6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 26, 2018

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

May 25, 2018

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-06

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 other parameters. The JRC may at any time update the parameters through another request-response exchange secured by OSCORE. This specification defines the Constrained Join Protocol and its CBOR (Concise Binary Object Representation) data structures, 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.

Status of This Memo

This Internet-Draft will expire on November 26, 2018.

Copyright Notice

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 the Constrained Join Protocol (CoJP) 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 and update them at a later time, a new CoAP option, and configures different layers of the 6TiSCH protocol stack for the join process to occur in a secure manner.

The Constrained Join Protocol defined in this document is generic and can be used as-is in modes of IEEE Std 802.15.4 other than TSCH, that 6TiSCH is based on. The Constrained Join Protocol may as well be used in other (low-power) networking technologies where efficiency in terms of communication overhead and code footprint is important. In such a case, it may be necessary to register configuration parameters specific to the technology in question, through the IANA process. The overall join process described in Section 5 and the configuration of the stack is, however, specific to 6TiSCH.

The Constrained Join Protocol 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 CoJP. How the PSK is installed is out of scope of this

document: this may happen through the one-touch provisioning process or by a key exchange protocol that may precede the execution of the 6TiSCH Join protocol.

When the pledge seeks admission to a 6TiSCH network, it first synchronizes to it, by initiating the passive scan defined in [IEEE802.15.4]. 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, link-layer short address and other parameters. 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 specifications using the Concise data definition language (CDDL) [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

The following terms defined in [RFC6775] are also used throughout this document:

- 6LoWPAN Border Router (6LBR)

The term "6LBR" is used interchangeably with the term "DODAG root" defined in [RFC6550], assuming the two entities are co-located, as recommended by [I-D.ietf-6tisch-architecture].

The term "pledge", as used throughout the document, explicitly denotes non-6LBR devices attempting to join over an IEEE Std 802.15.4 network interface. The device that attempts to join as the 6LBR of the network and does so over another network interface is explicitly denoted as the "6LBR pledge". When the text equally applies to the pledge and the 6LBR pledge, the "(6LBR) pledge" form is used.

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]. 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 (6LBR) 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 Std 802.15.4 device. This identifier is used to generate the IPv6 addresses of the (6LBR) 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 (6LBR) 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 (6LBR) 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 (6LBR) pledge, and that the two devices can exchange the parameters.

Parameters that are provisioned to the (6LBR) pledge include:

- Pre-Shared Key (PSK). The JRC additionally needs to store the pledge identifier 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 (6LBR) pledge SHOULD be provisioned with a unique PSK.

- Optionally, a network identifier. Provisioning the network identifier is RECOMMENDED. However, due to the operational constraints the network identifier may not be known at the time when the provisioning is done. In case this parameter is not provisioned to the pledge, the pledge attempts to join one network at a time, which significantly prolongs the join process. In case this parameter is not provisioned to the 6LBR pledge, the 6LBR pledge can receive it from the JRC as part of the join protocol.

- Optionally, any non-default algorithms. The default algorithms are specified in Section 9.5. When algorithm identifiers are not exchanged, the use of these default algorithms is implied.

Additionally, the 6LBR pledge that is not co-located with the JRC needs to be provisioned with:

- Global IPv6 address of the JRC. This address is used by the 6LBR pledge to address the JRC during the join process. The 6LBR pledge may also obtain the IPv6 address of the JRC through other available mechanisms, such as DHCPv6, GRASP, mDNS, the use of which is out of scope of this document. Pledges do not need to be provisioned with this address as they discover it dynamically during the join process.

5. Join Process 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]. 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. The Enhanced Beacon provides the L2 address of the JP and it may also provide its link-local IPv6 address.

2. The pledge configures its link-local IPv6 address and advertises it to the JP using Neighbor Discovery. This step may be omitted if the link-local address has been derived from a known unique interface identifier, such as an EUI-64 address.

