6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2017

M. Vucinic
Inria
J. Simon
Linear Technology
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
March 12, 2017

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-02

Abstract

This document describes the minimal mechanisms required to support secure enrollment of a pledge, a device being added to an IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) network. It assumes that the pledge has been provisioned with a credential that is relevant to the deployment - the "one-touch" scenario. The goal of this configuration is to set link-layer keys, and to establish a secure end-to-end session between each pledge and the join registrar who may use that to further configure the pledge. Additional security behaviors and mechanisms may be added on top of this minimal framework.

Table of Contents

1. Introduction

This document describes the minimal feature set for a new device, termed pledge, to securely join a 6TiSCH network. As a successful outcome of this process, the pledge is able to securely communicate with its neighbors, participate in the routing structure of the network or establish a secure session with an Internet host.

When a pledge seeks admission to a 6TiSCH [RFC7554] network, it first needs to synchronize to the network. The pledge then configures its link-local IPv6 address and authenticates itself, and also validates that it is joining the right network. At this point it can expect to interact with the network to configure its link-layer keying material. Only then may the node establish an end-to-end secure session with an Internet host using OSCOAP [I-D.ietf-core-object-security] or DTLS [RFC6347]. Once the application requirements are known, the node interacts with its peers to request additional resources as needed, or to be reconfigured as the network changes [I-D.ietf-6tisch-6top-protocol].

This document presumes a network as described by [RFC7554], [I-D.ietf-6tisch-6top-protocol], and [I-D.ietf-6tisch-terminology]. It assumes the pledge pre-configured with either a:

- pre-shared key (PSK),
- raw public key (RPK),
- or a locally-valid certificate and a trust anchor.

As the outcome of the join process, the pledge expects one or more link-layer key(s) and optionally a temporary network identifier.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words may also appear in this document in lowercase, absent their normative meanings.

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-terminology], [RFC7252], [I-D.ietf-core-object-security], and [I-D.ietf-anima-bootstrapping-keyinfra]. The following terms are imported: pledge, join proxy, join registrar/coordinator, drop ship, imprint, enrollment, ownership voucher.

Pledge: the prospective device, which has the identity provided to at the factory.

Joined Node: the prospective device, after having completed the join process, often just called a Node.

Join Proxy (JP): a stateless relay that provides connectivity between the pledge and the join registrar/coordinator.

Join Registrar/Coordinator (JRC): central entity responsible for authentication and authorization of joining nodes.

3. One-Touch Assumptions

This document assumes the one-touch scenario, where devices are provided with some mechanism by which a secure association may be made in a controlled environment. There are many ways in which this might be done, and detailing any of them is out of scope for this document. But, some notion of how this might be done is important so that the underlying assumptions can be reasoned about.

Some examples of how to do this could include:

- JTAG interface
- serial (craft) console interface
- pushes of physical buttons simultaneous to network attachment

o unsecured devices operated in a Faraday cage

There are likely many other ways as well. What is assumed is that there can be a secure, private conversation between the Join Registrar/Coordinator, and the pledge, and that the two devices can exchange some trusted bytes of information.

4. Join Overview

This section describes the steps taken by a pledge in a 6TiSCH network. When a previously unknown device seeks admission to a 6TiSCH [RFC7554] network, the following exchange occurs:

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE8021542015]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which plays the role of Join Proxy (JP) for the pledge, and when it can expect to receive a frame.

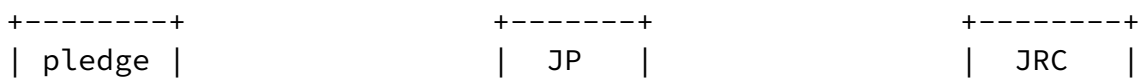
2. The pledge configures its link-local IPv6 address and advertizes it to Join Proxy (JP).

3. The pledge sends packets to JP in order to securely identify itself to the network. These packets are directed to the Join Registrar/Coordinator (JRC), which may be co-located on the JP or another device.

4. The pledge receives one or more packets from JRC (via the JP) that sets up one or more link-layer keys used to authenticate subsequent transmissions to peers.

From the pledge's perspective, minimal joining is a local phenomenon - the pledge only interacts with the JP, and it need not know how far it is from the DAG root, or how to route to the JRC. Only after establishing one or more link-layer keys does it need to know about the particulars of a 6TiSCH network.

The handshake is shown as a transaction diagram in Figure 1:



