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

M. Vucinic, Ed.
University of Montenegro
J. Simon
Analog Devices
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
March 05, 2018

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-05

Abstract

This document describes the minimal framework required for a new device, called "pledge", to securely join a 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) network. The framework requires that the pledge and the JRC (join registrar/coordinator, a central entity), share a symmetric key. How this key is provisioned is out of scope of this document. Through a single CoAP (Constrained Application Protocol) request-response exchange secured by OSCORE (Object Security for Constrained RESTful Environments), the pledge requests admission into the network and the JRC configures it with link-layer keying material and a short link-layer address. This specification defines the message format, a new Stateless-Proxy CoAP option, and configures the rest of the 6TiSCH communication stack for this join process to occur in a secure manner. Additional security mechanisms may be added on top of this minimal framework.

Table of Contents

1. Introduction

This document presumes a 6TiSCH network as described by [RFC7554] and [RFC8180]. By design, nodes in a 6TiSCH network [RFC7554] have their radio turned off most of the time, to conserve energy. As a consequence, the link used by a new device for joining the network has limited bandwidth [RFC8180]. The secure join solution defined in this document therefore keeps the number of over-the-air exchanges for join purposes to a minimum.

The micro-controllers at the heart of 6TiSCH nodes have a small amount of code memory. It is therefore paramount to reuse existing protocols available as part of the 6TiSCH stack. At the application layer, the 6TiSCH stack already relies on CoAP [RFC7252] for web transfer, and on OSCORE [I-D.ietf-core-object-security] for its end-to-end security. The secure join solution defined in this document therefore reuses those two protocols as its building blocks.

This document defines a secure join solution for a new device, called "pledge", to securely join a 6TiSCH network. The specification defines a 6TiSCH Join Protocol (6JP) used by the pledge to request admission into a network managed by the JRC, and for the JRC to configure the pledge with the necessary parameters, a new CoAP option, and configures different layers of the 6TiSCH protocol stack for the join process to occur in a secure manner.

It assumes the presence of a JRC (join registrar/coordinator), a central entity. It further assumes that the pledge and the JRC share a symmetric key, called PSK (pre-shared key). The PSK is used to configure OSCORE to provide a secure channel to 6JP. How the PSK is installed is out of scope of this document.

When the pledge seeks admission to a 6TiSCH network, it first synchronizes to it, by initiating the passive scan defined in [IEEE802.15.4-2015]. The pledge then exchanges messages with the JRC; these messages can be forwarded by nodes already part of the 6TiSCH network. The messages exchanged allow the JRC and the pledge to mutually authenticate, based on the PSK. They also allow the JRC to configure the pledge with link-layer keying material and a short link-layer address. After this secure join process successfully completes, the joined node can interact with its neighbors to request additional bandwidth using the 6top Protocol [I-D.ietf-6tisch-6top-protocol] and start sending the application traffic.

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 [RFC8152].

The specification also includes a set of informative examples using the CBOR diagnostic notation [I-D.ietf-cbor-cddl].

The following terms defined in [I-D.ietf-6tisch-terminology] are used extensively throughout this document:

- pledge
- joined node
- join proxy (JP)
- join registrar/coordinator (JRC)
- enhanced beacon (EB)
- join protocol
- join process

In addition, we use the generic terms "network identifier" and "pledge identifier". See Section 3.

3. Identifiers

The "network identifier" uniquely identifies the 6TiSCH network in the namespace managed by a JRC. Typically, this is the 16-bit Personal Area Network Identifier (PAN ID) defined in [IEEE802.15.4-2015]. Companion documents can specify the use of a different network identifier for join purposes, but this is out of scope of this specification. Such identifier needs to be carried within Enhanced Beacon (EB) frames.

The "pledge identifier" uniquely identifies the pledge in the namespace managed by a JRC. The pledge identifier is typically the globally unique 64-bit Extended Unique Identifier (EUI-64) of the

IEEE 802.15.4 device. This identifier is used to generate the IPv6 addresses of the pledge and to identify it during the execution of the join protocol. For privacy reasons, it is possible to use an identifier different from the EUI-64 (e.g. a random string). See Section 12.

4. One-Touch Assumption

This document assumes a one-touch scenario. The pledge is provisioned with certain parameters before attempting to join the network, and the same parameters are provisioned to the JRC.

There are many ways by which this provisioning can be done. Physically, the parameters can be written into the pledge using a number of mechanisms, such as a JTAG interface, a serial (craft) console interface, pushing buttons simultaneously on different devices, over-the-air configuration in a Faraday cage, etc. The provisioning can be done by the vendor, the manufacturer, the integrator, etc.

Details of how this provisioning is done is out of scope of this document. What is assumed is that there can be a secure, private conversation between the JRC and the pledge, and that the two devices can exchange the parameters.