3. The pledge sends a Join Request to the JP in order to securely identify itself to the network. The Join Request is forwarded to the JRC.

4. In case of successful processing of the request, the pledge receives a Join Response from the JRC (via the JP). The Join Response contains configuration parameters necessary for the pledge to join the network.

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:

```
     +--------+                 +-------+                 +--------+
     | pledge |                 |  JP   |                 |  JRC   |
     |        |                 |       |                 |        |
     +--------+                 +-------+                 +--------+
        |                          |                          |
        |<---Enhanced Beacon (1)---|                          |
        |                          |                          |
        |<-Neighbor Discovery (2)->|                          |
        |                          |                          |
        |-----Join Request (3a)----|----Join Request (3a)---->| \
        |                          |                          | | CoJP
        |<----Join Response (3b)---|----Join Response (3b)----| /
        |                          |                          |
```

Figure 1: Overview of a successful join process. CoJP stands for Constrained Join Protocol.

As other nodes in the network, the 6LBR node plays the role of the JP. The 6LBR may in addition be co-located with the JRC.

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], 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 9.1.3.

Once the pledge selects the EB, it synchronizes to it and transitions into a low-power mode. It follows the provided schedule which indicates the slots that 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 - Constrained Join Protocol (CoJP) Execution

The pledge triggers the join exchange of the Constrained Join Protocol (CoJP). The join exchange consists of two messages: the Join Request message (Step 3a), and the Join Response message conditioned on the successful security processing of the request (Step 3b). All CoJP messages are exchanged over a secure channel that provides confidentiality, data authenticity and replay protection.

5.3.1. Step 3a - Join Request

The Join Request is a message sent from the pledge to the JP, and which the JP forwards to the JRC. The pledge indicates in the Join Request the role it requests to play in the network as well as the identifier of the network it requests to join. 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.

5.3.2. Step 3b - Join Response

The Join Response is sent by the JRC to the pledge, and is forwarded through the JP. 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 forward the message.

The Join Response contains different parameters needed by the pledge to become a fully operational network node. For example, these parameters are the link-layer key(s) currently in use in the network,

the short link-layer address assigned to the pledge, the IPv6 address of the JRC needed by the pledge to operate as the JP, and others.

5.4. The Special Case of the 6LBR Pledge Joining

The 6LBR pledge performs Section 5.3 of the join process described above, just as any other pledge, albeit over another network interface. There is no JP intermediating the communication between the 6LBR pledge and the JRC, as described in Section 7. The other steps of the described join process do not apply to the 6LBR pledge. How the 6LBR pledge obtains an IPv6 address and triggers the execution of the CoJP protocol is out of scope of this document.

6. Link-layer Configuration

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

The pledge does not initially do any authenticity check of the EB frames, as it does not possess the link-layer key(s) in use. 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 unsecured frames for the duration of the join process. This behavior may be implemented by setting the "secExempt" attribute in the IEEE Std 802.15.4 security configuration tables. 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 verify the freshness nor verify the address of the JP. This opens up a possibility of DoS attack, as discussed in Section 11.

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, as the attacker may fill JP's neighbor table and prevent the discovery of legitimate neighbors. 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 using link-local addressing, 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 IPv6 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 9.1.2; it uses it once joined in order to operate as a JP.

As a special case, the 6LBR pledge is expected to have an additional network interface that it uses in order to obtain the configuration parameters from the JRC and start advertising the 6TiSCH network. This additional interface needs to be configured with a global IPv6 address, by a mechanism that is out of scope of this document. The 6LBR pledge uses this interface to directly communicate with the JRC using global IPv6 addressing.

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 the RPL DIOs [RFC6550] to its IPv6 address. The pledge learns the address of the JRC once joined and upon the reception of the first RPL DIO message, and uses it to operate as a JP.

7.1. Identification of Join Request Traffic

The join request traffic that is proxied by the Join Proxy (JP) 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 distributed 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 do so in response to the join request traffic. In order to permit the intermediate nodes to avoid this, the traffic needs to be tagged.