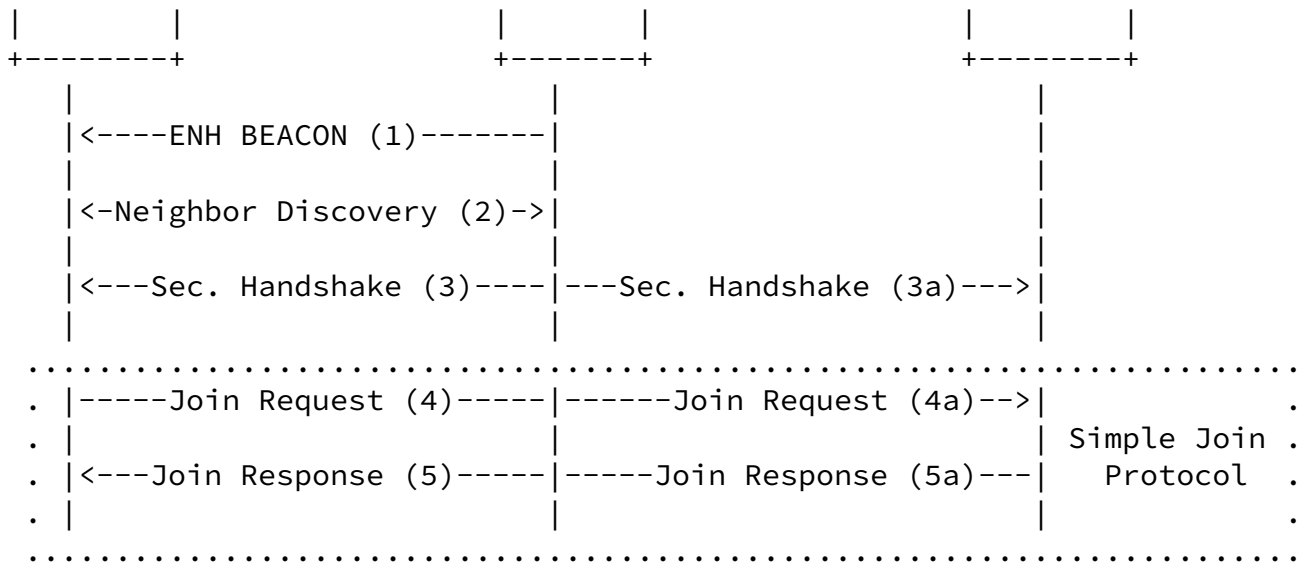


Figure 1: Overview of the join process.

The details of each step are described in the following sections.

4.1. Step 1 - Enhanced Beacon

Due to the channel hopping nature of 6TiSCH, transmissions take place on physical channels in a circular fashion. For that reason, Enhanced Beacons (EBs) are expected to be found by listening on a single channel. However, because some channels may be blacklisted, a

new pledge must listen for Enhanced Beacons for a certain period on each of the 16 possible channels. This search process entails having the pledge keep the receiver portion of its radio active for the entire period of time.

Once the pledge hears an EB from a JP, it synchronizes itself to the joining schedule using the cells contained in the EB. The selection of which beacon to start with is outside the scope of this document. Implementers SHOULD make use of information such as: whether the L2 address of the EB has been tried before, any Network Identifier [I-D.richardson-6tisch-join-enhanced-beacon] seen, and the strength of the signal. The pledge can be configured with the Network Identifier to seek when it is configured with the PSK.

Once a candidate network has been selected, the pledge can transition

into a low-power duty cycle, waking up only when the provided schedule indicates shared slots which the pledge may use for the join process.

At this point the pledge may proceed to step 2, or continue to listen for additional EBs.

A pledge which receives only Enhanced Beacons containing Network ID extensions [I-D.richardson-6tisch-join-enhanced-beacon] with the initiate bit cleared, SHOULD NOT proceed with this protocol on that network. The pledge SHOULD consider that it is in a network which manages join traffic, it SHOULD switch to [I-D.ietf-6tisch-dtsecurity-secure-join].

4.2. Step 2 - Neighbor Discovery

At this point, the pledge forms its link-local IPv6 address based on EUI64 and may register it at JP, in order to bootstrap the IPv6 neighbor tables. The Neighbor Discovery exchange shown in Figure 1 refers to a single round trip Neighbor Solicitation / Neighbor Advertisement exchange between the pledge and the JP. The pledge may further follow the Neighbor Discovery (ND) process described in Section 5 of [RFC6775].

4.3. Step 3 - Security Handshake

The security handshake between pledge and JRC uses Ephemeral Diffie-Hellman over COSE (EDHOC) [I-D.selander-ace-cose-ecdhe] to establish the shared session secret used to encrypt the Simple Join Protocol.

The security handshake step is OPTIONAL in case PSKs are used, while it is REQUIRED for RPKs and certificates.

When using certificates, the process continues as described in [I-D.selander-ace-cose-ecdhe], but MAY result in no network key being returned. In that case, the pledge enters a provisional situation where it provides access to an enrollment mechanism described in [I-D.ietf-6tisch-dtsecurity-secure-join].

If using a locally relevant certificate, the pledge will be able to validate the certificate of the JRC via a local trust anchor. In

that case, the JRC will return networks keys as in the PSK case. This would typically be the case for a device which has slept so long that it no longer has valid network keys and must go through a partial join process again.

In case the handshake step is omitted, the shared secret used for protection of the Simple Join Protocol in the next step is the PSK.

A consequence is that if the long-term PSK is compromised, keying material transferred as part of the join response is compromised as well. Physical compromise of the pledge, however, would also imply the compromise of the same keying material, as it is likely to be found in node's memory.

4.3.1. Pre-Shared Symmetric Key

The Diffie-Hellman key exchange and the use of EDHOC is optional, when using a pre-shared symmetric key. This cuts down on traffic between JRC and pledge, but requires pre-configuration of the shared key on both devices.