Parameters that are provisioned to the pledge include:

- Pre-Shared Key (PSK). The JRC additionally needs to store the identifier of the pledge bound to the given PSK. The PSK SHOULD be at least 128 bits in length, generated uniformly at random. It is RECOMMENDED to generate the PSK with a cryptographically secure pseudorandom number generator. Each pledge SHOULD be provisioned with a unique PSK.

- Optionally, a network identifier. Provisioning the network identifier to the pledge is RECOMMENDED, as it significantly speeds up the join process. In case this parameter is not provisioned, the pledge attempts to join one network at a time.

- Optionally, any non-default algorithms. Mandatory to implement and default algorithms are specified in Section 9.6.

5. Join Overview

This section describes the steps taken by a pledge in a 6TiSCH network. When a pledge seeks admission to a 6TiSCH network, the following exchange occurs:

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE802.15.4-2015]. 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 advertises it to the join proxy (JP).

3. The pledge sends a Join Request to the JP in order to securely identify itself to the network. The Join Request is directed to the JRC, which may be co-located on the JP or another device.

4. In case of successful processing of the request, the pledge receives a join response from JRC (via the JP) that sets up one or more link-layer keys used to authenticate and encrypt subsequent transmissions to peers, and a short link-layer address for the pledge.

From the pledge's perspective, joining is a local phenomenon - the pledge only interacts with the JP, and it needs not know how far it is from the 6LBR, 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 join process is shown as a transaction diagram in Figure 1:

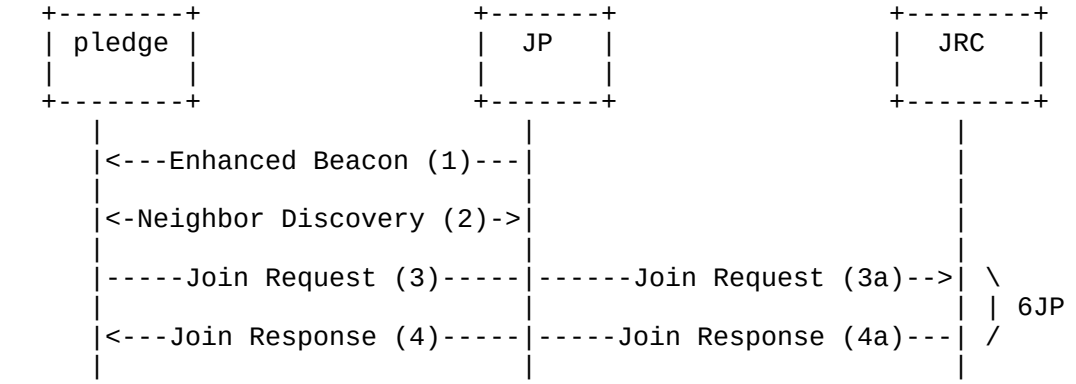

Figure 1: Overview of a successful join process. 6JP stands for 6TiSCH Join Protocol.

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

5.1. Step 1 - Enhanced Beacon

The pledge synchronizes to the network by listening for, and receiving, an Enhanced Beacon (EB) sent by a node already in the network. This process is entirely defined by [IEEE802.15.4-2015], and described in [RFC7554].

Once the pledge hears an EB, it synchronizes to the joining schedule using the cells contained in the EB. The pledge can hear multiple EBs; the selection of which EB to use is out of the scope for this document, and is discussed in [RFC7554]. Implementers should make use of information such as: what network identifier the EB contains, whether the source link-layer address of the EB has been tried before, what signal strength the different EBs were received at, etc. In addition, the pledge may be pre-configured to search for EBs with a specific network identifier.

If the pledge is not provisioned with the network identifier, it attempts to join one network at a time, as described in Section 9.3.

Once the pledge selects the EB, it synchronizes to it and transitions into a low-power mode. It deeply duty cycles its radio, switching the radio on when the provided schedule indicates slots which the pledge may use for the join process. During the remainder of the join process, the node that has sent the EB to the pledge plays the role of JP.

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

5.2. Step 2 - Neighbor Discovery

The pledge forms its link-local IPv6 address based on the interface identifier, as per [RFC4944]. The pledge MAY perform the Neighbor Solicitation / Neighbor Advertisement exchange with the JP, as per Section 5.5.1 of [RFC6775]. The pledge and the JP use their link-local IPv6 addresses for all subsequent communication during the join process.

Note that Neighbor Discovery exchanges at this point are not protected with link-layer security as the pledge is not in possession of the keys. How JP accepts these unprotected frames is discussed in Section 6.

5.3. Step 3 - Join Request

The Join Request is a message sent from the pledge to the JP, and which the JP forwards to the JRC. The JP forwards the Join Request to the JRC on the existing 6TiSCH network. How exactly this happens is out of scope of this document; some networks may wish to dedicate specific slots for this join traffic.