[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 9.1.3). AF42 has lower drop probability than AF43, giving join response traffic priority in buffers over join request 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 CoJP join exchange in Figure 1 is carried over CoAP [RFC7252] and the secure channel provided by OSCORE [I-D.ietf-core-object-security]. The (6LBR) 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. 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. 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 9.1.1.

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 (6LBR) 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 as per Section 3 of [I-D.ietf-core-object-security].

- the Master Secret MUST be the PSK.

- the Master Salt MUST be empty.

- the ID of the pledge MUST be set to the byte string 0x00. This identifier is used as the OSCORE Sender ID in the security context derivation, as the pledge initially plays the role of a CoAP client.

- the ID of the JRC MUST be set to the byte string 0x4a5243 ("JRC" in ASCII). This identifier is used as the OSCORE Recipient ID in

the security context derivation, as the JRC initially plays the role of a CoAP server.

- the ID Context MUST be set to the pledge identifier.

- 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 Authenticated Encryption with Associated Data (AEAD) algorithm due to the nonce reuse.

This OSCORE security context is used for initial joining of the (6LBR) pledge, where the (6LBR) pledge acts as a CoAP client, as well as for any later parameter updates, where the JRC acts as a CoAP client and the joined node as a CoAP server, as discussed in Section 9.2. A (6LBR) pledge is expected to have exactly one OSCORE security context with the JRC.

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 (6LBR) pledge and the JRC is rare, making security outweigh the cost of writing to persistent memory.

9. Constrained Join Protocol (CoJP)

Constrained Join Protocol (CoJP) is a lightweight protocol over CoAP [RFC7252] and a secure channel provided by OSCORE [I-D.ietf-core-object-security]. CoJP allows the (6LBR) 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, or advertising it in the case of 6LBR pledge. The JRC may update the parameters at any time, by reaching out to the joined node that formerly acted as a (6LBR) pledge. For example, network-wide rekeying can be implemented by updating the keying material on each node.

This section specifies how the CoJP messages are mapped to CoAP and OSCORE, CBOR data structures carrying different parameters, transported within CoAP payload, and the parameter semantics and processing rules.

CoJP 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.

```
               +-----------------------------------+
               |  Constrained Join Protocol (CoJP) |
               +-----------------------------------+
               +-----------------------------------+  \
               |         Requests / Responses      |  |
               |-----------------------------------|  |
               |               OSCORE              |  | CoAP
               |-----------------------------------|  |
               | Messaging Layer / Message Framing |  |
               +-----------------------------------+  /
               +-----------------------------------+
               |                UDP                |
               +-----------------------------------+
```

Figure 2: Abstract layering of CoJP.

When a (6LBR) pledge requests admission to a given network, it undergoes the CoJP join exchange that consists of:

- the Join Request message, sent by the (6LBR) pledge to the JRC, potentially proxied by the JP. The Join Request message and its mapping to CoAP is specified in Section 9.1.1.

- the Join Response message, sent by the JRC to the (6LBR) 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 (6LBR) pledge is authorized to join the network. The Join Response message is potentially proxied by the JP. The Join Response message and its mapping to CoAP is specified in Section 9.1.2.

When the JRC needs to update the parameters of a joined node that formerly acted as a (6LBR) pledge, it executes the CoJP parameter update exchange that consists of:

- the Parameter Update message, sent by the JRC to the joined node that formerly acted as a (6LBR) pledge. The Parameter Update message and its mapping to CoAP is specified in Section 9.2.1.

- the Parameter Update Response message, sent by the joined node to the JRC in response to the Parameter Update message to signal successful reception of the updated parameters. The Parameter Update Response message and its mapping to CoAP is specified in Section 9.2.2.

The payload of CoJP messages is encoded with CBOR [RFC7049]. The CBOR data structures that may appear as the payload of different CoJP messages are specified in Section 9.3.