It is REQUIRED to use unique PSKs for each pledge. If there are multiple JRCs in the network (such as for redundancy), they would have to share a database of PSKs.

4.3.2. Asymmetric Keys

The Security Handshake step is required, when using asymmetric keys. Before conducting the Diffie-Hellman key exchange using EDHOC [I-D.selander-ace-cose-ecdhe] the pledge and JRC need to receive and validate each other's public key certificate. As detailed above, this can only be done for locally relevant (LDevID) certificates. IDevID certificates require entering a provisional state as described in [I-D.ietf-6tisch-dtsecurity-secure-join].

When RPKs are pre-configured at pledge and JRC, they can directly proceed to the handshake.

4.4. Step 4 - Simple Join Protocol - Join Request

The Join Request that makes part of the Simple Join Protocol is sent from the pledge to the JP using the shared slot as described in the EB, and forwarded to the JRC. Which slot the JP uses to transmit to the JRC is out of scope: some networks may wish to dedicate specific slots for this join traffic.

The join request is typically authenticated/encrypted end-to-end using AES-CCM-16-64-128 algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3. Algorithm negotiation is described in detail in [I-D.selander-ace-cose-ecdhe].

The nonce is derived from the shared secret, the pledge's EUI64 and a monotonically increasing counter initialized to 0 when first starting.

4.5. Step 5 - Simple Join Protocol - Join Response

The Join Response that makes part of the Simple Join Protocol is sent from the JRC to the pledge through JP that serves as a stateless relay. Packet containing the Join Response travels on the path from JRC to JP using pre-established routes in the network. The JP delivers it to the pledge using the slot information from the EB. JP operates as the application-layer proxy and does not keep any state to relay the message. It uses information sent in the clear within the join response to decide where to forward to.

The join response is typically authenticated/encrypted using AES-CCM-16-64-128 algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3.

The nonce is derived from the shared secret, pledge's EUI64 and a monotonically increasing counter matching that of the join request.

The join response contains one or more link-layer key(s) that the pledge will use for subsequent communication. Each key that is provided by the JRC is associated with an 802.15.4 key identifier. In other link-layer technologies, a different identifier may be substituted. Join Response optionally also contains an IEEE 802.15.4 short address [IEEE8021542015] assigned to pledge by JRC.

5. Architectural Overview and Communication through Join Proxy

The protocol in Figure 1 is implemented over Constrained Application Protocol (CoAP) [RFC7252]. The Pledge plays the role of a CoAP client, JRC the role of a CoAP server, while JP implements CoAP forward proxy functionality [RFC7252]. Since JP is also likely a

constrained device, it does not need to implement a cache but rather process forwarding-related CoAP options and make requests on behalf of pledge that is not yet part of the network.

The pledge communicates with a Join Proxy (JP) over link-local IPv6 addresses. The pledge designates a JP as a proxy by including in the CoAP requests to the JP the Proxy-Scheme option with value "coap" (CoAP-to-CoAP proxy). The pledge MUST include the Uri-Host option with its value set to the well-known JRC's alias - "6tisch.arpa". The pledge does not need to learn the actual IPv6 address of JRC at any time during the join protocol. The JP knows the address of the JRC, via a provisioning process that occured when the JP, acting as a pledge, joined. The initial bootstrap of the DODAG root would require explicit provisioning of the JRC address.

5.1. Stateless-Proxy CoAP Option

The CoAP proxy by default keeps per-client state information in order to forward the response towards the originator of the request (client). This state information comprises CoAP token, but the implementations also need to keep track of the IPv6 address of the host, as well as the corresponding UDP source port number. In the setting where the proxy is a constrained device and there are potentially many clients, as in the case of JP, this makes it prone to Denial of Service (DoS) attacks, due to the limited memory.

The Stateless-Proxy CoAP option (c.f. Figure 2) allows the proxy to insert within the request the state information necessary for relaying the response back to the client. Note that the proxy still needs to keep some state, such as for performing congestion control or request retransmission, but what is aimed with Stateless-Proxy option is to free the proxy from keeping per-client state.

Stateless-Proxy option is critical, Safe-to-Forward, not part of the cache key, not repeatable and opaque. When processed by OSCOAP, Stateless-Proxy option is neither encrypted nor integrity protected.

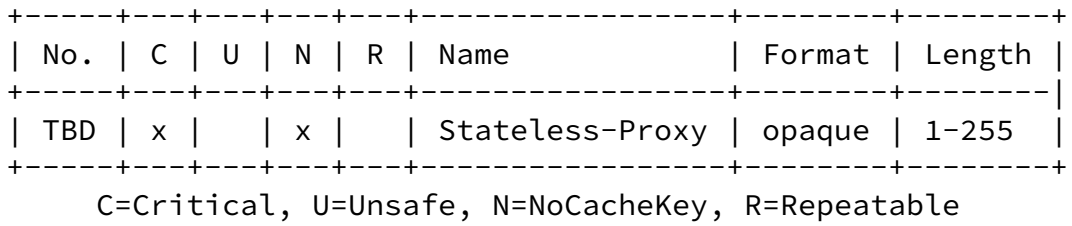


Figure 2: Stateless-Proxy CoAP Option

Upon reception of a Stateless-Proxy option, the CoAP server MUST echo it in the response. The value of the Stateless-Proxy option is

internal proxy state that is opaque to the server. Example state information includes IPv6 address of the client, its UDP source port, and the CoAP token. For security reasons, the state information MUST be authenticated, MUST include a freshness indicator (e.g. a sequence number or timestamp) and MAY be encrypted. The proxy may use an appropriate COSE structure [I-D.ietf-cose-msg] to wrap the state information as the value of the Stateless-Proxy option. The key used for encryption/authentication of the state information may be known only to the proxy.