The Join Request is authenticated/encrypted end-to-end using an AEAD (Authenticated Encryption with Associated Data) algorithm from [RFC8152] and a key derived from the PSK, the pledge identifier and a request-specific constant value. Algorithms which MUST be implemented are specified in Section 9.6.

The nonce used when securing the Join Request is derived from the PSK, the pledge identifier and a monotonically increasing counter initialized to 0 when first starting.

Join Request message is specified in Section 9.1, while the details on security processing can be found in Section 7 of [I-D.ietf-core-object-security].

5.4. Step 4 - Join Response

The Join Response is sent by the JRC to the pledge, and is forwarded through the JP as it serves as a stateless relay. The packet containing the Join Response travels from the JRC to JP using the operating routes in the 6TiSCH network. The JP delivers it to the pledge. The 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 authenticated/encrypted end-to-end using an AEAD algorithm from [RFC8152]. The key used to protect the response is different from the one used to protect the request (both are derived from the PSK, as explained in Section 8.1). The response is protected using the same nonce as in the 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. The Join Response also contains an IEEE 802.15.4 short address [IEEE802.15.4-2015] assigned by the JRC to the pledge, and optionally the IPv6 address of the JRC.

Join Response message is specified in Section 9.2, while the details on security processing can be found in Section 7 of [I-D.ietf-core-object-security].

6. Link-layer Configuration

In an operational 6TiSCH network, all frames MUST use link-layer frame security [RFC8180]. The IEEE 802.15.4 security attributes MUST include frame authenticity, and MAY include frame confidentiality (i.e. encryption).

As specified in [RFC8180], the network uses a key termed as K1 to authenticate EBs and a key termed as K2 to authenticate and optionally encrypt DATA and ACKNOWLEDGMENT frames. The keys K1 and K2 MAY be the same key (same value and IEEE 802.15.4 index). How the JRC communicates these keys to 6LBR is out of scope of this specification.

The pledge does not initially do any authenticity check of the EB frames, as it does not know the K1 key. The pledge is still able to parse the contents of the received EBs and synchronize to the network, as EBs are not encrypted [RFC8180].

When sending frames during the join process, the pledge sends unencrypted and unauthenticated frames. The JP accepts these frames (using the "exempt mode" in 802.15.4) for the duration of the join process. How the JP learns whether the join process is ongoing is out of scope of this specification.

As the EB itself cannot be authenticated by the 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 opens up a possibility of DoS attack, as discussed in Section 11. Beacon authentication keys are discussed in [RFC8180].

7. Network-layer Configuration

The pledge and the JP SHOULD keep a separate neighbor cache for untrusted entries and use it to store each other’s information during the join process. Mixing neighbor entries belonging to pledges and nodes that are part of the network opens up the JP to a DoS attack. How the pledge and the JP decide to transition each other from untrusted to trusted cache, once the join process completes, is out of scope. One implementation technique is to use the information whether the incoming frames are secured at the link layer.

The pledge does not communicate with the JRC at the network layer. This allows the pledge to join without knowing the IPv6 address of

the JRC. Instead, the pledge communicates with the JP at the network layer, and with the JRC at the application layer, as specified in Section 8.

The JP communicates with the JRC over global IPv6 addresses. The JP discovers the network prefix and configures its global IPv6 address upon successful completion of the join process and the obtention of link-layer keys. The pledge learns the actual IPv6 address of the JRC from the Join Response, as specified in Section 9.2; it uses it once joined in order to operate as a JP.

The JRC can be co-located on the 6LBR. In this special case, the IPv6 address of the JRC can be omitted from the Join Response message for space optimization. The 6LBR then MUST set the DODAGID field in RPL DIOs [RFC6550] to its IPv6 address. The pledge learns the address of the JRC once joined and upon the reception of a first RPL DIO message, and uses it to operate as a JP.

Before the 6TiSCH network is started, the 6LBR MUST be provisioned with the IPv6 address of the JRC.

7.1. Identification of Join Request Traffic

The join request traffic that is proxied by the Join Proxy comes from unauthenticated nodes, and there may be an arbitrary amount of it. In particular, an attacker may send fraudulent traffic in attempt to overwhelm the network.

When operating as part of a [RFC8180] 6TiSCH minimal network using reasonable scheduling algorithms, the join request traffic present may cause intermediate nodes to request additional bandwidth. An attacker could use this property to cause the network to overcommit bandwidth (and energy) to the join process.