9.1. Join Exchange

This section specifies the messages exchanged when the (6LBR) pledge requests admission and configuration parameters from the JRC.

9.1.1. Join Request Message

The Join Request message 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". This is an anycast type of identifier of the JRC that is resolved to its IPv6 address by the JP or the 6LBR pledge.

- 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 kid context is set to the ID context, which in turn is set to the pledge identifier. The OSCORE kid context allows the JRC to retrieve the security context for a given pledge.

- The payload is a Join_Request CBOR object, as defined in Section 9.3.1.

9.1.2. Join Response Message

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

- The response Code is 2.04 (Changed).

- The payload is a Configuration CBOR object, as defined in Section 9.3.2.

9.1.3. 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). 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 9.4.

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.

9.2. Parameter Update Exchange

During the network lifetime, parameters returned as part of the Join Response may need to be updated. One typical example is the update of link-layer keying material for the network, a process known as rekeying. This section specifies a generic mechanism when this parameter update is initiated by the JRC.

At the time of the join, the (6LBR) pledge acts as a CoAP client and requests the network parameters through a representation of the "/j" resource, exposed by the JRC. In order for the update of these parameters to happen, the JRC needs to asynchronously contact the joined node. The use of the CoAP Observe option for this purpose is not feasible due to the change in the IPv6 address when the pledge becomes the joined node and obtains a global address.

Instead, once the (6LBR) pledge receives and successfully validates the Join Response and so becomes a joined node, it switches its CoAP role and becomes a server. The joined node exposes the "/j" resource that is used by the JRC to update the parameters. Consequently, the JRC operates as a CoAP client when updating the parameters. The request/response exchange between the JRC and the (6LBR) pledge happens over the already-established OSCORE secure channel.

9.2.1. Parameter Update Message

The Parameter Update message that the JRC sends to the joined node SHALL be mapped to a CoAP request:

o The request method is POST.

o The type is Confirmable (CON).

- 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. When a joined node receives a request with the Sender ID set to 0x4a5243 (ID of the JRC), it is able to correctly retrieve the security context with the JRC.

- The payload is a Configuration CBOR object, as defined in Section 9.3.2.

The JRC has implicit knowledge on the global IPv6 address of the joined node, as it knows the pledge identifier that the joined node used when it acted as a pledge, and the IPv6 network prefix. The JRC uses this implicitly derived IPv6 address of the joined node to directly address CoAP messages to it.

9.2.2. Parameter Update Response Message

The Parameter Update Response message that the joined node sends to the JRC SHALL be mapped to a CoAP response:

- The response Code is 2.04 (Changed).

- The payload is empty.

9.3. CoJP Objects

This section specifies the structure of CoJP CBOR objects that may be carried as the payload of CoJP messages. Some of these objects may be received both as part of the CoJP join exchange when the device operates as a (CoJP) pledge, or the parameter update exchange, when the device operates as a joined (6LBR) node.

9.3.1. Join Request Object

The Join_Request structure is built on a CBOR map object.

The set of parameters that can appear in a Join_Request object is summarized below. The defined labels can be found below, the details of this registry are in section "CoJP Parameters" registry Section 13.2.

- role: The identifier of the role that the pledge requests to play in the network once it joins, encoded as an unsigned integer. Possible values are specified in Table 1. This parameter MAY be included. In case the parameter is omitted, the default value of 0, i.e. the role "6TiSCH Node", MUST be assumed.

- network identifier: The identifier of the network, as discussed in Section 3, encoded as a CBOR byte string. This parameter may appear both in the Join Request and in the Join Response. When present in the Join Request, it hints to the JRC the network that the pledge is requesting to join, enabling the JRC to manage multiple networks. The pledge obtains the value of the network identifier from the received EB frames. This parameter MUST be included in a Join_Request object if the role parameter is set to "6TiSCH Node". This parameter MAY be included if the role parameter is set to "6LBR". The inclusion of this parameter by the 6LBR pledge depends on whether the parameter was exchanged during the one-touch process, which in turn depends on the operational constraints.

