented with one device (centralized) or with multiple entities (distributed). In either case, mutual authentication is established with each physical server entity with which a role is implemented. Note that in the above description, the proxy does not solely act as a relay node. For more detailed rationale, see the relevant detailed descriptions further in this document. This also provides some insight into what happens in case the initial set-up requirements are not met or some other out-of-sync behavior occurs and suggest some optimization in case server-related information is already available with the proxy node (caching).

When a device rejoins the network in the same authorization domain, the authorization step could be omitted if the server distributes the authorization state for the device to the proxys when the device initially joined the network. However, this generally still requires the exchange of updated configuration information, e.g., related to time schedules and bandwidth allocation.

```
{joining node}      {neighbor}                 {server, etc.}
+---------+         +---------+                 +---------+
|  Node   |         |  Proxy  |              +--|    CA   |e.g., certificate
|    A    |         |    B    |              |  +---------+       issuance
+---------+         +---------+              |  +---------+
    |                    |                   +--|Authoriz.|e.g., membership
    |<----Beaconing------|                   |  +---------+         test
    |                    |                   |  +---------+
    |<--Authentication-->|                   +--| Routing |e.g., IP address
    |                    |<--Authorization-->|  +---------+       assignment
    |<-------------------|                   |  +---------+
    |                    |                   +--| Gateway |e.g., backbone,
    |------------------->|                   |  +---------+      cloud
    |                    |<--Configuration-->|  +---------+
    |<-------------------|                   +--|Bandwidth|e.g., PCE
                                                +---------+      schedule

    .                    .                   .
    .                    .                   .
```

Figure 1: Network joining, with only authorization by third party

1.3.2. Join protocol description

NOTE: the description below considers the scenario where devices have credentials on board and where the neighbor does not simply act as a relay node only. Other scenarios will be considered in future versions of this draft.

1. Upon hearing the enhanced beacon, the joining node stores the PAN descriptor.

2. The joining node uses local criteria, including information contained in the PAN desciptor, to determine whether it wishes to join the network.

3. The joining node sends the first join protocol message to the neighbor node. This message corresponds to one or more unsecured MAC data frames. This message includes the joining node's key contribution and credentials.

4. The neighbor node processes the incoming join message from the joining node and, depending on local criteria (including a check that this is a join message), grants the joining node temporary diplomatic immunity status ("exempt stauts") from a MAC perspective (if not granted, this simply results in a rejected incoming frame at the MAC layer).

5. The neighbor node performs some checks on the incoming message. If successful, it sends a first return join protocol message to the joining node. This message corresponds to one or more unsecured MAC data frames. This message includes the neighbor node's key contribution and credentials. It may also include the server's cached first return join protocol message info. At this point, the neighbor node is capable of deriving the shared key with the joining node based on inputs received and locally maintained status information.

6. The joining node performs some checks on the incoming message (including that it received this message from the neighbor node and that this is a join message). If succesful, it derives a shared key with the neighbor node and may derive a shared key with the server (it may also postpone the latter till required ["lazy evaluation"]).

7. The joining node sends a second join protocol message (a key confirmation message) to the neighbor node and may include some other information (so-called piggy-backed info). The piggy-backed information includes configuration information to be passed from the joining node to the neighbor node. This message corresponds to one or more unsecured MAC data frames.

8. The joining node sends a similar second join protocol message (another key confirmation message, including piggy-backed information) to the server. The piggy-backed information includes configuration information to be passed from the joining node to the server that allows the server to check the joining node's true credentials and some network-relevant parameters (including the ASN number and the joining node's local schedule maintained with the neighbor node). This message corresponds to one or more unsecured MAC data frames. This message may be combined with the message sent to the neighor node, since it travels along the same initial communication path.

9. The neighbor node checks the received second join protocol message (the key confirmation message and received piggy-backed info), including that this message originated from the same device as the previous join protocol message and that this message is a join message. If successful, it clears the "exempt status" attribute of the joining node in the DeviceDescriptor (thereby, lifting diplomatic immunity status for the joining device) and adds the {data key, joining node} pair to its KeyDescriptor list. It also stores policy-related attributes for this key. It may update some additional state, based on the piggy-backed info received from the joining device. The clearing of the "exempt status" flag means that it will only

accept incoming secured frames from the joining node from that moment onwards.