The Join Proxy is aware of what traffic is join request traffic, and so can avoid allocating additional bandwidth itself. The Join Proxy SHOULD implement a bandwidth cap on outgoing join request traffic. This cap will not protect intermediate nodes as they can not tell join request traffic from regular traffic. Despite the bandwidth cap implemented separately on each Join Proxy, the aggregate join request traffic from many Join Proxies may cause intermediate nodes to decide to allocate additional cells. It is undesirable to to so in response to the join request traffic. In order to permit the intermediate nodes to avoid this, the traffic needs to be tagged in some way.

[RFC2597] defines a set of per-hop behaviors that may be encoded into the Diffserv Code Points (DSCPs). The Join Proxy SHOULD set the DSCP

of join request packets that it produces as part of the relay process to AF43 code point (See Section 6 of [RFC2597]).

A Join Proxy that does not set the DSCP on traffic forwarded should set it to zero so that it is compressed out.

A Scheduling Function (SF) running on 6TiSCH nodes SHOULD NOT allocate additional cells as a result of traffic with code point AF43. Companion SF documents SHOULD specify how this recommended behavior is achieved.

7.2. Identification of Join Response Traffic

The JRC SHOULD set the DSCP of join response packets addressed to the Join Proxy to AF42 code point. Join response traffic can not be induced by an attacker as it is generated only in response to legitimate pledges (see Section 9.3). AF42 has lower drop probability than AF43, giving join response traffic priority in buffers over join request traffic.

When the JRC is not co-located with the 6LBR, then the code point provides a clear indication to the 6LBR that this is join response traffic.

Due to the convergecast nature of the DODAG, the 6LBR links are often the most congested, and from that point down there is progressively less (or equal) congestion. If the 6LBR paces itself when sending join response traffic then it ought to never exceed the bandwidth allocated to the best effort traffic cells. If the 6LBR has the capacity (if it is not constrained) then it should provide some buffers in order to satisfy the Assured Forwarding behavior.

Companion SF documents SHOULD specify how traffic with code point AF42 is handled with respect to cell allocation.

8. Application-level Configuration

The Join Request/Join Response exchange in Figure 1 is carried over CoAP [RFC7252] and secured using OSCORE [I-D.ietf-core-object-security]. The pledge plays the role of a CoAP client; the JRC plays the role of a CoAP server. The JP implements CoAP forward proxy functionality [RFC7252]. Because the JP can also be a constrained device, it cannot implement a cache. Rather, the JP processes forwarding-related CoAP options and makes requests on behalf of the pledge, in a stateless manner by using the Stateless-Proxy option defined in this document.

The pledge designates a JP as a proxy by including the Proxy-Scheme option in CoAP requests it sends to the JP. The pledge also includes in the requests the Uri-Host option with its value set to the well-known JRC's alias, as specified in Section 9.1.

The JP resolves the alias to the IPv6 address of the JRC that it learned when it acted as a pledge, and joined the network. This allows the JP to reach the JRC at the network layer and forward the requests on behalf of the pledge.

The JP MUST add a Stateless-Proxy option to all the requests that it forwards on behalf of the pledge as part of the join process.

The value of the Stateless-Proxy option is set to the internal JP state, needed to forward the Join Response message to the pledge. The Stateless-Proxy option handling is defined in Section 10.

The JP also tags all packets carrying the Join Request message at the network layer, as specified in Section 7.1.

8.1. OSCORE Security Context

Before the pledge and the JRC may start exchanging CoAP messages protected with OSCORE, they need to derive the OSCORE security context from the parameters provisioned out-of-band, as discussed in Section 4.

The OSCORE security context MUST be derived at the pledge and the JRC as per Section 3 of [I-D.ietf-core-object-security].

- the Master Secret MUST be the PSK.
- the Master Salt MUST be the pledge identifier.
- the Sender ID of the pledge MUST be set to byte string 0x00.
- the Recipient ID (ID of the JRC) MUST be set to byte string 0x01.
- the Algorithm MUST be set to the value from [RFC8152], agreed out-of-band by the same mechanism used to provision the PSK. The default is AES-CCM-16-64-128.
- the Key Derivation Function MUST be agreed out-of-band. Default is HKDF SHA-256 [RFC5869].

The derivation in [I-D.ietf-core-object-security] results in traffic keys and a common IV for each side of the conversation. Nonces are constructed by XOR'ing the common IV with the current sequence number

and sender identifier. For details on nonce construction, refer to [I-D.ietf-core-object-security].

Implementations MUST ensure that multiple CoAP requests to different JRCs result in the use of the same OSCORE context, so that the sequence numbers are properly incremented for each request. The pledge typically sends requests to different JRCs if it is not provisioned with the network identifier and attempts to join one network at a time. A simple implementation technique is to instantiate the OSCORE security context with a given PSK only once and use it for all subsequent requests. Failure to comply will break the confidentiality property of the AEAD algorithm due to the nonce reuse.