The CDDL fragment that represents the text above for the Join_Request follows.

```
Join_Request = {
    ? 1 : uint              ; role
    ? 5 : bstr              ; network identifier
}
```

Name	Value	Description	Reference
6TiSCH Node	0	The pledge requests to play the role of a regular 6TiSCH node, i.e. non-6LBR node.	[[this document]]
6LBR	1	The pledge requests to play the role of 6LoWPAN Border Router (6LBR).	[[this document]]

Table 1: Role values.

9.3.2. Configuration Object

The Configuration structure is built on a CBOR map object. The set of parameters that can appear in a Configuration object is summarized below. The defined labels can be found below, the details of this registry are in section "CoJP Key Usage Registry" Section 13.3.

- link-layer key set: An array encompassing a set of cryptographic keys and their identifiers that are currently in use in the network, or that are scheduled to be used in the future. The encoding of individual keys is described in Section 9.3.2.1. The link-layer key set parameter MAY be included in a Configuration

object. When present, the link-layer key set parameter MUST contain at least one key. How the keys are installed and used differs for the 6LBR and other nodes. When 6LBR receives this parameter, it MUST remove any old keys it has installed from the previous key set and immediately install and start using the new keys for all outgoing and incoming traffic. When a non-6LBR node receives this parameter, it MUST install the keys, use them for any incoming traffic matching the key identifier, but keep using the old keys for all outgoing traffic. A non-6LBR node accepts any frames for which it has keys: both old and new keys. Upon reception and successful security processing of a link-layer frame secured with a key from the new key set, a non-6LBR node MUST remove any old keys it has installed from the previous key set. From that moment on, a non-6LBR node MUST use the keys from the new key set for all outgoing traffic. In the case when the pledge is joining for the first time, before sending the first outgoing frame secured with a received key, the pledge needs to successfully complete the security processing of an incoming frame. To do so, the pledge can wait to receive a new frame or it can also store an EB frame that it used to find the JP and use it for immediate security processing upon reception of the key set. The described mechanism permits the JRC to provision the new key set to all the nodes while the network continues to use the existing keys. When the JRC is certain that all (or enough) nodes have been provisioned with the new keys, then the JRC updates the 6LBR. In the special case when the JRC is co-located with the 6LBR, it can simply trigger the sending of a new broadcast frame (e.g. EB), secured with a key from the new key set. The frame goes out with the new key, and upon reception and successful security processing of the new frame all receiving nodes will switch to the new active keys. Outgoing traffic from those nodes will then use the new key, which causes an update of additional peers, and the network will switch over in a flood-fill fashion.

- link-layer short address: IEEE Std 802.15.4 short address assigned to the pledge. The short address structure is described in Section 9.3.2.2. The link-layer short address parameter MAY be included in a Configuration object. When a node receives this parameter as part of the Parameter Update message, it MUST update its link-layer short address to the one received.

- JRC address: the IPv6 address of the JRC, encoded as a byte string, with the length of 16 bytes. If the length of the byte string is different than 16, the parameter MUST be discarded. If the JRC is not co-located with the 6LBR and has a different IPv6 address than the 6LBR, this parameter MUST be included. In the special case where the JRC is co-located with the 6LBR and has the same IPv6 address as the 6LBR, this parameter MAY be included. If

the JRC address parameter is not present in the Join Response, this indicates that the JRC has the same IPv6 address as the 6LBR. The joined node can then discover the IPv6 address of the JRC through network control traffic. See Section 7.

- network identifier: the identifier of the network, as discussed in Section 3, encoded as a byte string. When present in the Join Response, this parameter is only valid when received by the 6LBR pledge. The parameter indicates to the 6LBR the value of the network identifier it should advertise at the link layer. This parameter MUST NOT be included in the Join Response if the role parameter from the corresponding Join Request indicated 0, i.e. the role "6TiSCH Node". In the case where the corresponding Join_Request object does not contain the network identifier parameter, this parameter MUST be included. When the corresponding Join_Request object does contain the network identifier parameter, this parameter MAY be included in the Configuration object. This may happen if the JRC decides to overwrite the network identifier provisioned during the one-touch process. The value of the network identifier parameter from the Configuration object SHOULD take precedence over the value provisioned during the one-touch process.