Once the proxy has received the CoAP response with Stateless-Proxy option present, it decrypts/authenticates it, checks the freshness indicator and constructs the response for the client, based on the information present in the option value.

Note that a CoAP proxy using the Stateless-Proxy option is not able to return 5.04 Gateway Timeout error in case the request to the server times out. Likewise, if the response to the proxy's request does not contain the Stateless-Proxy option, for example when the option is not supported by the server, the proxy is not able to return the response to the client.

6. Security Handshake

In order to derive a shared session key, pledge and JRC run the EDHOC protocol [I-D.selander-ace-cose-ecdhe]. During this process, pledge and JRC mutually authenticate each other and verify authorization information before proceeding with the Simple Join Protocol. In case certificates are used for authentication, this document assumes that a special certificate with role attribute set has been provisioned to the JRC. This certificate is verified by pledge in order to authorize JRC to continue with the join process. How such a certificate is issued to the JRC is out of scope of this document.

Figure 3 details the exchanges between the pledge and JRC that take place during the execution of the security handshake. Format of EDHOC messages is specified in [I-D.selander-ace-cose-ecdhe].

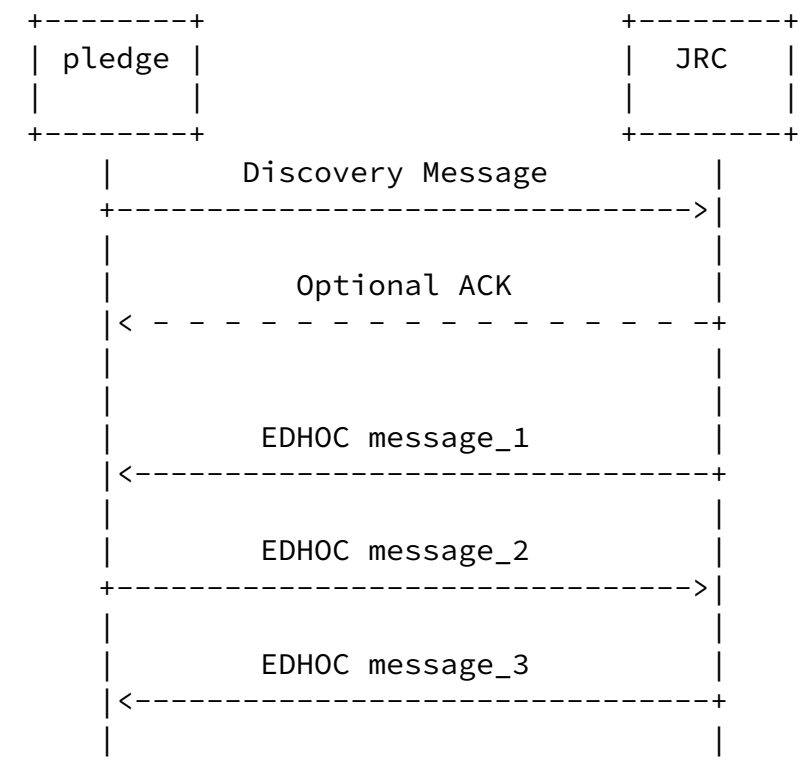


Figure 3: Transaction diagram of the security handshake.

6.1. Discovery Message

Pledge triggers the security handshake by sending a discovery message to the JRC. This initial message does not make part of the EDHOC handshake. JRC is the initiator of the EDHOC run and is able to schedule the execution of many concurrent enrollments at will by acknowledging the request and sending a separate, delayed response. The Discovery Message SHALL be mapped to a CoAP request:

- The request method is POST.
- The Proxy-Scheme option is set to "coap".
- The Uri-Host option is set to "6tisch.arpa".
- The Uri-Path option is set to "edhoc".
- The payload is optional and contains pledge's EUI-64.

7. Simple Join Protocol Specification

Simple Join Protocol is a single round trip protocol (c.f. Figure 4) that facilitates secure enrollment of a pledge, based on a shared symmetric secret. In case the pledge was provisioned by an

asymmetric key (certificate or RPK), Simple Join Protocol is preceded by a security handshake, described in Section 6. When the pledge is provisioned with a PSK, Simple Join Protocol may be run directly.

Pledge and JRC MUST protect their exchange end-to-end (i.e. through the proxy) using Object Security of CoAP (OSCOAP) [I-D.ietf-core-object-security].