10. The server checks the received second join protocol message (key confirmation message and received piggy-backed info). If successful, it adds the {data key, joining node} pair to its locally maintained list of end-to-end keying material and includes policy-related attributes for this key. It sends its own second return join protocol message (another key confirmation message, including piggy-backed configuration information) to the joining node. This is actually sent to the neighbor node it received the first join protocol message from, who in turn forwards this to the joining node (here, the neighbor node acts in storing mode and knows the local network topology the server may not know (yet)). NOTE: this requires the neighbor node to remember some information pertaining to the joining node (mainly, the {data key, joining node} pair of the KeyDescriptor and the local communication schedule with the joining node). This may include an explicit notification to the neighbor node that the joining node is authorized to join the network. If so, this authorization part of this message is secured, using end-to-end security between the server and the neighbor node.

11. The neighbor node checks the authorization-related info, if indeed contained in this message (if denied, it may clear the joining node related info from its tables). If successful, it forwards this information along with its own second return join protocol message (key confirmation message and piggy-backed info) to the joining node. Obviously, this can be done separately as well, but travels over the same (single hop) communication path.

12. The joining node checks the received second join protocol message (the key confirmation and piggy-backed info) from its neighbor node. If successful, it adds the {data key, neighbor node} pair to its KeyDescriptor list. It also stores policy-related attributes for this key. If not successful, it clears its local table with info pertaining to the neighbor node.

13. The joining node checks the received second join protocol message (the key confirmation and piggy-backed info) from the server. If successful, it adds the {data key, server node} pair to its locally maintained list of end-to-end keying material and includes policy-related attributes for this. It may also update its local state, based on information contained in the piggy-backed info received from the server. Updates of local state may be subject to additional local criteria, such as consistency

of status information obtained from neighbor node and server node (e.g., pertaining to the ASN field, PAN identifier, or scheduling information). This may give rise to triggered alerts. If not successful, it clears its local table with info pertaining to the server node. Depending on local criteria, it may clear the table with info pertaining to the neighbor node.

1.3.3. Remarks

1. The join protocol above can be optimized in various ways, including first handling mutual authentication of local communication channels, prior to engaging in non-local communications so as to reduce time latencies in case of failure conditions. This is realized by having the neighbor node authenticate itself to the joining node before initiating non-local communications from the joining node to the server node along the communication path via the neighbor node (rather than at the end of this non-local communications). Since 10-hop communications may take roughly 2.5 minutes on a TSCH network and local communication time latencies take roughly 15 seconds, this could present a significant time saving (and reduced requirement on keeping state and energy consumption on the joining device).

2. The join protocol above takes only one non-local communication between the neighbor node and the server node. This assumes that the neighbor node is able to cache security-related information from the server. Since this includes certificate-related information of the server node (which may require more than one classical 802.15.4 MAC frame to carry), this may present significant communication time latency savings. Obviously, an additional long-haul round trip may be required should this cached information be stale (keeping this information in sync is a responsibility of the neighbor node). With caching, this turns the join protocol described above into the most efficient possible, in terms of communication time latencies involved. At the same time, this protocol has very strong security properties, unmatched by legacy protocols [...].

3. The join protocol above assumes authentication of the joining node to the neighbor node, before non-local traffic takes place. This assists in thwarting denial-of-service attacks on "das Hinterland" of the neighbor node triggered by joining nodes with improper credentials (unparsable certs). While this check is an authentication check only and *not* a fine-grained authorization check, this could be complemented by additional local "sanity checks" on the neighbor node (device white listing, etc.), thus allowing extensibility to more fine-grained authorization

filtering mechanisms. (Further details are outside scope of this document, but may be described later.)

4. The join protocol above assumes authentication of the neighbor node to the joining node (i.e., the neighbor node is not simply a relay node). This potentially assists in thwarting denial of service attacks on the joining node itself, primarily since it may allow the joining node to conclude it joined an improper network based on local communications only (if the neighbor node presented an unparsable cert or did not properly authenticate), rather than having to await a nonlocal verdict via the server that may take a long time to materialize. Here, again, more fine-grained authorization checks may be realized in scenarios where the joining node has more local intelligence to draw from. (Again, further details are outside scope of this document for now.)