8.1.1. Persistency

Implementations MUST ensure that mutable OSCORE context parameters (Sender Sequence Number, Replay Window) are stored in persistent memory. A technique that prevents reuse of sequence numbers, detailed in Section 6.5.1 of [I-D.ietf-core-object-security], MUST be implemented. Each update of the OSCORE Replay Window MUST be written to persistent memory.

This is an important security requirement in order to guarantee nonce uniqueness and resistance to replay attacks across reboots and rejoins. Traffic between the pledge and the JRC is rare, making security outweigh the cost of writing to persistent memory.

9. 6TiSCH Join Protocol

6TiSCH Join Protocol (6JP) is a lightweight protocol over CoAP [RFC7252] and a secure channel provided by OSCORE [I-D.ietf-core-object-security]. 6JP allows the pledge to request admission into a network managed by the JRC, and for the JRC to configure the pledge with the parameters necessary for joining the network. These parameters are: link-layer keys in use, IEEE 802.15.4 short address assigned to the pledge, and the IPv6 address of the JRC.

This section specifies the 6JP bindings to CoAP and OSCORE, 6JP message formats and the semantics of different fields.

6JP relies on the security properties provided by OSCORE. This includes end-to-end confidentiality, data authenticity, replay protection, and a secure binding of responses to requests.

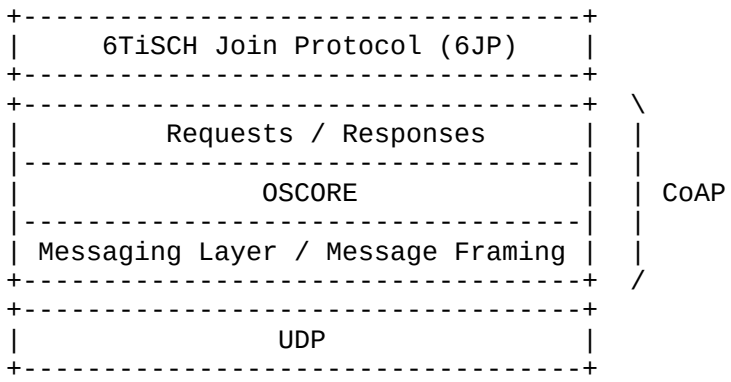

Figure 2: Abstract layering of 6JP.

6JP consists of two messages:

- the Join Request message, sent by the pledge to the JRC, proxied by the JP. The Join Request message and its mapping to CoAP is specified in Section 9.1.

- the Join Response message, sent by the JRC to the pledge if the JRC successfully processes the Join Request using OSCORE and it determines through a mechanism that is out of scope of this specification that the pledge is authorized to join the network. The Join Response message is proxied by the JP. The Join Response message and its mapping to CoAP is specified in Section 9.2.

The payload of 6JP messages is encoded with CBOR [RFC7049], with some parameters being optional. The first byte of the CBOR-encoded byte string contains the CBOR major type and additional information (e.g. the number of elements in an array). In case of an array, the CBOR decoder decides based on this additional information if a certain optional parameter is present or not.

9.1. Specification of the Join Request

The Join Request the pledge sends SHALL be mapped to a CoAP request:

- The request method is POST.

- The type is Non-confirmable (NON).

- 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]. The OSCORE security context used is the one derived in Section 8.1. The OSCORE Context Hint SHALL be set to the pledge identifier. The OSCORE Context Hint allows the JRC to retrieve the security context for a given pledge.

- The payload is a CBOR array [RFC7049] containing, in order:

  * Byte string, containing the identifier of the network that the pledge is attempting to join. This enables the JRC to manage multiple 6TiSCH networks.

```
request_payload = [
    network_identifier : bstr,
]
```

9.2. Specification of the Join Response

The Join Response the JRC sends SHALL be mapped to a CoAP response:

- The response Code is 2.04 (Changed).

- The payload is a CBOR array [RFC7049] containing, in order:

  * the COSE Key Set, specified in [RFC8152], containing one or more link-layer keys. The mapping of individual keys to 802.15.4-specific parameters is described in Section 9.2.1.

  * the link-layer short address to be used by the pledge. The format of the short address follows Section 9.2.2.

  * optionally, the IPv6 address of the JRC, encoded as a byte string, with the length of 16 bytes. If the IPv6 address of the JRC is not present in the Join Response, this indicates the JRC is co-located with the 6LBR, and has the same IPv6 address as the 6LBR. See Section 7.

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

9.2.1. Link-layer Keys Transported in the COSE Key Set

Each key in the COSE Key Set [RFC8152] SHALL be a symmetric key. The first key in the COSE Key Set SHALL be used as the K1 key from [RFC8180]. The second key in the COSE Key Set SHALL be used as the K2 key from [RFC8180]. In the case where the network uses the same key for K1 and K2, the COSE Key Set SHALL carry a single key.