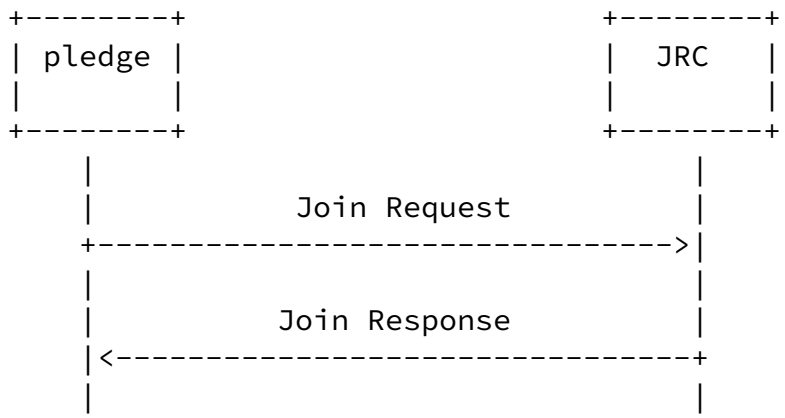


Figure 4: Transaction diagram of the Simple Join Protocol.

7.1. OSCOAP Security Context Instantiation

The OSCOAP security context MUST be derived at pledge and JRC as per Section 3.2 of [I-D.ietf-core-object-security] using HKDF [RFC5869] as the key derivation function.

- Master Secret MUST be the secret generated by the run of EDHOC as per Appendix B of [I-D.selander-ace-cose-ecdhe], or the PSK in case EDHOC step was omitted.

- Sender ID of the pledge MUST be set to the concatenation of its EUI-64 and byte string 0x00.

- Recipient ID (ID of JRC) MUST be set to the concatenation of pledge's EUI-64 and byte string 0x01. The construct uses pledge's EUI-64 to avoid nonce reuse in the response in the case same PSK is shared by a group of pledges.

- Algorithm MUST be set to AES-CCM-16-64-128 from [I-D.ietf-cose-msg]. CoAP messages are therefore protected with an 8-byte CCM authentication tag and the algorithm uses 13-byte long nonces.

The hash algorithm that instantiates HKDF MUST be SHA-256 [RFC4231]. The derivation in [I-D.ietf-core-object-security] results in traffic

keys and static IVs for each side of the conversation. Nonces are constructed by XOR'ing the static IV with current sequence number. The context derivation process occurs exactly once.

Implementations MUST ensure that multiple CoAP requests to different JRCs result in the use of the same OSCOAP context so that sequence numbers are properly incremented for each request. This may happen in a scenario where there are multiple 6TiSCH networks present and the pledge tries to join one network at a time.

7.2. Specification of Join Request

Message Join Request SHALL be mapped to a CoAP request:

- The request method is GET.

- The Proxy-Scheme option is set to "coap".

- The Uri-Host option is set to "6tisch.arpa".
- The Uri-Path option is set to "j".
- The object security option SHALL be set according to [I-D.ietf-core-object-security] and OSCOAP parameters set as described above.

7.3. Specification of Join Response

If OSCOAP processing is a success and the pledge is authorized to join the network, message Join Response SHALL be mapped to a CoAP response:

- The response Code is 2.05 (Content).
- Content-Format option is set to application/cbor.
- The payload is a CBOR [RFC7049] array containing, in order:
  - COSE Key Set, specified in [I-D.ietf-cose-msg], containing one or more link-layer keys. The mapping of individual keys to 802.15.4-specific parameters is described in Section 7.3.1.
  - Optional short address that is assigned to the pledge. The format of the short address follows Section 7.3.2.


```
payload = [
    COSE_KeySet,
    ? short_address,
]
```

7.3.1. Link-layer Keys Transported in COSE Key Set

Each key in the COSE Key Set [I-D.ietf-cose-msg] SHALL be a symmetric key. If "kid" parameter of the COSE Key structure is present, the corresponding keys SHALL belong to an IEEE 802.15.4 KeyIdMode 0x01 class. In that case, parameter "kid" of COSE Key structure SHALL be

used to carry IEEE 802.15.4 KeyIndex value. If the "kid" parameter is not present in the transported key, the application SHALL consider the key to be an IEEE 802.15.4 KeyIdMode 0x00 (implicit) key. This document does not support IEEE 802.15.4 KeyIdMode 0x02 and 0x03 class keys.

7.3.2. Short Address

Optional "short_address" structure transported as part of the join response payload represents IEEE 802.15.4 short address assigned to the pledge. It is encoded as CBOR array object, containing in order:

- Byte string, containing the 16-bit address.
- Optional lease time parameter, "lease_asn". The value of the "lease_asn" parameter is the 5-byte Absolute Slot Number (ASN) corresponding to its expiration, carried as a byte string in network byte order.

```
short_address = [
    address : bstr,
    ? lease_asn : bstr,
]
```

It is up to the joined node to request a new short address before the expiry of its previous address. The mechanism by which the node requests renewal is the same as during join procedure, as described in Section 10. The assigned short address is used for configuring both Layer 2 short address and Layer 3 addresses.