5. The join protocol above includes mutual authentication between the joining node and the neighbor node and establishment of a shared "link key" (to use 802.15.4 parlance) between these two devices. This may be useful in case one wishes to trigger time synchronization between the joining node and the neighbor node contingent on frames secured using this pair-wise key only. This would strengthen TSCH security compared to that provided by the current 802.15.4e-2012 specification (which allows time synchronization to be also triggered by frames secured using a network-wide key, thereby opening the network to attacks by a single random compromised node, rather than a specific compromised node [the "clock tower" node] only.)

6. The join protocol above can also be "weakened", e.g., by removing authentication of the neighbor node to the joining node or vice-versa. As already said, this might open the protocol to wide-spread denial of service attacks on the network (in case the neighbor node simply forwards any joining node traffic, without inspection) or denial of service attacks on the joining node (in case the neighbor node is a bogus node or a node of an alien network). In some settings, though, practical trade-offs may favor such a "weakened" approach, e.g., if one wishes to "sprinkle" in sufficiently many neighbor nodes to guarantee connectivity to the server during initial deployment. If so, one should still have a fall-back strategy in place should denial of service attacks become a reality. (NOTE: These "weakened" versions will be analyzed in more detail in a later version of this draft.)

7. The join protocol above does not impose requirements on the security of the communication path between the neighbor node and

the server, except that "it should be there" (i.e., there is connectivity) although there may be additional requirements to counter, e.g., denial of service attacks on communications between neighbor node and server. (An exception here is if the server returns authorization-related information to the neighbor node [which we required to be secured], but which we will ignore for now.) Such minimization of dependencies between the join protocol and the routing protocol may be beneficial for use cases where one wishes to facilitate "random" installation process flows. Obviously, once a node is part of the network, it should be able to route packets (but that is not part of the join protocol itself, but next-stage phase).

8. The join protocol above tries to embrace a design where the order of joining would be mostly orthogonal to routing protocol topology considerations, if it all possible. In particular, it is aimed to take into account that not all installations follow the pattern where one has an operational network and where all non-local communications during the join protocol not of the type {joining node - neighbor router} are within the operational network (i.e., one would like to facilitate scenarios other than a tree-like structure, where network is built from tree root up onwards [this is highly relevant in building control settings]).

9. The join protocol above exchanges piggy-backed information between joining node, neighbor node, and server. This conceptually would allow very agressive implementations of the routing protocol, where one intertwines routing and join processes, by including some of the routing-related attributes as opaque strings in the piggy-backed fields. It should be noted that the join protocol already supports the routing tree of the existing network and the "new tree branch" {joining node - neighbor node}, so all "upwards routes" to the pre-existing tree roots are inherited right away. The only routes that may need defining are those towards the newbee joining node. For reliability reasons, this does require the joining node to have successfully concluded the join protocol first. As such, there seems to be no technical reason to intertwine these protocols: one should simply perform routing-related operations only *after* the join protocol ran its full course.

10. The join protocol above allows the neighbor node to influence with which server the joining node communicates, thus allowing a distributed implementation of the server.

11. The join protocol above assumes that the server arbitrages the correct value of supposedly common network parameters, such as the PAN identifier and ASN field. Here, one should note that

the neighbor node can indicate, e.g., any PAN identifier and any ASN entry to its liking in its beacon, which does not necessarily correspond to the "common world view" hereof by the server.

12. The join protocol above could in theory result in a node joining the network only locally (i.e., forming a two-node network with the neighbor node only), without the server or any other nodes becoming aware of this. This scenario could arrise if the joining node is unaware of some server-related context information and if the neighbor node simply usurps the server role itself. The impact of this "hidden node" type scenario depends on higher-layer, end-to-end design details. From a MAC perspective, this could simply mean that the two-node {joining node, neighbor node} network is conceptually represented by this neighbor node, where the internal structure of this two-node network remains hidden for other nodes.

1.4. Routing Behavior

TBD.

2. IANA Considerations

There is no IANA action required for this document.

3. Acknowledgments

TBD. Kris Pister provided the filtering example details for enhanced beacon frames. Yoshi Ohba, Subir Das, Giuseppe Piro, Pascal Thubert, and Kris Pister kindly provided feeback on previous versions of this document.

4. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-02 (work in progress), July 2014.

Author's Address

Rene Struik
Struik Security Consultancy

Email: rstruik.ext@gmail.com