If the COSE Key Set carries more than 2 keys, the implementation SHOULD consider the response as malformed.

If the "kid" parameter of the COSE Key structure is present, the corresponding key SHALL be used as IEEE 802.15.4 KeyIdMode 0x01 (index). In that case, parameter "kid" of the COSE Key structure SHALL be used to carry the IEEE 802.15.4 KeyIndex value.

If the length of the "kid" parameter is more than 1 byte (length defined by [IEEE802.15.4-2015]), the implementation SHOULD consider the response as malformed.

If the "kid" parameter is not present in the transported key, the implementation 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. In the case that the response is considered malformed, the implementation SHOULD indicate to the user through an out-of-band mechanism the presence of an error condition.

9.2.2. Short Address

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

- Byte string, containing the 16-bit address.
- Optionally, the 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 9.4.

9.3. Error Handling and Retransmission

Since the Join Request is mapped to a Non-confirmable CoAP message, OSCORE processing at the JRC will silently drop the request in case of a failure. This may happen for a number of reasons, including failed lookup of an appropriate security context (e.g. the pledge attempting to join a wrong network), failed decryption, positive replay window lookup, formatting errors (possibly due to malicious alterations in transit). Silently dropping the Join Request at the JRC prevents a DoS attack where an attacker could force the pledge to attempt joining one network at a time, until all networks have been tried.

Using a Non-confirmable CoAP message to transport the Join Request also helps minimize the required CoAP state at the pledge and the Join Proxy, keeping it to a minimum typically needed to perform CoAP congestion control. It does, however, introduce some complexity as the pledge needs to implement a retransmission mechanism.

The following binary exponential back-off algorithm is inspired by the one described in [RFC7252]. For each Join Request the pledge sends while waiting for a Join Response, the pledge MUST keep track of a timeout and a retransmission counter. For a new Join Request, the timeout is set to a random value between TIMEOUT_BASE and (TIMEOUT_BASE * TIMEOUT_RANDOM_FACTOR), and the retransmission counter is set to 0. When the timeout is triggered and the retransmission counter is less than MAX_RETRANSMIT, the Join Request is retransmitted, the retransmission counter is incremented, and the timeout is doubled. Note that the retransmitted Join Request passes new OSCORE processing, such that the sequence number in the OSCORE context is properly incremented. If the retransmission counter reaches MAX_RETRANSMIT on a timeout, the pledge SHOULD attempt to join the next advertised 6TiSCH network. If the pledge receives a Join Response that successfully passes OSCORE processing, it cancels the pending timeout and processes the response. The pledge MUST silently discard any response not protected with OSCORE, including error codes. For default values of retransmission parameters, see Section 9.5.

If all join attempts to advertised networks have failed, the pledge SHOULD signal to the user the presence of an error condition, through some out-of-band mechanism.

9.4. Rekeying and Rejoining

This specification handles initial keying of the pledge. For reasons such as rejoining after a long sleep, expiry of the short address, or node-initiated rekeying, the joined node MAY send a new Join Request using the already-established OSCORE security context. The JRC then responds with up-to-date keys and a (possibly new) short address. How the joined node decides when to rekey is out of scope of this document. Mechanisms for rekeying the network are defined in companion specifications.

9.5. Parameters

6JP uses the following parameters:

Name	Default Value
TIMEOUT_BASE	10 s
TIMEOUT_RANDOM_FACTOR	1.5
MAX_RETRANSMIT	4

The values of TIMEOUT_BASE, TIMEOUT_RANDOM_FACTOR, MAX_RETRANSMIT may be configured to values specific to the deployment. The default values have been chosen to accommodate a wide range of deployments, taking into account dense networks.

9.6. Mandatory to Implement Algorithms

The mandatory to implement AEAD algorithm for use with OSCORE is AES-CCM-16-64-128 from [RFC8152]. This is the algorithm used for securing 802.15.4 frames, and hardware acceleration for it is present in virtually all compliant radio chips. With this choice, CoAP messages are protected with an 8-byte CCM authentication tag, and the algorithm uses 13-byte long nonces.

The mandatory to implement hash algorithm is SHA-256 [RFC4231].

The mandatory to implement key derivation function is HKDF [RFC5869], instantiated with a SHA-256 hash.

10. Stateless-Proxy CoAP Option

The CoAP proxy defined in [RFC7252] keeps per-client state information in order to forward the response towards the originator of the request. This state information includes at least the CoAP token, the IPv6 address of the host, and the UDP source port number. If the JP used the stateful CoAP proxy defined in [RFC7252], it would be prone to Denial-of-Service (DoS) attacks, due to its limited memory.

The Stateless-Proxy CoAP option Figure 3 allows the JP to be entirely stateless. This option inserts, in the request, the state information needed for relaying the response back to the client. The proxy still keeps some general state (e.g. for congestion control or request retransmission), but no per-client state.