7.3.3. Error Handling

In the case JRC determines that pledge is not supposed to join the network (e.g. by failing to find an appropriate security context), it should respond with a 4.01 Unauthorized error. Upon reception of a 4.01 Unauthorized, the pledge SHALL attempt to join the next advertised 6TiSCH network. If all join attempts have failed at

pledge, the pledge SHOULD signal to the user by an out-of-band mechanism the presence of an error condition.

In the case that the JRC determines that the pledge is not (yet)

authorized to join the network, but a further zero-touch process might permit it, the JRC responds with a 2.05 (Content) code, but the payload contains the single CBOR string "prov" (for "provisional"). No link-layer keys or short address is returned.

This response is typically only expected when in asymmetric certificate mode using 802.1AR IDevID certificates. But for reasons of provisioning or device reuse, this could occur even when a one-touch PSK authentication process was expected.

8. Link-layer Requirements

In an operational 6TiSCH network, all frames MUST use link-layer frame security. The frame security options MUST include frame authentication, and MAY include frame encryption.

Link-layer frames are protected with a 16-byte key, and a 13-byte nonce constructed from current Absolute Slot Number (ASN) and the source (the JP for EBs) address, as shown in Figure 5:

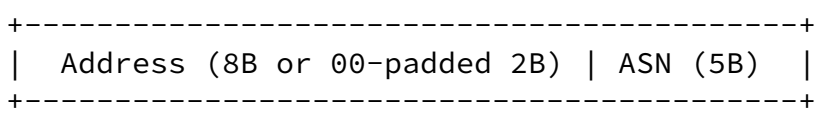


Figure 5: Link-layer CCM* nonce construction

The pledge does not initially do any authentication of the EB frames, as it does not know the K1 key. When sending frames, the pledge sends unencrypted and unauthenticated frames. JP accepts these frames (exempt mode in 802.15.4) for the duration of the join process. How JP learns whether the join process is ongoing is out of scope of this specification.

As the EB itself cannot be authenticated by pledge, an attacker may craft a frame that appears to be a valid EB, since the pledge can neither know the ASN a priori nor verify the address of the JP. This permits a Denial of Service (DoS) attack at the pledge. Beacon authentication keys are discussed in [I-D.ietf-6tisch-minimal].

9. Asymmetric Keys

Certificates or pre-configured RPKs may be used to exchange public keys between the pledge and JRC. The key pair is generated using

elliptic curve secp256r1, and the certificate containing the public key is signed using ECDSA. (XXX: would be nice to move to EdDSA)

The certificate itself may be a compact representation of an X.509 certificate, or a full X.509 certificate. Compact representation of X.509 certificates is out of scope of this specification. The certificate is signed by a root CA whose certificate is installed on all nodes participating in a particular 6TiSCH network, allowing each node to validate the certificate of the JRC or pledge as appropriate.

10. Rekeying and Rejoin

This protocol handles initial keying of the pledge. For reasons such as rejoining after a long sleep, or expiry of the short address, the joined node MAY send a new Join Request over the previously established secure end-to-end session with JRC. JRC responds with up-to-date keys and a short address. The node may also use the Simple Join Protocol exchange for node-initiated rekeying. How node learns that it should be rekeyed is out of scope. Additional work, such as in [I-D.richardson-6tisch-minimal-rekey] can be used.

11. Key Derivations

When EDHOC is used to derive keys, the cost of the asymmetric operation can be amortized over any additional connections that may be required between the node (during or after joining) and the JRC.

Each application SHOULD use a unique session key. EDHOC was designed with this in mind. In order to accomplish this, the EDHOC key derivation algorithm can be run with a different label. Other users of this key MUST define the label.

12. Security Considerations

In case PSKs are used, this document mandates that the pledge and JRC are pre-configured with unique keys. The uniqueness of generated nonces is guaranteed under the assumption of unique EUI64 identifiers for each pledge. Note that the address of the JRC does not take part in nonce construction. Therefore, even should an error occur, and a PSK shared by a group of nodes, the nonces constructed as part of the different responses are unique. The PSK is still important for authentication of the pledge and authentication of the JRC to the pledge. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible. The well known problem with Bluetooth headsets with a "0000" pin applies here. The design differentiates between nonces constructed for requests and nonces constructed for responses by different sender identifiers (0x00 for pledge and 0x01

for JRC).

Being a stateless relay, JP blindly forwards the join traffic into the network. While the exchange between pledge and JP takes place over a shared cell, join traffic is forwarded using dedicated cells on the JP to JRC path. In case of distributed scheduling, the join traffic may therefore cause intermediate nodes to request additional bandwidth. (EDNOTE: this is a problem that needs to be solved) Because the relay operation of JP is implemented at the application layer, JP is the only hop on the JP-6LBR path that can distinguish join traffic from regular IP traffic in the network. It is therefore recommended to implement stateless rate limiting at JP: a simple bandwidth (in bytes or packets/second) cap would be appropriate.

The shared nature of the "minimal" cell used for join traffic makes the network prone to DoS attacks by congesting the JP with bogus radio traffic. As such an attacker is limited by emitted radio power, redundancy in the number of deployed JPs alleviates the issue and also gives the pledge a possibility to use the best available link for join. How a network node decides to become a JP is out of scope of this specification.