- network prefix: the IPv6 network prefix, encoded as a byte string. The length of the byte string determines the prefix length. This parameter is only valid when received by the 6LBR pledge. The parameter indicates to the 6LBR the value of the IPv6 network prefix. This parameter MAY be included in the Join Response if the role parameter from the corresponding Join_Request object indicated 1, i.e. the role "6LBR". This parameter MUST NOT be included in the Join Response if the role parameter from the corresponding Join_Request object indicated 0, i.e. the role "6TiSCH Node".

The CDDL fragment that represents the text above for the Configuration follows. Structures Link_Layer_Key and Short_Address are specified in Section 9.3.2.1 and Section 9.3.2.2.

```
Configuration = {
    ? 2 : [ +Link_Layer_Key ],   ; link-layer key set
    ? 3 : Short_Address,         ; link-layer short address
    ? 4 : bstr                   ; JRC address
    ? 5 : bstr                   ; network identifier
    ? 6 : bstr                   ; network prefix
}
```

Name	Label	CBOR type	Description	Reference
role	1	unsigned integer	Identifies the role parameter.	[[this document]]
link-layer key set	2	array	Identifies the array carrying one or more link-level cryptographic keys.	[[this document]]
link-layer short address	3	array	Identifies the assigned link-layer short address	[[this document]]
JRC address	4	byte string	Identifies the IPv6 address of the JRC	[[this document]]
network identifier	5	byte string	Identifies the network identifier parameter	[[this document]]
network prefix	6	byte string	Identifies the IPv6 prefix of the network	[[this document]]

Table 2: Join Response map labels.

9.3.2.1. Link-Layer Key

The Link_Layer_Key structure encompasses the parameters needed to configure the link-layer security module: the value of the cryptographic key, the key identifier, the link-layer algorithm identifier, and the security level and the frame types that it should be used with, both for outgoing and incoming security operations.

For encoding compactness, Link_Layer_Key object is not enclosed in a top-level CBOR object. Rather, it is transported as a consecutive group of CBOR elements, with some being optional. To be able to decode the keys that are present in the link-layer key set, and to identify individual parameters of a single Link_Layer_Key object, the CBOR decoder needs to differentiate between elements based on the CBOR type. For example, when the decoder determines that the current element in the array is a byte string, it is certain that it is processing the last element of a given Link_Layer_Key object.

The set of parameters that can appear in a Link_Layer_Key object is summarized below, in order:

- key_index: The identifier of the key, encoded as a CBOR unsigned integer. This parameter MUST be included. The parameter uniquely identifies the key and is used to retrieve the key for incoming traffic. In case of [IEEE802.15.4], the decoded CBOR unsigned integer value sets the "secKeyIndex" parameter that is signaled in all outgoing and incoming frames secured with this key. If the decoded CBOR unsigned integer value is larger than the maximum link-layer key identifier, which is 255 in [IEEE802.15.4]), the key is considered invalid. Additionally, in case of [IEEE802.15.4], the value of 0 is considered invalid. In case the key is considered invalid, the implementation MUST discard the key and attempt to decode the next key in the array.

- key_usage: The identifier of the link-layer algorithm, security level and link-layer frame types that can be used with the key, encoded as a CBOR unsigned or negative integer. This parameter MAY be included. Possible values and the corresponding link-layer settings are specified in IANA "CoJP Key Usage" registry (Section 13.3). In case the parameter is omitted, the default value of 0 from Table 3 MUST be assumed.