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

No.	C	U	N	R	Name	Format	Length
TBD	x		x		Stateless-Proxy	opaque	1-255

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 3: 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 the 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 [RFC8152] 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 a 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 a 5.04 Gateway Timeout Response Code 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.

11. Security Considerations

This document recommends that the pledge and JRC are provisioned with unique PSKs. The nonce used for the Join Request and the Join Response is the same, but used under a different key. The design differentiates between keys derived for requests and keys derived for responses by different sender identifiers (0x00 for pledge and 0x01 for JRC). Note that the address of the JRC does not take part in nonce or key construction. Even in the case of a misconfiguration in which the same PSK is used for several pledges, the keys used to protect the requests/responses from/towards different pledges are different, as they are derived using the pledge identifier as Master Salt. The PSK is still important for mutual authentication of the pledge and the JRC. 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.

Being a stateless relay, the JP blindly forwards the join traffic into the network. A simple bandwidth cap on the JP prevents it from forwarding more traffic than the network can handle. This forces attackers to use more than one Join Proxy if they wish to overwhelm the network. Marking the join traffic packets with a non-zero DSCP allows the network to carry the traffic if it has capacity, but encourages the network to drop the extra traffic rather than add bandwidth due to that traffic.

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

At the beginning of the join process, 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 will either fail the security check or time out. The pledge may implement a blacklist in order to filter out undesired EBs and try to join using the next seemingly valid EB. This blacklist alleviates the issue, but is effectively limited by

the node's available memory. Bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode.

12. Privacy Considerations

The join solution specified in this document relies on the uniqueness of the pledge identifier within the namespace managed by the JRC. This identifier is transferred in clear as an OSCORE Context Hint. The use of the globally unique EUI-64 as pledge identifier simplifies the management but comes with certain privacy risks. The implications are thoroughly 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 EUI-64 as the pledge identifier are minimal. In addition, the Join Response message contains a short address which is assigned by the JRC to the pledge. The assigned short address SHOULD be uncorrelated with the long-term pledge identifier. The short address is encrypted in the response. Once the join process completes, the new node uses the short addresses for all further layer 2 (and layer-3) operations. This mitigates the aforementioned privacy risks as the short layer-2 address (visible even when the network is encrypted) is not traceable between locations and does not disclose the manufacturer, as is the case of EUI-64.

13. 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.

13.1. CoAP Option Numbers Registry

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

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

14. 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, and to Klaus Hartke for providing input on the Stateless-Proxy CoAP option. The authors would also like to thank Francesca Palombini, Ludwig Seitz and John Mattsson for participating in the discussions that have helped shape the document.

15. References

15.1. Normative References

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-08 (work in progress), January 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>.

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <https://www.rfc-editor.org/info/rfc2597>.

[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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7252>.

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

15.2. Informative References

[I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 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-09 (work in progress), June 2017.

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-ietf-cbor-cddl-02 (work in progress), February 2018.

[I-D.richardson-6tisch-minimal-rekey]
Richardson, M., "Minimal Security rekeying mechanism for 6TiSCH", draft-richardson-6tisch-minimal-rekey-02 (work in progress), August 2017.

[IEEE802.15.4-2015]
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, <https://www.rfc-editor.org/info/rfc4231>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

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

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7721>.

[RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <https://www.rfc-editor.org/info/rfc8180>.

Appendix A. Example

Figure 4 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the traffic keys and nonces from the PSK. It uses the instantiated context to protect the Join Request addressed with a Proxy-Scheme option, the well-known host name of the JRC in the Uri-Host option, and its EUI-64 as pledge identifier and OSCORE Context Hint. Triggered by the presence of a Proxy-Scheme option, the 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. The JP has learned the IPv6 address of the JRC when it acted as a pledge and joined the network. Once the JRC receives the request, it looks up the correct context based on the Context Hint parameter. It reconstructs OSCORE's external Additional Authenticated Data (AAD) needed for verification based on:

- the Version of the received CoAP header.

- the Algorithm value agreed out-of-band, default being AES-CCM-16-64-128 from [RFC8152].
- the Request ID being set to the value of the "kid" field of the received COSE object.
- the Join Request sequence number set to the value of "Partial IV" field of the received COSE object.
- Integrity-protected options received as part of the request.

Replay protection is ensured by OSCORE and through persistent handling of mutable context parameters. Once the JP receives the Join Response, it authenticates the Stateless-Proxy option before deciding where to forward. The JP sets its internal state to that found in the Stateless-Proxy option, and forwards the Join Response to the correct pledge. Note that the JP does not possess the key to decrypt the COSE object (join_response) present in the payload. The Join Response is matched to the Join Request and verified for replay protection at the pledge using OSCORE processing rules. In this example, the Join Response does not contain the IPv6 address of the JRC, the pledge hence understands the JRC is co-located with the 6LBR.