At the time of the join, the pledge has no means of verifying the content in the EB and has to accept it at "face value". In case the pledge tries to join an attacker's network, the join response message in such cases will either fail the security check or time out. The pledge may implement a blacklist in order to filter out undesired beacons and try to join the next seemingly valid network. The blacklist alleviates the issue but is effectively limited by the node's available memory. Such bogus beacons will prolong the join time of the pledge and so the time spent in "minimal" [I-D.ietf-6tisch-minimal] duty cycle mode.

13. Privacy Considerations

This specification relies on the uniqueness of EUI64 that is transferred in clear as part of the security context identifier. (EDNOTE: should we say IID here?) Privacy implications of using such long-term identifier are discussed in [RFC7721] and comprise correlation of activities over time, location tracking, address scanning and device-specific vulnerability exploitation. Since the join protocol is executed rarely compared to the network lifetime,

long-term threats that arise from using EUI64 are minimal. In addition, the join response message contains an optional short address which can be assigned by JRC to the pledge. The short address is independent of the long-term identifier EUI64 and is encrypted in the response. For that reason, it is not possible to correlate the short address with the EUI64 used during the join. Use of short addresses once the join protocol completes mitigates the aforementioned privacy risks. In addition, EDHOC may be used for

identity protection during the join protocol by generating a random context identifier in place of the EUI64 [I-D.selander-ace-cose-ecdhe].

14. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[[this document]]" with the RFC number of this specification.

This document allocates a well known name under the .arpa name space according to the rules given in: [RFC3172]. The name "6tisch.arpa" is requested. No subdomains are expected. No A, AAAA or PTR record is requested.

14.1. CoAP Option Numbers Registry

The Stateless-Proxy option is added to the CoAP Option Numbers registry:

Number	Name	Reference
TBD	Stateless-Proxy	[[this document]]

15. Acknowledgments

The work on this document has been partially supported by the European Union's H2020 Programme for research, technological development and demonstration under grant agreement No 644852, project ARMOUR.

The authors are grateful to Thomas Watteyne and Goeran Selander for

reviewing the draft. The authors would also like to thank Francesca Palombini and Ludwig Seitz for participating in the discussions that have helped shape the document.

16. References

16.1. Normative References

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-ietf-core-object-security-01 (work in progress), December 2016.

[I-D.ietf-cose-msg]
Schaad, J., "CBOR Object Signing and Encryption (COSE)", draft-ietf-cose-msg-24 (work in progress), November 2016.

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

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <http://www.rfc-editor.org/info/rfc3172>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

16.2. Informative References

[I-D.ietf-6tisch-6top-protocol]

Wang, Q. and X. Vilajosana, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-03 (work in progress), October 2016.

[I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.

[I-D.ietf-6tisch-minimal]
Vilajosana, X., Pister, K., and T. Watteyne, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-21 (work in progress), February 2017.

[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-08 (work in progress), December 2016.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-04 (work in progress), October 2016.

[I-D.richardson-6tisch-join-enhanced-beacon]
Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join Information", draft-richardson-6tisch-join-enhanced-beacon-01 (work in progress), March 2017.

[I-D.richardson-6tisch-minimal-rekey]
Richardson, M., "Minimal Security rekeying mechanism for 6TiSCH", draft-richardson-6tisch-minimal-rekey-01 (work in progress), February 2017.

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral

Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-04 (work in progress), October 2016.

[IEEE8021542015]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <http://www.rfc-editor.org/info/rfc4231>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <http://www.rfc-editor.org/info/rfc6775>.

[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, <http://www.rfc-editor.org/info/rfc7554>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <http://www.rfc-editor.org/info/rfc7721>.

Appendix A. Example

Figure 6 illustrates a join protocol exchange in case PSKs are used. Pledge instantiates the OSCOAP context and derives the traffic keys and nonces from the PSK. It uses the instantiated context to protect the CoAP request addressed with Proxy-Scheme option and well-known host name of JRC in the Uri-Host option. The example assumes a JP that is already aware of JRC's IPv6 address and does not need to resolve the well-known "6tisch.arpa" host name. Triggered by the presence of Proxy-Scheme option, JP forwards the request to the JRC and adds the Stateless-Proxy option with value set to the internally needed state, authentication tag, and a freshness indicator. Once JRC receives the request, it looks up the correct context based on the Sender ID (sid) parameter. It reconstructs OSCOAP's external Additional Authenticated Data (AAD) needed for verification based on:

- Version field of the received CoAP header.
- Code field of the received CoAP header.
- Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msg].
- Request ID being set to pledge's EUI-64 concatenated with 0x00.
- Request Sequence number set to the value of "Partial IV" of the received COSE object.