- key_value: The value of the cryptographic key, encoded as a byte string. This parameter MUST be included. If the length of the byte string is different than the corresponding key length for a given algorithm specified by the key_usage parameter, the key MUST be discarded and the decoder should attempt to decode the next key in the array.

The CDDL fragment that represents the text above for the Link_Layer_Key follows.

```
Link_Layer_Key = (
      key_index          : uint,
    ? key_usage          : uint / nint,
      key_value          : bstr,
)
```

Name	Value	Algorithm	Description	Reference
6TiSCH-K1K2-ENC-MIC-32	0	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, ENC-MIC-32 for DATA	[[this document]]

			and ACKNOWLEDGMENT.	
6TiSCH-K1K2-ENC-MIC-64	1	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC-128	2	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC-32	3	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC-64	4	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC-128	5	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1-MIC-32	6	IEEE802154-AES-CCM-128	Use MIC-32 for EBs.	[[this document]]
6TiSCH-K1-MIC-64	7	IEEE802154-AES-CCM-128	Use MIC-64 for EBs.	[[this document]]
6TiSCH-K1-MIC-128	8	IEEE802154-AES-CCM-128	Use MIC-128 for EBs.	[[this document]]
6TiSCH-K2-MIC-32	9	IEEE802154-AES-	Use MIC-32	[[this d

		CCM-128	for DATA and ACKNOWLEDGMENT.	ocument]]
6TiSCH-K2-MIC-64	10	IEEE802154-AES-CCM-128	Use MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-MIC-128	11	IEEE802154-AES-CCM-128	Use MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC-32	12	IEEE802154-AES-CCM-128	Use ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC-64	13	IEEE802154-AES-CCM-128	Use ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC-128	14	IEEE802154-AES-CCM-128	Use ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]

Table 3: Key Usage values.

9.3.2.2. Short Address

The Short_Address object represents an address assigned to the pledge that is unique locally in the network. It is encoded as a CBOR array object, containing, in order:

o address: The assigned locally-unique address, encoded as a byte string. This parameter MUST be included. In case of [IEEE802.15.4], if the length of the byte string is different than 2, the address is considered invalid. In case of [IEEE802.15.4], the value of this parameter is used to set the short address of IEEE Std 802.15.4 module. In case the address is considered

invalid, the decoder MUST silently ignore the Short_Address object.

- lease_time: The validity of the address in seconds after the reception of the CBOR object, encoded as a CBOR unsigned integer. This parameter MAY be included. The node MUST stop using the assigned short address after the expiry of the lease_time interval. It is up to the JRC to renew the lease before the expiry of the previous interval. The JRC updates the lease by executing the Parameter Update exchange with the node and including the Short_Address in the Configuration object, as described in Section 9.2. In case the address lease expires, the node SHOULD initiate a new join exchange, as described in Section 9.1. In case this parameter is omitted, the value of positive infinity MUST be assumed, meaning that the address is valid for as long as the node participates in the network.

The CDDL fragment that represents the text above for the Short_Address follows.

```
Short_Address = [
      address           : bstr,
    ? lease_time        : uint
]
```

9.4. Parameters

CoJP 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.

9.5. 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 IEEE Std 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.

The Stateless-Proxy CoAP option (see Figure 3) allows the proxy to be entirely stateless. The proxy inserts this option in the request to carry 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. For security reasons, the option value MUST be authenticated, MUST include a freshness indicator (e.g. a sequence number or timestamp) and MAY be encrypted. The proxy may use a 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, and the client eventually times out.

11. Security Considerations

This document recommends that the (6LBR) 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. 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 (6LBR) 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 temporary 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 kid context. 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 (6LBR) 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	\[\[this document\]\]

13.2. CoJP Parameters Registry

This section defines a sub-registries within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Parameters Registry".

The columns of the registry are:

Name: This is a descriptive name that enables an easier reference to the item. It is not used in the encoding.

Label: The value to be used to identify this parameter. The label is an unsigned integer.

CBOR type: This field contains the CBOR type for the field.