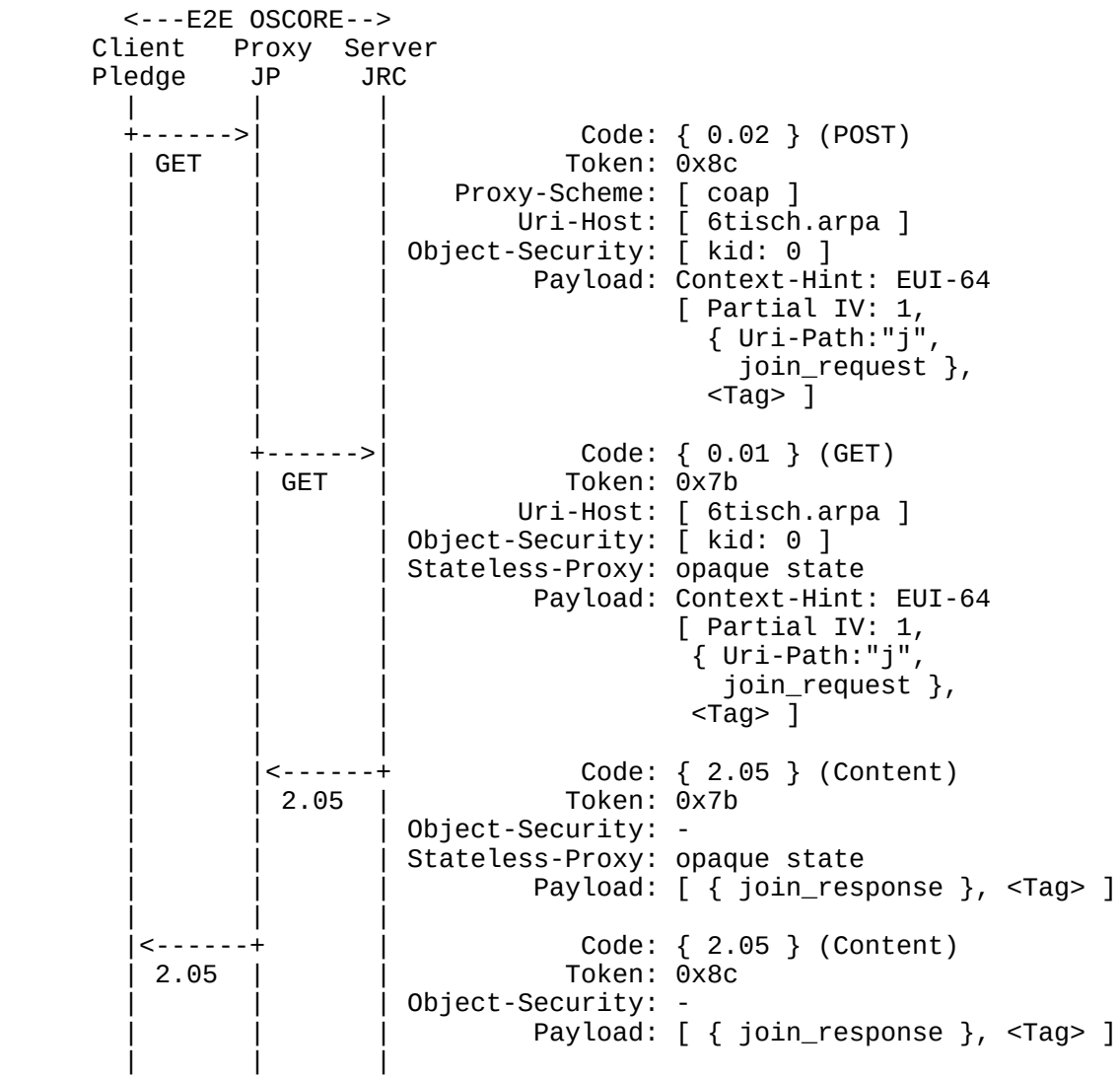

Figure 4: Example of a successful join protocol exchange. { ... } denotes encryption and authentication, [...] denotes authentication.

Where join_request is:

```
join_request:
[
    h'cafe' / PAN ID of the network pledge is attempting to join /
]
```

The join_request encodes to h'8142cafe' with a size of 4 bytes.

And join_response is:

```
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 /
    ]
]
```

The join_response encodes to h'8281a301040241012050e6bf4287c2d7618d6a9687445ffd33e68142af93' with a size of 30 bytes.

Authors' Addresses

Malisa Vucinic (editor)
University of Montenegro
Dzordza Vasingtona bb
Podgorica 81000
Montenegro

Email: malisav@ac.me

Jonathan Simon
Analog Devices
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jonathan.simon@analog.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