Replay protection is ensured by OSCOAP and the tracking of sequence numbers at each side. In the example below, the response contains sequence number 7 meaning that there have already been some attempts to join under a given context, not coming from the pledge. Once JP receives the response, it authenticates the Stateless-Proxy option before deciding where to forward. JP sets its internal state to that found in the Stateless-Proxy option. Note that JP does not posses the key to decrypt the COSE object present in the payload so the join_response object is opaque to it. The response is matched to the

request and verified for replay protection at pledge using OSCOAP processing rules.

```
     <--E2E OSCOAP-->
     Client  Proxy Server
```

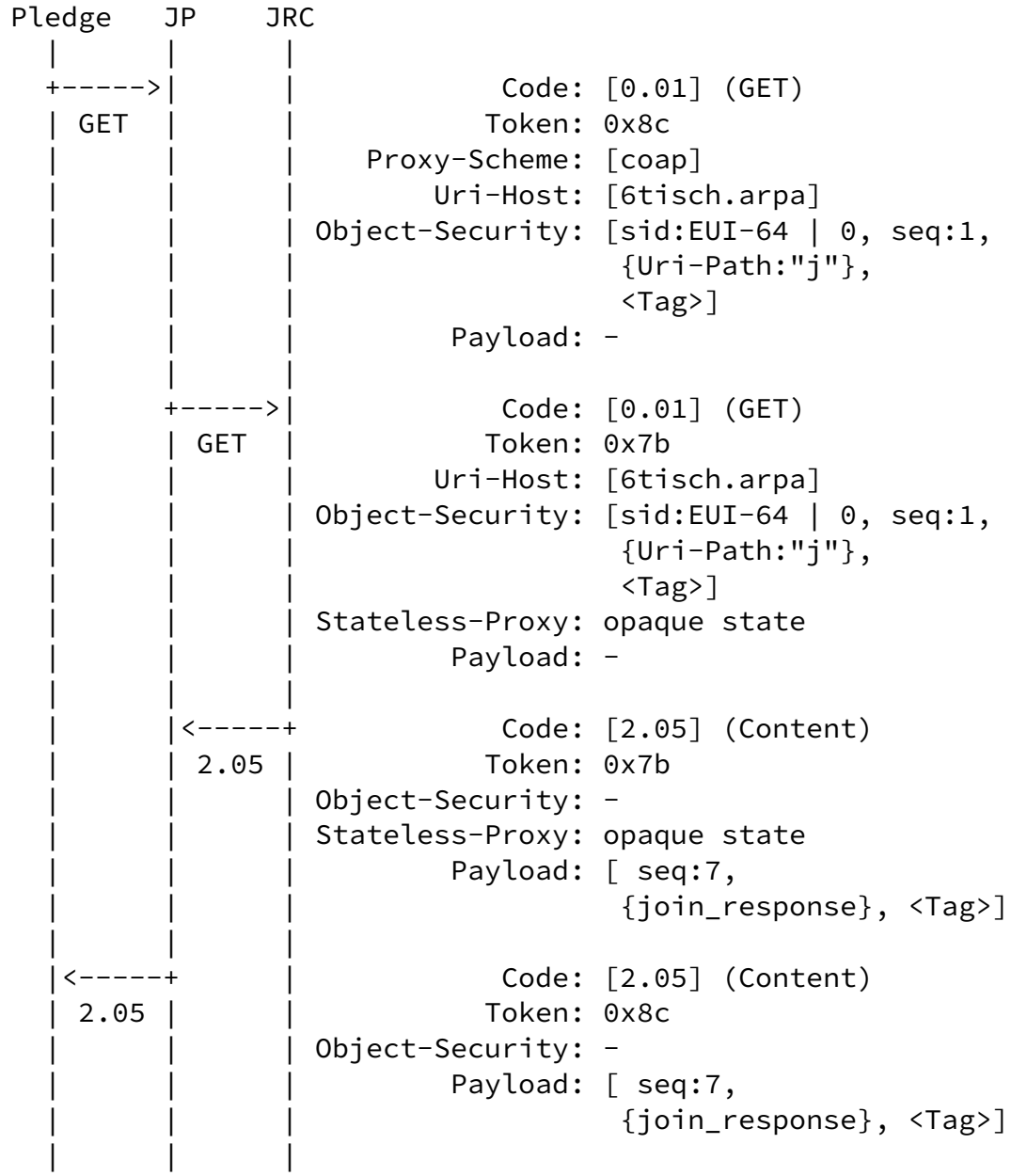


Figure 6: Example of a join protocol exchange with a PSK. {} denotes encryption and authentication, [] denotes authentication.

Where join_response is as follows.

```
join_response:
[
    [   / COSE Key Set array with a single key /
        {
             1 : 4, / key type symmetric /
             2 : h'01', / key id /
            -1 : h'e6bf4287c2d7618d6a9687445ffd33e6' / key value /
        }
    ],
    [
        h'af93' / assigned short address /
    ]
]
```

Encodes to h'8281a301040241012050e6bf4287c2d7618d6a9687445ffd33e68142af93' with a size of 30 bytes.

Authors' Addresses

Malisa Vucinic
Inria
2 Rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

Jonathan Simon
Linear Technology
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jsimon@linear.com

Kris Pister
University of California Berkeley
512 Cory Hall
Berkeley, CA 94720
USA

Email: pister@eecs.berkeley.edu

Michael Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z5V7
Canada

Email: mcr+ietf@sandelman.ca