Description: This field contains a brief description for the field.

Reference: This field contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 2.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

13.3. CoJP Key Usage Registry

This section defines a sub-registries within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Key Usage Registry".

The columns of this registry are:

Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Value: This is the value used to identify the key usage setting. These values MUST be unique. The value is an integer.

Algorithm: This is a descriptive name of the link-layer algorithm in use and uniquely determines the key length. The name is not used in the encoding.

Description: This field contains a description of the key usage setting. The field should describe in enough detail how the key is to be used with different frame types, specific for the link-layer technology in question.

References: This contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 3.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

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, Goeran Selander, Xavier Vilajosana, Pascal Thubert 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.

The IANA considerations for the three created registries is copied verbatim from RFC8392 at the suggestion of Mike Jones.

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-13 (work in progress), May 2018.

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

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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-11 (work in progress), March 2018.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-14 (work in progress), April 2018.

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

[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.

[IEEE802.15.4]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4 Standard for Low-Rate Wireless Networks", n.d..

[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 kid context. 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. 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 kid context parameter. OSCORE data authenticity verification ensures that the request has not been modified in transit. In addition, replay protection is ensured 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 CBOR object (configuration) 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.

```
   <---E2E OSCORE-->
 Client      Proxy     Server
 Pledge       JP        JRC
   |          |          |
   |  Join    |          |            Code: { 0.02 } (POST)
   | Request  |          |           Token: 0x8c
   +--------->|          |    Proxy-Scheme: [ coap ]
   |  POST    |          |        Uri-Host: [ 6tisch.arpa ]
   |          |          | Object-Security: [ kid: 0 ]
   |          |          |         Payload: kid_context: EUI-64
   |          |          |                  [ Partial IV: 1,
   |          |          |                    { Uri-Path:"j",
   |          |          |                      join_request },
   |          |          |                      <Tag> ]
   |          |          |
   |          |  Join    |            Code: { 0.01 } (GET)
   |          | Request  |           Token: 0x7b
   |          +--------->|        Uri-Host: [ 6tisch.arpa ]
   |          | POST     | Object-Security: [ kid: 0 ]
   |          |          | Stateless-Proxy: opaque state
   |          |          |         Payload: kid_context: EUI-64
   |          |          |                  [ Partial IV: 1,
   |          |          |                    { Uri-Path:"j",
   |          |          |                      join_request },
   |          |          |                      <Tag> ]
   |          |          |
   |          |  Join    |            Code: { 2.05 } (Content)
   |          | Response |           Token: 0x7b
   |          |<---------+ Object-Security: -
   |          | 2.04     | Stateless-Proxy: opaque state
   |          |          |         Payload: [ { configuration }, <Tag> ]
   |          |          |
   |  Join    |          |            Code: { 2.05 } (Content)
   | Response |          |           Token: 0x8c
   |<---------+          | Object-Security: -
   | 2.04     |          |         Payload: [ { configuration }, <Tag> ]
   |          |          |
```

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

Where the join_request object is:

```
  join_request:
  {
      5 : h'cafe' / PAN ID of the network pledge is attempting to join /
  }
```

Since the role parameter is not present, the default role of "6TiSCH Node" is implied.

The join_request object encodes to h'a10542cafe' with a size of 5 bytes.

And the configuration object is:

```
   configuration:
   {
       2 : [           / link-layer key set /
             1,        / key_index /
             h'e6bf4287c2d7618d6a9687445ffd33e6' / key_value /
           ],
       3 : [           / link-layer short address /
             h'af93'   / assigned short address /
           ]
   }
```

Since the key_usage parameter is not present in the link-layer key set object, the default value of "6TiSCH-K1K2-ENC-MIC-32" is implied. Similarly, since the lease_time parameter is not present in the link-layer short address object, the default value of positive infinity is implied.

The configuration object encodes to

h'a202820150e6bf4287c2d7618d6a9687445ffd33e6038142af93' with a size of 26 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